


# Centro Universitario de la Defensa en la Escuela Naval Militar

**TRABAJO FIN DE GRADO**

*Implementación de un sistema de comunicaciones de voz para canales de baja tasa binaria.*

**Grado en Ingeniería Mecánica**

**ALUMNO:** Santiago de Loresecha Palma

**DIRECTORES:** José María Núñez Ortuño

Milagros Fernández Gavilanes

**CURSO ACADÉMICO:** 2022-2023

Universidade Vigo




Centro Universitario de la Defensa
en la Escuela Naval Militar

**TRABAJO FIN DE GRADO**

# *Implementación de un sistema de comunicaciones de voz para canales de baja tasa binaria.*

**Grado en Ingeniería Mecánica**

Intensificación en Tecnología Naval

Cuerpo General.

Universidade Vigo

# RESUMEN

Las comunicaciones submarinas suponen un factor crucial en el ámbito militar, ya que permiten la transmisión segura de información, situándose como un elemento clave en el desarrollo estratégico, operacional y táctico de la misión. La necesidad de comunicaciones fiables ha llevado al desarrollo de protocolos como el JANUS por parte de la OTAN.

En el ámbito de las radiocomunicaciones en HF (*High Frequency*) de muy larga distancia, se utilizan protocolos de comunicaciones digitales caracterizados por emplear señales débiles en recepción, pero robustas frente a los desvanecimientos del canal. En concreto, el modo FT8 es el más empleado en la actualidad. Es utilizado en situaciones de propagación esporádica en las que las aperturas suelen ser breves y las señales muy débiles.

Dada la analogía entre el canal radio para comunicaciones ionosféricas (mecanismo principal de propagación en HF) y el canal submarino, el protocolo FT8 se presenta como una opción interesante para las comunicaciones militares submarinas, ya que ofrece una gran robustez y tasa de éxito en la transmisión de señales en condiciones difíciles y baja potencia. Asimismo, su capacidad para operar en anchos de banda muy estrechos resulta beneficioso para reducir el riesgo de detección y evitar interferencias con otras señales.

El objetivo principal del presente trabajo es desarrollar una aplicación para la comunicación de mensajes simples. Para ello, se emplea una técnica de conversión voz-texto-datos y se hace uso del protocolo de comunicaciones FT8 para la transmisión del mensaje. Posteriormente, se procede a realizar el proceso inverso en recepción, donde la aplicación recibe los datos y muestra al usuario por pantalla la información recibida.

Se realiza el montaje completo para la comprobación del correcto funcionamiento del sistema. Finalmente, se comprueba la cantidad de ruido que es capaz de aguantar el protocolo realizando dos experimentos diferentes.

## PALABRAS CLAVE

Protocolo FT8, SNR, JANUS, Corpus, M-GFSK

# CONTENIDO

Contenido ..... 1
Índice de Figuras ..... 3
Índice de Tablas ..... 5
1 Introducción y objetivos ..... 7
2 Estado del arte ..... 9
2.1 Historia de la acústica submarina ..... 9
2.2 Características principales de las comunicaciones submarinas ..... 13
2.3 Métodos actuales de las comunicaciones submarinas. ..... 14
2.3.1 Teléfono submarino ..... 14
2.3.2 Extremely low frequency (ELF) ..... 15
2.3.3 TARF (Comunicación RF acústica traslacional) ..... 16
2.3.4 Sistema de comunicaciones Calisto ..... 16
2.3.5 SECOMSAT/COMSATSUB ..... 17
2.4 Protocolo de comunicación JANUS ..... 17
2.5 Familia de protocolos “Weak Signal FT8” ..... 19
2.5.1 Modulación digital FSK ..... 20
2.5.2 CPFSK (Continuos Phase FSK) ..... 21
2.5.3 Modulación GFSK ..... 22
2.6 Tablas Hash ..... 23
2.6.1 Función Hash ..... 23
2.6.2 Tamaño de la tabla ..... 23
2.6.3 Gestión de colisiones mediante listas encadenadas ..... 24
3 Desarrollo del TFG ..... 25
3.1 Generalidades ..... 25
3.2 Lenguaje de programación, entorno de programación y librerías empleadas ..... 26
3.2.1 Python ..... 26
3.2.2 Pycharm ..... 26
3.2.3 Instalación de paquetes ..... 27
3.2.4 Reconocimiento de voz ..... 29
3.3 Implementación del proceso de comunicación ..... 31
3.3.1 Emisión ..... 31
3.3.2 Recepción ..... 43
3.3.3 Protocolo de comunicaciones FT8 ..... 47

4 Resultados y Validación ..... 55
4.1 Búsqueda de límite de relación señal a ruido (SNR) ..... 55
4.1.1 Búsqueda del límite de SNR con archivos de audio. ..... 55
4.1.2 Búsqueda del límite de la SNR con recepción en vivo. ..... 57
5 Conclusiones y líneas futuras ..... 60
5.1 Conclusiones ..... 60
5.2 Líneas futuras ..... 60
5.2.1 Lematización ..... 60
5.2.2 Etiquetado ..... 61
5.2.3 Pattern ..... 62
6 Bibliografía ..... 63
Anexo I: Implicaciones Sociales y Ambientales ..... 65
Anexo II: Reflexiones Éticas y Sociales ..... 67

# ÍNDICE DE FIGURAS

Figura 1. Sistema de defensa mediante boyas. Fuente: [10] ....10
Figura 2. Hidrófono direccional JP. Fuente: [12]....11
Figura 3. Torpedo MK-24. Fuente: [12]....11
Figura 4. Canal SOFAR. Fuente: [13]....12
Figura 5. Ejemplo sonar de casco. Fuente: [15] ....12
Figura 6. Esquema capas submarinas. Fuente: [15] ....13
Figura 7. Protocolo Janus. Fuente: [17] ....18
Figura 8. Modulación BFSK. Fuente: [19]....19
Figura 9. Gráfico Frecuencia-Tiempo de JANUS según CMRE. Fuente: [23] ....19
Figura 10. Esquema frecuencias FSK. Fuente propia. ....21
Figura 11. Problemática frecuencial FSK. Fuente propia. ....21
Figura 12. Funcionamiento de osciladores CPFSK. Fuente propia. ....22
Figura 13. Modulación GFSK. Fuente: [21] ....22
Figura 14. Explicación tabla Hash. Fuente: [16]....23
Figura 15. Gestión de colisiones tablas Hash. Fuente: [16] ....24
Figura 16. Esquema general del proyecto. Fuente propia. ....25
Figura 17. Proceso instalación de librerías. Fuente propia.....28
Figura 18.Proceso instalación de librerías. Fuente propia.....28
Figura 19. Proceso instalación de librerías. Fuente propia.....29
Figura 20. Modelos de bases de datos de Whisper. Fuente [27]:. ....30
Figura 21. Tasa de error de palabras en función del idioma. Fuente: [27]....31
Figura 22. Esquema emisión. Fuente propia. ....32
Figura 23. Esquema del progreso de código de emisión. Parte 1: Corpus. Fuente propia. ....33
Figura 24. Modalidades estudiadas para el reconocimiento de datos. Fuente propia. ....34
Figura 25. Librerías reconocimiento voz "online". Fuente propia. ....34
Figura 26. Reconocimiento "online" de la información. Fuente propia.....35
Figura 27. Librerías reconocimiento de voz "offline". Fuente propia.....35
Figura 28. Reconocimiento de voz "offline". Fuente propia.....36
Figura 29. Librerías reconocimiento de voz "en vivo". Fuente propia. ....37
Figura 30. Directorios y archivos temporales. Fuente propia. ....37
Figura 31. Función "Transcribe". Fuente propia. ....37
Figura 32. Función para detección de palabra de detención. Fuente propia. ....37
Figura 33. Explicación función "transcribe". Fuente propia. ....38

Figura 34. Esquema del progreso de código de emisión. Parte 2: Whisper. Fuente propia. .....39
Figura 35. Tabla corpus. Fuente propia. .....40
Figura 36. Tabla hash Corpus. Fuente propia. .....41
Figura 37. Explicación Tabla Hash. Fuente propia. .....41
Figura 38. Funcionamiento de la tabla Hash. Fuente propia. .....42
Figura 39. Esquema del progreso de código de emisión. Parte 3: Tabla Hash. Fuente propia. .....42
Figura 40. Librerías de recepción de datos. Fuente propia. .....43
Figura 41. Recepción de datos. Fuente propia. .....44
Figura 42. Obtención de datos en recepción. Fuente propia. .....44
Figura 43. Recepción de datos "en vivo" Fuente propia. .....45
Figura 44. Obtención de palabras en recepción. Fuente propia. .....46
Figura 45. Esquema del progreso del código. Proceso emisión y recepción. Fuente propia. .....47
Figura 46. Esquema del progreso de código. Proceso de transmisión. Fuente propia. .....47
Figura 47. Función gaussian_boxcar. Fuente propia. .....48
Figura 48. Modulación GFSK .....48
Figura 49. Función modulation_waveform. Fuente propia. .....49
Figura 50. Función mgfsk. Fuente propia. .....50
Figura 51. Función simulate. Fuente propia. .....51
Figura 52. Gestión mediante colas. Fuente propia. .....52
Figura 53. Frecuencias de emisión. Fuente propia. .....52
Figura 54. Salto de frecuencias. Fuente propia. .....53
Figura 55. Ejemplo datos de emisión de un mensaje. Fuente propia. .....53
Figura 56. Transmisión de la señal. Fuente propia. .....53
Figura 57. Experimento SNR= 0 dB. Fuente propia. .....55
Figura 58. Experimento SNR= -10 dB. Fuente propia. .....56
Figura 59. Experimento SNR= -15 dB. Fuente propia. .....56
Figura 60. Experimento SNR= -19 dB. Fuente propia. .....56
Figura 61. Experimento SNR= -18 dB. Fuente propia. .....57
Figura 62. Recepción de datos en vivo. SNR= -15 dB. Fuente propia. .....58
Figura 63. Recepción de datos en vivo. SNR= -17 dB. Fuente propia. .....58
Figura 64. Recepción de datos en vivo. SNR= -19 dB. Fuente propia. .....59
Figura 65. Lematizadores. Fuente propia. .....61
Figura 66. Lematizador spaCy. Fuente propia. .....61
Figura 67. Texto lematizado por spaCy [32]. Fuente propia. .....61
Figura 68. Librería pattern. Fuente propia. .....62

# ÍNDICE DE TABLAS

Tabla 1. Frecuencias usadas en las comunicaciones submarinas .....14

# 1 INTRODUCCIÓN Y OBJETIVOS

La mar se encuentra íntimamente ligada a la historia del ser humano, su evolución y crecimiento económico. Este medio se presenta, por tanto, como uno de los objetos de estudio más importantes para el desarrollo de la sociedad. En la actualidad, según datos de la Conferencia de las Naciones Unidas sobre el Comercio y Transporte (UNCTAD), más de un 90% de las actividades económicas se transportan vía marítima.

La seguridad para el correcto desarrollo de dichas actividades económicas resulta esencial para el bienestar de la sociedad, razón por la que los distintos países invierten grandes cantidades de dinero en sus marinas de guerra y policías marítimas. El desarrollo de la tecnología e ingeniería naval se presenta como una de las necesidades básicas para la prosperidad de un país.

Las tecnologías submarinas adquieren una gran importancia en esta materia de defensa. Submarinos nucleares, convencionales e incluso vehículos submarinos no tripulados (UUV), se han convertido en el foco de atención para mantener el control del mar.

Las comunicaciones se presentan como el talón de Aquiles de las operaciones submarinas. La coordinación y transferencia de información permite la integración en operaciones navales y la máxima explotación de las misiones submarinas. Sin embargo, nos encontramos ante un factor importante de indiscreción y fuente de peligrosidad para la unidad. Es por ello que las comunicaciones submarinas son uno de los principales focos de estudio para esta área de la guerra.

El surgimiento de esta modalidad de defensa conlleva la necesidad de asentar y asegurar comunicaciones seguras, fiables y robustas, materia en la que la OTAN toma acción adoptando el protocolo JANUS. Esta adopción supuso el primer protocolo de comunicaciones submarinas que es reconocido a nivel internacional, ya que fue estandarizado por la OTAN mediante un STANAG *(Standardization Agreement*), es decir, un acuerdo de normalización.

En el ámbito de las radiocomunicaciones en HF (*High Frequency*) de muy larga distancia, se han introducido recientemente protocolos de comunicaciones digitales que emplean señales débiles (WS del inglés *Weak Signals*) en recepción, pero que son robustas frente a los desvanecimientos del canal. Existen modos como WSPR, JT4, JT9, FT8, etc. que se han hecho muy populares, y el modo FT8, en particular, es el más empleado en la actualidad. Se utiliza en situaciones de propagación esporádica, en las que las aperturas suelen ser breves, y las señales muy débiles. Según sus creadores, FT8 puede decodificar señales con relación señal a ruido de hasta –27 dB con anchos de banda reducidos.

Dada la analogía entre el canal radio para comunicaciones ionosféricas (mecanismo principal de propagación en HF) y el canal submarino, en este trabajo se pretende investigar la aplicación del

protocolo FT8, lo que permitiría aumentar el alcance de la comunicación submarina o, alternativamente, reducir el tamaño y complejidad de los transductores.

El objetivo principal del presente trabajo es desarrollar una aplicación para la comunicación de mensajes simples. Para ello, se emplea un apartado para la explicación del sistema de emisión y recepción de la información, donde se explica la conversión voz-texto-datos en la parte del sistema de emisión y su recibimiento y decodificación en recepción. Se expone posteriormente el uso del protocolo de comunicaciones FT8 para la transmisión del mensaje como sustituto al protocolo JANUS. Finalmente, se realiza un estudio sobre la capacidad de establecer comunicaciones en presencia de ruido.

# 2 ESTADO DEL ARTE

## 2.1 Historia de la acústica submarina

Dada la importancia de la mar en materia militar, económica, etc., la evolución de la ingeniería naval se ha situado como uno de los centros de atención de las distintas sociedades a lo largo de la historia. Por ende, la acústica submarina y su propagación por este medio llamaron la atención de quienes descubrieron la posibilidad de establecer un canal de comunicación subacuático.

Para atender a la primera evidencia del estudio del sonido o de su propagación como tal, debemos remontarnos a la Antigua Grecia, siglo IV a.C. Aristóteles centró la música como uno de sus múltiples objetos de estudio, realizando intuiciones certeras sobre el comportamiento del sonido. El filósofo discípulo de Platón fue capaz de deducir que la razón por la que el sonido se podía propagar residía en el desplazamiento y el golpeo de una masa de aire sobre otra continua [5].

Leonardo Da Vinci nos proporcionó una de las primeras referencias a este tipo de comunicaciones, cuando en 1490 escribía: “Si paras tu barco e introduces el extremo de un tubo en el agua, y aplicas el oído al otro extremo, oirás barcos que se encuentran a gran distancia de ti”.

De esta forma, Leonardo Da Vinci descubrió, en su esencia, los principios básicos del funcionamiento de un sonar pasivo. Demostró que todo buque realiza cierto ruido al navegar y que dicho sonido se puede transmitir en un medio de propagación como es el agua [6].

Fue Marín Merssene quien, en 1636, publicó “*L’Harmonie Universelle*”, obra con la que se constató los aspectos relacionados con la naturaleza y el comportamiento del sonido. Desarrolla contenido centrándose en aspectos relacionados con la naturaleza del sonido, la voz, instrumentos armónicos etc. [7].

Isaac Newton publicó “*Philosofiae Naturalis Principa Mathematica*”, donde centró sus estudios en la acústica proponiendo una teoría de propagación del sonido a través de la compresión de las partículas del aire. Es decir, defendió que la propagación del sonido a través de cualquier fluido dependía de las propiedades físicas del propio fluido, como puede ser la elasticidad o la densidad [8].

Jean-Baptiste le Rond d’Alembert publicó “*Recherches sur la courbe que fomr une corde Tenduë mise en vibration*” [5] en 1747. En dicha publicación, el autor uso ecuaciones diferenciales para modelar la vibración de una cuerda tensa, demostrando así que la velocidad de propagación de las ondas en la cuerda es dependiente de la amplitud de vibración. Este descubrimiento supuso un paso esenciar en el estudio matemático de la propagación del sonido, asentando posteriormente la teoría de las ondas sonoras.

El físico Jean-Daniel Colladon en 1826 realizaron la primera medición precisa de la velocidad del sonido en el agua. Usaron un hidrófono para medir el tiempo que tardaba el sonido en recorrer una distancia conocida en el agua. El experimento se realizó concretamente, en el Lago de ginebra, en Suiza. Como resultado, se dedujo que la velocidad del sonido era cercana a 1435 metros por segundo, cercana al valor real [9].

Durante la Primera Guerra Mundial, la acústica submarina se convirtió en uno de los aspectos fundamentales de estudio para las marinas de guerra de ambos bandos. Se experimentó con múltiples tecnologías de acústica para mejorar la detección de buques enemigos y evitar ser detectados.

El HMS Birmingham, buque de la Royal Navy, uso un hidrófono con el objetivo de detectar submarinos alemanes (U-boots). Los aliados hicieron uso de estos dispositivos de sonido para detecta submarinos y minas. El operador debía mover mecánicamente el receptor para determinar la dirección desde la que llegaba el sonido. En respuesta, dichos submarinos comenzaron a portar los mismos dispositivos, generando, a su vez, dispositivos generadores de ruido para confundir a los enemigos.

Los británicos desarrollaron un dispositivo denominado ASDIC, el cual emitía pulsos de sonido para detectar la presencia y ubicación de objetos bajo el agua. Se convertiría en el precursor del sonar moderno.

Durante el periodo de entreguerras, se produjeron importantes avances en términos de acústica submarina. Se realizaron avances significativos en 1919 con el desarrollo del sonar activo, ASDIC, afinando la tecnología y llegando a plantear el sonar pasivo.

Elias Klein determino que los bordes en las hélices sufrían cavitación, causando vibraciones ruidosas a bordo de la nave. También se desarrolló una tecnología de defensa portuaria (Figura 1) que consistía en un conjunto de boyas acústicas ancladas, las cuales proporcionaban alertas y localizaban naves furtivas [10].

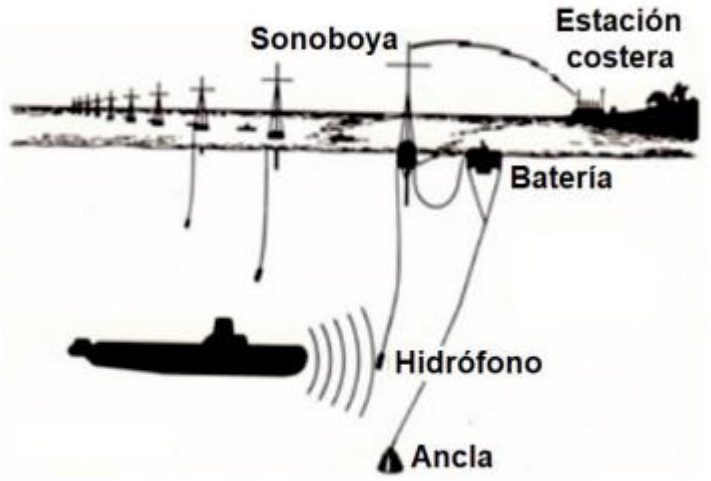



**Figura 1. Sistema de defensa mediante boyas. Fuente: [10]**

Los esfuerzos británicos se centraron en silenciar la firma de sus barcos hasta el punto de que no pudieran ser escuchados por escucha pasiva.

En el periodo previo a la Segunda Guerra Mundial, la Marina de los Estados Unidos reconoció el atraso en sus sistemas de acústica submarina, implicando a las universidades de Columbia, Harvard, Universidad de California, Instituto de Tecnología de Massachusetts (MIT), y otras tantas organizaciones reconocidas a crear grupos de trabajo especializados para la mejora de acústica submarina.

Como resultado, uno de los laboratorios elaboró un hidrófono direccional (Figura 2) que podía ser portado a bordo de un submarino, el cual se presentó como el primer sistema de vigilancia de banda

ancha de la flota. Esto permitió a los submarinos americanos sentir y medir la dirección de las naves enemigas.



**Figura 2. Hidrófono direccional JP. Fuente: [12]**

Así mismo, la acústica submarina se empezó a introducir, no solo en los mecanismos de detección de unidades, sino en los sistemas de armas. El equipo liderado en la Universidad de Harvard, desarrollo un torpedo de propulsión eléctrica (Figura 3) que era guiado por un sonar autónomo instalado delante de la ojiva, usado posteriormente en operaciones en el Atlántico.



**Figura 3. Torpedo MK-24. Fuente: [12]**

Durante la Segunda Guerra Mundial, se realizaron grandes descubrimientos sobre la propagación del sonido en el entorno submarino. La marina de los Estados Unidos se centró en investigar los factores que influían en la ecuación de propagación del sonido. Se dedujo que la absorción, pérdidas por reflexión, ruido ambiental, o parámetros relacionados con la emisión y la recepción influían en la propagación, afectando directamente en la ecuación del sonar.

Las capas submarinas también fueron sujeto de estudio durante la contienda. El batitermógrafo fue un instrumento esencial, a partir del cual se dedujo la temperatura en función de la profundidad, influyendo enormemente en el estudio de la propagación en función de estos parámetros.

Se descubrió el canal SOFAR (*SOund Fixing And Ranging*), (Figura 4) capa de agua en el océano que tiene una densidad y velocidad única, creado por la disminución de temperatura y aumento de presión. Esta capa se encontraba a una cota de alrededor de mil metros, y se usó principalmente como una herramienta de salvamento. Los sobrevivientes de los buques que se hundían dejaban caer una carga explosiva que se podía llegar a escuchar en estaciones de escucha a grandes distancias. Asimismo, se usó como un canal de búsqueda y detección de submarinos enemigos.

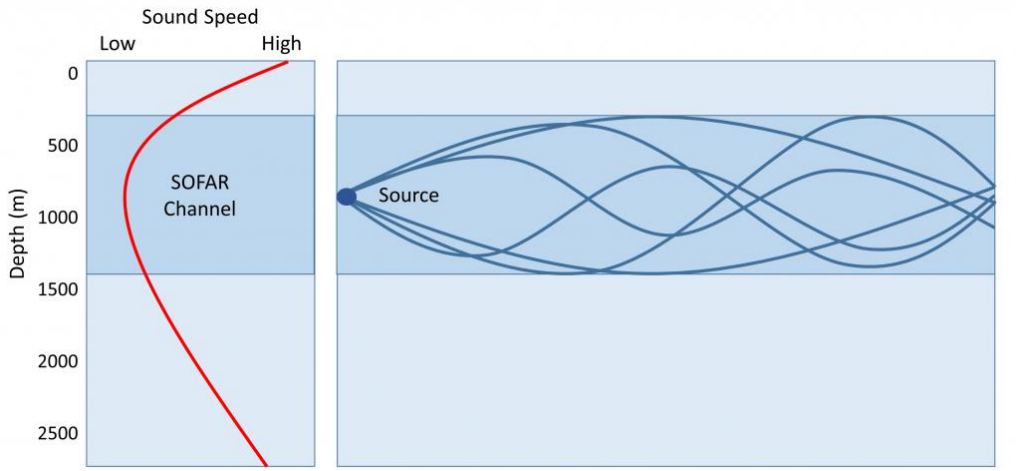



**Figura 4. Canal SOFAR. Fuente: [13]**

Así mismo, fueron descubiertas otras capas submarinas que fueron importantes para el desarrollo de la contienda. Ejemplo de ello puede ser la termoclina, donde la temperatura disminuye rápidamente en función de la profundidad y provoca una refracción del sonido, llegando a distorsionar señales.

La capa de fondo, la cual hace referencia al hecho de que está en contacto con el fondo del océano, también fue descubierta, donde las señales podían ser reflejadas y hacer que los submarinos fueran detectables a largas distancias.

El desarrollo del sonar de haz estrecho se produjo con el fin de la Segunda Guerra Mundial, situándose como uno de los grandes avances de la acústica submarina. Con este avance, se logró emitir un haz de sonido enfocado en una dirección específica, en lugar de emitir de forma omnidireccional (sonar convencional). Esto proporciono una mayor resolución espacial y la capacidad de detectar objetivos más pequeños a distancias mayores [14].

El conocimiento en materia de acústica submarina relacionado con las marinas de guerra resulta hoy en día esencia. Actualmente, y tras todos los avances explicados, podemos encontrar sonares pasivos y activos en los cascos de los buques de guerra, submarinos, o sonares remolcados (Figura 5). Se han desarrollado sonoboyas para la detección de submarinos y minas subacuáticas, y se han desarrollado torpedos con sistemas de guiados acústicos altamente eficientes. A su vez, todos estos sistemas poseen modalidades tales como sistemas direccionales u omnidireccionales, o la aplicación de los mismos en diferentes bandas de frecuencia.




**Figura 5. Ejemplo sonar de casco. Fuente: [15]**

Además, se presenta un conocimiento exhaustivo de las capas submarinas, sus características, y de la ecuación de propagación del sonido bajo el agua (Figura 6).

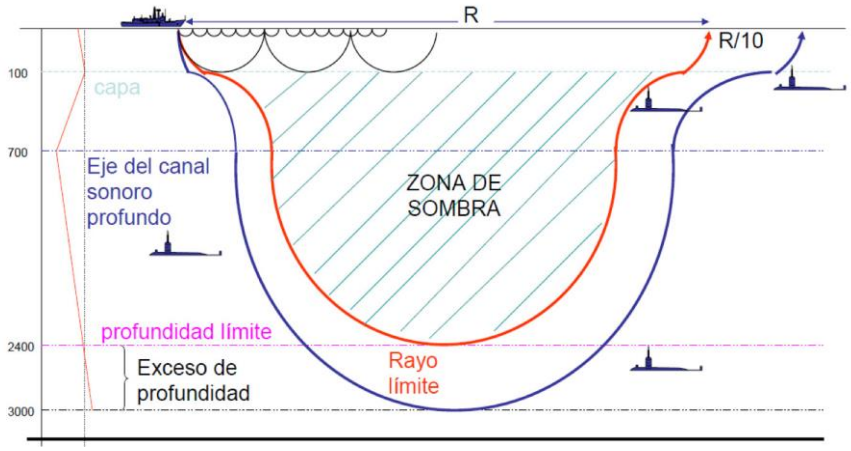



**Figura 6. Esquema capas submarinas. Fuente: [15]**

## 2.2 Características principales de las comunicaciones submarinas

Las comunicaciones submarinas difieren, en gran medida, a las de las otras fuerzas. Presentan un desafío de coordinación que influye directamente en el resultado de la misión y las operaciones.

La indiscreción se puede ver afectada de forma visual (exposición de mástiles) y por la transmisión de energía electromagnética. La necesidad de coordinación y transferencia de información limitan su integración en las operaciones navales.

Se destacan los siguientes aspectos como principales características de las comunicaciones submarinas:

- Tasa de transmisión:
  La velocidad en la que el sistema es capaz de transmitir datos por debajo del agua resulta francamente baja en relación con las comunicaciones de superficie, por ejemplo, comunicaciones vía satélite.

- Interferencia ambiental:
  Las comunicaciones submarinas enfrentan la problemática de los ruidos marinos, atenuación de la señal etc., aspectos que implementan una dificultad superior al establecimiento de comunicaciones.

- Fiabilidad:
  Las comunicaciones submarinas no se caracterizan por presentar una fiabilidad elevada. El medio de propagación, así como problemas ambientales o técnicos afectan directamente a la transmisión de datos.

- Equipos especializados y alto coste:
  Los equipos que debe portar una unidad submarina para ser capaz de establecer comunicaciones con otra, ya sea en el mismo medio o distinto, son altamente especializados para el entorno. Así pues, las comunicaciones submarinas se caracterizan por ser costosas, entre otras cosas, por su complejidad de implementación y tecnología avanzada que requiere.

- Limitaciones técnicas:

Las comunicaciones submarinas presentan limitaciones en términos de ancho de banda, baja tasa de transmisión e interferencia ambiental.

Por otra parte, los submarinos poseen ciertas limitaciones que se deben principalmente a las antenas:

- Discreción.
- Limitación de número de mástiles.
- Periodos de silencio obligados.
- Longitud de las antenas.
- Altura respecto a la superficie.
- Limitación en potencia.
- Diagrama de radiación de antena fuertemente afectado por la disposición de otros mástiles

Para salvaguardar la capacidad furtiva de los submarinos, debemos atender a la siguiente tabla:

| SIGLAS | DENOMINACIÓN | MARGEN FRECUENCIA | LONGITUD DE ONDA |
|---|---|---|---|
| EHF | EXTRA HIHG FREQUENCY | 30-300 GHz | 10 - 1 mm |
| SHF | SUPER HIHG FREQUENCY | 3-30 GHz | 10 - 1 cm |
| UHF | ULTRA HIHG FREQUNECY | 300 MHz - 3 GHz | 100 -10 cm |
| VHF | VERY HIGH FREQUENCY | 30 - 300 MHz | 10 - 1 m |
| HF | HIGH FREQUENCY | 3 - 30 MHz | 100 - 10 m |
| MF | MEDIUM FREQUENCY | 300 KHz - 3 MHz | 1.000 - 100 m |
| LF | LOW FREQUENCY | 30 - 300 KHz | 10 - 1 Km |
| VLF | VERY LOW FREQUENCY | 3 - 30 KHz | 100 - 10 Km |
| VF | VOICE FREQUENCY | 300 Hz - 3 KHz | 1.000 - 100 Km |
| ELF | EXTRA LOW FREQUENCY | 30-300 Hz | 10.000 - 1.000 Km |
| ULF | ULTRA LOW FREQUENCY | 3 - 30 Hz | 100.000 - 10.000 Km |

**Tabla 1. Frecuencias usadas en las comunicaciones submarinas**

Debido a la triangulación a larga distancia, se suelen restringir las comunicaciones en bandas de HF, VHF y UHF. Se recomienda hacer uso de las bandas LF o inferior (sección 2.3.2), soliendo usar la banda SHF para comunicaciones SATCOM (sección 2.3.5).

## 2.3 Métodos actuales de las comunicaciones submarinas.

La discreción supone el objetivo fundamental del arma submarina. Sin embargo, toda unidad militar debe disponer de métodos de comunicación, ya sea para recibir órdenes, reportar una situación táctica del entorno determinada, o comunicarse con unidades aliadas. Estas necesidades entran directamente en conflicto, pues un submarino al comunicarse delata su posición.

La problemática conduce a las comunicaciones submarinas a situarse como uno de los aspectos más importantes en el desarrollo e ingeniería de estas embarcaciones subacuáticas.

### *2.3.1 Teléfono submarino*

El teléfono submarino fue desarrollado por primera vez por la Marina de los Estados Unidos dotando de ello a los submarinos y a múltiples embarcaciones de superficie relacionadas con la guerra submarina

(ASW, *Anti-Submarine Warfare*). Fue denominado como AN/BQC-1 o "Gertrude", *brevity word*[1] a la que se le asocia de forma oficial hoy en día.

Para entender el funcionamiento del teléfono submarino, se procede a explicar sus modos de operación:

1. Modo de voz:
   Se debe modular la voz a una frecuencia determinada (8,3 kHz a 11,08 kHz) con banda lateral única (SSB). El receptor debe filtrar la señal para poder eliminar la señal portadora y una de las bandas laterales, recuperando así el audio original.
   Este modo está limitado en distancia (unas 500 yds), dependiendo del nivel de ruido y condiciones de sonido.
2. Código morse:
   El teléfono submarino presenta la capacidad de establecer comunicaciones con el código morse.
3. Modo ping:
   Pensado para ser detectado durante una emergencia en la mar para el rescate del submarino. El teléfono emitía un ping omnidireccional haciendo uso de una frecuencia de 24,26 kHz. El rango de los pings es, aproximadamente de unas 5000 yds.

Asimismo, se debe resaltar que la información transmitida por teléfono submarino no presentaba ningún tipo de encriptado, por lo que las comunicaciones podían ser fácilmente interpretadas si eran detectadas por un submarino o buzos enemigos.

El AN/WQC-2 fue desarrollado durante la Guerra Fría y usado por unidades hasta los buques actuales. Este último modelo hace uso de un esquema de modulación y frecuencias muy similar al Gertrude original.

### *2.3.2 Extremely low frequency (ELF)*

Con el surgimiento de los misiles balísticos durante la Guerra Fría, las necesidades tácticas de las diferentes marinas aumentaron su exigencia. Los cuarteles generales presentaban la necesidad de establecer conexión con las unidades sumergidas y a distancias mayores a las que presentaba el alcance del teléfono submarino.

Se define entonces a la Frecuencia Extremadamente Baja (ELF) como la radiación electromagnética con frecuencias pertenecientes al rango 3-30 Hz. Este tipo de comunicaciones no ha sido extensamente acogido por los diferentes sistemas civiles debido a la dificultad de construcción de antenas que pudieran radiar ondas tan largas [7].

Dada la longitud de onda extremadamente larga, este tipo de onda es capaz de difractar alrededor de grandes obstáculos, sin quedarse bloqueadas por cadenas montañosas. Además, es capaz de propagarse sobre la curvatura de la Tierra, superando así el horizonte.

En el ámbito militar si presentaron adopción por aquellos países que requirieran dichas necesidades tácticas y pudieran construir las instalaciones necesarias para la radiación de este tipo de onda. Se encuentra su utilidad para el arma submarina debido a que las ondas ELF son capaces de atravesar el agua de mar y alcanzar las profundidades operativas de los submarinos.

La marina de China construyó la instalación ELF más grande del mundo, siendo sus dimensiones parecidas a la de la ciudad de Nueva York.

[1]Las *Brevity Words* son un conjunto de términos simples que son usados en el entorno de las comunicaciones militares para definir, de forma clara y concisa, algún aspecto de interés de mayor longitud al hablar en claro.

Se deduce, por tanto, que una de las principales dificultades para adoptar este tipo de comunicaciones con los submarinos son las dimensiones de la antena para emitir la gama de frecuencias ELF. La necesidad de cables extremadamente largos ha conducido a implantar distintos medios, como el alargamiento eléctrico, para construir estas estaciones de radio con menores tamaño.

Otro aspecto negativo es el impacto ecológico que supone la sola construcción de las instalaciones. Además, desde finales de la década de los 70 se han planteado numerosos estudios sobre si la exposición a campos electromagnéticos ELF producen efectos negativos en la salud. Los campos magnéticos inducen campos eléctricos y corrientes en el cuerpo del ser humano que, si se diera la intensidad necesaria, provocan estimulaciones nerviosas y musculares, así como cambios en la excitabilidad de las células del sistema nervioso central. Los estudios realizados llegaron a demostrar como el ELF a niveles de kV/m determinados creaba una sensación de hormigueo perceptible por los seres humanos delatándose, especialmente, en los brazos [8].

### *2.3.3 TARF (Comunicación RF acústica traslacional)*

Los submarinos presentan serias dificultades para realizar comunicaciones con unidades que sean externas al medio submarino, ya que se usan distintas señales que únicamente funcionan en sus respectivos medios. Ante esta problemática, se planteó un sistema de comunicaciones liderado por el MIT, cuya meta final es disponer de la capacidad de establecer comunicaciones entre aviones y submarinos.

Se presentan dos principales inconvenientes: las señales acústicas usadas en entornos submarinos se reflejan en su gran mayoría al alcanzar la superficie, sin llegar a atravesarlo. Por otro lado, las comunicaciones de señales de radio que son usadas a través del aire como medio de propagación, se atenúan muy rápido en el agua.

Ante la problemática comentada, los científicos del MIT plantean una ingeniosa solución. La señal sonar provocada por la unidad submarina se dirigirá a la superficie de manera que provoque pequeñas vibraciones. Se colocará un receptor altamente sensible que es capaz de leer dichas vibraciones, obteniendo el contenido del mensaje.

El principal problema que se presenta no es otro que las propias ondas naturales. Para solventarlo, el sistema tiene un sofisticado algoritmo de procesamiento de señales. Este sistema permite decodificar señales con ondas inferiores a 16 mm.

Este sistema permitirá a los submarinos comunicarse con unidades en tierra y aéreas, sin llegar a comprometer su discreción al mantenerse sumergido. Por otra parte, los investigadores afirman que las aplicaciones no son únicamente militares. Los aviones perdidos bajo el agua que lleven incorporada una baliza, por ejemplo, en la “caja negra” serán capaces de delatar su situación haciendo uso de estas comunicaciones [10].

### *2.3.4 Sistema de comunicaciones Calisto*

Este sistema de comunicaciones, desarrollado por la marina germana, proporciona al arma submarina la capacidad de establecer y mantener enlace y comunicaciones desde un submarino en inmersión profunda sin perder capacidad de maniobra.

El sistema de comunicaciones Calisto consta de una boya de comunicación, la cual incorpora instalaciones de recepción UHF SATCOM, UHF, VHF, HF, GPS y ESM (*Electronic Support Measures*). Además, está unida a un mástil retráctil (dispositivo de elevación).

El comandante del submarino puede establecer, dentro de los parámetros permitidos, la velocidad y la inmersión requeridas. Se determina así la cantidad de cable (fibra óptica) largado, posicionando, de esta forma, la boya y la antena en su correcta posición [9].

### *2.3.5 SECOMSAT/COMSATSUB*

Se trata de un sistema de comunicaciones satélite, donde el arma submarina aprovecha la capacidad gubernamental del SPAINSAT (satélite geoestacionario, centrado en la Península Ibérica) para iluminar el 43% de la Tierra. SECOMSAT frece un sistema de comunicaciones discreto, presentando una nula deteccion electromagnética, ya que emite rayo directo al satélite. Además, no deja huella térmica en el mástil, debido a que su amplificador de potencia se encuentra en el interior del submarino.

Para el uso del SATCOM deberemos mantener el submarino en cota periscópica (14 metros de profundidad). Es por ello que debemos tener en cuenta que nos encontramos ante una de las situaciones tácticas menos deseables, debido a la exposición del submarino, lo que supone una alta probabilidad de pérdida de factor sorpresa en sus operaciones.

Además, la subida a cota periscópica es una maniobra peligrosa, pero imprescindible para los submarinos convencionales, por lo que paralelamente podremos efectuar la carga de baterías y renovación del aire.

## 2.4 Protocolo de comunicación JANUS

La naturaleza del presente trabajo tiene por objetivo estudiar la posibilidad de usar otro protocolo de comunicación como alternativa al protocolo JANUS . Es por ello que, a continuación, se explicará de forma más detalladas sus características y funcionamiento.

JANUS es un protocolo estandarizado por la OTAN para comunicaciones submarinas que ha sido diseñado y aprobado por el Centro de Investigación y Experimentación Marítima CMRE. Se trata de un protocolo que ha estandarizado la OTAN mediante una STANAG (*Standardization Agreement*) [12], siendo este el primer protocolo de comunicaciones submarinas “entendible” a nivel internacional.

Las comunicaciones aéreas y terrestres no presentan un buen funcionamiento bajo el agua, por lo que ninguno de los protocolos disponibles se podía exportar fácilmente al ámbito submarino. Ante esta situación, el CMRE comenzó hace un poco más de diez años a desarrollar una forma de codificación digital submarina, estableciendo un denominador común entre todas las comunicaciones submarinas.

JANUS ha sido diseñado para ser usado tanto en el ámbito civil como en el militar (Figura 7). Se trata de una forma de codificar información en sonido de forma simple, de modo que los equipos, tanto nuevos como existentes puedan cumplir con esta estandarización.

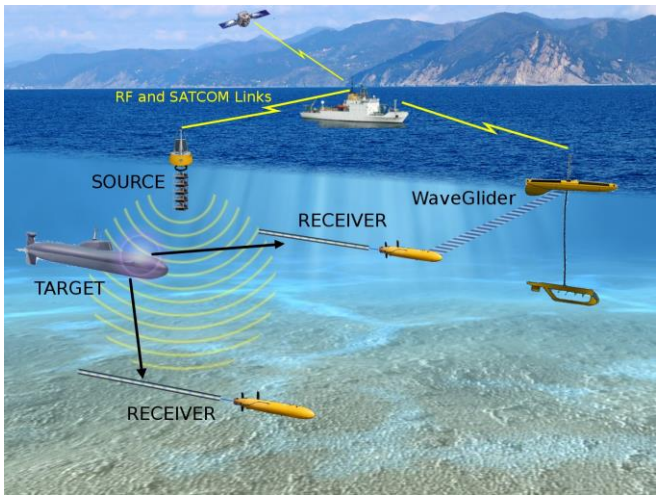



**Figura 7. Protocolo Janus. Fuente: [17]**

JANUS hace referencia al dios romano de las aberturas y las puertas, haciendo referencia a esa capacidad que nos permite para comunicarnos entre todas las naciones. Este protocolo combina la necesidad operativa militar de la OTAN con la de estandarización en el ámbito civil. Antes de JANUS, no existían capacidades interoperables entre los módems de los diferentes fabricantes, no se estableció ningún protocolo específico ni estructura de mensajes. protocolo combina la necesidad operativa militar de la OTAN con la de estandarización en el ámbito civil [13].

Los propósitos principales son anunciar la presencia de un nodo y establecer contacto inicial entre dos nodos. El protocolo usa un servidor denominado JANUS Gateway, que actúa como intermediario entre los participantes de un enlace. Así pues, cada participante se conectará al servidor JANUS y enviará sus datos a través de él. El servidor JANUS es el encargado de enviar esos datos a los demás participantes.

En primer lugar, JANUS usará una frecuencia comúnmente conocida (11,5 kHz). Una vez que los dispositivos estén vinculados, estos cambiaran a una frecuencia secundaria u otro medio de transmisión diferente dependiendo de velocidades de datos requeridas, SNR (relación señal a ruido), etc. Durante estas comunicaciones, los mensajes portarán bits de paridad y corrección de errores.

El protocolo fue concebido para ser altamente robusto. Hace uso de una modulación por desplazamiento de frecuencia binaria (BFSK, Figura 8). Esto es una técnica de modulación digital donde se presentan los bits de información como cambios de frecuencia entre dos predefinidas. Se trata de una técnica fácil de implementar y es robusta ante ruido e interferencia, aunque presenta una eficiencia espectral limitada.

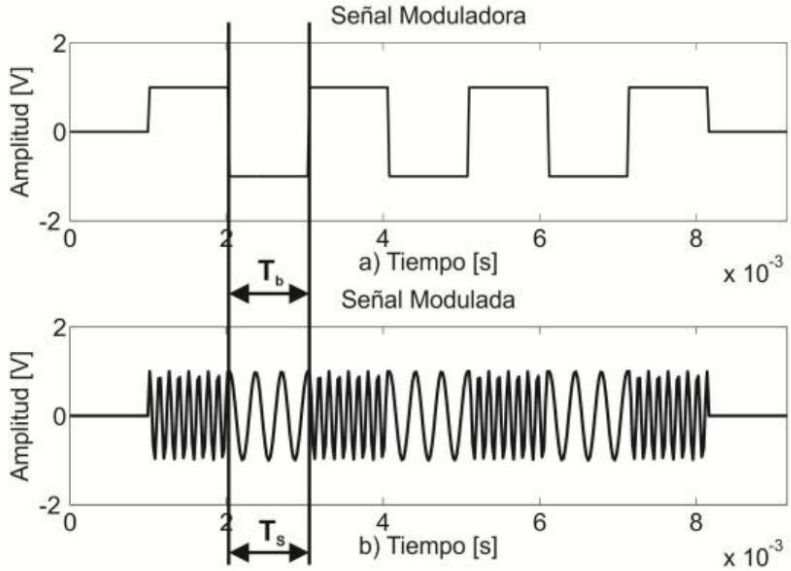



**Figura 8. Modulación BFSK. Fuente: [19]**

Por otra parte, el salto de frecuencia (FH) se usa para saltar de un canal a otro según un patrón predeterminando. Esto ayuda a evitar la interferencia.

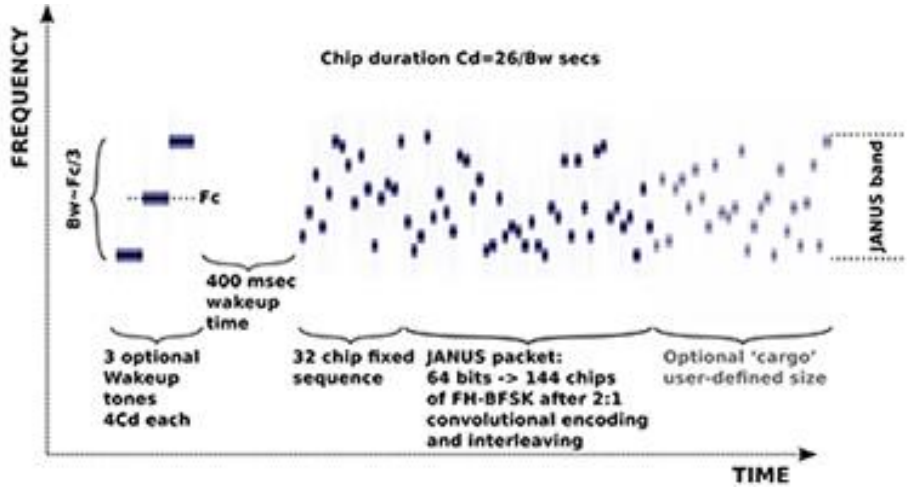



**Figura 9. Gráfico Frecuencia-Tiempo de JANUS según CMRE. Fuente: [23]**

Según las especificaciones del CMRE (Figura 9), el protocolo de comunicación JANUS comienza el enlace estableciendo tres tonos en las frecuencias Fc-Bandwidth/2, FC y Fc+Ancho de Banda/2 Hz. Estos pulsos se usan como un mensaje "despierta" para indicar al resto de equipos que comiencen a escuchar. El tiempo entre estos pulsos no debe ser inferior a 0,4 segundos. De esta forma, la energía reverberante (reflexiones de objetos y limites medios) se disipa [14].

A continuación, el sistema puede realizar un barrido de modulación HFM (barrido de modulación de frecuencia hiperbólica). Este tipo de modulación permite

## 2.5 Familia de protocolos "Weak Signal FT8"

FT8 es una familia de protocolos de transmisión de señales débiles creado por el Premio Nobel de Física en 1993 Joseph Hooteon Taylor. Este protocolo está pensado para la transmisión de datos en bandas de frecuencias difíciles, especialmente en condiciones de propagación adversas, como puede ser

la mar. Se coloca como una de las opciones más eficientes en términos de ancho de banda y tiempo de transmisión para las comunicaciones [15].

Se trata de uno de los protocolos más usados en la actualidad, siendo muy común entre los radioaficionados, especialmente en el ámbito de las señales débiles. Encontramos a una amplia comunidad de usuarios donde cada individuo contribuye a su desarrollo y mejora.

Algunas de sus características más importantes [16], a las que atenderemos posteriormente, son:

- Longitud del mensaje: 75 bits + 12 bits de CRC (Verificación de redundancia cíclica)
- Secuencia de transmisión/recepción del mensaje (T / R): 15 segundos
- Ancho de banda: 47 Hz
- 8-FSK (*Frequency Shift Keying*), Velocidad de inserción = separación de tono = 5,86 Hz
- Duración de la transmisión 13,48 segundos
- Límite de decodificación -20 dB

Una vez introducido el concepto de FT8, pasaré a comentar la modulación de dicho protocolo. Pero para llegar a ella, tendremos que ir poco a poco explicando otro tipo de modulaciones que dan lugar a la empleada por este protocolo.

En este sentido, la modulación es una técnica que nos permite desplazar las frecuencias de "Banda base" al rango de frecuencias que sea más adecuado para realizar la transmisión de la señal. De esta forma, el usuario será capaz de adaptar la señal al medio de transmisión y multiplexar varias señales en banda base de forma simultánea [22]. Es decir, la modulación tiene como propósito fundamental la eficiencia y fiabilidad de la transmisión mediante un canal de comunicación.

El protocolo FT8 utiliza una modulación M-GFSK. Sin embargo, para alcanzar una explicación clara de dicho protocolo, debemos comenzar entendiendo el concepto de funcionamiento de la modulacion FSK y CFSK, la cuales se expondrán a continuación.

### *2.5.1 Modulación digital FSK*

La modulación por desplazamiento de frecuencia (FSK, *Frequency Shift Keying*) es una de las técnicas de modulación digital que hemos visto. Al hacer uso de ella, la información digital se transmitirá a través de cambios en la frecuencia de la señal portadora.

En la modulación digital FSK (Figura 10) se hace uso de dos frecuencias diferentes para los bits "0" y "1". El índice de modulación (m) se define como la razón entre la desviación máxima de frecuencias $\Delta f$ y la frecuencia de la señal moduladora (fm). Dependiendo del valor de esta razón nos encontramos con los siguientes casos:

1. Si las frecuencias para los dos bits son "parecidas", la desviación de frecuencia $\Delta f$ es baja, por lo que el ancho de banda de la señal modulada $\Delta B$ será bajo. Si su índice de modulación es menor a la unidad, será conocido como "FM de Banda estrecha"
2. Por el contrario, si la diferencia entre las frecuencia a las que se emiten los bits "0" y "1" es muy alta, $\Delta f$ será alto y la señal modulada ocupará un mayor ancho de banda. Esto se debe a que un cambio de frecuencia mayor en la señal modulada repercute en un mayor cambio en la frecuencia de la señal portadora, lo que significa que se necesitara un ancho de banda mayor para transportar la señal modulada. De la misma forma, el índice de modulación, m, será mayor a la unidad, y será conocido como "FM de Banda Ancha".

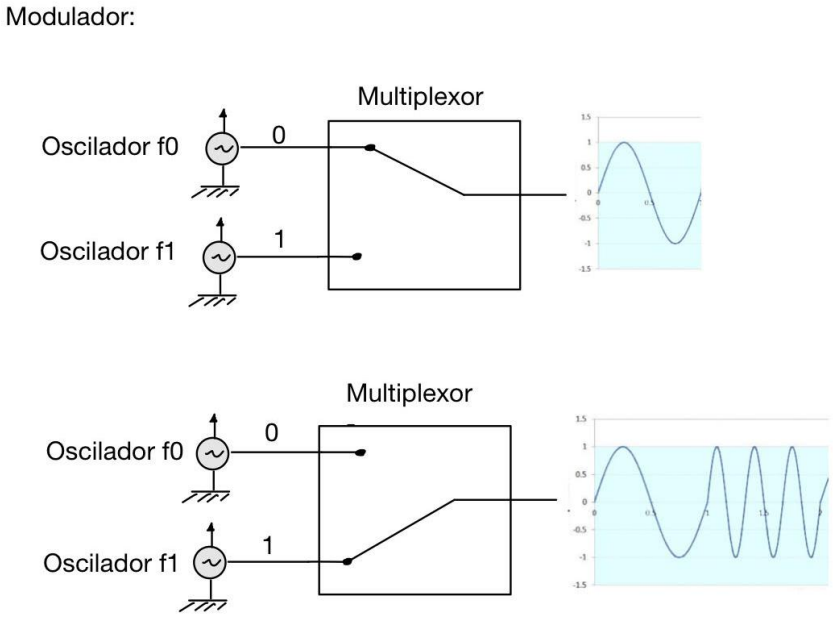



**Figura 10. Esquema frecuencias FSK. Fuente propia.**

## *2.5.2 CPFSK (Continuos Phase FSK)*

La problemática de la modulación FSK reside en el abrupto cambio frecuencial que se produce en la transición de la emisión de un bit al otro (Figura 11). El cambio de un oscilador a otro, es decir, entre bits, puede llegar a generar interferencia con canales adyacentes y reducir la eficiencia del espectro.

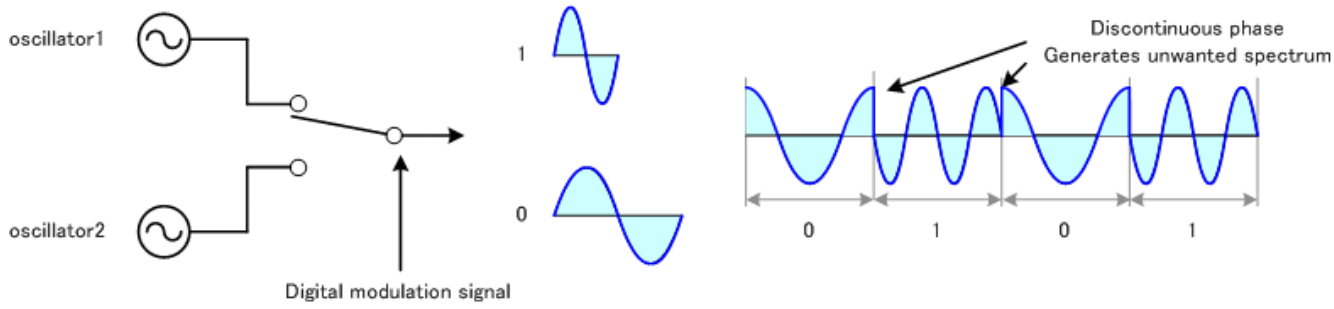



**Figura 11. Problemática frecuencial FSK. Fuente propia.**

La solución se presenta gracias al CPFSK, el cual mantiene la continuidad de la fase entre los bits. Esto permite un mayor aprovechamiento del ancho de banda disponible, ya que permite una mayor tasa de trasferencia de datos en el mismo ancho de banda que la modulación FSK convencional.

Es decir, la solución propuesta por CPFSK radica en cambiar los dos osciladores de la FSK por un único oscilador (Figura 12). Este oscilador será capaz de cambiar su frecuencia gracias al VCO (*Voltage-Controlled-Oscillator).* Este VCO recibirá una tensión de corriente continua, y, en función de esa tensión, el VCO variará la frecuencia emitida. Al ser el mismo oscilador, la transición entre bits no genera un cambio abrupto en la frecuencia. Esto nos proporciona una reducción del ancho de banda.

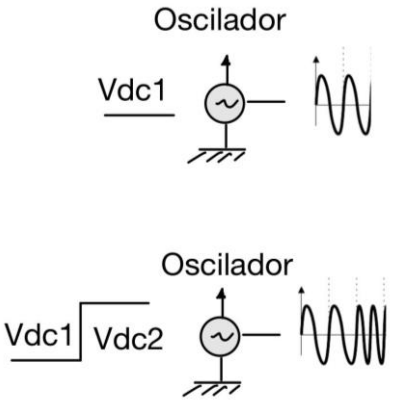



**Figura 12. Funcionamiento de osciladores CPFSK. Fuente propia.**

La solución se presenta gracias al CPFSK, el cual mantiene la continuidad de la fase entre los bits. Esto permite un mayor aprovechamiento del ancho de banda disponible, ya que permite una mayor tasa de trasferencia de datos en el mismo ancho de banda que la modulación FSK convencional.

### *2.5.3 Modulación GFSK*

Finalmente, se procede a la explicación de la modulación GFSK, la cual se usa como protocolo de comunicaciones en este trabajo.

Se ha explicado anteriormente cómo la modulación CPFSK modificaba la estructura de producción de la señal para lograr una transición suave entre las dos frecuencias, sin discontinuidades abruptas o interrupciones. La modulación GFSK se presenta como una evolución de esta última, donde la transición entre frecuencias, a saber, entre bits, se realiza ajustando la señal modulada a una función gaussiana (Figura 13). Esta modificación nos proporciona una mayor robustez frente al ruido y a la distorsión de la señal.

El perfil de frecuencia de la señal modulada se asemeja a una campana gaussiana, distribución de probabilidad continua que es usada comúnmente en la estadística para describir la distribución de valores aleatorios. Esta es la forma que proporciona una mayor suavidad en la transición entre las dos frecuencias. El uso de esta técnica nos proporciona una gran fiabilidad en la transmisión de datos.

Para llevar a cabo esta modulación, GFSK también hace uso del oscilador controlado por voltaje (VCO), el cual varía la frecuencia de la señal portadora en respuesta a una señal moduladora.

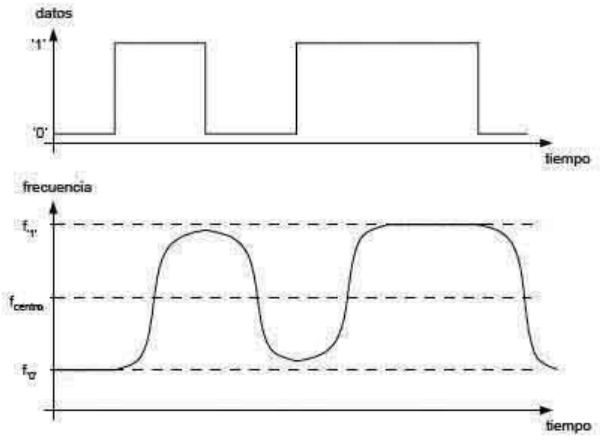



**Figura 13. Modulación GFSK. Fuente: [21]**

## 2.6 Tablas Hash

Las estructuras o tablas Hash son usadas en el ámbito de la programación para realizar búsquedas e inserción de forma eficiente para un número muy elevado de datos. Se trata de una tabla que almacena un conjunto de pares "clave-valor" [16].

- Clave es única para cada elemento de la tabla
- El dato es el que se use para buscar un determinado valor.

Es decir, la tabla Hash es una función que, dada una clave de un elemento, devuelve el índice de la tabla hash donde se encuentre almacenado dicho elemento (Figura 14). La implementación de una tabla hash está basada en los siguientes elementos:

- Tabla de tamaño razonable para almacenar pares (clave-valor)
- La propia función hash que nos devolverá el índice para acceder a la posición de la tabla al proporcionar la clave.
- Un procedimiento para tratar las colisiones, es decir, los casos en los que la función hash devuelva el mismo índice para dos claves distintas.

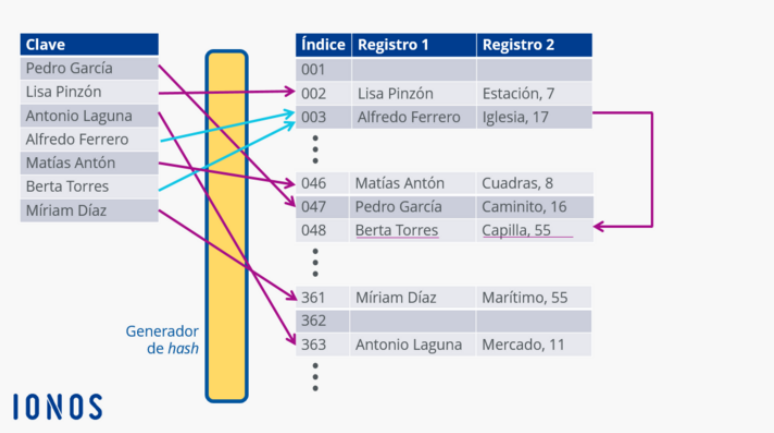



**Figura 14. Explicación tabla Hash. Fuente: [16]**

### *2.6.1 Función Hash*

Debemos resaltar que la ganancia en eficiencia que presenta la tabla Hash respecto a una lista encadenada se basa en que la longitud de los elementos de colisión es menor que el número total de elementos. Es decir, en el hipotético caso en el que todos los elementos están en una única cadena de colisión, la tabla tendrá la misma eficiencia que la lista encadenada.

Esta observación sugiere que, además de producir un índice en la tabla, para un número de elementos elevado debe producir dichos índices los más dispersos posibles.

### *2.6.2 Tamaño de la tabla*

Se trata de otro parámetro al que debemos atender para obtener el mayor rendimiento de la tabla. Analizando la casuística de los casos extremos entenderemos las ineficiencias posibles. A saber, si el tamaño de la tabla es demasiado pequeño, la tabla se comportaría de nuevo como una lista encadenada. Si por el contrario la tabla presenta un tamaño enorme, el malgasto de memoria resulta evidente, donde además encontraremos un número muy elevado de posiciones cuya lista de colisión está vacía.

Para el estudio de este tamaño de tabla, se suele recurrir a lo que se denomina densidad. Por este término entendemos el resultado que surge al dividir el número de elementos de la tabla por su número de entradas. Un resultado elevado de densidad denota una longitud excesiva de las listas de colisiones.

Por el contrario, un valor reducido indica una longitud muy corta de las cadenas de colisión, lo que indica una infrautilización de la memoria.

En las aplicaciones más complejas con el uso del hash se incluyen dos umbrales para el valor de la densidad. Opera de tal forma que con cada inserción o borrado de elemento se consulta estos valores. Si la densidad supera el valor máximo, se proporciona un espacio para una tabla mayor, se borran los elementos de la tabla antigua y se inserta en la nueva, donde obviamente encontraremos una densidad inferior. Por su parte, si al borrar un elemento la densidad es menor que el umbral mínimo, se realza la operación contraria, una tabla más pequeña.

De esta forma se consigue mantener la densidad en un rango de valores óptimos. Destacar que, si bien es cierto que el redimensionamiento de la tabla es una operación cara en términos de tiempo de CPU, las operaciones futuras de búsqueda rentabilizan la ejecución.

### *2.6.3 Gestión de colisiones mediante listas encadenadas*

La solución propuesta trata de combinar la lista encadenada con la estructura de la tabla Hash. A saber, cada posición de la tabla no almacenará un único elemento, sino una lista encadenada que a su vez contiene todos aquellos elementos que devolverá con un resultado idéntico.

Atendiendo a la siguiente ilustración (Figura 15) observamos cómo en las posiciones donde no se haya insertado ningún elemento, la función nos devolverá un NULL. Por otra parte, queda claro cómo dada una clave, las listas pueden presentar diferentes longitudes.

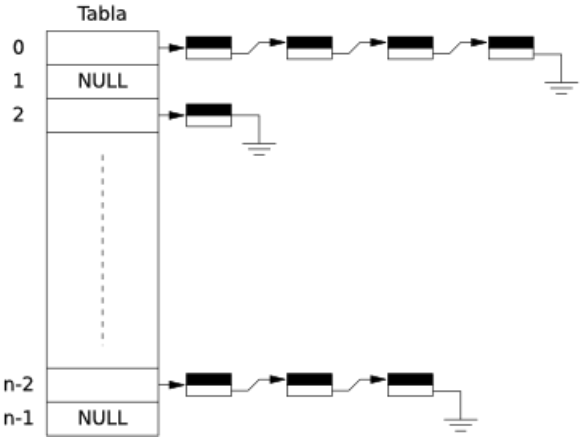



**Figura 15. Gestión de colisiones tablas Hash. Fuente: [16]**

# 3 DESARROLLO DEL TFG

## 3.1 Generalidades

Antes de comenzar, se proporciona una pequeña visión general sobre el desarrollo del presente trabajo. Se procede a emplear una estructura deductiva, donde el lector entienda la razón y naturaleza de cada una de las funciones incorporadas y sus respectivas modificaciones.

El objeto fundamental del proyecto es la comunicación o transmisión de datos haciendo uso del protocolo WS FT8 (sección 2.5), es decir, sustituyendo al actual JANUS (sección 2.4). Se trata de aplicar un protocolo de comunicaciones (con medio de propagación aire) que es usado por radioaficionados, pero en un entorno submarino.

El motivo del desarrollo del presente trabajo es la robustez que el protocolo presenta frente al ruido. Tanto es así, que puede llegar a realizar la correcta comunicación de la información con una potencia de ruido mayor a la de la propia señal, esto es, con una SNR negativa.

Para lograr el correcto entendimiento del lector, se establece la Figura 16.a modo de esquema conceptual, donde se puede apreciar, de forma sencilla, dónde se ubica en el proceso de construcción del sistema de comunicaciones.

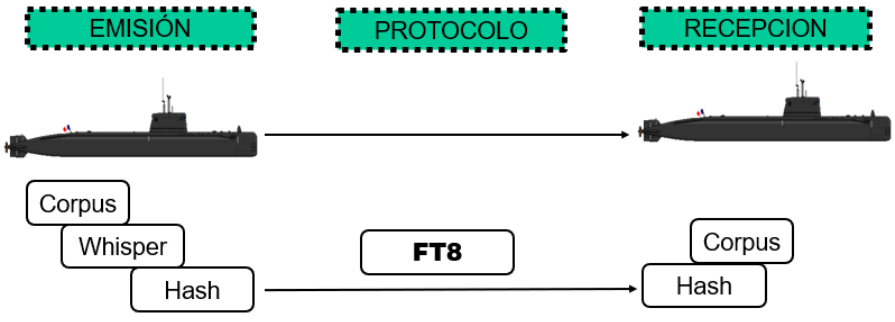



**Figura 16. Esquema general del proyecto. Fuente propia.**

Tal y como se puede observar, el proceso cuenta con tres grandes partes diferenciadas: Emisión, transferencia de la información mediante el protocolo, y la recepción del mensaje. Destacar que para cada gran etapa comentada se han realizado varios modelos, sin embargo, se muestra en el esquema la última de las versiones de cada etapa. La explicación general del codigo se agrupa de la siguiente forma:

1. Emisión y recepción del mensaje:
   Parte orientada a la programación, que proporciona el código necesario para realizar la transmisión y recepción de la información. Se realiza la explicación de estas dos partes de forma seguida debido a la similitud que presentan en sus estructuras de código.
   La intención en este apartado es tratar de reconocer las palabras que desea transmitir el usuario y elaborar el mensaje de la manera más eficiente y compatible con los requerimientos del protocolo para su transmisión.
2. Transmisión del mensaje:
   Durante este apartado se procede a estudiar el protocolo de transmisión de la información.

Sin embargo, antes de comenzar con la exposición del codigo, es necesario explicar las herramientas que se utilizan durante la realización del mismo. Por lo tanto, se da paso a un apartado para describir de forma sencilla el lenguaje elegido, así como el entorno de programación y librerías empleadas para la elaboración del codigo.

## 3.2 Lenguaje de programación, entorno de programación y librerías empleadas

### *3.2.1 Python*

A la hora de programar, se hace uso de un lenguaje de programación. Este se encarga de establecer un enlace entre el programador y la máquina. Dentro del amplio abanico de opciones, se ha escogido Python debido a su popularidad y sencillez.

Python es un lenguaje de programación de alto nivel y de propósito general [15]. Busca en su esencia claridad en la sintaxis gracias a su estructura que aporta una alta importancia a la legibilidad del código.

Se trata de un lenguaje de fácil lectura, multiplataforma y de código abierto. No requiere la declaración explícita de los tipos de datos, aspecto que aporta rapidez y flexibilidad al código. Se puede definir este lenguaje como multiparadigma imperativo y funcional. A saber, se trata de un código que, excepto los bucles y estructuras condicionales, sigue unas instrucciones de forma secuencial. El programador debe establecer, en primer lugar, las funciones y emplear las variables a las que se le pueden asignar distintos valores.

El enfoque de la programación orientada a objetos es otra de las características destacables de este lenguaje. Python permite la creación de clases y objetos, lo que permite una programación más estructurada y modular. Además, no necesita un proceso de compilación para ejecutarlo, pues se trata de un lenguaje interpretado.

Python cuenta además con una amplia selección de módulos de código abierto que pueden integrarse fácilmente gracias a la comunidad de usuarios. Esto coloca a Python como una de las principales herramientas para la creación rápida de prototipos.

### *3.2.2 Pycharm*

PyCharm es el entorno elegido para el desarrollo del presente trabajo. Se trata de un entorno de desarrollo integrado (IDE) par Python. Fue presentado en 2010 a los usuarios por la firma de software JetBrains.

Se trata de una aplicación multiplataforma, por lo que se puede instalar en los tres principales sistemas operativos: macOS, Linux o Windows [18]. Su objetivo fundamental es optimizar el desarrollo de Python, generando un entorno donde el usuario es capaz de construir aplicaciones sencillas a proyectos de gran envergadura. Dispone de una Edición Comunitaria que se publica bajo la licencia de Apache, siendo esta totalmente gratuita. Una profesional, la cual requiere una financiación para su suscripción, y finalmente una educativa.

Algunas de sus principales características son:

- Editor inteligente:
  Integrado en la aplicación, permite al usuario comprender y leer de forma rápida y eficiente los diferentes códigos.

- Autocompletar código:
  Pycharm proporciona al usuario un completado inteligente de código.

- Análisis de código:
  Alerta a los usuarios sobre posibles errores o problemas en el código.

- Depurador integrado:
  Permite revisar, probar y corregir el código. Proporciona la capacidad a los desarrolladores de ejecutar el código y examinar el estado de las variables en cualquier momento.

- Vistas:
  Ofrece la capacidad de buscar segmentos específicos, muestra la estructura del proyecto y permite localizar segmentos clave en poco tiempo con su modo lente.

- Refactorización:
  Brinda la posibilidad de examinar y cambiar estructura del código, como variables u organización, sin alterar el resultado final.

- Plantillas:
  Permiten crear segmentos de código comunes con poco esfuerzo.

### *3.2.3 Instalación de paquetes*

La gestión de paquetes se realiza desde el repositorio oficial de Python `Pypi`. Se trata de un repositorio público donde el usuario puede encontrar numerosos paquetes, librerías, herramientas etc. que la propia comunidad de usuarios aporta de forma desinteresada.

El sistema de gestión se realiza mediante el comando `pip`, el cual permite instalar, actualizar o administrar los distintos paquetes de software. De esta forma, el usuario deberá escribir en la terminal o consola de Python `pip install` para instalar, `pip uninstall` para desinstalar, etc.

Esta utilización de paquetes resulta esencial para el desarrollo de aplicaciones de Python, ya que nos permite instalar librerías y paquetes que son fundamentales para la creación de aplicaciones sin que el usuario deba desarrollar el código desde cero.

Si bien es cierto que lo previamente explicado es el procedimiento común de instalación, en este caso, al hacer uso de PyCharm, la gestión de paquetes se realiza de la siguiente forma:

1. El usuario debe dirigirse a la parte superior izquierda de la aplicación, y en la opción de "*File*", entrar en el apartado de "*Settings*", tal y como podemos apreciar en la Figura 17.

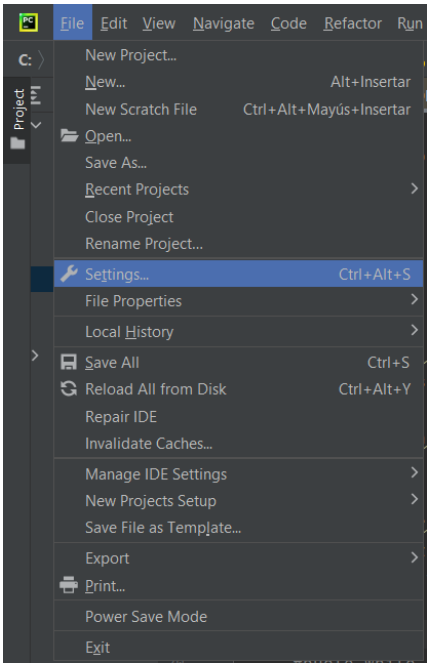



**Figura 17. Proceso instalación de librerías. Fuente propia.**

2. Tras abrirse la ventana de *settings*, se debe buscar el apartado del proyecto y entrar el en intérprete de Python. Porteriormente, se debe pulsar la opción "+" que se muestra en la Figura 18.

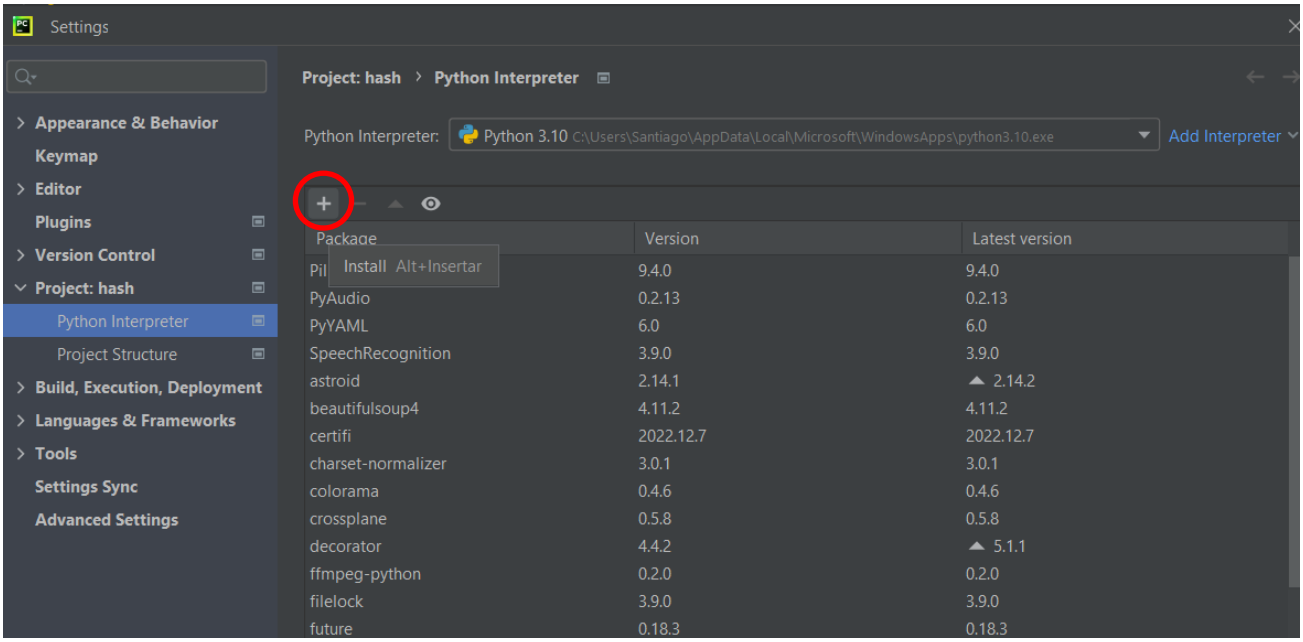



**Figura 18.Proceso instalación de librerías. Fuente propia.**

3. Finalmente, el usuario debe buscar el paquete en el que está interesado y descargarlo (Figura 19).

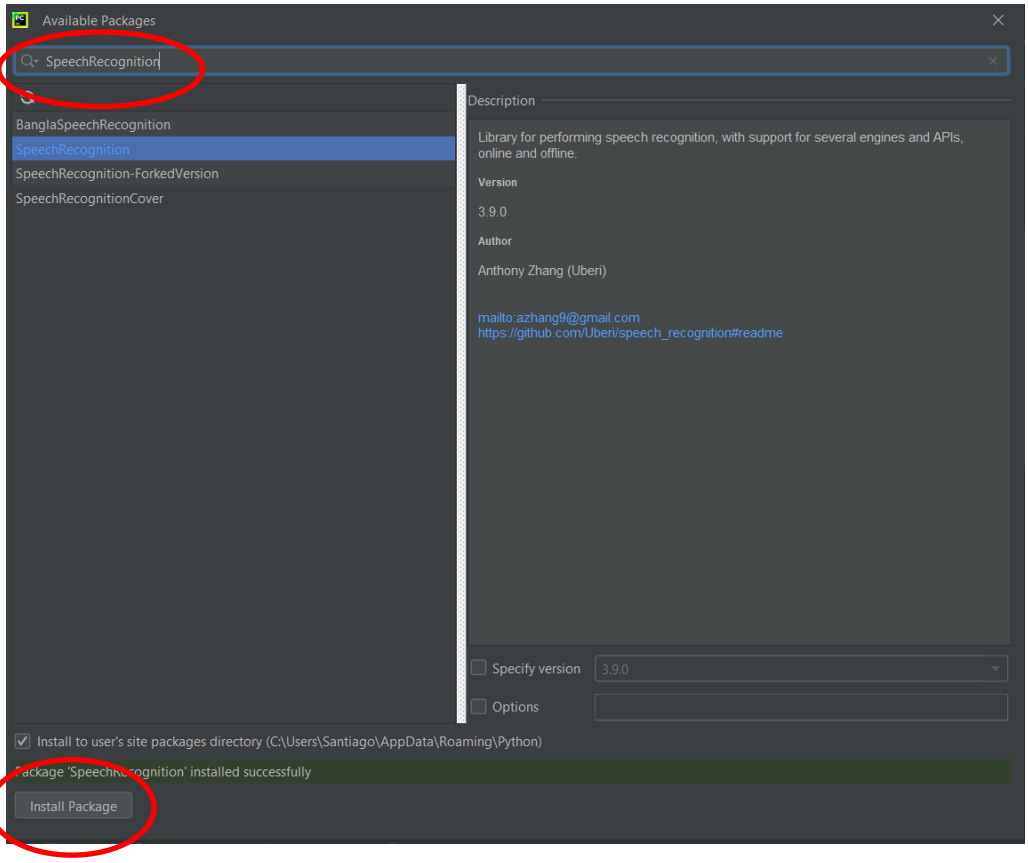



**Figura 19. Proceso instalación de librerías. Fuente propia.**

### *3.2.4 Reconocimiento de voz*

El reconocimiento automático de voz (ASR, en inglés *Automatic Speech Recognition*) permite transcribir el lenguaje hablado a texto. Este reconocimiento se realiza implementando una serie de librerías especializadas como pueden ser Google Speech recognition o Whisper.

Estas librerías proporcionan una API (*Application Programing Interface*), lo que se traduce como una serie de reglas y especificaciones que determinan cómo se puede interactuar con software o aplicaciones. De esta forma el usuario es capaz de integrar, por ejemplo, OpenAI Whisper en el código [19].

De forma general, el reconocimiento en Python consta de los siguientes pasos:

1. Capturar el audio:
   Se hace uso del micrófono para capturar el audio o proporcionaremos un archivo mp3 o WAP

2. Procesamiento del audio:
   Se extraen las características del audio más relevantes para el reconocimiento de voz mediante su procesamiento.

3. Reconocimiento de voz:
   El audio, una vez procesado, se compara con una base de datos predeterminada y se devuelve la palabra que el audio se supone ha dicho.

A continuación, se procede a realizar un estudio más específico de la biblioteca de Whisper, pues es la elegida para realizar el reconocimiento de voz de forma *offline*.

Whisper consiste en un modelo de reconocimiento de voz de propósito general que es capaz de realizar su función en un ámbito multilingüe, así como la traducción e identificación del idioma. Trabaja

en varias tareas de procesamiento de voz, tales como el propio reconocimiento de voz multilingüe, traducción de voz, identificación del idioma o detección de actividad de voz. Todas estas tareas se representan como una serie de *tokens*, lo que permite que un solo modelo reemplace muchas etapas diferentes en una canalización de procesamiento de voz.

Concretamente, en este caso, Whisper posee hasta cinco posibles modelos para la transcripción de audio procesado (Figura 20). El uso de un tamaño u otro dependerá de la velocidad relativa necesaria, así como del espacio que el usuario pretenda ocupar.

| Size | Parameters | English-only model | Multilingual model | Required VRAM | Relative speed |
|---|---|---|---|---|---|
| tiny | 39 M | `tiny.en` | `tiny` | ~1 GB | ~32x |
| base | 74 M | `base.en` | `base` | ~1 GB | ~16x |
| small | 244 M | `small.en` | `small` | ~2 GB | ~6x |
| medium | 769 M | `medium.en` | `medium` | ~5 GB | ~2x |
| large | 1550 M | N/A | `large` | ~10 GB | 1x |

**Figura 20. Modelos de bases de datos de Whisper. Fuente [27]:.**

El rendimiento que proporciona Whisper depende, en gran medida, del idioma. En el siguiente gráfico se observa un desglose del WER (tasa de error de palabras) por cada idioma (Figura 21). Destacar que cuanto menor sea el WER, mejor rendimiento se obtiene [19].

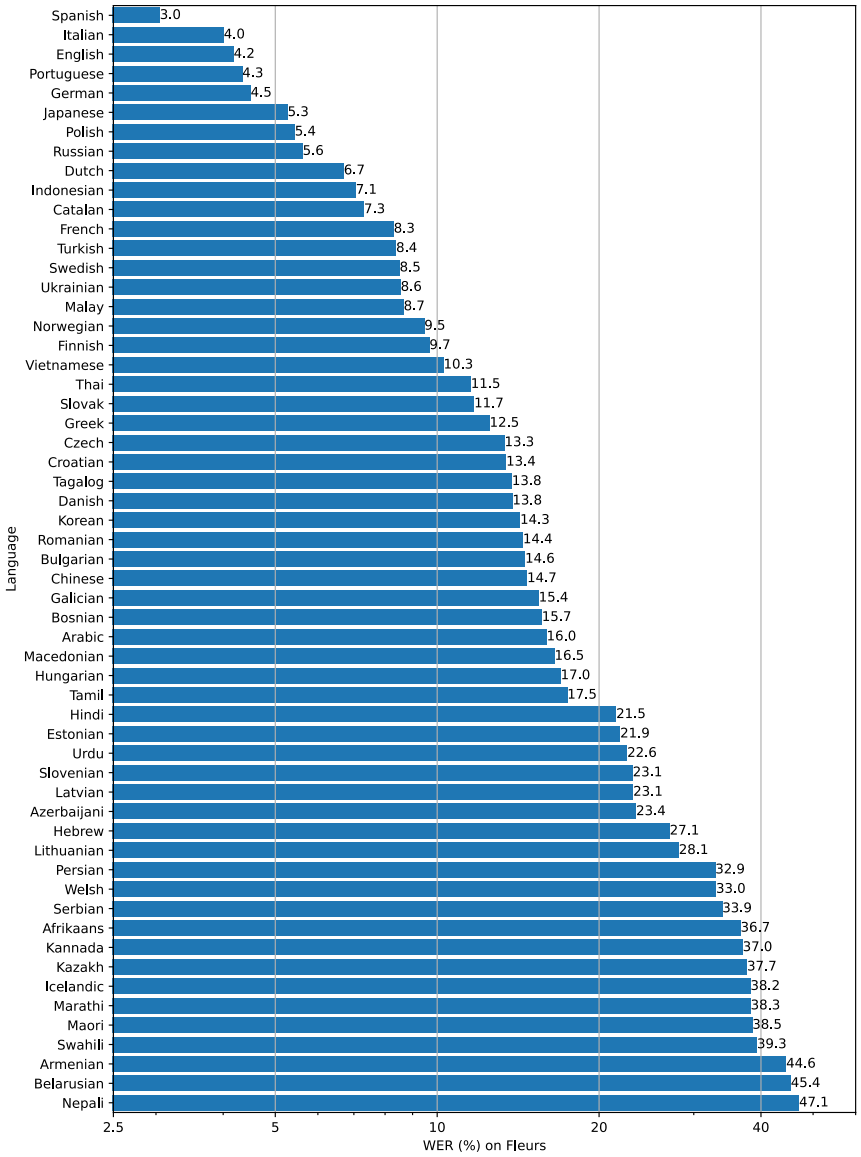



**Figura 21. Tasa de error de palabras en función del idioma. Fuente: [27]**

## 3.3 Implementación del proceso de comunicación

Una vez expuestas las herramientas que se necesitan para la elaboración del sistema, se procede a la explicación de su estructura. Tres son, en ese sentido, las partes que lo constituyen: emisión, protocolo FT8 para su propagación, y recepción.

Ahora bien, dadas las similitudes, tanto de herramientas como de estructura, se comienza explicando la emisión y recepción de la información, dando paso posteirormente a la explicación del protocolo FT8.

### *3.3.1 Emisión*

Tal y como se menciona anteriormente, Python es el lenguaje de programación seleccionado (sección **¡Error! No se encuentra el origen de la referencia.**) y PyCharm, el entorno de programación (sección **¡Error! No se encuentra el origen de la referencia.**).

Al realizar la programación del código, se afrontan los siguientes problemas:

1. ¿Qué información se puede transmitir?

2. ¿Cómo reconoce el código los datos que quiere transmitir el usuario?
3. ¿Cómo se debe transmitir el mensaje para que el dispositivo en recepción sea compatible?

Una vez expuestas las cuestiones que se deben afrontar, se establece la Figura 22 a modo de subíndice. Cada uno de los escalones mostrados hace referencia a la última de las versiones de la solución adoptada ante cada pregunta. De esta forma, el lector es capaz de identificar, de forma sencilla, en qué parte del proceso de emisión se encuentra.

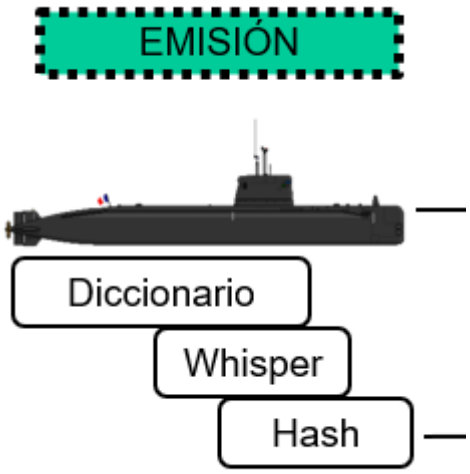



**Figura 22. Esquema emisión. Fuente propia.**

### *3.3.1.1 Diccionario*

En primer lugar, se procede a resolver la cuestión relativa a qué información se puede transmitir. Para ello, el primer paso a realizar es proporcionar al código un vocabulario de palabras o diccionario, que permite consultar si dispone, o no, de las palabras producidas por el emisor.

Este diccionario se define como un conjunto lo más extenso posible de palabras [21] que atiende a una naturaleza común (el ámbito submarino), sin ser este último un aspecto exclusivo. La intención, pues, es realizar un diccionario apto para comunicaciones submarinas.

Si bien es cierto que el objeto del proyecto no es tanto la creación de este vocabulario, sino su codificación y transmisión, se ha creído oportuno la creación de un pequeño corpus submarino que sirva de prueba de concepto para aportar un nivel de aplicación lo más real posible a su hipotética utilización.

Para ello, se debe entender que en materia de defensa existen numerosos términos que se pueden introducir en nuestro diccionario, sin embargo, presentan cierto grado de confidencialidad. Se resalta este aspecto porque, si bien es cierto no se va a introducir contenido confidencial, resultaría sencillo introducirlo si en alguna operación se llegase a usar.

Así pues, el contenido del diccionario se basa en dos fuentes fundamentales:

- Real Academia Española:
  Aquí se pueden encontrar corpus de referencia del español actual en base a la frecuencia de uso de las palabras. Esto es, los listados de las 1000, 5000, 10000 y listado total de las palabras más frecuentes de la lengua española.

- *Joint Brevity Words Publication* APP-7 (E):

  Publicación de la OTAN que proporciona el conjunto de *Brevity Words* usadas, en este caso, para el entorno submarino.

Las *Brevity Words* son un conjunto de términos simples que se usan en el entorno de las comunicaciones militares para definir, de forma clara y concisa, algún aspecto de interés de mayor longitud al hablar en claro. Para su entendimiento, ponemos una serie de ejemplos:

- Caddilac: Sonoboya direccional activa
- Goblin: objeto sumergido identificado
- Cold: contacto sonar perdido.

De esta forma, combinando las dos fuentes explicadas, se dispone de un vocabulario que sirve de biblioteca base para los interlocutores. Este vocabulario se almacena en un fichero de texto `txt`, al cual se accede y busca cada una de las palabras pronunciadas por el usuario, aspecto que se comenta posteriormente.

El diccionario es, en ese sentido, la solución de la primera de las cuestiones planteadas Figura 23, relativa a la creación de un diccionario de palabras:




**Figura 23. Esquema del progreso de código de emisión. Parte 1: Corpus. Fuente propia.**

### *3.3.1.2 Reconocimiento de voz y transcripción.*

Se encuentra, pues, ante la segunda problemática que se debe resolver. El sistema ha de ser capaz de realizar el reconocimiento de voz para transcribir el lenguaje hablado a texto. Para realizar dicha tarea, se ha partido de un código básico sobre el cual, se han añadido mejoras.

El modelo inicial se presenta como un sistema dependiente de la conexión a internet, el cual se modifica posteriormente para que se pueda usar el sistema de forma *offline*. El reconocimiento de datos *offline* se realiza, en primer lugar, mediante la utilización de archivos de audio `.wav`, dotando finalmente al sistema la capacidad de procesamiento de los datos sin la necesidad de emplear directamente dichos archivos.

A continuación, se muestra un esquema (Figura 24) de las diferentes implementaciones que se han añadido al sistema para mejorar el reconocimiento de voz y su transcripción.

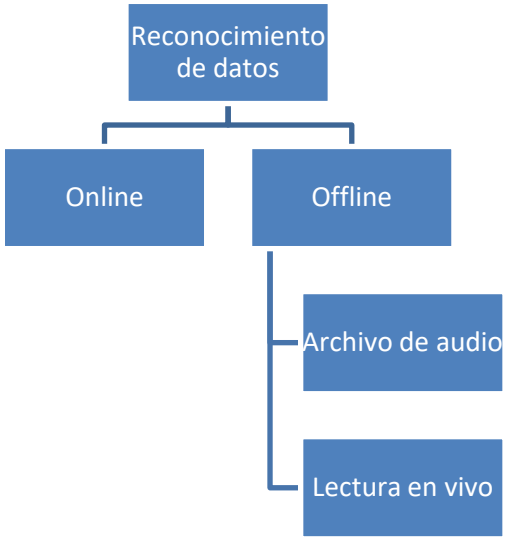



**Figura 24. Modalidades estudiadas para el reconocimiento de datos. Fuente propia.**

La metodología seguida para explicar cada una de las implementaciones es común para cada una de ellas:

- En primer lugar, se presentan las librerías a usar, explicando de forma resumida su función.
- A continuación, se da paso al cuerpo del código, exponiendo detalladamente cada una de las acciones.
- Las librerías, funciones, etc. que se expliquen en un código se dan por expuestas en los posteriores.

### *3.3.1.2.1 Reconocimiento modalidad "online"*

En una implementación inicial, se trató de realizar este reconocimiento mediante la librería de *Speech Recognition de Google*. Dicha librería permite al código realizar el reconocimiento de voz de forma "online".

Para ello, se importan las librerías que se requieren durante la ejecución del código (Figura 25). Al importar estos módulos se prepara al entorno para usarlos posteriormente en el código:

```
import speech_recognition as sr


import time
```

**Figura 25. Librerías reconocimiento voz "online". Fuente propia.**

Destacar que estas librerías deben ser previamente instaladas (sección 3.2.3).

- `speech_recognition`: permite reconocer voz a través de diferentes fuentes de entrada, la cual se especifica posteriormente.
- `time`: proporciona la función para trabajar con el tiempo, de forma que se pueda medir o esperar un tiempo determinado.

```
reconocimiento = sr.Recognizer()


lista_palabras = []


start_time = time.time()
print("¿Qué desea transmitir?")

while time.time() - start_time < 7:
    # bucle while de 7 segundos
    with sr.Microphone() as source:
        audio = reconocimiento.listen(source)
        try:
            lista_palabras = reconocimiento.recognize_google(audio, language="es-ES").split()


        except:
            pass
```

**Figura 26. Reconocimiento "online" de la información. Fuente propia.**

Atendiendo a la Figura 26, el objeto `Recognizer` permite escuchar y reconocer las palabras emitidas por el usuario. La variable `lista_palabras` almacena las palabras detectadas por el emisor.

Se establece un temporizador de siete segundos para permitir al usuario hablar. En el bucle `while`, establece la fuente de audio del micrófono como `source`, y más tarde, se hace uso del método `listen` para grabar el audio de la fuente.

Si el audio es correctamente reconocido por el `recognize` de Google, las palabras se agregan a la `lista_palabras`. En cambio, si el audio no se reconoce correctamente, el código simplemente continúa ejecutando el bucle.

### *3.3.1.2.2 Reconocimiento modalidad "offline"*

Atendiendo al entorno natural donde se realizan las comunicaciones, esto es, el submarino, resulta de obvia evidencia la falta de conexión a internet. De esta forma, el siguiente paso que se ha realizado es la transición del reconocimiento de voz de forma *online* a *offline*, es decir, sin la necesidad de disponer de conexión a internet.

- ***Mediante uso de grabaciones de audio***

Para realizar el reconocimiento de voz de forma *offline*, el primer modelo que se plantea es mediante el uso de ficheros con grabaciones de audio. Para ello, hicimos uso de la librería Whisper (Figura 27).

```
import pywhisper
import wave
import numpy as np
```

**Figura 27. Librerías reconocimiento de voz "offline". Fuente propia.**

- `pywhisper`: proporciona la capacidad de transcribir el mensaje emitido por el usuario [34].

- `wave`: librería que permite leer y escribir archivos de audio. `wav` (*Waveform Audio File Format*) [34].
- `numpy`: librería que proporciona una funcionalidad avanzada de computación numérica. [34]

Este primer sistema de la modalidad *offline* trata de transcribir el mensaje que desea emitir el usuario a partir de un archivo `.wav` (Figura 28).

```
# Cargar el modelo pre-entrenado
model = pywhisper.load_model("base", device="cpu")

# Realizar la transcripción del audio
try:
    result = model.transcribe("C:\\Users\\Santiago\\Desktop\\tfg2\\comunicaciones\\transmision\\audio.wav")
    text = result['text']  # Acceder al valor 'text' en el diccionario 'result'
    lista_palabras = text.split()
    print(lista_palabras)
except ValueError:
    print("No se pudo entender el audio")
```

**Figura 28. Reconocimiento de voz "offline". Fuente propia.**

En primer lugar, se carga el modelo de reconocimiento `base` de voz mediante la función `pywhisper.load.model` y se almacena en la variable `model`. Durante el proyecto se opta por utilizar el modelo `base` debido a su eficiencia en relación la capacidad de los equipos disponibles, así como las necesidades de los experimentos a realizar. Es importante destacar que el modelo puede ser modificado en cualquier momento si el usuario lo desea, atendiendo a los modelos expuestos en la Figura 20

A continuación, se transcribe el mensaje grabado por el usuario que se encuentra en el archivo de audio ubicado en la dirección que aparece en el código (`C:\Users\Santiago\Desktop\tfg2\comunicaciones\transmision\audio.wav`). Una vez realizado, y haciendo uso de la función `model.transcribe()`, se transcribe el audio y se almacena en la variable `result`.

Finalmente, mediante `text.split()` se divide la cadena de texto en una lista de palabras, almacenándolas en lista_palabras. Si se produce algún error durante el proceso, se imprimirá por pantalla el aviso de que no se pudo entender el audio.

- ***Reconocimiento modalidad en vivo***

Se trata de la versión definitiva creada para el reconocimiento de audios transmitidos por el usuario. Whisper nos proporciona la capacidad de realizar el reconocimiento de voz y su procesamiento de forma *offline*. Se debe reseñar que, aunque parezca que la tarea se realiza directamente sobre el audio pronunciado, la realidad es que esta librería realiza sus funciones creando ficheros de audio temporales, los cuales son almacenados en archivos y directorios, que, de la misma forma, también son temporales.

Es por ello que se implementa una última modificación para conseguir así, un método de reconocimiento de voz y procesamiento de datos de forma *offline* y sin la necesidad de gestionar archivos de audio [24].

```python
import io
from pydub import AudioSegment
import speech_recognition as sr
import whisper
import tempfile
import os
```

**Figura 29. Librerías reconocimiento de voz "en vivo". Fuente propia.**

A continuación, se explica la utilidad de las librerías importadas para la correcta ejecución de esta funcionalidad del sistema, ilustradas en la Figura 29.

- `io` proporciona las herramientas necesarias para trabajar con flujos de datos en Python
- `pydub` permite manipular y gestionar paquetes de audio
- `whisper` es la librería seleccionada para el reconocimiento de voz usando modelos de lenguaje
- `tempfile` proporciona las herramientas para la creación de directorios y archivos temporales.
- `os` permite interactuar con el sistema operativo.

```python
lista_palabras = []

temp_dir = tempfile.mkdtemp()
save_path = os.path.join(temp_dir, "temp.wav")
r = sr.Recognizer()
```

**Figura 30. Directorios y archivos temporales. Fuente propia.**

Una vez importadas las librerías, es necesario crear un directorio temporal usando la función `mkdtemp`, del módulo `tempfile`, tal y como se ilustra en la Figura 30. Esto es necesario para almacenar los archivos de audio de forma temporal, los cuales son usados para la grabación y su posterior transcripción. De este modo, el archivo temporal de audio se almacena en el directorio temporal con el nombre `temp.wav`. Estos archivos son automáticamente eliminados una vez cerrado el programa.

Una vez creados tanto el fichero de audio temporal como la carpeta que lo va a contener, se realiza la captura de audio y su transcripción. Para ello, se llama a la función denominada `transcribe`, estableciendo los valores mostrados en la Figura 31 para cada parámetro.

```python
transcribe(model="base", language="spanish", mic_energy=100, pause_duration=0.5, mic_dynamic_energy=True,
           stop_word="cambio")
```

**Figura 31. Función "Transcribe". Fuente propia.**

A este respecto, la función `transcribe` toma seis argumentos: `model`, `language`, `mic_energy`, `pause_duration`, `mic_dynamic_energy` y `stop_word`.

```python
def check_stop_word(predicted_text: str, stop_word: str) -> bool:
    import re
    pattern = re.compile('[\W_]+', re.UNICODE)
    return pattern.sub('', predicted_text).lower() == stop_word
```

**Figura 32. Función para detección de palabra de detención. Fuente propia.**

Con esta función (Figura 32) se consigue establecer una palabra para detener la transcripción de texto. De esta forma, se indica al sistema que se ha finalizado el mensaje de emisión por parte del usuario

Concretamente, esta función toma el argumento `predicted_text`, el cual representa el texto transcrito por el reconocimiento de voz, mientras que `stop_word` es una cadena texto que representa la palabra clave que se establece para parar la transcripción del texto. A continuación, se explica en que consiste la función `transcribe`, ilustrada en la Figura 33.

```
def transcribe(model, language, mic_energy, pause_duration, mic_dynamic_energy, stop_word):
    # there are no english models for large
    if model != "base" and language == 'spanish':
        model = model + ".es"
    audio_model = whisper.load_model(model)

    with sr.Microphone(sample_rate=16000) as source:
        print("Que desea transmitir")
        start_time = time.time()
        while True:
            if time.time() - start_time >= 7:
                break

            audio = r.listen(source)
            data = io.BytesIO(audio.get_wav_data())
            audio_clip = AudioSegment.from_file(data)
            audio_clip.export(save_path, format="wav")

            if language == 'spanish':
                result = audio_model.transcribe(save_path, language='spanish')
            else:
                result = audio_model.transcribe(save_path)

            predicted_text = result["text"]
            lista_palabras.extend(predicted_text.split())  # dividir la oración en palabras y agregar a la lista
            print("Text: " + predicted_text)
        if check_stop_word(predicted_text, stop_word):
            break

print("Ha finalizador la transmision del mensaje")
```

**Figura 33. Explicación función "transcribe". Fuente propia.**

En primer lugar, se comprueba qué modelo se debe usar en función del idioma en el que emite el usuario y, posteriormente, se carga el modelo de reconocimiento haciendo uso del módulo de Whisper. El uso de un modelo u otro depende de la velocidad relativa necesaria así como del espacio que el usuario pretenda ocupar (Figura 20).

El audio grabado se convierte en un objeto `AudioSegment` usando el módulo `pydub`, guardándose como un archivo de audio temporal `.wav` en la ruta definida anteriormente. A continuación, se transcribe el archivo `.wav` haciendo uso del reconocimiento de voz.

La transcripción se almacena en la variable `predicted_text`, y se agrega a la lista de palabras mediante la función `extend`.

El bucle se interrumpe cuando se cumpla alguna de las dos condiciones:

1. El temporizador alcanza el tiempo de 7siete segundos
2. Se detecta la palabra **cambio**, establecida como `stop_word`.

En ambos casos se imprime un mensaje por pantalla indicando al usuario que ha finalizado la transmisión del mensaje.

Una vez finalizada esta parte, el sistema tendrá almacenadas las palabras que desea transmitir el usuario en una variable. Se completa así, la segunda de las cuestiones expuestas al principio del apartado 3.3.1, relativa a cómo puede el código reconocer los datos que quería transmitir el usuario. Se alcanza, de esta forma, el segundo de los escalones del esquema de emisión (Figura 34).




**Figura 34. Esquema del progreso de código de emisión. Parte 2: Whisper. Fuente propia.**

### *3.3.1.3 Generación del mensaje de emisión*

En este apartado se trata la tercera y última parte de las soluciones propuestas ante las problemáticas expuestas para la creación de la información a transmitir.

Una vez almacenadas las palabras que el usuario desea transmitir, es necesario realizar una codificación de las mismas para que se puedan emitir mediante el protocolo ft8, ya que no es posible enviarlas de forma directa. Para poder realizar esa codificación, se debe comprobar si se encuentran en el vocabulario común entre emisor y receptor.

Cabe destacar que el contenido propuesto para el diccionario es cerrado y común tanto para los usuarios de emisión como para los de recepción. Es decir, las posibles palabras contenidas en el mensaje están incluidas en el diccionario de sendas partes, ya que usan el mismo vocabulario.

Si bien es cierto que en un primer momento pudiéramos crear este diccionario de forma automática e interactiva preguntando al usuario por pantalla si desea añadir cierta palabra que no estuviera contenida en él, es importante recalcar que nunca se debería hacer durante una transmisión, ya que, tal y como se comentó, este vocabulario tiene que ser idéntico a ambos lados de la comunicación.

Así pues, se establece el diccionario como un conjunto de palabras máximas conocidas por todas las partes, inalterable y previamente establecido.

Para la comprobación de la existencia o no de las palabras en el vocabulario común, esto es, el corpus, se proponen dos alternativas, siendo la primera la primitiva y presentando la segunda como la opción más eficiente y óptima para nuestra aplicación.

- ***Mediante el uso de listas.***

```python
def tabla_corpus(word, corpus, datos_emision):
    with open(corpus) as f:
        lista_corpus = [word.strip() for word in f]

    if word in lista_corpus:
        print(str(word) + " - " + str(lista_corpus.index(word) + 1))
        datos_emision.append(lista_corpus.index(word) + 1)

    else:
        print("La palabra '" + word + "' no se reconoce.")
```

**Figura 35. Tabla corpus. Fuente propia.**

En este caso, tal y como se observa en la Figura 35, se define la función `tabla_corpus`, la cual tomará tres argumentos: una palabra, el archivo de texto del corpus, y una lista denominada `datos_emision`.

Se abre el archivo de texto y, si la palabra solicitada por el usuario se encuentra en la `lista_corpus`, agrega su índice a la lista `datos_emision`. Si, por el contrario, la palabra no se encuentra en el corpus, se imprime por pantalla al usuario que la palabra no se reconoce. Cabe destacar que al índice se le suma una unidad, ya que el código identifica la primera palabra con la posición cero.

Posteriormente, se llama a la función y se imprime por pantalla los números que representan los índices a los que están asociadas las palabras. Estos números son almacenados en la variable `datos_emision`.

- ***Mediante el uso de tabla HASH.***

La segunda alternativa para comprobar la compatibilidad del mensaje y la composición del mismo son las denominadas Tablas Hash.

Como se ha comentado en la sección 2.6, las estructuras o tablas Hash son usadas en el ámbito de la programación para realizar búsquedas e inserción de forma eficiente para un número muy elevado de datos. Se trata de una tabla que almacena un conjunto de pares “clave-valor” [16].

Es decir, la tabla Hash es una función que, dada una clave de un elemento, devuelve, en este caso, el índice de la tabla hash donde se encuentre almacenado dicho elemento, tal y como podemos apreciar en la Figura 36. La implementación de una tabla hash está basada en los siguientes elementos:

- Clave es única para cada elemento de la tabla. En este caso, la clave es cada palabra del diccionario, contenidas en el archivo `corpus_1000.txt`.
- El valor es el número de línea en el que aparece o está posicionada la palabra, es decir, su índice en el archivo.

Cabe destacar que el diccionario que contiene las palabras (`corpus_1000.txt`) es denominado de tal forma debido a que contiene las mil palabras mas usadas en nuestro idioma. Dicho diccionario ha sido adquirido a partir de la RAE (Real Academia Española) [33].

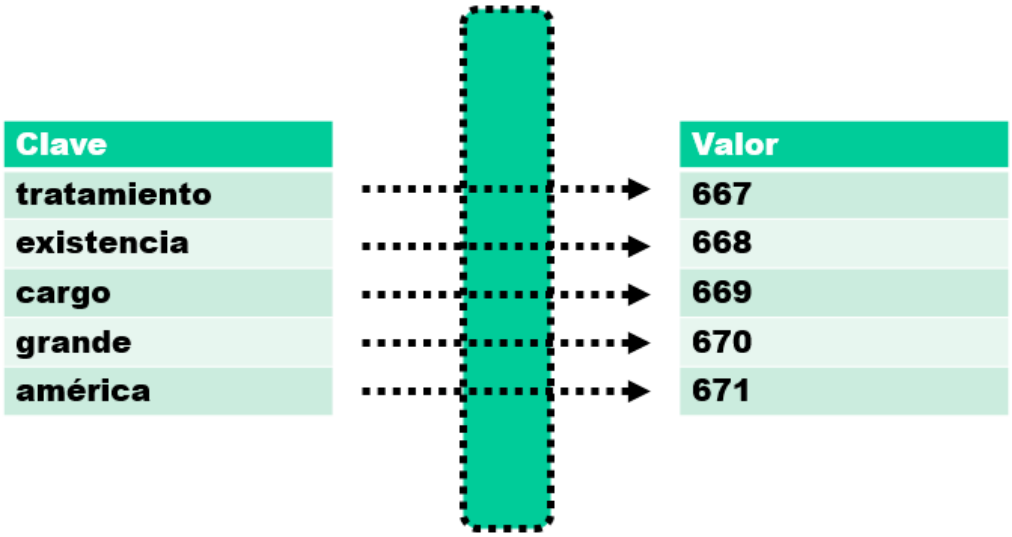



**Figura 36. Tabla hash Corpus. Fuente propia.**

Tomando como referencia la Figura 37, la tabla hash es construida mediante dos bucles `for` que se proceden a describir a continuación:

- En primer lugar, se lee cada línea del archivo, y con la función `enumerate`, se cuenta el número de línea en cada iteración del bucle. Posteriormente, se itera sobre cada palabra, se asocia con su número de línea correspondiente en el diccionario haciendo uso de la sintaxis `hash[Word] = i +1`, ya que empieza contando por el 0.
  De esta forma, al final de la ejecución del bucle, se crea una tabla hash que asocia cada palabra del archivo con la línea correspondiente.

```
hash = {}
with open("corpus_1000.txt", "r") as f:
    for i, line in enumerate(f):
        for word in line.strip().split(" "):
            hash[word] = i+1
```

**Figura 37. Explicación Tabla Hash. Fuente propia.**

- En segundo lugar, se genera otro bucle *for*, con el cual se trata de buscar cada palabra solicitada por el usuario, contenida en `lista_palabras`, en la tabla hash. Posteriormente, se devuelve su número de línea correspondiente en caso de que se encuentre en el archivo.

```
n3_list = []

for word in lista_palabras:
    try:
        print("La palabra '{}' tiene asignada la posicion {}".format(word, hash[word]))
        n = [hash[word.rstrip('.')]]
        n3 = str(hash[word.rstrip('.')]).zfill(3)
        n3_list.append(n3)
    except KeyError:
        # La palabra no está en el fichero
        print("La palabra '{}' no se encuentra en el corpus.".format(word))

n3_str = ",".join(n3_list)

n3_str_con_espacios = " ".join(n3_list)
```

**Figura 38. Funcionamiento de la tabla Hash. Fuente propia.**

Se ha hecho uso de la estructura `try-except` para manejar la casuística posible, es decir, que la palabra se encuentre, o no, en el corpus, tal y como se muestra en la Figura 38. Es importante explicar que:

- La variable `n` almacena el índice que corresponde a cada palabra.
- La variable `n3` introduce los datos de la variable `n` con tres dígitos. Es decir, el número "3" almacenado en la variable `n`, se almacena en `n3` como "003". Además, se convierten estos números a cadenas de texto para procesarlo posteriormente. La razón de esta conversión se explica posteriormente.
- Los datos se almacenan en la variable `n3_list`.

Los datos, considerados como cadenas de texto, se separan en grupos de tres y se guardan en una variable denominada `n3_str_con_espacios`.

Finalmente, se alcanza el último de los escalones para lograr la parte de emisión del mensaje (Figura 39). De esta forma, el sistema es capaz de generar el mensaje decodificable con el receptor y compatible con el protocolo de comunicaciones FT8.

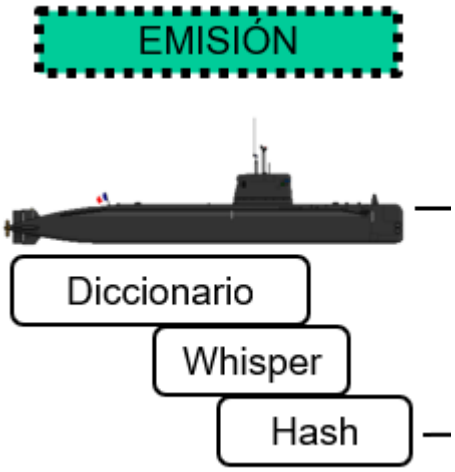



**Figura 39. Esquema del progreso de código de emisión. Parte 3: Tabla Hash. Fuente propia.**

### *3.3.2 Recepción*

Una vez finalizada la explicación del sistema de emisión, se da paso a la presentación del sistema de recepción. El presente apartado se divide en dos grandes bloques, presentando en primer lugar las dos formas planteadas para recibir los mensajes, y en segundo lugar la explicación de su correspondiente decodificación.

#### *3.3.2.1 Métodos de recepción*

La recepción del mensaje se realiza atendiendo a dos necesidades para la elaboración de resultados, a saber, el estudio de la capacidad máxima de relación señal a ruido (SNR) que es capaz de soportar, y la propia elaboración funcional del código.

La recepción de mensajes mediante archivos es la destinada al estudio cuantitativo de la capacidad de relación señal a ruido (SNR) que es capaz de soportar el sistema. El procedimiento se ha realizado mediante la utilización de archivos de audio, con el objeto de producir ruido de forma artificial. Esto permite enviar el mensaje con los parámetros de potencia de ruido, potencia de señal y la relación entre ellos (SNR) perfectamente conocidos, sin modificar su valor con señales externas que pudieran surgir durante la propagación de la señal.

Por su parte, la recepción de mensajes en vivo es la destinada a la prueba funcional del código, tratando de simular su aplicación en una situación práctica lo más realista posible.

##### *3.3.2.1.1 Recepción mediante archivos*

En ese apartado se procede a realizar la explicación del proceso de recepción a partir de archivos de audio, destinada al estudio cuantitativo de la relación señal a ruido (SNR). Para ello, se muestran las librerías usadas durante el proceso en la Figura 40:

```
import ft8
import datetime
import time
import scipy.io.wavfile
```

**Figura 40. Librerías de recepción de datos. Fuente propia.**

- `ft8`: Librería que implementa la demodulación de la señal recibida según el protocolo ft8.
- `scipy.io.wavfile`: Proporciona las funciones para leer y escribir archivos de audio en el formato de audio.wav.
  Esta librería permite cargar los archivos, procesar y analizarlos, obteniendo, por ejemplo, el análisis de la SNR.

```
seconds = datetime.datetime.now().second
while seconds not in {0, 15, 30, 45}:
    time.sleep(0.01)
    seconds = datetime.datetime.now().second

sample_rate, samples = scipy.io.wavfile.read('audio.wav')

start = time.time()
analysis = ft8.SpectralAnalysis(samples, sample_rate)
end = time.time()

ft8.Signal.demap_max_symbols = 1

signals = []

for i, candidate in enumerate(analysis.candidate_list):
    try:
        s = ft8.Signal(candidate, analysis)
        signals.append(s)
    except ValueError:
        pass
```

**Figura 41. Recepción de datos. Fuente propia.**

A continuación, el código mostrado en la Figura 41 espera a que los segundos actuales sean múltiplos de quince antes de comenzar. De esta forma, se consigue una mayor sincronización entre las partes. Posteriormente, se lee el archivo .`wav` usando la función `scipy.io.wavfile`, la cual devuelve los datos del audio, así como la tasa de muestreo, almacenando la información en las variables `samples` y `sample_rate`.

Para cada "candidato", se intenta decodificar con `ft8.signal`, de forma que, si se crea exitosamente, se agrega la información a la lista `signals`, de lo contrario, se salta de candidato.

Finalmente, se almacenan todas las señales decodificadas exitosamente en la lista `signals`.

```
for i, s in enumerate(signals):
    print("{:6d} {:5.1f}s {:7.0f}Hz {:3.0f}".format(i, s.offset, s.freq, s.snr))

for s in signals:
    words = [word for word in s.msg.text.split() if word.isdigit()]
    numeros_recibidos = ''.join(words)
    print("Numeros recibidos: ", numeros_recibidos)
```

**Figura 42. Obtención de datos en recepción. Fuente propia.**

Atendiendo a la Figura 42, se extraen los parámetros offset, frecuencia y relación señal a ruido (SNR), para posteriormente, recorrer la lista `signals` y extraer los números de cada mensaje recibido. Dichos números son concatenados y almacenados en la variable `numeros_recibidos`.

### *3.3.2.1.2 Recepción en vivo*

El siguiente código mostrado en la Figura 43, tiene como objetivo realizar la decodificación de la señal recibida obteniendo los datos sin la necesidad de gestionar paquetes de audio, esto es, en vivo.

```
sd.default.device = 1 # Micrófono #check with sd.query_devices()
print('Recording signal...')
duration = 15  # seconds
sample_rate = 24000
samples = sd.rec(int(duration * sample_rate), samplerate=sample_rate, channels=1, dtype='int16')
sd.wait()
samples = samples.reshape((-1,))

start = time.time()
analysis = ft8.SpectralAnalysis(samples, sample_rate)
end = time.time()
print("Analysis time = {:.1f} ms".format(1000*(end - start)))

ft8.Signal.demap_max_symbols = 1
signals = []
```

**Figura 43. Recepción de datos "en vivo" Fuente propia.**

Para comenzar, se configura el dispositivo de audio predeterminado como el micrófono, imprimiendo posteriormente por pantalla que se está grabando el audio para indicar al usuario que la aplicación comienza a escuchar. La duración del audio dura quince segundos (cumpliendo con el estándar del protocolo), a una tasa de muestreo de 24000 Hz. Se hace uso de la biblioteca `sounddevice` para realizar la grabación de audio, almacenándolo posteriormente en la variable `simples`. Finalmente, se almacenan los datos en la lista vacía `signals`.

Siguiendo la lógica de la metodología anterior, mostrada en la Figura 42, los números recibidos son concatenados y almacenados en la variable `numeros_recibidos`, finalizando así esta modalidad de recepción.

Cabe destacar que se expone únicamente la parte que difiere a la recepción mediante archivos de audio, siendo el resto de código el mismo, por lo que se dará por explicado.

Una vez recibida la información, se debe implantar un sistema para ser capaces de decodificarla y mostrar al usuario el mensaje en claro. De esta forma, damos paso al proceso de decodificación.

### *3.3.2.2 Decodificación*

```
try:
    numeros_enteros = []
    for i in range(0, len(numeros_recibidos), 3):
        numeros_enteros.append(int(numeros_recibidos[i:i+3]))
    print(numeros_enteros)
    hash = {}
    with open("corpus_1000.txt", "r") as f:
        for i, line in enumerate(f):
            for word in line.strip().split(" "):
                hash[i+1] = word

    palabras_recibidas = []
    for num in numeros_enteros:
        try:
            word = hash[int(num)]
            palabras_recibidas.append(word)
            print("El número '{}' hace referencia a la palabra '{}'".format(num, word))
        except KeyError:
            # El número no está en la tabla hash
            print("El número '{}' no está asociado a ninguna palabra del fichero".format(num))

    print((palabras_recibidas))
except NameError:
    print("Límite de SNR superado")
```

**Figura 44. Obtención de palabras en recepción. Fuente propia.**

A continuación, el código de la Figura 44 trata de convertir los valores numéricos almacenados previamente a las palabras que lo representan.

Para realizar dicha tarea, el primer paso consiste en convertir la información de tres caracteres de la variable `numeros_recibidos` en un número entero, agregándolo, posteriormente a la lista `numeros_enteros` e imprimiéndola por pantalla. Esto se realiza gracias al bucle `for`, donde recorremos la lista `numeros_recibidos` en incrementos de 3, convirtiéndolos en un entero con la función `int()`.

Posteriormente, se hace uso de la tabla hash previamente explicada en contexto en la sección (2.6) y en ejecución (3.3.1.3) para descifrar el significado de cada uno de los números enteros. La clave es el número (añadiéndole la unidad, pues empieza a contar en cero), y el valor, la palabra.

Antes de terminar, se agregan las palabras a la variable `palabras_recibidas`, imprimiendo por pantalla el número y la palabra a la que hace referencia. Finalmente, se imprime por pantalla la variable `palabras_recibidas`, siendo este, el mensaje final.

Si se supera la señal a ruido soportada, la variable `palabras_recibidas` quedará vacía y se imprimirá por pantalla indicándoselo al usuario.

Una vez completados los dos procesos expuestos de recepción de la información, el sistema es capaz de generar el mensaje en la parte de emisión (sección: 3.3.1) compatible con el protocolo y recibir el mensaje decodificándolo correctamente en recepción. En ese sentido, se alcanza la situación mostrada en la Figura 45.

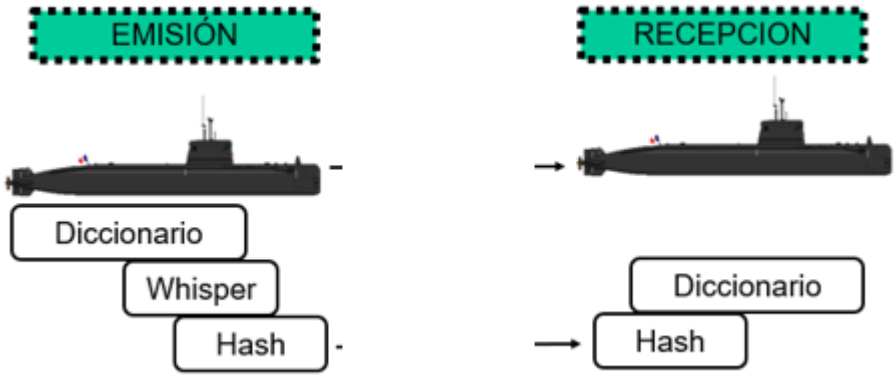



**Figura 45. Esquema del progreso del código. Proceso emisión y recepción. Fuente propia.**

### *3.3.3 Protocolo de comunicaciones FT8*

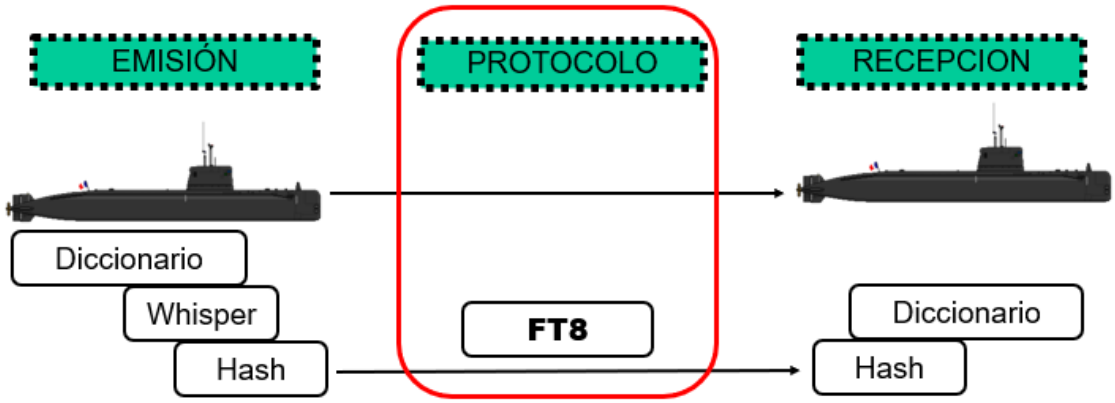



**Figura 46. Esquema del progreso de código. Proceso de transmisión. Fuente propia.**

En este apartado se expone la cuestión relativa al siguiente gran bloque del proceso de comunicaciones propuesto, mostrado en la Figura 46. La emisión, por consiguiente, ha sido correctamente realizada. Tres son, en ese sentido, las partes completadas durante el proceso.

1. Se ha elaborado un diccionario común entre las partes, a saber, emisor y receptor.
2. Se ha capturado el audio emitido por voz del emisor y se han transcrito esos datos.
3. Se han codificado dichos datos transcritos en función de la posición de la palabra en el corpus.

Es resumen, durante el proceso de comunicación, el protocolo recibirá un dato parecido al siguiente: "*016 840 001 325*". Se comienza entonces en este punto de partida para la explicación del protocolo FT8.

```python
def gaussian_boxcar(b, t):
    """Function to calculate gaussian filtered boxcar pulse waveform to perform M-GFSK modulation.

    The boxcar function is assumed to be centered around t = 0 with width 1.0 and amplitude 1.0.
    This function will return the value of the gaussian filtered boxcar at time t.
    The parameter t can be a scalar or a numpy array.
    The parameter b controls the bandwidth of the gaussian filter."""
    c = b * np.pi * np.sqrt(2.0 / np.log(2.0))
    return 0.5 * (scipy.special.erf(c * (t + 0.5)) - scipy.special.erf(c * (t - 0.5)))
```

**Figura 47. Función gaussian_boxcar. Fuente propia.**

En primer lugar, se define la función llamada `gaussian_boxcar`, mostrada en la Figura 47. Esta función permite calcular la forma de onda filtrada con una función gaussiana para poder realizar la modulación M-GFSK (sección 2.5.3), con lo que se obtiene una modulación parecida a la siguiente Figura 48:

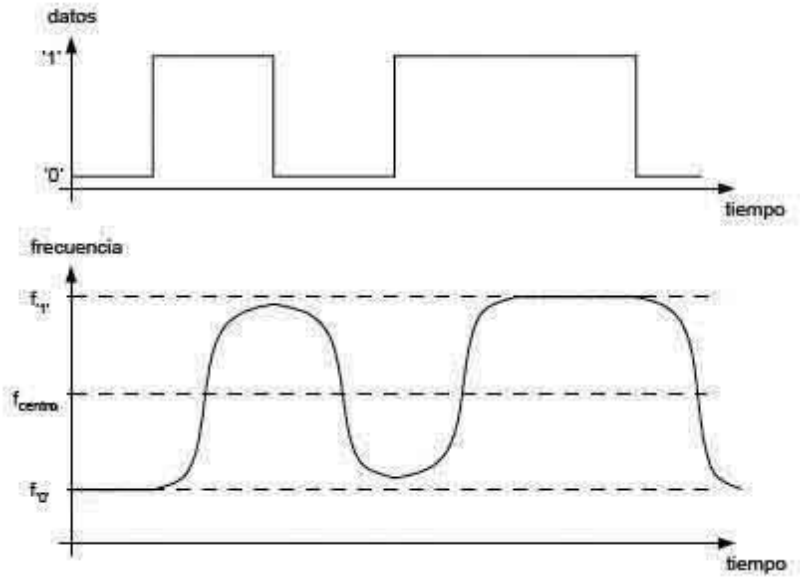



**Figura 48. Modulación GFSK**

Si bien es cierto ya se ha explicado detalladamente la modulación GFSK, no se ha hecho referencia a su modificación M-GFSK, la cual se usa para modular nuestra señal. Tan solo se debe añadir que la M-GFSK se caracteriza por tener una menor desviación de frecuencia (*Minimum Gaussian Frequency Shift Keying*).

Debido a la forma de la onda más suave y continua de la señal modulada, la M-GFSK presenta un ancho de banda ligeramente mayor que la GFSK, sin embargo, proporciona una mayor robustez contra interferencias.

```python
def modulation_waveform(symbols, samples_per_symbol, pulse):
    """Generate a modulation waveform from a list of symbols using the provided pulse waveform

    The pulse waveform must have a length that is equal to three symbol periods."""

    assert pulse.size == 3 * samples_per_symbol
    w = np.zeros((len(symbols) + 4) * samples_per_symbol)
    s = 0

    # Pad start and end of waveform with first and last symbol respectively
    for symbol in ([symbols[0]] + symbols + [symbols[-1]]):
        w[s: s + pulse.size] += symbol * pulse
        s += samples_per_symbol

    return w[samples_per_symbol * 2: samples_per_symbol * (2 + len(symbols))]
```

**Figura 49. Función modulation_waveform. Fuente propia.**

A continuación, con la función definida como `modulation_waveform`, mostrada en la Figura 49, se genera la forma de la onda a partir de una lista de símbolos usando el filtro gaussiano como forma de onda proporcionada. Así, `Symbols` es la lista que contiene los símbolos digitales que se desean transmitir, mientras que `simples_per_symbol` es la cantidad de muestras que se utilizan para representar cada símbolo.

En la función, se comprueba que la longitud de la forma de la onda del pulso sea igual a tres veces la cantidad de muestras por símbolo. Finalmente, la función itera a través de los símbolos de la lista `symbols`, agregándo a cada uno la forma de la onda de modulación donde se eliminan los dos primeros y últimos símbolos, los cuales son agregados para asegurar que la onda comience y finalice en un estado estable.

```python
def mgfsk(if_freq, offset, symbols, pulse, sample_rate):

    two_pi = 2.0 * np.pi
    samples_per_symbol = int(sample_rate / ft8.baud_rate)
    total_samples = ft8.tr_period * sample_rate
    cycle = 2.0 * np.pi / sample_rate
    dphase_symbol = cycle * ft8.freq_shift
    dphase_if_freq = cycle * if_freq

    # Calculate signal phases
    phases = np.zeros(total_samples)
    phase = 0
    sample = int(sample_rate * (ft8.start_delay + offset))
    mod_wave = modulation_waveform(symbols, samples_per_symbol, pulse)
    dphases = mod_wave * dphase_symbol + dphase_if_freq
    for dphase in dphases:
        phase += dphase
        phase %= two_pi
        phases[sample] = phase
        sample += 1
    return np.sin(phases)
```

**Figura 50. Función mgfsk. Fuente propia.**

En la Figura 50 se muestra la función `mgfsk`, que toma como entrada la frecuencia, el desplazamiento de tiempo, una lista de símbolos, una forma de pulso, y una tasa de muestreo.

Se trata de sintetizar la señal moduladora (M-GFSK) en un portador de frecuencia. Es decir, crea la señal de radiofrecuencia que contiene la información que ha sido modulada anteriormente por el filtro gaussiano.

Cada fase se obtiene con la frecuencia de la portadora y el cambio de fase debido al símbolo actual, reduciéndose posteriormente al rango de valores pertenecientes entre [0, 2π]. La frecuencia del portador resultante se establece en la frecuencia `if_freq`, siendo el resultado final la señal modulada que puede ser emitida por un transmisor radio.

```python
def simulate(signals, noise_db=30, sample_rate=24000):
    samples_per_symbol = int(sample_rate / ft8.baud_rate)
    total_samples = sample_rate * ft8.tr_period

    noise_sigma = 10.0 ** (noise_db / 20)
    noise_power = 2500 / sample_rate * 2 * noise_sigma * noise_sigma

    t = np.linspace(-1.5, 1.5, 3 * samples_per_symbol, endpoint=False)
    filtered_boxcar = gaussian_boxcar(ft8.gaussian_bandwidth, t)

    samples = np.random.normal(0, noise_sigma, total_samples)

    for message, if_freq, snr, offset in signals:
        tx_power = noise_power * 10 ** (snr / 10)
        amplitude = np.sqrt(2 * tx_power)
        symbols = message.encode()
        samples += amplitude * mgfsk(if_freq, offset, symbols, filtered_boxcar, sample_rate)


    samples.round()
    wavedata = samples.astype('int16')

    return wavedata
```

**Figura 51. Función simulate. Fuente propia.**

La función `simulate` mostrada en la Figura 51 trata de simular una señal de radio que contiene las señales FT8 y ruido. Se debe atender a los siguientes parámetros:

- `Signals`: lista de tuplas donde se contiene la señal de FT8 a simular, incluyendo el mensaje a enviar, la frecuencia `if_freq`, previamente definida, la relación señal a ruido `snr` y el desfase temporal.
- `Noise_db`: potencia de ruido en decibelios.
- Se calcula el pulso de onda filtrado por M-GFSK usando la función `gaussian_boxcar`, el cual se aplica a los símbolos de cada señal mediante la función `mgfsk` generando la señal modulada M-GFSK en el portador de frecuencia `if_freq`.

Se puede añadir ruido a las señales para crear la señal conjunta. Finalmente, la señal se convierte en una serie de 16 bits y se devuelve como resultado de la función.

```
MAX_SIZE = 12
cola = deque()
while len(mensaje) > MAX_SIZE:
    cola.append(mensaje[:MAX_SIZE])
    mensaje = mensaje[MAX_SIZE:]


if len(mensaje) > 0:
    cola.append(mensaje)

signals = []
freq = 50
while len(cola) > 0 and freq <= 5000:  # agregamos condición para la frecuencia máxima
    mensaje = cola.popleft()
    signals.append((ft8.TextMessage(mensaje), freq, 0, 0))
    freq += 100  # aumenta la frecuencia en 60 Hz para el siguiente mensaje

if freq > 5000:  # si la frecuencia superó el límite, imprimir la última palabra
    ultima_palabra = mensaje.split()[-1]
    print(f"No se pudo emitir a partir de la palabra '{ultima_palabra}' debido a la frecuencia máxima alcanzada.")
```

**Figura 52. Gestión mediante colas. Fuente propia.**

Para cumplir con el estándar del protocolo, se establece la máxima cantidad de caracteres por mensaje en 12. La cola, por consiguiente, se presenta como una herramienta para la gestión de información en caso de que se supere esta cantidad.

El código mostrado en la Figura 52 divide los mensajes en fragmentos de un tamaño máximo de 12 caracteres. Si el mensaje supera este valor, se divide en otros fragmentos y se añaden a la cola. En caso de que el mensaje sea igual o menor a la cantidad establecida, se agrega por completo a la cola.

Cabe destacar que, al trabajar entre las frecuencias de 50 y 5000 Hz, si se requieren 60 Hz para cada canal, se obtiene una capacidad de emisión de 82 canales. Es decir, se puede transmitir simultáneamente 82 canales con 4 palabras, esto es, 328 palabras en un solo mensaje.

```
signals = []
freq = 50
while len(cola) > 0 and freq <= 5000:  # agregamos condición para la frecuencia máxima
    mensaje = cola.popleft()
    signals.append((ft8.TextMessage(mensaje), freq, 0, 0))
    freq += 60

if freq > 5000:  # si la frecuencia superó el límite, imprimir la última palabra
    ultima_palabra = mensaje.split()[-1]
    print(f"No se pudo emitir a partir de la palabra '{ultima_palabra}' debido a la frecuencia máxima alcanzada.")
```

**Figura 53. Frecuencias de emisión. Fuente propia.**

A continuación, se proceden a enviar los fragmentos almacenados a través de señales de audio, tal y como se observa en la Figura 53. Para ello, se establece la frecuencia mínima de emisión en 50 Hz, cumpliendo, de esta forma, con el estándar del protocolo.

Con el bucle `while`, se consigue enviar cada uno de los mensajes aumentando en 60 Hz su frecuencia para cada grupo de 12 caracteres. Como parámetro, es importante remarcar que se puede variar la `snr` del código.

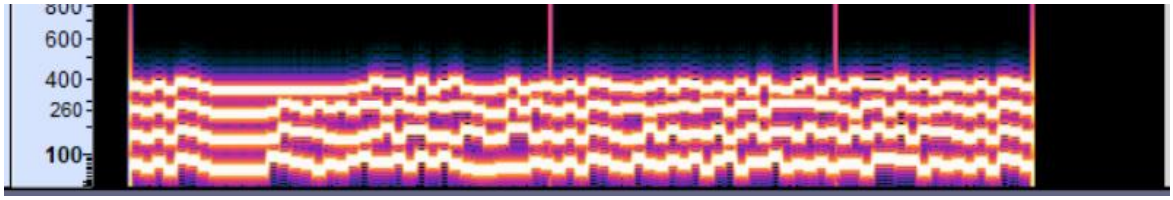



**Figura 54. Salto de frecuencias. Fuente propia.**

Se observa en la Figura 54 cómo se emiten cuatro mensajes, el primero en la frecuencia 50 Hz aumentando la misma hasta el último centrado en 350 Hz. Es decir, el emisor trata de emitir un mensaje que contiene, al menos, cuarenta y ocho caracteres

Comentar que en dicha imagen se ha establecido el aumento de frecuencia entre mensajes en 100 Hz para la correcta visualización y comprensión del lector, pues se presenta con una mayor claridad el salto entre frecuencias.

```
[(ft8.TextMessage('HOLA'), 500, -10, 0),
```

**Figura 55. Ejemplo datos de emisión de un mensaje. Fuente propia.**

A modo de ejemplo, la Figura 55 muestra el mensaje "*Hola*" emitido en una frecuencia de 500 Hz con una relación de señal a ruido (SNR) de -10 dB.

```
wavedata = simulate(signals, noise_db=0, sample_rate=sample_rate)
# Write to WAV file
scipy.io.wavfile.write('audio.wav', sample_rate, wavedata)

for i in range(1, 2):
    print('Generating burst nr. {}'.format(i))
    # Wait for TX slot
    print(' Waiting for TX slot...')
    seconds = datetime.datetime.now().second
    while seconds not in {0, 15, 30, 45}:
        time.sleep(0.01)
        seconds = datetime.datetime.now().second

    # Play FT8
    sd.play(wavedata, sample_rate)
    sd.wait()

    # Wait for TX slot
    print(' Waiting for TX slot...')
    seconds = datetime.datetime.now().second
    while seconds not in {0, 15, 30, 45}:
        time.sleep(0.01)
        seconds = datetime.datetime.now().second
```

**Figura 56. Transmisión de la señal. Fuente propia.**

Finalmente se simula la transmisión de la señal de audio con la función `simulate`. Para el estudio de la señal, se genera un archivo de audio `.wav` donde se guarda el resultado, tal y como se muestra en la Figura 56.

La gestión de tiempo se realiza mediante un bucle `while`, donde se envia el mensaje siempre que sean múltiplos de 15 segundos. De esta forma, se consigue estandarizar las comunicaciones entre las partes. Así, para emitir el siguiente paquete, se espera otro intervalo de 15 segundos.

La biblioteca `sounddevice` permite reproducir la señal de audio en el dispositivo por la salida de audio predeterminada, la cual se puede modificar.

# 4 RESULTADOS Y VALIDACIÓN

## 4.1 Búsqueda de límite de relación señal a ruido (SNR)

### *4.1.1 Búsqueda del límite de SNR con archivos de audio.*

El principal objetivo de este experimento es buscar y detectar el límite de relación señal a ruido (SNR). Para ello, se trata de emitir un mensaje disminuyendo la SNR hasta que el dispositivo en recepción sea incapaz de descodificar el mensaje.

Para lograr dicha búsqueda, se ha hecho uso del código relativo a la recepción de la información mediante archivos (sección 3.3.2.1.1). De esta forma, se consiguen introducir los parámetros de forma manual y asegurar que son esos mismos valores los que llegan a la recepción, sin ser estos modificados durante la transmisión por ruidos ambientales.

Se comienza introduciendo una SNR de 0 dB, es decir, introduciendo una potencia de ruido igual a la de la señal, tal y como se muestra en la Figura 57:

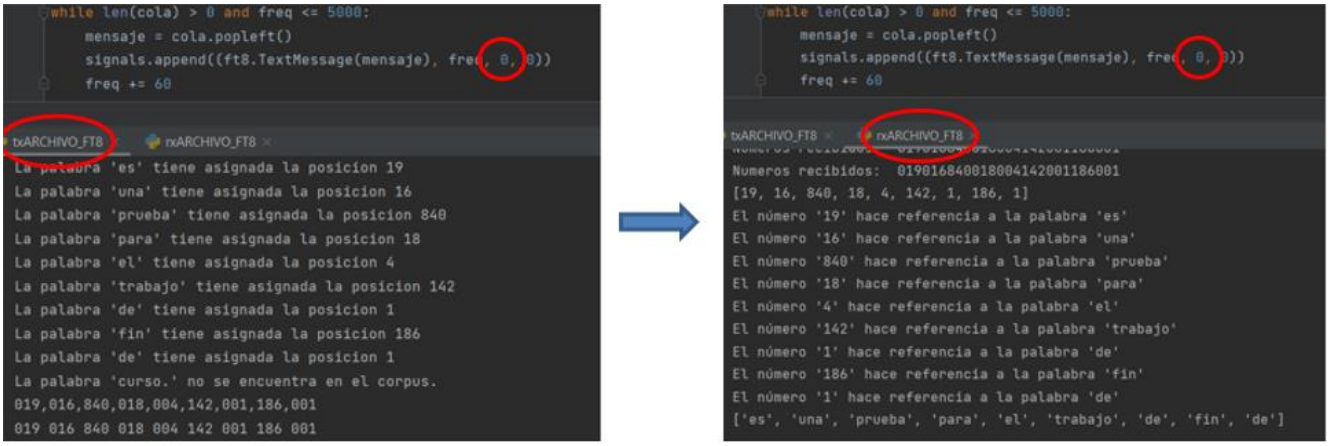



**Figura 57. Experimento SNR= 0 dB. Fuente propia.**

Tras comprobar su correcta recepción, la intención es ir disminuyendo el valor de la SNR hasta alcanzar el punto donde el dispositivo en recepción sea incapaz de decodificar el mensaje. De esta forma, se disminuye el valor a de la SNR a -10 dB, como se muestra en la Figura 58, que, tras obtener una correcta recepción de la información, se continua disminuyendo hasta -15 dB (Figura 59).

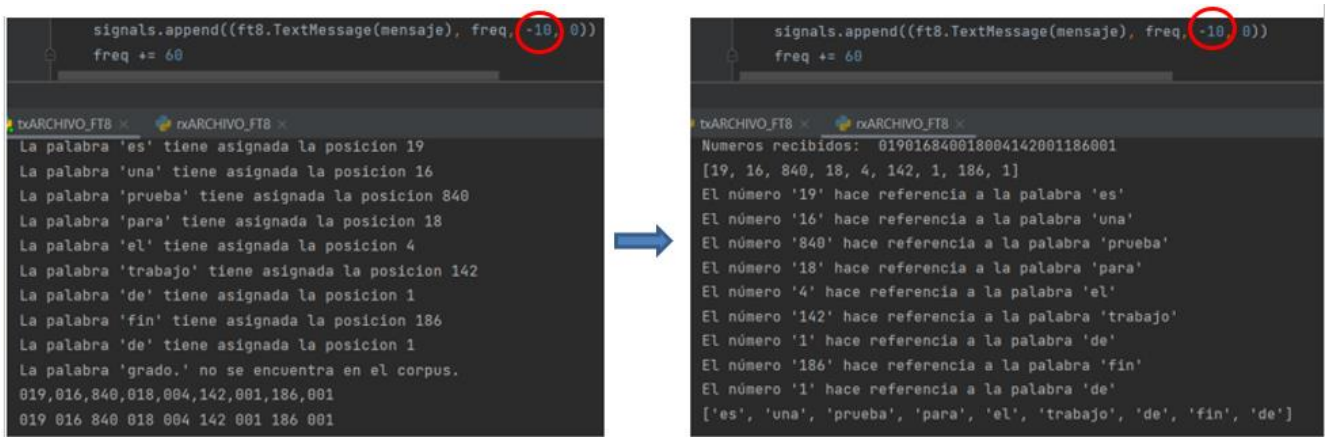



**Figura 58. Experimento SNR= -10 dB. Fuente propia.**

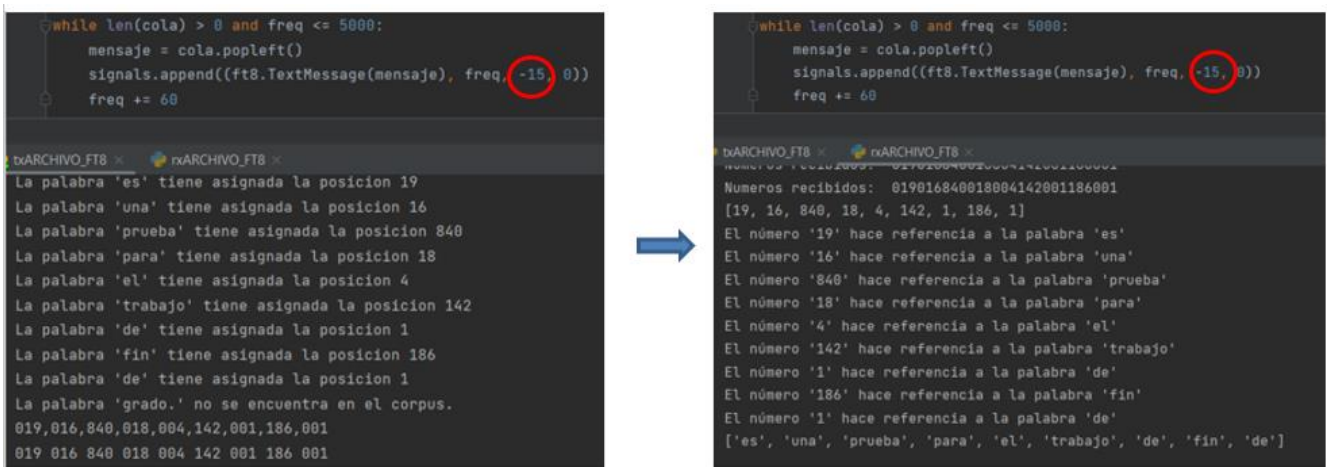



**Figura 59. Experimento SNR= -15 dB. Fuente propia.**

Tras detectar que con una SNR de -19 dB no se consigue detectar la información, como se ilustra en la Figura 60, se aumenta la relación hasta conseguir decodificar el mensaje. Este límite se encuentra en -18 dB, como se puede apreciar en la Figura 61.




**Figura 60. Experimento SNR= -19 dB. Fuente propia.**

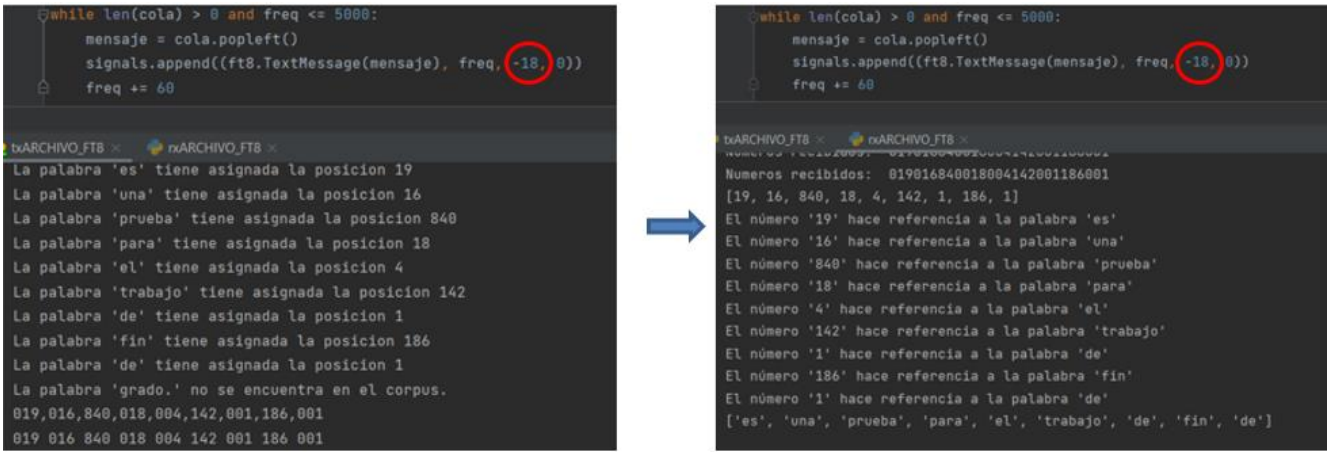



**Figura 61. Experimento SNR= -18 dB. Fuente propia.**

Una SNR de -18 dB significa que la potencia de la señal es 18 decibelios menor que la potencia de ruido. Atendiendo a las expresiones mostradas en Ecuación 1 y Ecuación 2, se observa la relación existente entre las potencias de señal y ruido:

$$SNR = 10 \times log_{10} \frac{P_{señal}}{P_{ruido}}$$

**Ecuación 1. Ecuación SNR. Fuente propia.**

$$0{,}0126 = \frac{P_{señal}}{P_{ruido}}$$

**Ecuación 2. Resultado SNR mediante archivos de audio. Fuente propia.**

Por tanto, se puede concluir que la potencia de la señal es aproximadamente 0,0126 veces la potencia del ruido.

### *4.1.2 Búsqueda del límite de la SNR con recepción en vivo.*

El segundo de los experimentos busca establecer el límite de relación señal a ruido mediante la recepción de datos en vivo, es decir, recibiendo la información sin la gestión de archivos de audio. Se presenta entonces como el experimento último y más real posible, pues si bien es cierto la anterior (sección 4.1.1) busca la exactitud del resultado, este trata de asemejarlo lo más posible a la realidad.

El proceso se realiza de la siguiente forma. En primer lugar, se ha obtenido el ruido producido por un motor de combustión, el cual simula el ruido ambiental. A continuación, se coge nuestro mensaje y se unifica mediante la aplicación de Audacity. En dicha aplicación, el usuario es capaz de establecer cuanto mayor o menor es la potencia de emisión de una de las señales respecto a la otra, es decir, podemos fijar la SNR.

Así pues, se comienza con la emisión de un mensaje y, posteriormente, se va disminuyendo esa relación hasta determinar el limite donde se deja de recibir por completo la información en recepción:

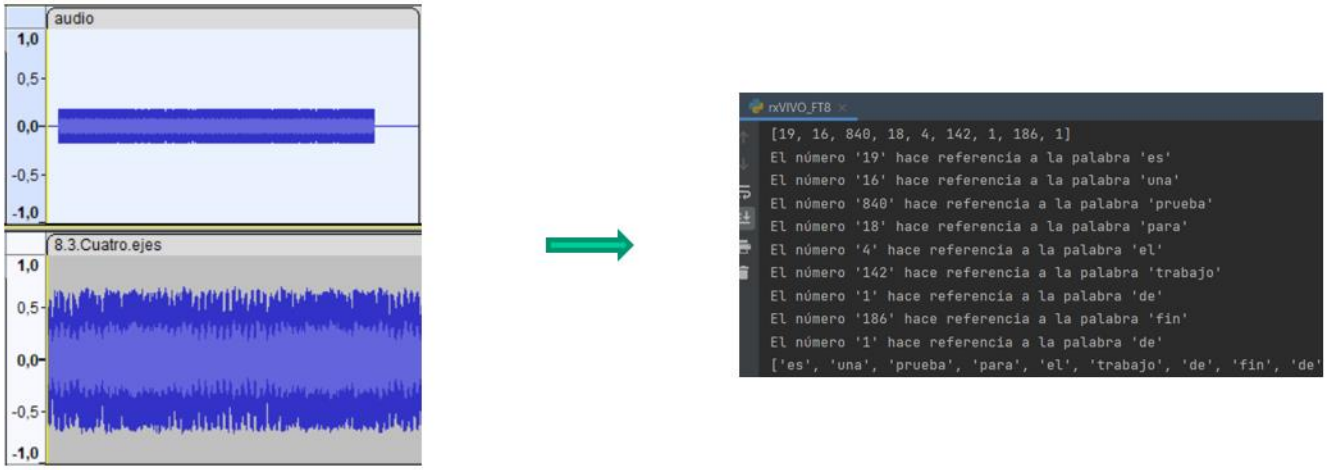



**Figura 62. Recepción de datos en vivo. SNR= -15 dB. Fuente propia.**

Se observa cómo la información es perfectamente decodificada con una SNR de -15 dB (Figura 62). El usuario recibe en vivo (sección 3.3.2.1.2) el conjunto de números en su sistema producidos por el emisor, que en este caso son:

```
[19, 16, 840, 18, 4, 142, 1, 186, 1]
```

Posteriormente, y haciendo uso del sistema explicado en la sección 3.3.2.2 para su correcta decodificación, se muestra por pantalla al usuario la información recibida.

Sin embargo, si se continúa disminuyendo la SNR, se detecta que existen pérdidas de paquetes en recepción. Esto ocurre cuando la SNR alcanza un valor de -17 dB, como se puede observar en la Figura 63. Recepción de datos en vivo. SNR= -17 dB.




**Figura 63. Recepción de datos en vivo. SNR= -17 dB. Fuente propia.**

Se puede observar cómo, atendiendo a la Figura 64, la pérdida total de información se produce cuando SNR= -19 dB.




**Figura 64. Recepción de datos en vivo. SNR= -19 dB. Fuente propia.**

Finalmente, se deduce que gracias al protocolo usado somos capaces de decodificar correctamente la información del mensaje con un ruido ambiental quince veces superior al de la potencia de la señal de información. Además, si se continúa disminuyendo la SNR, se irán perdiendo información en recepción, hasta alcanzar una SNR= -19 dB.

Se establece entonces el límite para una correcta transmisión en -15 dB, es decir, la potencia de la señal es aproximadamente 0,177 veces la potencia del ruido, como se puede apreciar en la Ecuación 3.

$$0{,}177 = \frac{P_{señal}}{P_{ruido}}$$

**Ecuación 3. Resultado SNR con recepción en vivo. Fuente propia.**

# 5 CONCLUSIONES Y LÍNEAS FUTURAS

## 5.1 Conclusiones

Durante el presente trabajo, se ha llevado a cabo un proyecto para desarrollar un sistema que permita el uso del protocolo FT8 para el ámbito de las comunicaciones submarinas. El sistema consiste en un proceso donde la información emitida por voz por el usuario es reconocida, transcrita, codificada y enviada mediante este protocolo y, finalmente, decodificada en recepción. En ese sentido, el sistema propuesto cumple con los objetivos planteados inicialmente en el trabajo.

Asimismo, se han realizado una serie de pruebas con un objetivo doble. En primer lugar, y de forma indirecta, se ha tratado de averiguar si efectivamente el sistema funcionaba y realizaba las tareas propuestas. En segundo lugar, se ha buscado qué cantidad de ruido es capaz de soportar el sistema, realizando esta medición con un doble experimento.

Cabe destacar que, aunque se han cumplido los objetivos propuestos, no se han podido llevar a cabo los experimentos en un entorno submarino, debido a la falta de material proporcionado para tal fin. Este hecho abre la posibilidad de considerar esta línea de investigación en el futuro, con el objetivo de mejorar y optimizar aún más el sistema y, por tanto, las comunicaciones y su hipotética implementación.

## 5.2 Líneas futuras

El sistema propuesto hace uso de un diccionario formado por mil palabras, donde cada palabra es identificada directamente por la posición que ocupa en el índice. Esta metodología de trabajo se muestra ineficiente y primitiva, pues para cada categoría gramatical de una misma forma se emplean numerosos espacios del índice.

Es decir, si se desea transmitir niño, niña, niñez, niñas, se emplean cuatro espacios del índice tratando de transmitir una misma forma.

Existen múltiples formas para optimizar este proceso. A continuación, se expone de forma conceptual una posible implementación, la cual constará de un proceso de lematización, etiquetado y recomposición de la información.

### *5.2.1 Lematización*

La lematización es el proceso que permite transformar las palabras a su forma canónica o lema correspondiente. Por ejemplo, las palabras “*corrí*”, “*niñito*”, “*bonitas*”, se reducen a sus lemas “*correr*”, “*niño*” y “*bonito*”. Este proceso permite unificar las palabras que comparten mismo lema, reduciendo el tamaño del diccionario o manteniendo el tamaño, pero aumentando la cantidad de palabras a emitir.

Como herramienta propuesta para su hipotética implementación, se ha elegido el módulo de ConTexto [31], el cual consta de dos lematizadores distintos, cada uno apoyado en una librería de PLN (Procesamiento de lenguaje natural) distinta: spaCy y Stanza.

ConTexto proporciona una serie de ejemplos (Figura 65) con cada uno de los lematizadores, tratando de realizar el proceso con la oración contenida en la variable `texto`.

```
import time
from contexto.lematizacion import LematizadorSpacy, LematizadorStanza
from contexto.lematizacion import lematizar_texto

# textos de prueba
texto = 'esta es una prueba para ver si las funciones son correctas y funcionan bien. Perritos y gatos van a la casita'
```

**Figura 65. Lematizadores. Fuente propia.**

En este caso, se estudia el uso de spaCy únicamente, debido a su sencillez y eficacia. Como se puede apreciar en la Figura 66, el usuario es capaz de elegir tanto el lenguaje como el tamaño del modelo. ConTexto proporciona para la mayoría de los lenguajes tres posibles modelos atendiendo a su maño: pequeño, mediano y grande.

```
# Lematización con librería Spacy
texto_lematizado = lematizar_texto(texto, 'auto', dim_modelo='small')
print(texto_lematizado)
```

**Figura 66. Lematizador spaCy. Fuente propia.**

Tras completar el proceso, la información inicial transmitida por el usuario se almacena de la forma que ilustra la Figura 67:

```
este ser uno probar parir ver si los funcionar ser correcto y funcionar bien perro
y gato ir a lo casita
```

**Figura 67. Texto lematizado por spaCy [32]. Fuente propia.**

Esta herramienta permite reducir todas las formas de una palabra a una única categoría gramatical, reduciendo, de esta forma, el diccionario. Sin embargo, para que el receptor entienda bien el mensaje que se desea transmitir, se debe acompañar cada una de esas palabras con una etiqueta que nos indique la categoría gramatical o flexión a usar (dependiendo de si se trata de un verbo o un sustantivo/adjetivo/etc.). Por este motivo, damos paso al etiquetado.

### *5.2.2 Etiquetado*

El etiquetado gramatical es el proceso que permite asignar una etiqueta a cada una de las palabras del texto, indicando, mediante la cual, a que categoría pertenece. La optimización, por consiguiente, reside en esta etiqueta.

El etiquetador propuesto presenta de manera incremental cuatro posibles formas de analizar las palabras.

1. Etiquetador que por defecto establezca una palabra desconocida como un nombre común, masculino, singular y asigne la etiqueta correspondiente a todas las palabras. Se trata del etiquetador más simple.
2. Etiquetador basado en unigramas que aprende a partir de la lista de oraciones, usando el etiquetador anterior como respaldo. Los unigramas son la forma más sencilla de análisis de

palabras, donde se considera cada palabra del texto de forma individual, sin tener en cuenta su relación con palabras adyacentes.
Por ejemplo, si la oración solicitada por el usuario es "Un AF es un alumno de quinto curso", nuestro etiquetador analizaría: "Un", "AF", "es", "un", "alumno", "de", "quinto", "curso".

3. El etiquetador basado en bigramas considerará cada par de palabras adyacentes en el texto.
   En este caso, nuestro análisis sería: "Un AF", "AF es", "es un", "un alumno", "alumno de", "de quinto", "quinto curso".
4. El etiquetador basado en trigramas considerará cada trio de palabras adyacentes.
   De nuevo, nuestro análisis consistiría en la terna de palabras: "Un AF es", "AF es un", "es un alumno", "un alumno de", "alumno de quinto", "de quinto curso".

ConTexto proporciona una serie de etiquetas morfológicas a modo de ejemplo:

- da0ms0: determinante artículo masculino singular
- Ncms000: nombre común masculino singular
- Aq0cs0: adjetivo calificativo de género común singular

De esta formar, se consigue informar en recepción del tipo de categoría gramatical enviada.

### *5.2.3 Pattern*

Con el sistema propuesto, la información recibida por el usuario se presenta en forma de una palabra canónica junto a un etiquetador que indica su categoría gramatical. Ahora bien, el sistema receptor necesita una herramienta que sea capaz de adquirir dicha información y mostrar al usuario el mensaje tal y como lo transmitió el usuario emisor.

La herramienta propuesta para la realización de esta tarea es Pattern [36], la cual permite realizar la recomposición del mensaje. Esta librería nos permite conjugar las distintas palabras en función de los parámetros que le proporcionemos. A continuación, se proporciona un ejemplo del uso de la librería:

```
from pattern.es import conjugate
from pattern.es import INFINITIVE, PRESENT, PAST, SG, SUBJUNCTIVE, PERFECTIVE

print conjugate('ser', INFINITIVE)
print conjugate('ser', PRESENT, 1, SG, mood=SUBJUNCTIVE)
print conjugate('ser', PAST, 3, SG)
print conjugate('ser', PAST, 3, SG, aspect=PERFECTIVE)
```

**Figura 68. Librería pattern. Fuente propia.**

Podemos observar en la Figura 68 cómo trata de conjugar la forma canónica de un unico verbo en distintas categorías gramaticales, devolviendo al usuario receptor:

"ser, sea, era, fue"

Esta herramienta permite ocupar un único espacio en nuestro índice, en vez de cuatro (modelo actual)

# 6 BIBLIOGRAFÍA

[1] J. Rodríguez y V. Fernández, Cómo redactar el estado del arte de un trabajo, Editorial Genios, 2010.

[2] P. Martínez y A. García, Cómo escribir una buena memoria de TFG, Publicaciones del 2000, 2013.

[3] A. Pérez, Cómo escribir una bibliografía, Nuevas publicaciones.

[4] «Web de La Moncloa,» [En línea]. Available: http://www.lamoncloa.gob.es. [Último acceso: 13 enero 2015].

[5] «SIMON,» [En línea]. Available: https://bricoladores.simonelectric.com/blog/bid/401768/Datos-y-curiosidades-sobre-el-sonido. [Último acceso: 10 ENERO 2023].

[6] «Acustica Web,» [En línea]. Available: https://www.acusticaweb.com/curiosidades/blog/curiosidades/el-sonido-y-leonardo-da-vinci.html. [Último acceso: 10 Enero 2023].

[7] [En línea]. Available: https://www.criticadelibros.com/sin-clasificar/armonia-universal-marin-mersenne/. [Último acceso: 29 Enero 2023].

[8] «Hmong,» [En línea]. Available: https://hmong.es/wiki/Extremely_low_frequency. [Último acceso: 12 Enero 2023].

[9] «Frecuencia extremadamente baja,» [En línea]. Available: https://hmong.es/wiki/Extremely_low_frequency. [Último acceso: 26 Enero 2023].

[10] «New Courier,» [En línea]. Available: http://news-courier.com/informatics/news/data-technique-allows-sub-to-plane-communication-308083. [Último acceso: 06 Febrero 2023].

[11] «Comunidad submarinista latinoamericana,» [En línea]. Available: https://www.elsnorkel.com/2008/07/sistema-de-comunicaciones_1103.html. [Último acceso: 26 enero 2023].

[12] [En línea]. Available: https://nso.nato.int/nso/nsdd/main/list-promulg. [Último acceso: 06 Febrero 2023].

[13] [En línea]. Available: https://www.cmre.nato.int/rockstories-blog-display/398-janus-the-cmre-underwater-communication-protocol-becomes-a-nato-standard. [Último acceso: 6 Febrero 2023].

[14] [En línea]. Available: https://www.elettroamici.org/es/nuovo-standard-per-comunicazioni-subacquee/. [Último acceso: 08 Febrero 2023].

[15] «Unión de radioaficionados de Valladolid,» [En línea]. Available: http://www.ea1urv.es/web/ft8-ft4/. [Último acceso: 12 Febrero 2023].

[16] «EcuRed,» [En línea]. Available: https://www.ecured.cu/FT8. [Último acceso: 12 Febrero 2023].

[17] [En línea]. Available: https://www.python.org/. [Último acceso: 09 Febrero 2023].

[18] «Open BootCamp,» [En línea]. Available: https://open-bootcamp.com/aprender-programar/tipos-de-ide-pycharm. [Último acceso: 13 Febrero 2023].

[19] «PyPi,» [En línea]. Available: https://pypi.org/project/SpeechRecognition/. [Último acceso: 13 Febrero 2023].

[20] «GitHub,» [En línea]. Available: https://github.com/openai/whisper. [Último acceso: 13 Febrero 2023].

[21] «Universidad Complutense de Madrid,» [En línea]. Available: https://www.it.uc3m.es/pbasanta/asng/course_notes/ch07s07.html. [Último acceso: 10 Febrero 2023].

[22] Sistemas de Raciocomunicación. Tema 5: Modulaciones., Marín, Pontevedra, 2021/2022.

[23] «Real Academia Española (RAE),» [En línea]. Available: https://dle.rae.es/corpus?m=form. [Último acceso: 13 Febrero 2023].

[24] «Sondas Batimétricas; Cartografía submarina,» [En línea]. Available: https://www.lpi.tel.uva.es/~nacho/docencia/ing_ond_1/trabajos_06_07/io7/public_html/sonar2.html. [Último acceso: 10 Enero 2023].

[25] «CMRE NATO,» [En línea]. Available: https://www.cmre.nato.int/rockstories-blog-display/398-janus-the-cmre-underwater-communication-protocol-becomes-a-nato-standard. [Último acceso: 10 Enero 2023].

[26] «SIMON,» [En línea]. Available: https://bricoladores.simonelectric.com/blog/bid/401768/Datos-y-curiosidades-sobre-el-sonido. [Último acceso: 10 ENERO 2023].

[27] Sistemas de Radiocomunicaciones. Tema 5: Modulaciones..

# ANEXO I: IMPLICACIONES SOCIALES Y AMBIENTALES

El ruido causado por las actividades humanas se considera un tipo de contaminación, estando recogido por la legislación española [35]. Actualmente, las leyes se aplican en medio terrestre, pero se requiere de un desarrollo formal para controlar la contaminación acústica submarina. Resumimos la legislación nacional aplicable a las siguientes leyes:

- 37/2003 del Ruido
- 41/2010 de Protección marino
- 42/2007 del Patrimonio Natural y la Biodiversidad
- 9/2006 de Evaluación de Impacto Ambiental.

En el marco del derecho internacional, mediante el uso de instrumentos normativos y resoluciones de diversas situaciones como el programa de las Naciones Unidas para el Medio Ambiente, la Organización Marítima Internacional, la convención sobre derechos del Mar de las Naciones Unidas etc., se gestiona y conserva el entorno marino.

Estos textos y resoluciones reflejan una preocupación sobre el impacto no regulado de la contaminación acústica y se invoca el principio de precaución y la puesta en marcha de medidas de mitigación de impacto. España forma parte de la mayoría de los convenios internacionales y se le insta a tomar medidas en consecuencia.

El principal motivo por el cual las regulaciones existentes sobre la contaminación acústica en los entornos marinos no se aplican con frecuencia es por que históricamente se desconocía la importancia del sonido para la fauna marina y su papel en el ecosistema acuático.

Multitud de especies dependen del sonido para sus funciones vitales. Si bien es cierto que aún no se conoce el efecto del ruido en la vida marina al completo, se ha observado que puede tener impactos significantes, incluso la muerte, en mamíferos y otras especies.

Las medidas de mitigación, incluyendo mejoras tecnológicas o reducción de emisiones, tendrán un efecto positivo inmediato en la reducción de la contaminación acústica. En muchos países se aplican medidas en forma de guías de conducta, estando reguladas legislativamente en otros. Es práctica común que las actividades de ruido de alta intensidad deban someterse a procedimientos de evaluación de impacto antes de ser usadas legalmente. Otras, sin embargo, son medidas sencillas y de clara aplicación lógica.

En la Armada, es de obligado cumplimiento respetar el entorno marino, estando recogidas las actuaciones relativas al ruido submarino en la publicación AXP-01(E). Durante la realización de ejercicios de adiestramiento CASEX, de guerra ASW (*Anti-Submarine Warfare*), se debe tomar medidas de protección sonar. Cabe destacar que estas se tomarán respetando siempre la seguridad de las unidades y la situación táctica.

La política de la OTAN requiere la implementación de medidas de protección apropiadas para garantizar la protección de los mamíferos marinos en el marco de las actividades marítimas de la Alianza. Durante la planificación de las operaciones, se deberá realizar una evaluación de la mitigación de riesgos. Dicha evaluación consiste en la revisión de las ubicaciones primarias de los mamíferos marinos, áreas protegidas del medio ambiente y demás consideraciones que puedan ser necesarias según los factores ambientales de alta probabilidad conocidos.

En este sentido, y de conformidad con la normativa vigente, las unidades de la Armada deben evitar entrar con sonar activo en áreas donde encontrarán condiciones que en su conjunto puedan contribuir a un factor perjudicial para los mamíferos marinos. En dichas condiciones vienen recogidas algunas como batimetrías determinadas (ruptura de la plataforma continental, montañas submarinas, etc.), uso de

múltiples sonares durante periodos de tiempo o canales restringidos o salida limitada para mamíferos marinos.

Si las unidades navales deben llevar a cabo operaciones bajo las recogidas en el documento, se buscará la aprobación de la cadena de mando correspondiente, actuando, como mínimo, bajo los procedimientos nacionales.

En resumen, en presencia de mamíferos marinos:

- Reducir al máximo potencia sonar, y si es posible, apagarlo.
- Apagar sonares calables.
- No usar dispositivos explosivos.

# ANEXO II: REFLEXIONES ÉTICAS Y SOCIALES

Desde un punto de vista ético, el principal punto de reflexión se centra en el impacto de las comunicaciones en el entorno submarino. Las ondas de sonido generadas por las comunicaciones pueden llegar a ser perjudiciales para los animales que dependan del sonido para la realización de sus funciones vitales, como su orientación, comunicación o el equilibrio ecológico general del entorno submarino. Es importante, por tanto, que se lleven a cabo las medidas de prevención y control que minimicen dichos efectos. Se presenta, en ese sentido, como un deber ético y moral de cada individuo el respetar la normativa recogida por cada país, así como el uso de buenas prácticas para respetar al ecosistema.

En el plano social, atendiendo al entorno militar, el uso del protocolo FT8 puede mejorar la eficacia de las operaciones, permitiendo una comunicación más clara y rápida entre unidades, lo que a su vez mejora el éxito en las misiones. Además, el protocolo FT8 ofrece cierta seguridad en la transmisión de datos, lo que puede ser especialmente importante en el entorno militar.

En el ámbito civil, la incorporación de protocolo FT8 en las comunicaciones submarinas puede mejorar la conectividad y la seguridad en el establecimiento de las comunicaciones entre las personas. Esto puede ser útil, por ejemplo, en la exploración submarina, en la comunicación con submarinos civiles o con vehículos submarinos no tripulados. En general, el uso del protocolo FT8 puede mejorar significativamente las comunicaciones en el entorno submarino, tanto en el ámbito civil, como en el militar.