UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

Laboratorio 4

Integrantes: Marco Hernández

Curso: Redes de computadores

Profesor: Carlos González

Ayudante: Nicole Reyes

4 de Marzo de 2021

Tabla de contenidos

1. Introducción

En la experiencia anterior se estudió la modulación de una señal análoga, dentro de las cuales se encuentra la modulación por amplitud (AM) y la modulación por frecuencias (FM), que son técnicas que ayudan a ajustar la señal a un medio de transmisión deseado.

Sin embargo, al modulación también se extiende a las señales digitales, donde la información se transmite en forma binaria(0 y 1). Existen varias maneras de modular una señal digital, dentro de las cuales se encuentran la modulación digital por amplitud, por frecuencia, por fase, entre otras.

En la presente experiencia se generaran arreglos que representen una señal digital, la cual sera modulada por amplitud, en especifica su variación OOK, para después agregar ruido blanco gaussiano a la señal modulada, demodularla y ver los errores de lectura que se producen, comparando la señal demodulada con la señal original. Todas las actividades mencionadas anteriormente se desarrollaran por medio del desarrollo de un programa en el lenguaje de programación python, y se utilizaran las librerías scipy, random, numpy y matplotlib.

1.1. Objetivos

1. Implementar la función de modulación digital
2. Implementar Función de demodulación digital
3. Implementar una función que agregue ruido AWGN a una señal
4. Analizar la efectividad de las funciones anteriores en señales de largos grandes

2. Marco Teórico

A continuación se pasa a mencionar conceptos importantes para el desarrollo de la experiencia de laboratorio

2.1. Definición de señal

Como se menciona anteriormente, en la presente experiencia se trabajara con una señal de audio, pero antes de extraer toda la información del archivo, es necesario saber que es una señal. " *Una señal son ondas electromagnéticas (rango de frecuencias) propagadas a través de un medio de transmisión...*"Dakar (2010). Por lo tanto, una señal, es una energía, la cual es interpretada por diversos dispositivos y existen de distintos tipos como lo son

- Señales analógicas: Son usadas para la transmisión de elementos de vídeo y de sonido. Son señales de tipo continuo, con un comportamiento senoidal.(viu (2018))
- Señales digitales: La información de la señal se transmite utilizando códigos en binario, transformando las señales senoidales a ondas del tipo cuadradas, dando paso a una señal que no es continua. (viu (2018))

2.2. Modulación

La modulación es una técnica que se usa como base para las señales en sistemas de comunicación. La modulacion es un paso para poder ajustar las señales a los medios que se disponen para poder transmitirla

"Many applications require the use of signals that are not well matched to the required media.... We can often modify the signals to obtain a better match. Today we will introduce simple matching strategies based on modulation."(Muchas aplicaciones requieren del uso de señales que no se ajustan al medio requerido... Podemos de vez en cuando modificar las señales para obtener mejor partido de la señal. Hoy vamos a introducir estrategias de ajuste simples basadas en modulación). Oppenheim (2001)

La modulación digital cumple el principio anteriormente mencionado pero para señales digitales, existen muchos tipos de modulación para las señales digitales, algunas son

las siguientes:

- Modulación ASK: ´´La modulación por desplazamiento de amplitud, en inglés Amplitude-shift keying (ASK), es una forma de modulación en la cual se representan los datos digitales como variaciones de amplitud de la onda portadora en función de los datos a enviar."ASK (2020)

- Modulación FSK : ´´La modulación por desplazamiento de frecuencia o FSK —del inglés Frequency Shift Keying— es una técnica de modulación para la transmisión digital de información utilizando dos o más frecuencias diferentes para cada símbolo. La señal moduladora solo varía entre dos valores de tensión discretos formando un tren de pulsos donde uno representa un "1z el otro representa el "0".FSK (2020)

- Modulación PSK :´´La modulación por desplazamiento de fase o PSK (Phase Shift Keying) es una forma de modulación angular que consiste en hacer variar la fase de la portadora entre un número determinado de valores discretos. La diferencia con la modulación de fase convencional (PM) es que mientras en esta la variación de fase es continua, en función de la señal moduladora, en la PSK la señal moduladora es una señal digital y, por tanto, con un número de estados limitado."PSK (2020)

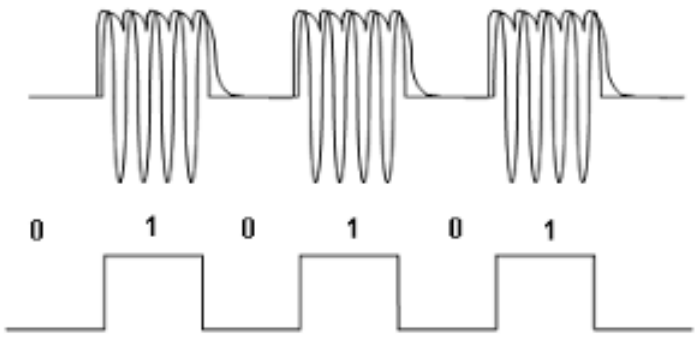


Figura 1: Ejemplo de modulación ASK.

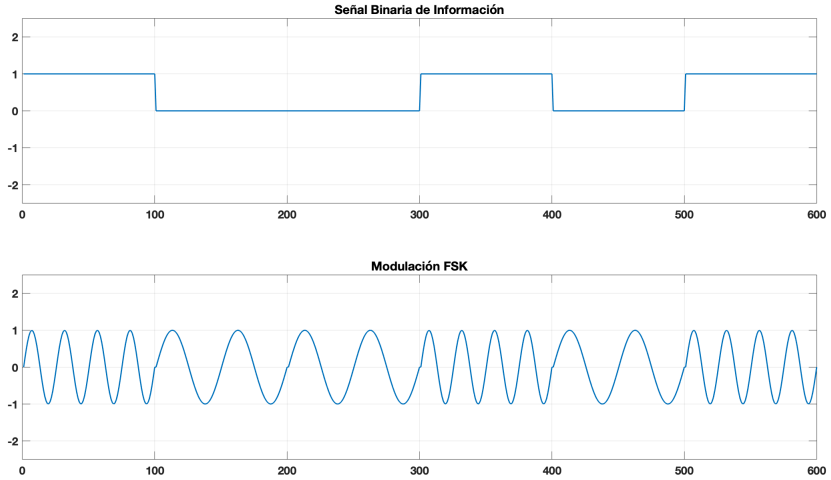


Figura 2: Ejemplo de modulación FSK.

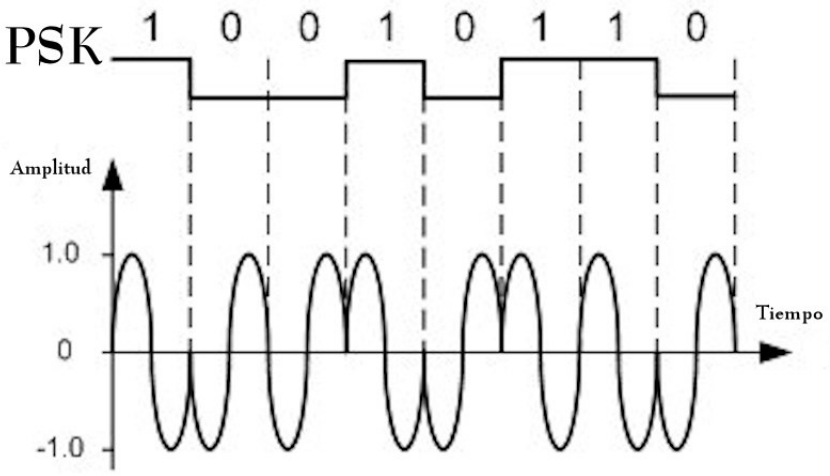


Figura 3: Ejemplo de modulación PSK.

2.3. Ruido AWGN

El ruido AWGN o también conocido como ruido blanco gaussiano, es una señal generada de manera aleatoria para hacer pruebas sobre un equipo y ver la tolerancia al ruido que puede tener :

´´Un generador de ruido produce ruido eléctrico (es decir, una señal aleatoria) para probar el equipo y medir parámetros clave, como la figura de ruido, la respuesta de

frecuencia, etc. Agregando ruido a propósito, los diseñadores pueden prepararse mejor y medir su efecto en el sistema."AWG (2020)

El ruido AWGN es un tipo especial de ruido blanco, el cual tiene la característica especial de que sigue una distribución estadística gaussiana o normal

´´... debido a la naturaleza aleatoria de una fuente de ruido, se utiliza un modelo matemático para calcular la probabilidad de eventos. La distribución gaussiana, o una distribución normal, tiene un promedio de cero en el dominio del tiempo, y se representa como una curva en forma de campana que es simétrica respecto al eje vertical central."AWG (2020)

2.4. Librerías de Python

En cuanto a las herramientas que se usaran a nivel de programación, para desarrollar la experiencia de laboratorio se usara el lenguaje de programación Python, con las herramientas que otorga la librería SciPy de python la cual se puede ver en sci (2020).

A diferencia de experiencias anteriores, se usara principalmente la librería numpy de python, para generar los números aleatorios y principalmente para la generación del arreglo con distribución normal para el ruido AWGN

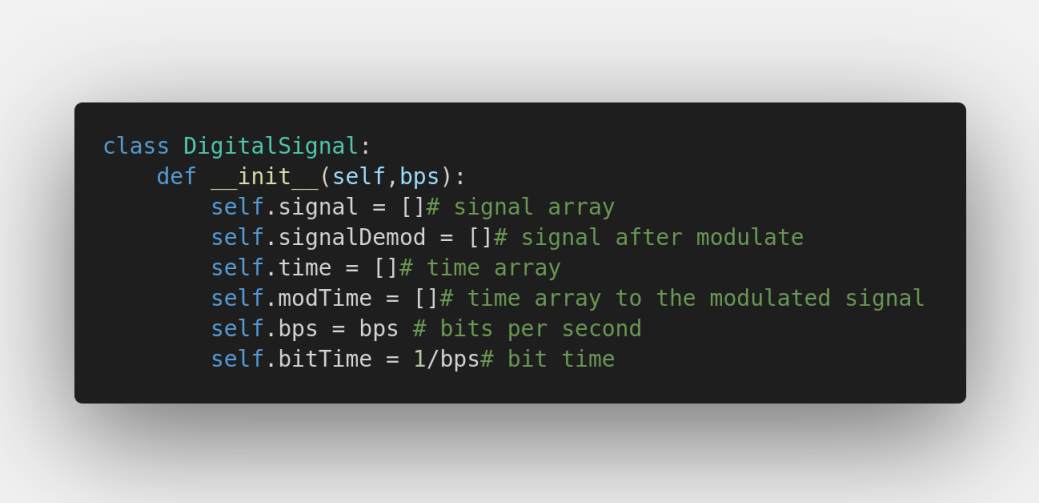
Para las representaciones gráficas de los resultados que se obtengan, se utilizara la librería matplotlib, la cual otorga todas las herramientas necesarias para desplegar gráficos con la información necesaria.

3. Desarrollo de la experiencia

A continuación se pasa a explicar como se llevaron a cabo las actividades propuestas, se detalla primer la solución y después se muestra el código creado para ello.

3.1. Estructura inicial

Para el desarrollo de la experiencia se crea una clase llamada *DigitalSignal*, la cual contendrá los atributos y métodos necesarios para manejar una señal digital, en ella se inicializan los arreglos de la señal, la señal modulada y demodulada, además de los arreglos de tiempo de cada una. Cabe destacar que para crear el objeto de una señal digital se debe pasar como argumento la tasa de bits por segundo de la señal, para después identificar el tiempo que tendrá cada bit



```
class DigitalSignal:
    def __init__(self,bps):
        self.signal = []# signal array
        self.signalDemod = []# signal after modulate
        self.time = []# time array
        self.modTime = []# time array to the modulated signal
        self.bps = bps # bits per second
        self.bitTime = 1/bps# bit time
```

Figura 4: Clase DigitalSignal.

Después, para generar la señal digital en si, se hará con el método *randomSignal* el cual requiere de un largo de arreglo, es decir, cuantos bits tendrá la señal y también que se le indique un numero que sera la semilla para generar la señal de manera aleatoria.

```
# method to generate a digital signal randomly with a previous seed determined by the user
    # inputs:
        # lenght: number ob bits to create the digital signal
        # seed: seed to the random numbers
    def randomSignal(self,lenght,seed):
        init = self.bitTime
        random.seed(seed)
        for i in range(0,lenght):
            self.signal.append(random.randrange(0,2))
            self.time.append(init)
            init= init + self.bitTime
```

Figura 5: Código método para generar señal digital.

3.2. Modulación Digital

Para llevar a cabo la modulación digital, se dejo a criterio libre elegir el tipo de modulación que se implementara, porque como se menciona anteriormente, existen muchos tipos de modulación. En la presente experiencia se opta por una modulación tipo ASK, pero un caso especial de este tipo de modulación que es la OOK, cuya particularidad es que cuando se lee uno de los signos, en este caso 0, y cuando sea 1, se usa una señal senoidal con una amplitud determinada, la señal cuando el símbolo que se lee es 1 es la siguiente:

```
A = 2 # amplitud del coseno

# function to generate signal when a bit of the digital signal is 1
# inputs:
    # t : time value
    # fc: frecuency of the cos function
# output: cos function value in t
def cosFunction(t,fc):
    return A*np.cos(2*np.pi*fc*t)
```

Figura 6: Código de función cuando la señal es 1.

Teniendo lo anterior en cuenta, a continuación se enseña la función de modulación:

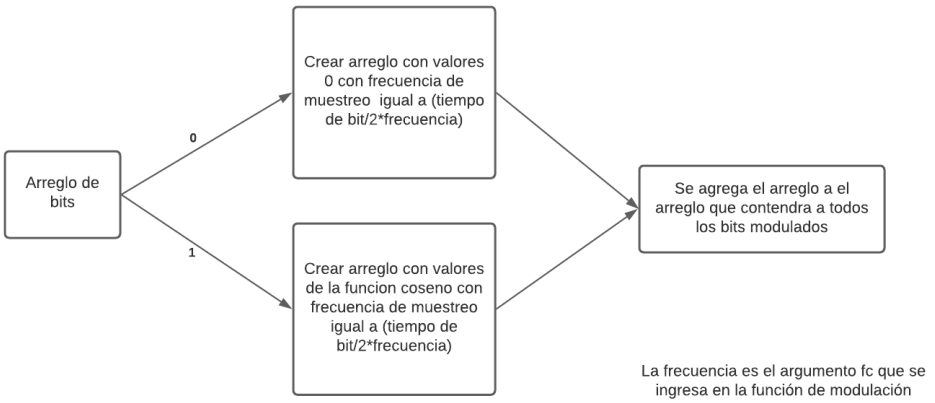


Figura 7: Código de función modulación.

```
# method to modulate by frequency a digital signal
        #fc : frecquency to apply to the cos
    def ookMod(self,fc):
        t = []
        init = 0
        i = 0
        modulated = []
        self.sampleStep = self.bitTime/(2*fc)
        for bit in self.signal:
            bitSignal = []
            tn = np.linspace(init,self.time[i], 2*fc)

            if (bit == 0):
                for t_i in tn:
                    value = 0.0 # when the bit is 0, the value is also 0
                    bitSignal.append(value)
                    t.append(t_i)
            elif (bit == 1):
                for t_i in tn:
                    value = cosFunction(t_i,fc)
# when the bit is 1, the value is the cos function in the time
                    bitSignal.append(value)
                    t.append(t_i)
            modulated.append(bitSignal)
            init = self.time[i]
            i+=1
        self.modTime = t
        self.signalModulated = np.array(modulated)
```

Figura 8: Código de función modulación.

Los resultados del código implementado son los siguientes :

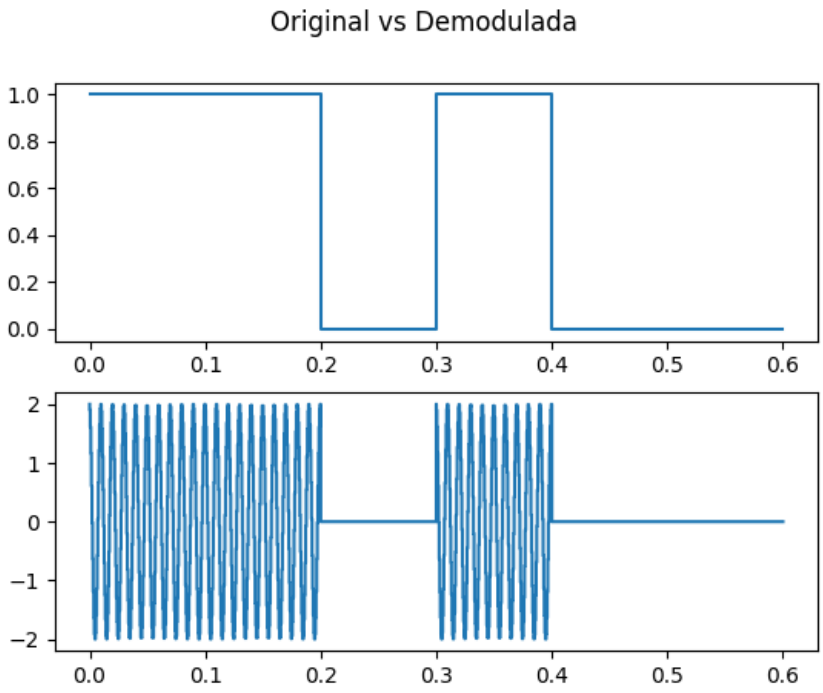


Figura 9: Resultados modulación.

3.3. Demodulación Digital

Una vez hecha la modulación, toca hacer la demodulación, al tratarse de una modulación OOK la demodulación se hace de manera simple, esta consiste en leer cada uno de los arreglos que representa cada bit y se calcula el valor máximo que se encuentra en cada arreglo, si este supera la amplitud de la señal cuando se trata de la función senoidal, el bit que se lee es un 1 y en caso contrario es un 0. El código que realiza esta actividad es la siguiente

```
# method to demodulate a ook modulated signal
        # inputs:
            # timeArray: time array of the signal modulated
            # signalArray: array of the signal to demodulate
        # output: array of signal demodulated
    def demodulate(self,timeArray,signalArray):
        maxValue = 0
        i = 0
        signalDemodulated = []
        for bitSignal in signalArray:
            maxValue = np.max(bitSignal)
            if maxValue >= A:
                signalDemodulated.append(1)
            else:
                signalDemodulated.append(0)
        return signalDemodulated
```

Figura 10: Código de función demodulación.

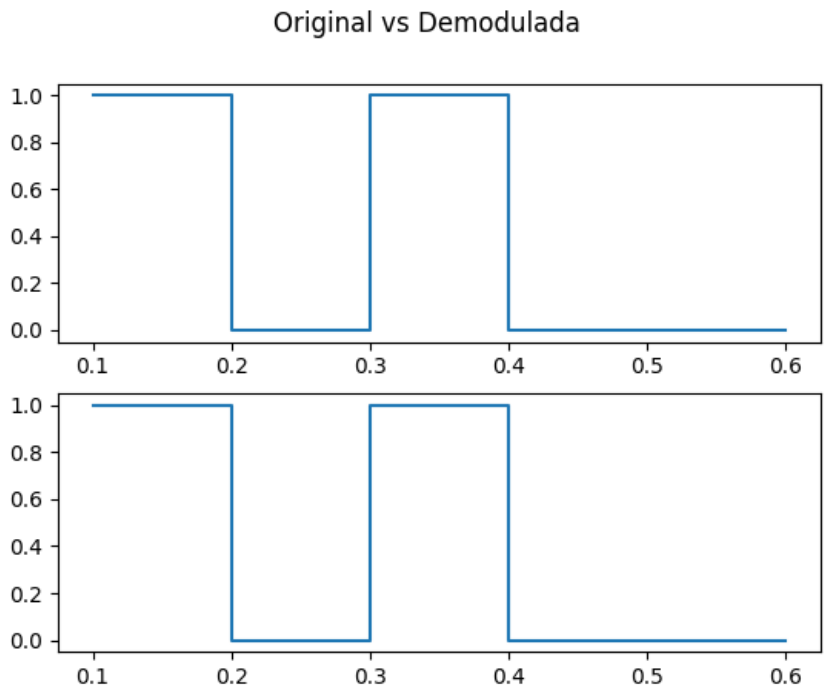


Figura 11: Resultados demodulación.

3.4. Simulador canal AWGN

El ultimo método que se incluye en la clase de señal digital es la que crea la simulación de un canal AWGN en la cual se entrega como parámetro de entrada la razón SNR en decibeles para poder calcular la magnitud del ruido y como este afecta a la señal, de manera que a menor SNR mayor es el ruido presente y por ende más difícil es en teoría extraer la información de la señal, El código del método que retorna una señal que pasa por un canal AWGN es el siguiente:

```python
#method to add AWGN noise to a modulated signal
    # inputs:
        # snr_db: level of noise expresed in SNR in decibels
    #output: array of signal modulated with noise
    def awgnAdd(self,snr_db):
        self.signalPower = np.array(self.signalModulated)**2
        self.signalPowerDb = 10 * np.log10(self.signalPower)
        sig_avg = abs(np.mean(self.signalPower))
        sig_avg_db = 10 * np.log10(sig_avg)
        noise_avg_db = sig_avg_db - snr_db
        noise_avg = 10 ** (noise_avg_db / 10)
        noise = np.random.normal(0, np.sqrt(noise_avg), len(self.modTime))
        signalWithNoise = []
        i = 0
        for bitSignal in self.signalModulated:
            noiseBitSignal = []
            for value in bitSignal:
                noiseBitSignal.append(value + noise[i])
                i+=1
            signalWithNoise.append(noiseBitSignal)
        return np.array(signalWithNoise)
```

Figura 12: Código agregar AWGN.

Y a continuación se tiene la misma señal modulada anteriormente, pero ahora sumándole ruido AWGN con un SNR de 10 decibeles

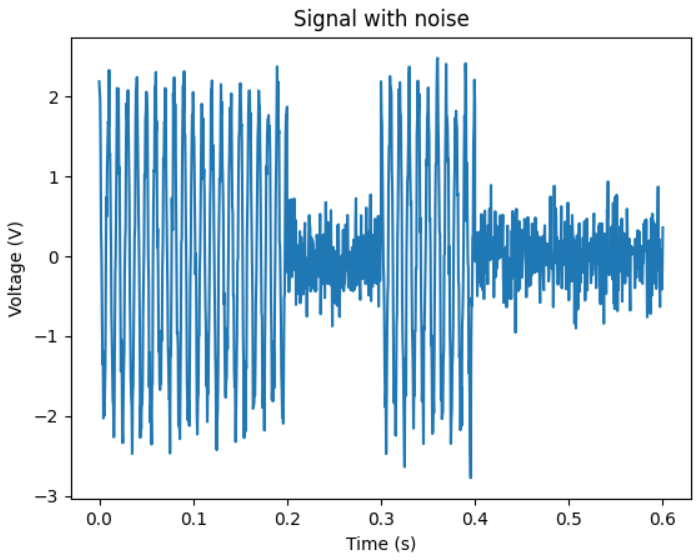


Figura 13: Señal con ruido AWGN.

3.5. Test de modulación y demodulación en señales grandes

Todo lo anterior lleva a esta actividad que consiste en primero generar arreglos que representan una señal digital de largo significativo(de 100000 bits en adelante) y aplicarles la modulación y demodulación para después calcular el BER o bits mal traducidos, que consta básicamente de una razón entre la cantidad de bits demodulados que difieren de la señal original, a continuación se muestra la función que calcula el BER:

```
# function to calculate number of diferent bits of a demodulated digital signal in compari
son to the original signal
# inputs:
    # original: original signal
    # demodulated: demodulated signal
def calculateBER(original, demodulated):
    diferent = 0
    for i in xrange(0,len(original)):
        if(original[i] != demodulated[i]):
            diferent+=1
    ber = diferent/len(original)
    return ber
```

Figura 14: Código calculo de BER.

Teniendo en cuenta lo anterior el proceso consta de tener una lista de 3 números que representan distintos números de bps para una señal digital y una lista con 10 números que representaran distintos niveles de SNR. Con cada numero de la lista de bps se usará una lista de largo de 100000 bits se modulara, se le agregara ruido AWGN, se demodulara y se calculara el BER, esto para cada uno de los 10 niveles de SNR que están en su respectiva lista, una vez hecho esto se pasara al siguiente bps y se le reajustara el tiempo de bit a la misma señal anterior y se repetirá el proceso. La finalidad de esto es generar un gráfico SNR vs BER para analizar el comportamiento de la demodulación a distintos niveles de ruido y distintos niveles de bps.

```
snr_levels = [1,1.2,2,2.5,3,4,5,7,8,8.5]# list with levels ob snr
    bps = [1,4,7] # list with diferent bps
    bersPerSignal = [] # list to add ber of each signal
    for b in bps:
        bers = [] # list to add bers of a signal with a especific bps
        if b == bps[0]:
            firstSignal = DigitalSignal(b)
            firstSignal.randomSignal(10**4,5)
        else:
            firstSignal.setBps(b) # adjust the time array to the new bps of the signal

        firstSignal.ookMod(10) # modulate signal with 10 HZ

        for level in snr_levels: # for each snr level
            noiseSignal = firstSignal.awgnAdd(level) # add noise
            demSignal = firstSignal.demodulate(firstSignal.modTime,noiseSignal) # demodulate signal
            ber = calculateBER(firstSignal.signal,demSignal) # calculate ber
            bers.append(ber) # add ber to bers list
        bersPerSignal.append(bers) # add bers list to the bersPerSignal

    plt.figure(5)
    plt.plot(snr_levels,bersPerSignal[0],label = str(bps[0]))
    plt.plot(snr_levels,bersPerSignal[1],label = str(bps[1]))
    plt.plot(snr_levels,bersPerSignal[2],label = str(bps[2]))
    plt.title('BER vs SNR')
    plt.ylabel('BER')
    plt.xlabel('SNR')
    plt.legend()
```

Figura 15: Código análisis de modulación y demodulación, señales grandes.

Los resultados de todo lo mencionado anteriormente se reflejan en el siguiente gráfico

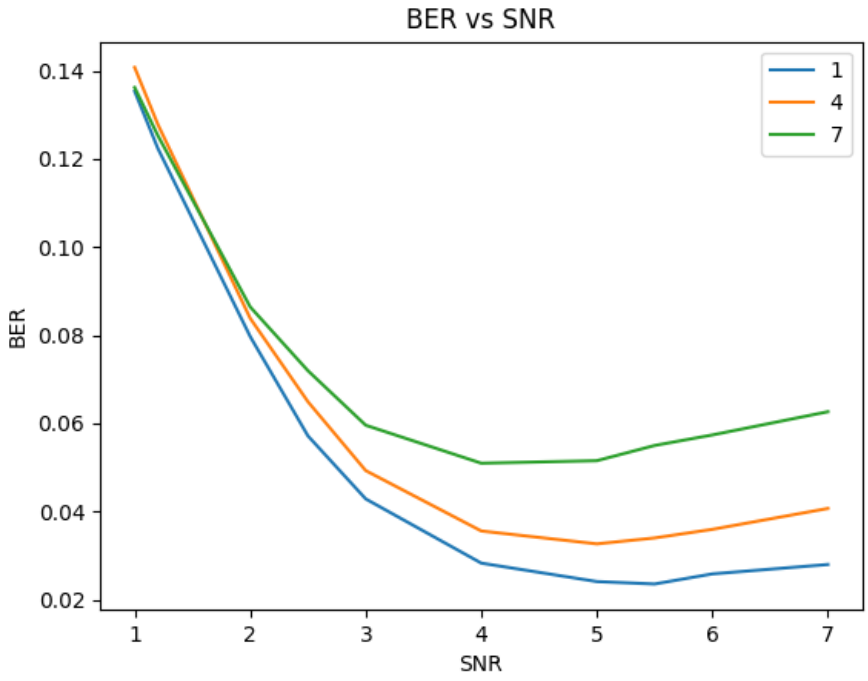


Figura 16: Gráfico SNR vs BER.

4. Análisis de resultados

En vista a todos los resultados expuestos anteriormente, primero que todo hay que reconocer las ventajas y desventajas de la modulación digital. La principal ventaja de la modulación digital es su simplicidad, dado que principalmente consiste en codificar el mensaje de diferentes maneras en caso de que sea 0 o 1, en el caso que se estudio en esta experiencia solo se debió aplicar un coseno cuando el bit que se leía al momento de modular era 1, y además, la modulación no es rígida, existen maneras de agrupar secuencias de bits y así hacer la modulación mas fácil. Viendo ahora las desventajas de la modulación digital, estas van en relación al tipo de modulación digital que se use, pero algo en común para todas es que al tratarse de bits, un error de lectura puede tener muchas mas consecuencias en comparación a un error de lectura en una modulación análoga. En el caso particular de la modulación que se aplica, se nota que este tipo de modulación es muy sensible al ruido, y esto no es bueno teniendo en cuenta de que los medios de transmisión actualmente tienen garantizada la presencia de niveles considerables de ruido.

En cuanto a la tasa de errores, esta como se ve en los resultados expuestos en el gráfico de SNR vs BER dependen primero que todo de la cantidad de ruido que presenta el medio, por ejemplo, en los niveles de SNR más bajos, estos representan medios con mayor prescencia de ruido y tiene sentido que sean los que dan una tasa de error mayor, las tres señales con distintos bits por segundo tienen más o menos el mismo comportamiento en medios con ruido considerable. pero la cantidad de bits por segundo pasa a ser de mayor importancia en niveles de SNR más altos, más o menos a partir de un SNR de 2 se empieza a apreciar las diferencias, siendo a mayor tasa de bits un error de lectura mas importante. En el gráfico se aprecia que con tasa de 7 bits por segundo hay más error que en las otras dos señales y la que tiene 1 bit por segundo tiene menos error, lo que tiene sentido debido a que hay más tiempo dedicado para cada bit lo que facilita que haya menos errores de lectura. En vista a todo lo mencionado anteriormente se puede tener una menor tasa de errores con menos bits por segundo y con medios que tengan una menor cantidad de ruido.

El ruido perjudica la tasa de datos de un sistema, dado que mientras más ruido haya, será más compleja la lectura y esta se complica más si son en tiempos acotados, y con

tasas de datos más grandes , se cometen más errores al momento de demodular. Y en base a los resultados obtenidos en la presente experiencia se puede decir que para grandes tasas de datos el ruido influye más cuando el bps es mayor.

Los principales usos de la modulación digital son para la transmisión de señales de medios de comunicación como lo son la radio y la televisión

5. Conclusión

Finalmente, evaluando todo lo que desarrollo durante la presente experiencia de laboratorio, primero cabe destacar que se logro la implementación para la generación de una señal digital aleatoria, con una determinada tasa de bps, para esto se creo una clase de python que tuviera los atributos que pueda tener una señal digital, También, como se muestra en la figura 9 se logra la modulación de una señal digital, esta vez por medio de una modulación OOK, siendo esta una variación especifica de la modulación ASK. Como se puede apreciar, cuando el bit es 1 en la modulación corresponde una función senoidal, mientras que cuando el bit es 0, en la señal demodulada también se tiene un0 constante. Esto hace considerar que se logro la modulación de manera adecuada. En cuanto a la demodulación ocurre más o menos lo mismo, se logra implementar un método que cumpla con la premisa de obtener la señal original a partir de de una señal modulada, tal y como se ve en la figura 11, donde se obtiene la misma señal original, sin embargo puede ser que el criterio de la demodulación, que consiste en encontrar el máximo valor en el intervalo de tiempo que le corresponde al bit modulado, no sea el más adecuado, pero a pesar de esto, se considera que la demodulación se logra. Para la implementación de una función que sume ruido AWGN, los resultados demuestran que se logro, dado que en la figura 13 se puede apreciar que la misma señal que se había modulado anteriormente ahora tiene una componente de ruido apreciable, y este fue generado a partir de la función *np.random.mormal()* que genero un arreglo con una distribución normal y con los parámetros obtenidos a partir de un SNR dado se le sumo al arreglo de la señal modulada. La principal dificultad para hacer la suma es que la estructura del arreglo de la señal modulada, porque es un arreglo de arreglos, donde cada arreglo interior tiene la información modulada de un bit.

Para el análisis de la modulación, la suma de ruido AWGN y demodulación de señales de tamaño significativo, los resultados son discutibles, dado que si bien se logran observar diferencias en las tres señales que se analizaron, se da la particularidad de que los BER tienen un comportamiento no esperado en valores de SNR a partir de 8 aproximadamente, lo que debe implicar que el algoritmo no es completamente adecuado para señales grandes, pero más allá de esto se consideran cumplidos todos los objetivos de la experiencia.

Bibliografía

(2020). Documentación scipy. [Online] https://www.scipy.org/docs.html.

(2020). Modulación por desplazamiento de amplitud. [Online] https://es.wikipedia.org/wiki/Modulaci%C3%B3n_por_desplazamiento_de_amplitud#:~:text=La%20modulaci%C3%B3n%20por%20desplazamiento%20de,de%20los%20datos%20a%20enviar.

(2020). Modulación por desplazamiento de fase. [Online] https://es.wikipedia.org/wiki/Modulaci%C3%B3n_por_desplazamiento_de_fase.

(2020). Modulación por desplazamiento de frecuencia. [Online] https://es.wikipedia.org/wiki/Modulaci%C3%B3n_por_desplazamiento_de_frecuencia.

(2020). ¿qué es el ruido blanco gaussiano aditivo awgn? [Online] https://www.adler-instrumentos.es/wp-content/uploads/2020/09/Qu%C3%A9-es-AWGN-Ruido-Blanco-Gaussiano-Aditivo.pdf.

Dakar, V. (2010). Fundamentos de telecomunicacion. [Online] https://www.itmerida.mx/panel/posgrado/archivos/mi/Fundamentos%20de%20Telecomunicaciones%20Unidad%201%20(1).pdf.

Lathi (2003). *Sistemas de comunicación*.

Oppenheim, A. (2001). Signals and systems. [Online] https://ocw.mit.edu.

viu (2018). Diferencias entre señal analogica y digital. [Online] https://www.universidadviu.com/diferencias-senal-analogica-digital/.