COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

MÁSTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIONES

TRABAJO FIN DE MÁSTER

IF digital con microprocesador/DSP para satélites de la serie AMSAT

Autor: Antonio Martos Herrero

Director: Pedro Olmos González

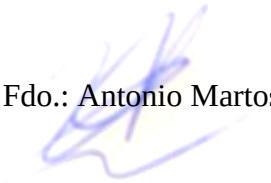
Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

IF digital con microprocesador/DSP para satélites de la serie AMSAT

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2020/21 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.



Fdo.: Antonio Martos Herrero Fecha: Madrid a 15 de junio de 2021

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Pedro Olmos González Fecha: ……/ ……/ ……

COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

MASTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIONES

TRABAJO FIN DE MÁSTER

IF digital con microprocesador/DSP para satélites de la serie AMSAT

Autor: Antonio Martos Herrero

Director: Pedro Olmos González

Madrid

Agradecimientos

Agradezco la labor y la dedicación de mi director y tutor del proyecto, D. Pedro Olmos González, quien me acompañado en todo momento y me ha guiado en los momentos de duda.

Agradezco también a la organización AMSAT, concretamente a D. Eduardo Alonso, por seguir tan de cerca mi trabajo y por su interés en que se consiguiera un resultado positivo.

IF DIGITAL CON MICROPROCESADOR/DSP PARA SATÉLITES DE LA SERIE AMSAT

Autor: Martos Herero, Antonio.
Director: Olmos González, Pedro.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

Palabras clave: Digital, DSP, analógico, AMSAT, DSP, señal, audio, FM, ADC, DAC, CORDIC, arcotangente, PLL, filtro, diezmador, oscilador, paso banda.

1. Introducción

Hoy en día, es cada vez más frecuente la sustitución de las tecnologías analógicas por tecnologías digitales. Esto se debe a que la "digitalización" ofrece numerosas ventajas frente a la implementación analógica como una mayor disponibilidad y accesibilidad a los procesos o que, en muchos casos, es más económico y producen resultados más controlables. [1]

Las señales de audio han de ser digitalizadas, guardadas y procesadas de una manera rápida y efectiva que permita reproducirlas en el "mundo real". Por ello, se han diseñado un tipo de microcontroladores diseñados específicamente para este tipo de sistemas llamados DSP.

DSP se corresponde con las siglas procesador digital de señal. Este tipo de dispositivos capturan las señales del mundo real (audio, vídeo, presión, etc.), las digitalizan y las transforman según los procesos que requiera la aplicación en concreto diseñada por el usuario. [2]

2. Definición del proyecto

El trabajo que se va a desarrollar consiste en el diseño de un receptor de audio digital que logre demodular una señal recibida. Esta señal estará modulada en frecuencia o FM con una frecuencia de portadora de 45 MHz o 455 kHz, a **especificar** en el diseño. Además, las bandas en FM ocuparán como máximo 15 kHz. Además, la señal moduladora no tendrá un ancho de banda superior a los 3 kHz.

Para la recepción de la señal se hará uso de un conversor Analógico-Digital o ADC aplicando la técnica del submuestreo con la que se muestreará la señal a una frecuencia muy por debajo de la frecuencia de la portadora escogida ya que el DSP que se usa tiene una frecuencia máxima de reloj de 80 MHz y no se podría realizar un sobremuestreo.

Posteriormente, se realizará el procesado de la señal. Se usarán osciladores para pasar la señal a banda base, filtros para quedarnos con las frecuencias deseadas, diezmadores para reducir el ruido de cuantificación y el propio bloque de demodulación de la señal. Para este último bloque se ha decidido usar el algoritmo de CORDIC que es parecido al de la arcotangente (más habitual) pero reduce notablemente el tiempo de procesamiento al realizar operaciones sencillas como sumas y productos (las divisiones consumen mucho tiempo), además de que no hay casi ejemplos de su uso.

El último bloque será un conversor Digital-Analógico o DAC para tener la señal a la salida y poder escuchar el audio.

Se usará Matlab para realizar pruebas y para ver visualmente algunos resultados. También, se usará un osciloscopio para comprobar los tiempos de los distintos sistemas, viendo la sincronía del sistema completo.

3. Descripción del modelo/sistema/herramienta

A continuación, se muestra el sistema a alto nivel que se va a implementar:

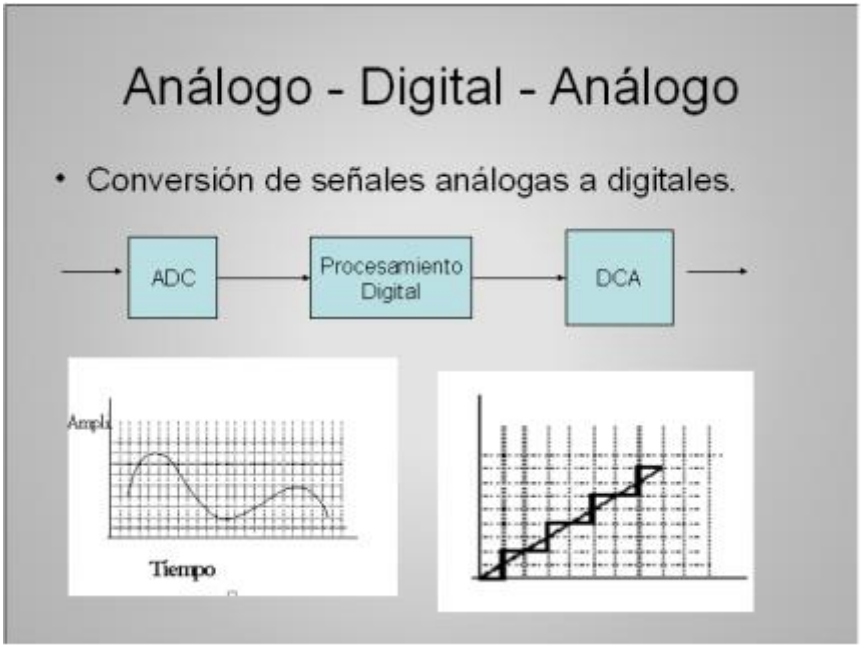
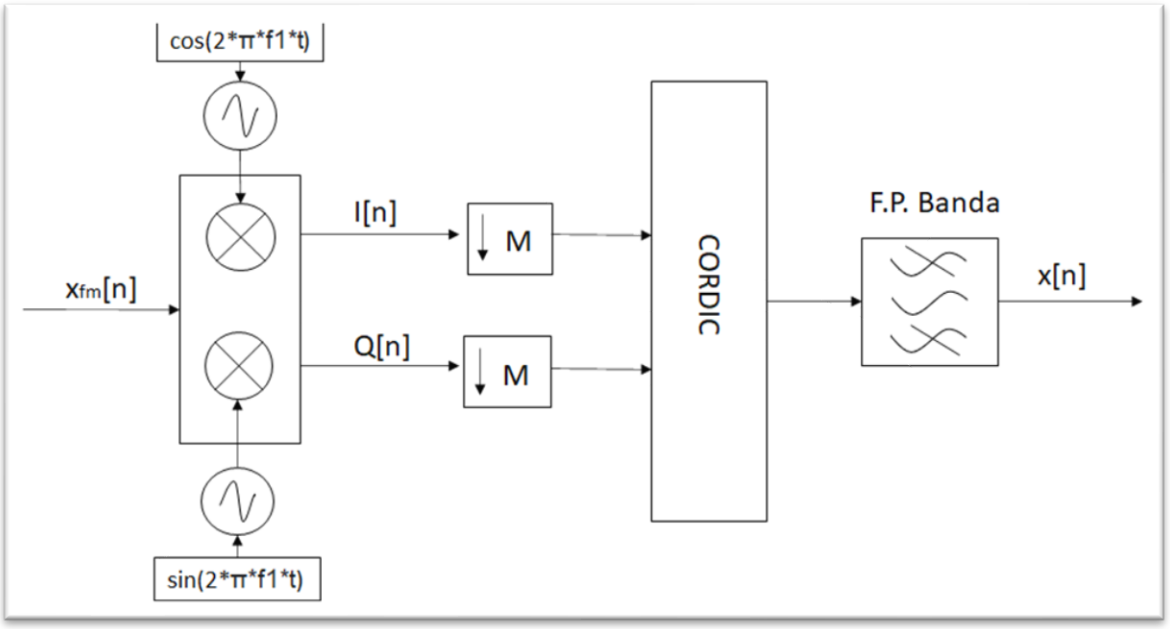


Ilustración 1 – Esquema general a implementar

El ADC convierte la señal analógica a digital para su posterior procesamiento y el DCA o DAC realiza el proceso contrario, convierte la señal de digital a analógico.

El diagrama de bloques de la parte de procesamiento digital es el siguiente:



Ilutración 2 – Esquema procesamiento digital

Se recibe la señal modulada, se multiplica por los osciladores para pasar a banda base con componentes I y Q. Posteriormente se realiza un diezmado para quitar ruido de cuantificación y se procesa con el algoritmo de CORDIC. Finalmente, se filtra la señal para obtener la señal original.

4. Resultados

En este apartado se van a mostrar resultados de los tres bloques principales del proyecto: ADC, procesamiento y DAC.

- **ADC**: Este apartado era quizás uno de los más importantes ya que muchos de los parámetros del sistema dependen de él. Ha consistido en elegir bien los parámetros para conseguir una buena frecuencia de muestreo.

Después de numerosas pruebas, se ha elegido una frecuencia de muestreo de 61256 Hz para una señal modulada a 455 kHz. Este bloque recoge 2048 muestras de las cuales únicamente se procesan 1024 que corresponden a un tiempo de 16,7 ms como se muestra a continuación.

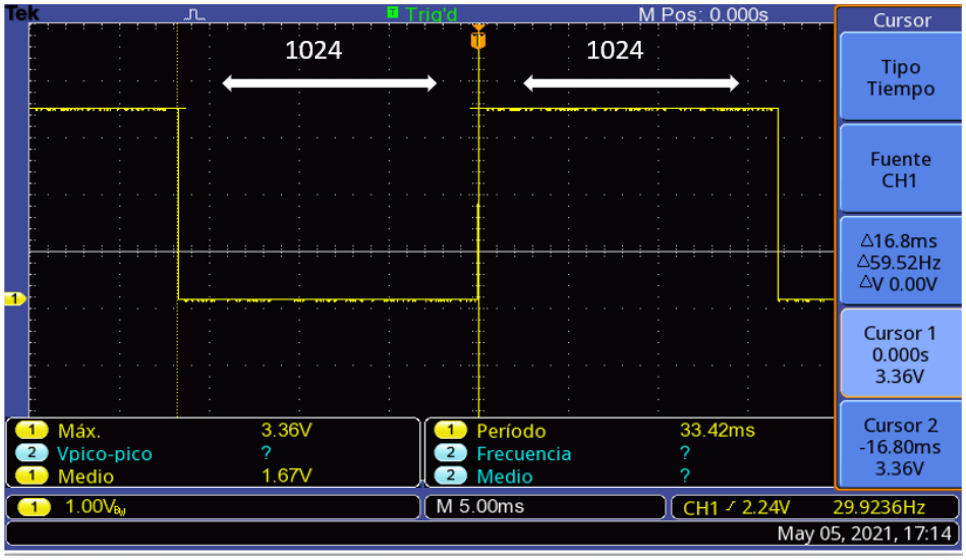


Ilustración 3 – Tiempos ADC para 2048 muestras

Con algunos cálculos, obtenemos que la frecuencia de la portadora en la primera banda de Nyquist quedaría entorno a los 26200 Hz.

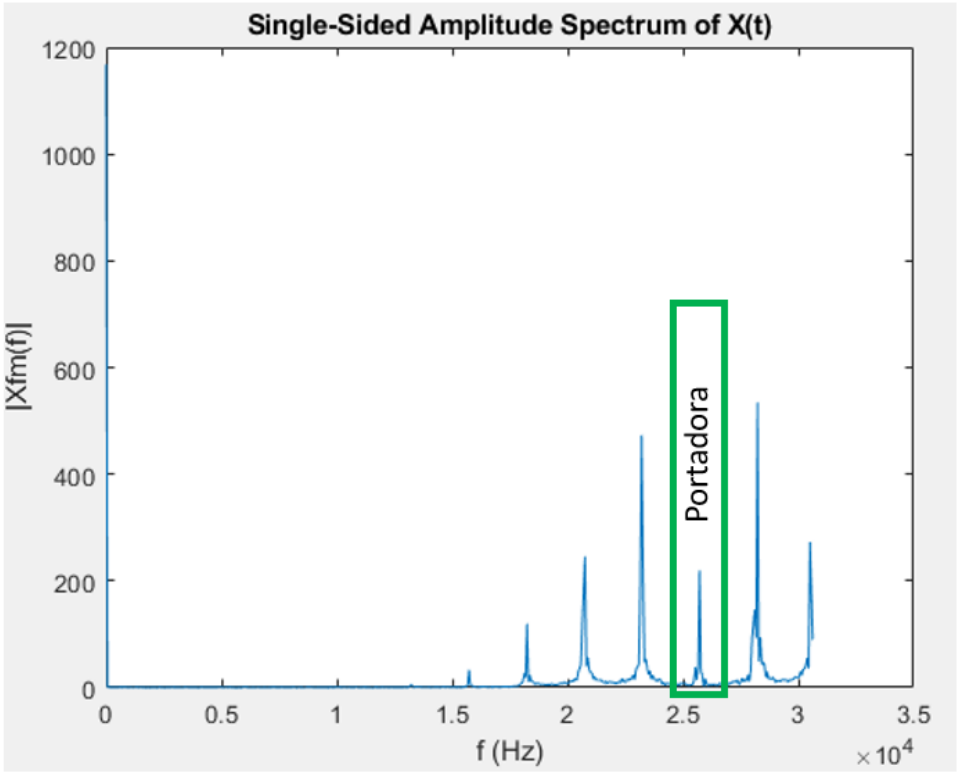


Ilustración 4 - Señal ADC en frecuencia

- **Procesamiento**: Con la señal obtenida se realiza el procesamiento mostrado en la ilustración 2, obteniendo la siguiente señal:

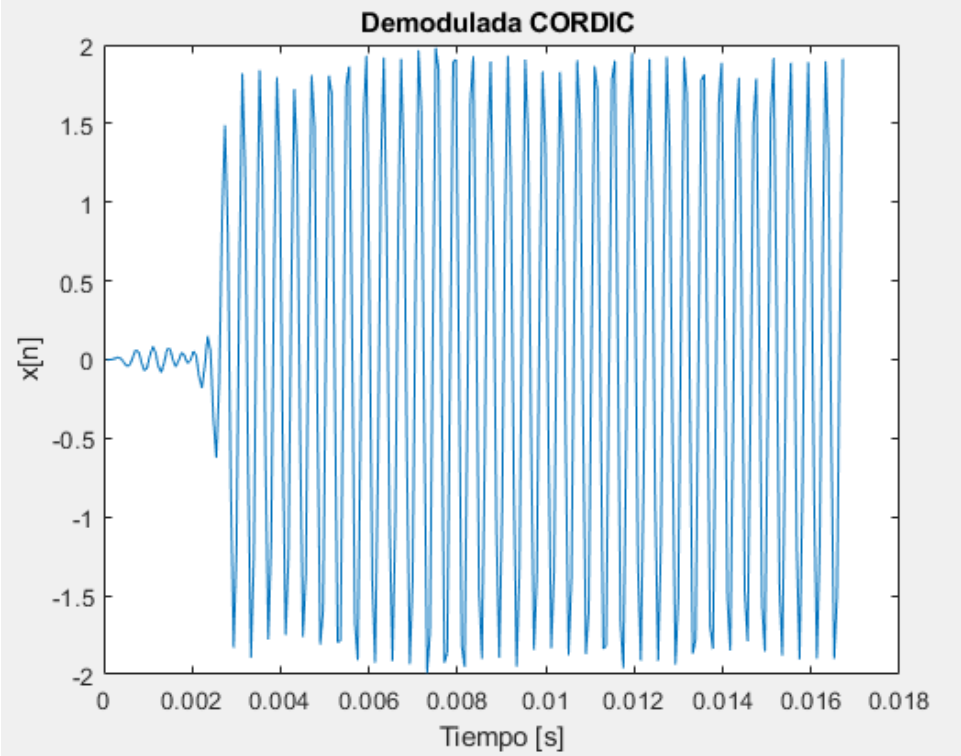


Ilustración 5 - Señal demodulada en el dominio del tiempo

La señal se corresponde con un tono que ocupa 16 ms (medio buffer del ADC) y tiene varios ciclos de la señal original. Al principio, la señal vale cero debido al retardo en la respuesta al impulso del filtro.

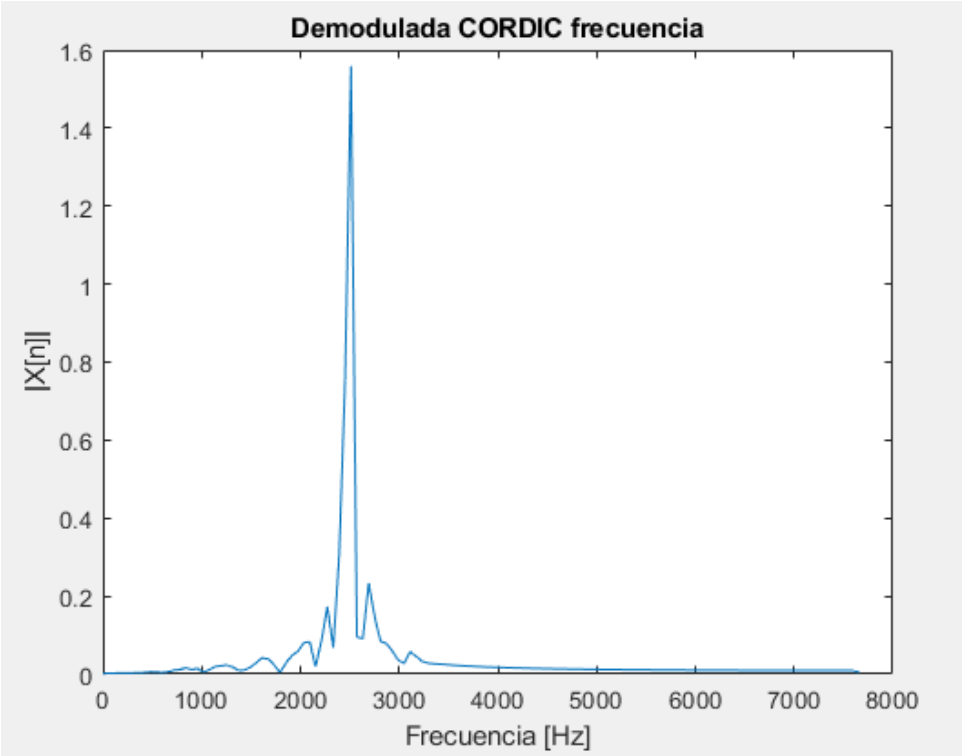


Ilustración 6 - Señal demodulada en el dominio de la frecuencia

Aunque se aprecian algunos picos de ruido en la transformada de Fourier de la señal, se ve claramente el tono principal entorno a los 2500 Hz, que es la señal moduladora original usada para esta prueba.

Obteniendo un tiempo de procesado en el DSP de 6.1 ms que se corresponde con la bajada del pulso de la ilustración 7:

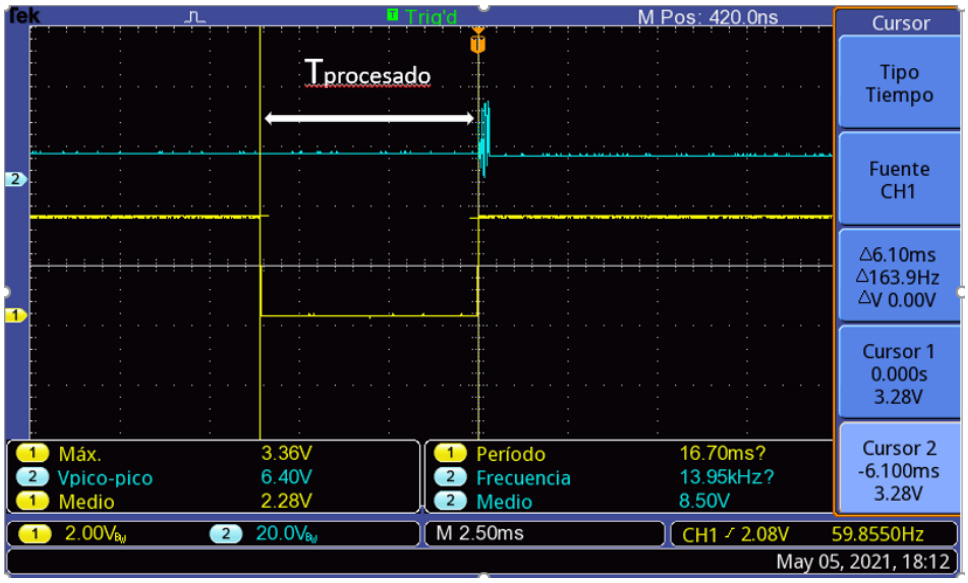


Ilustración 7 – Tiempo procesado de señal

- **DAC**: Este bloque únicamente reproduce la señal procesada de manera analógica. Queda fuera de este proyecto, pero se tendrá que realizar un filtrado analógico, además de eliminar el nivel medio de la señal.

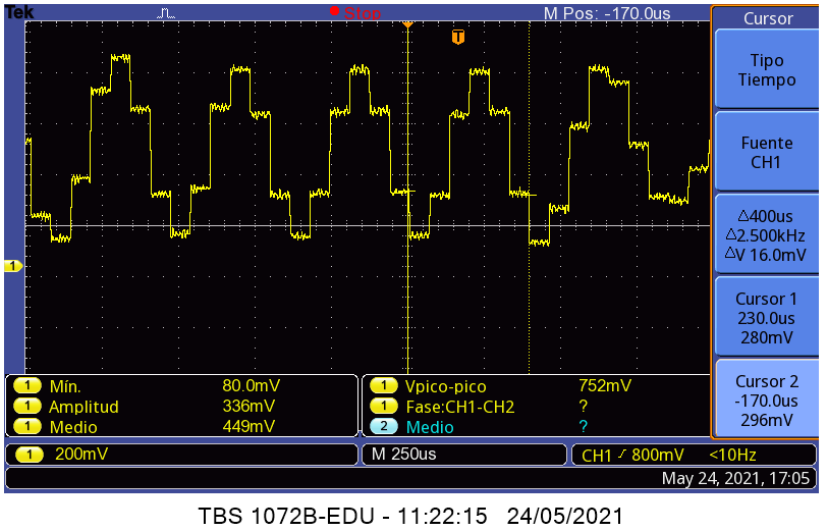


Ilustración 8 – Señal a la salida del DAC

Obtenemos una señal continua de 2500 Hz, aunque no se corresponde exactamente con la señal original. Se necesitarían más muestras para conseguir eliminar los “escalones” a la salida. Además, se observa cómo, al final de la imagen, hay una pequeño alliasing. Esto último es producido porque se están muestreando ciclos incompletos de la señal lo que hace que solape con la siguiente muestra del buffer.

5. Conclusiones

Se ha conseguido diseñar un sistema de recepción de audio completo minimizando los tiempos de ejecución de los procesos. Esto permite el funcionamiento de los tres principales bloques (ADC, procesamiento, DAC) a la vez sin que existan interrupciones entre ellos.

Según los resultados obtenidos, el sistema demodula bien las señales que se aproximan a la frecuencia de 3000 Hz, pero no lo hace tan bien a medida que se va disminuyendo la frecuencia de la moduladora. Esto puede ser debido a que se necesita un diseño de filtros más complejos o a que el algoritmo de CORDIC no funciona bien para todos los casos. También, puede que la selección de la frecuencia de los osciladores infiera en estos errores produciendo ciertas desviaciones en frecuencia.

Por ello, se propone para trabajos futuros el estudio de filtros más complejos además de la prueba de otro tipo de sistemas de demodulación como pueda ser el uso de un PLL o el algoritmo clásico de demodulación en banda base con retardo. Para el problema de los osciladores convendría eliminar su uso y recibir directamente las señales I e Q en el ADC. Además, se recomienda el diseño de dos filtros analógicos a la salida: uno paso alto para eliminar el nivel medio de la señal y otro paso bajo para quedarse únicamente con la banda de 0-3000 Hz.

6. Referencias

- [1] Digital, S. (2019, 7 septiembre). Las ventajas del proceso digital. SAJ Digital - Confira tudo o que acontece na Justiça brasileira. https://www.sajdigital.com/tribunal-de-justicia-es/ventajas-proceso-digital/?lang=es
- [2] A Beginner's Guide to Digital Signal Processing (DSP) | Design Center | Analog Devices. (s.f.).MyAnalog.

IF DIGITAL WITH MICROPROCESSOR/DSP FOR AMSAT SERIES SATELLITES

Autor: Martos Herero, Antonio.
Director: Olmos González, Pedro.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

ABSTRACT

Keywords: Digital, DSP, analogical, AMSAT, DSP, signal, audio, FM, ADC, DAC, CORDIC, arctangent, PLL, filter, decimator, oscillator, step, band

1. Introduction

Nowadays, analog technologies are increasingly being replaced by digital technologies. This is because "digitization" offers numerous advantages over analog implementation such as greater availability and accessibility of processes or that, in many cases, it is more economical and produces more controllable results. [1]

Audio signals have to be digitized, stored and processed in a fast and effective way that allows them to be reproduced in the "real world". Therefore, a type of microcontrollers specifically designed for this type of systems called DSP have been designed.

DSP stands for digital signal processor. This type of device captures real world signals (audio, video, pressure, etc.), digitizes them and transforms them according to the processes required by the specific application designed by the user. [2]

2. Project Definition

The work to be developed consists of the design of a digital audio receiver that can demodulate a received signal. This signal will be frequency modulated or FM with a carrier frequency of 45 MHz or 455 kHz, to be specified in the design. In addition, the FM bands will occupy a maximum of 15 kHz. In the same way, the modulated tones will have assigned frequencies no higher than 3 kHz.

To receive the signal, an analog-to-digital converter or ADC will be used, applying the undersampling technique to sample the signal at a frequency well below the frequency of the chosen carrier, since the DSP used has a maximum clock frequency of 80 MHz and oversampling would not be possible.

Subsequently, the signal processing will be performed. Oscillators will be used to pass the signal to baseband, filters to keep the desired frequencies, decimators to reduce the quantization noise and the signal demodulation block itself. For this last block it has been decided to use the CORDIC algorithm, which is similar to the arctangent algorithm (more common) but reduces significantly the processing time when

performing simple operations such as additions and products (divisions are very time consuming), and there are almost no examples of its use.

The last block will be a Digital-to-Analog converter or DAC to have the signal at the output and be able to listen to the audio.

Matlab will be used for testing and to visually see some results. Also, an oscilloscope will be used to check the timing of the different systems, seeing the synchrony of the whole system.

3. Description of the model/system/tool

The high-level system to be implemented is shown below:

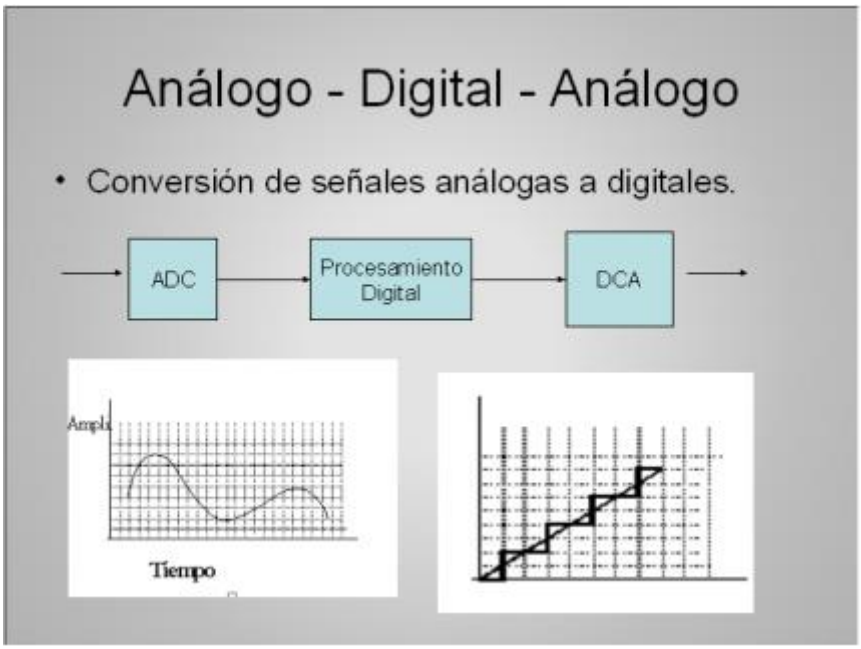


Illustration 1 – General scheme

The ADC converts the analog signal to digital for further processing and the DCA or DAC performs the reverse process, converting the signal from digital to analogic.

The block diagram of the digital processing part is as follows:

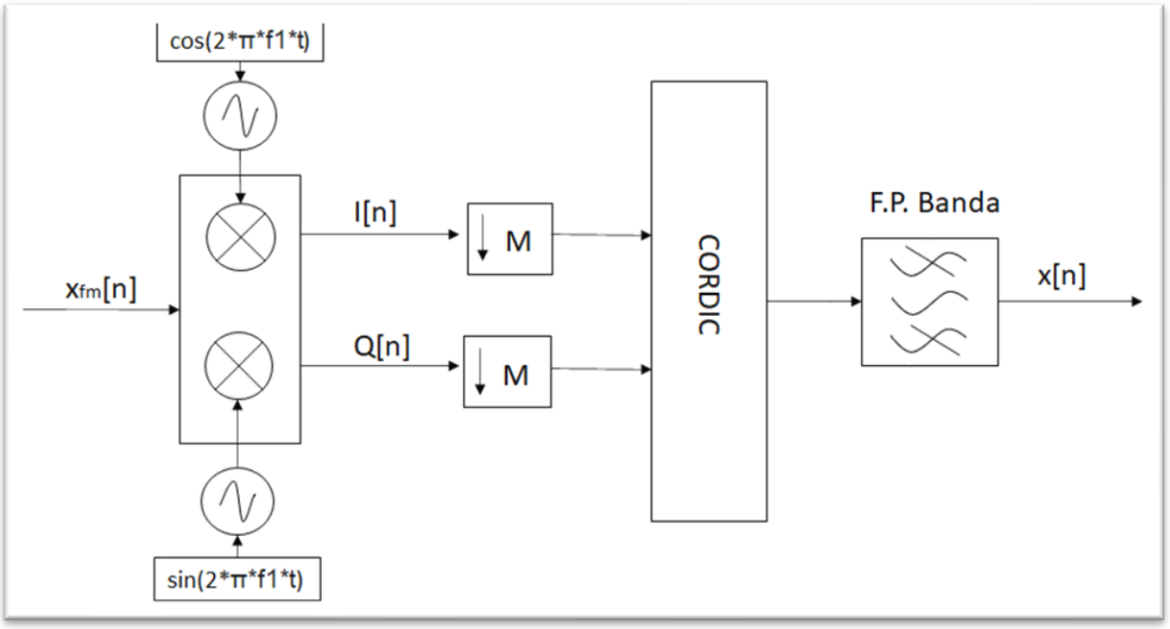


Illustration 2 – Digital processing schema

The modulated signal is received, multiplied by the oscillators to pass to baseband with I and Q components. Subsequently, it is decimated to remove quantization noise and processed with the CORDIC algorithm. Finally, the signal is filtered to obtain the original signal.

4. Results

This section will show results of the three main blocks of the project: ADC, processing and DAC.

ADC: This section was perhaps one of the most important since many of the system parameters depend on it. It consisted in choosing the right parameters to get a good sampling frequency.

After numerous tests, a sampling frequency of 61256 Hz was chosen for a signal modulated at 455 kHz. This block contains 2048 samples of which only 1024 are processed which corresponds to a time of 16.7 ms as shown below.

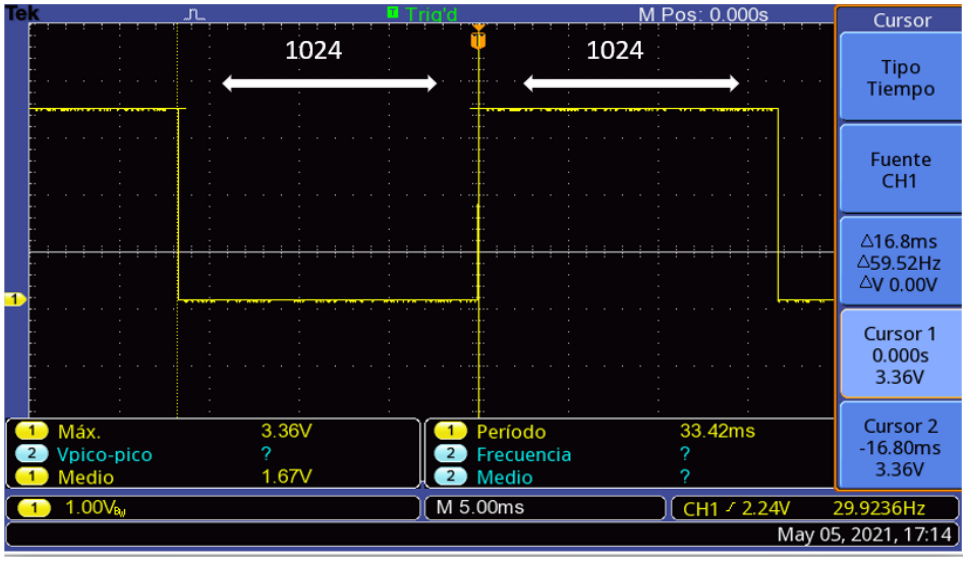


Illustration 3 - ADC times for 2048 samples

With some calculations, we obtain that the carrier frequency in the first Nyquist band would be around 26200 Hz.

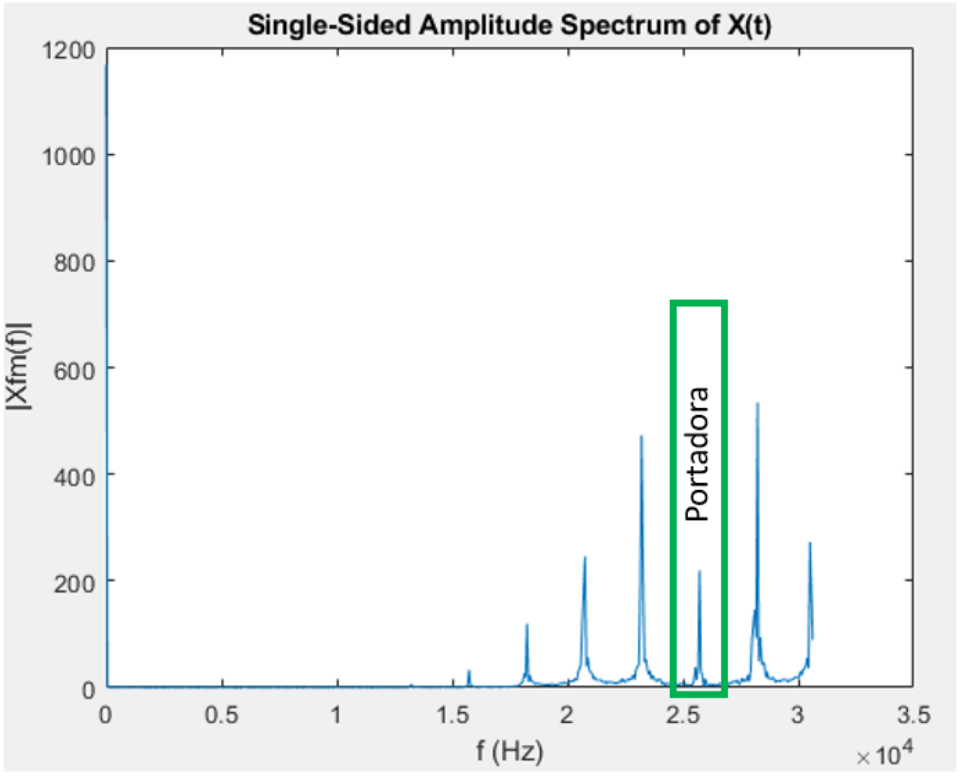


Illustration 4 - Frequency ADC signal

Processing: The signal obtained is processed as shown in Figure 2, obtaining the following signal:

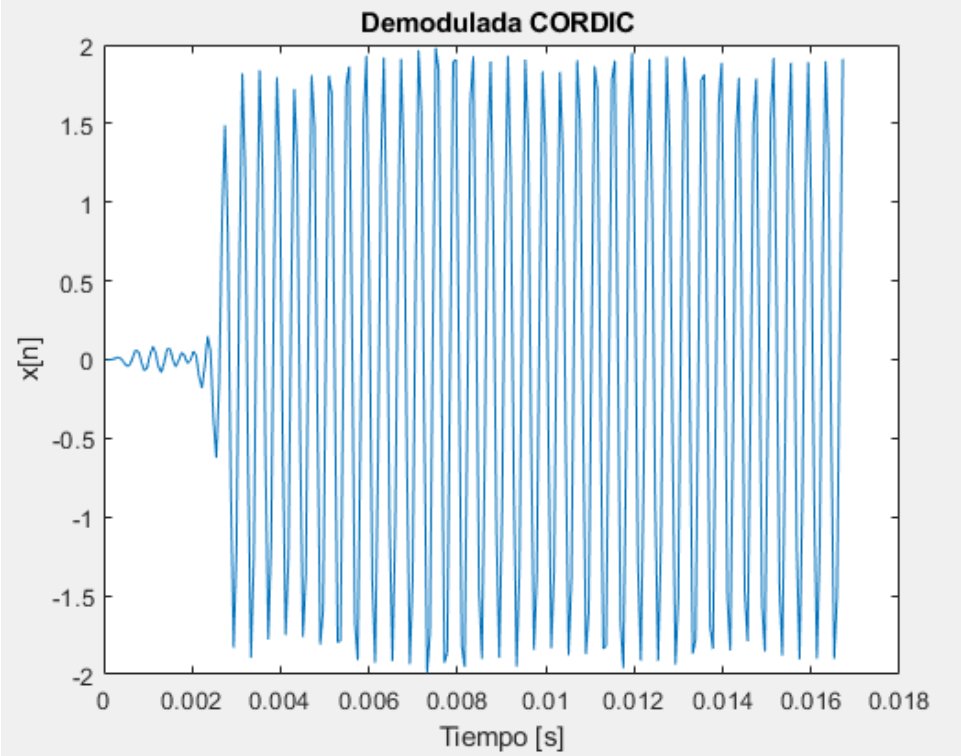


Ilustración 5 - Señal demodulada en el dominio del tiempo

The signal corresponds to a tone that occupies 16 ms (half buffer of the ADC) and has several cycles of the original signal. At the beginning, the signal is zero due to the delay in the impulse response of the filter.

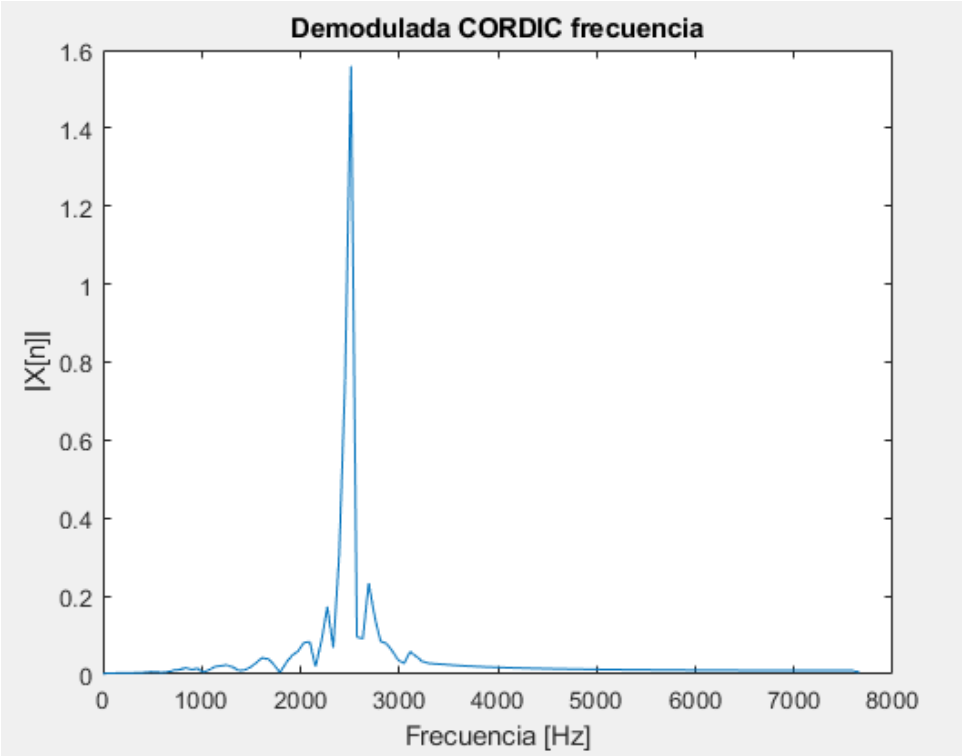


Illustration 6 - Demodulated signal in frequency domain

Although some noise peaks can be seen in the Fourier transform of the signal, the main tone is clearly visible around 2500 Hz, which is the original modulating signal used for this test.

Obtaining a processing time in the DSP of 6.1 ms, which corresponds to the pulse decay in Figure 7:

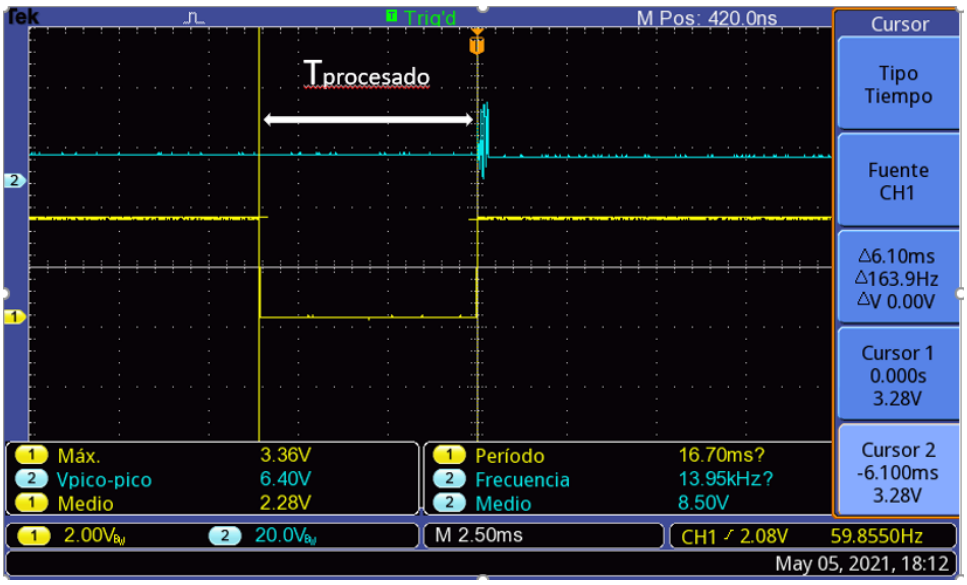


Illustration 7 – Signal processing time

- **DAC**: This block only reproduces the analog processed signal. It is outside the scope of this project, but an analog filtering will have to be performed, in addition to eliminating the average level of the signal.

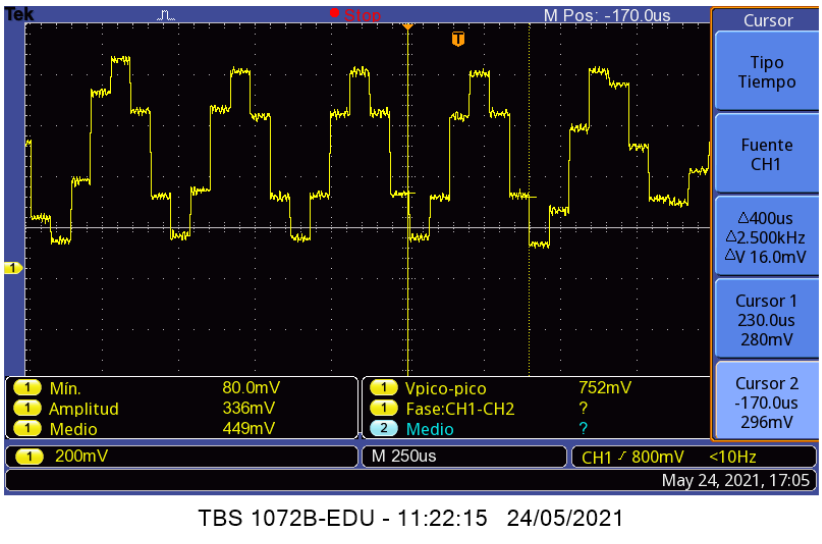


Illustration 8 - Signal at the output of the DAC

A continuous 2500 Hz signal is obtained, although it does not correspond exactly to the original signal. More samples would be needed to eliminate the "steps" at the output. In

addition, we can see how, at the end of the image, there is a small alliasing. The latter is produced because incomplete cycles of the signal are being sampled, which causes it to overlap with the next sample in the buffer.

5. Conclusions

It has been possible to design a complete audio reception system while minimizing process execution times. This allows the operation of the three main blocks (ADC, processing, DAC) at the same time without interruptions between them.

According to the results obtained, the system demodulates signals approaching the 3000 Hz frequency well, but does not do so as well as the modulator frequency decreases. This may be because a more complex filter design is needed or because the CORDIC algorithm does not work well for all cases. Also, it may be that the selection of the oscillator frequency influences these errors by producing certain frequency deviations.

Therefore, it is proposed for future work to study more complex filters in addition to testing other types of demodulation systems such as the use of a PLL or the classical algorithm of baseband demodulation with delay. For the oscillator problem it would be convenient to eliminate the use of oscillators and directly receive the I and Q signals in the ADC. In addition, it is recommended to design two analog filters at the output: one high pass to eliminate the middle level of the signal and another low pass to keep only the 0-3000 Hz band.

6. References

[1] Digital, S. (2019, 7 septiembre). Las ventajas del proceso digital. SAJ Digital - Confira tudo o que acontece na Justiça brasileira. https://www.sajdigital.com/tribunal-de-justicia-es/ventajas-proceso-digital/?lang=es

[2]A Beginner's Guide to Digital Signal Processing (DSP) | Design Center | Analog Devices. (s.f.).MyAnalog.https://www.analog.com/en/design-center/landing-pages/001/beginners-guide-to-dsp.html#:%7E:text=Although%20real%2Dworld%20signals%20can,a%20wide%20variety%20of%20applications.

Índice de la memoria

Índice de Figuras

Índice de Tablas

Capítulo 1. INTRODUCCIÓN

1.1 MOTIVACIÓN DEL PROYECTO

La organización AMSAT lleva poniendo satélites en órbita desde el año 1983. Todos estos satélites incluyen métodos de procesamiento de señales analógicas. Por ello, sería interesante desarrollar un sistema que incorpore procesamiento digital para comprobar si el funcionamiento es mejor que el actual. Además, al usar software y no hardware, los sistemas tendrían más tolerancia a fallos ya que el entorno no facilita la reparación. Otro punto importante, es que muchos de los DSP actuales son compatibles con la radiación y no habría que diseñar sistemas específicos con materiales resistentes a este tipo de condiciones.

Entrando en aspectos de software, existen numerosos algoritmos y metodologías para la demodulación de señales de FM. Uno de los métodos más habituales es el uso del algoritmo de la arcotangente. Como se explicará más adelante, este algoritmo realiza operaciones complejas que aumentan el tiempo de computación por lo que es conveniente la prueba de aproximaciones al mismo y ver si producen resultados similares reduciendo los tiempos de ejecución. Además, no es eficiente trabajar con formatos “double” o “float” cuando se tienen números decimales muy pequeños que pueden arrastrar errores significativos por lo que aparecen conceptos como “coma flotante” o representación en “Q-binario”.

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

2.1 ARM CORTEX -M4

Se trata de una gama de procesadores diseñados y dirigidos a mercados de control que demandan altas eficiencias en productos y facilidad de uso, además de capacidades de procesamiento de señales. La combinación de la alta eficiencia en el procesamiento de señales con su baja potencia, bajos costes y facilidad de uso hacen a esta gama un producto ideal para sistemas de audio, automatización, control de potencia, etc.

2.2 STM32XXXX

STM32 es una familia de microcontroladores de 32 bits basados en que procesador ARM-Cortex -M está diseñado para ofrecer nuevos grados de libertad para los usuarios de MCU. Ofrece productos que combinan un alto rendimiento, capacidades en tiempo real, procesado digital de señales con la librería CMIS DSP, operaciones con baja potencia/voltaje y fácil integración para entornos de desarrollo.

La organización de AMSAT está usando la placa con el procesador STM32F446. Desgraciadamente, se desconocía este dato y se compró la STM32L476 que ofrece capacidades similares en general a diferencia que tiene un reloj con menor frecuencia. Esto no es de importancia ya que la programación es la misma y los drivers parecidos por lo que se podrá migrar el entorno.

2.3 STM32CUBEIDE

Se trata de un entorno de similar a Eclipse y sirve para la programación de software en lenguajes básicos como son C o C++. También, ofrece una interfaz muy cómoda en la que el usuario puede configurar los distintos drivers del sistema. Hay que mencionar que se ha elegido la programación en C por su fácil compresión.

2.4 MATLAB

MATLAB combina un entorno de escritorio perfeccionado para el análisis iterativo y los procesos de diseño con un lenguaje de programación que expresa las matemáticas de matrices y arrays directamente.

Se realizarán pruebas sobre Matlab además de alguna verificación de resultados.

2.5 OTROS

A parte, se usarán herramientas de apoyo como Excel o Word para documentar el proyecto. También, se dispondrá de los generadores de señal para las pruebas en tiempo real además de materiales como cables, osciloscopio, etc.

Capítulo 3. ESTADO DE LA CUESTIÓN

En este apartado se abordarán los principales métodos existentes para la demodulación FM además de una pequeña introducción de por qué el uso de los DSP hoy en día es más habitual en sistemas de radioaficionados.

3.1 DEMODULACIÓN FM

La modulación FM o modulación en frecuencia es una de las múltiples técnicas que se utilizan para el tratamiento de señales y consiste en transmitir información a través de una portadora variando su frecuencia de manera que se transmiten las señales a frecuencias mucho más altas que equivale a facilitar un ancho de banda mayor y con ello mayor capacidad de transmisión de señales. Este tipo de técnicas son muy empleadas en aplicaciones como radiodifusión, televisión o en sistemas de vídeo analógicos.

Ahora bien, cuando se modula siempre es necesario realizar el proceso inverso para recuperar la señal original o señal muy parecida a esta para recibirla en la frecuencia deseada. En audio, es necesario realizar este proceso de recuperación ya que el cerebro humano no es capaz de relacionar todo el espectro de frecuencias existentes con audio y por ello solo escucha la banda de frecuencias de 20 a 20000 Hz. Existen múltiples formas de demodular este tipo de señales y algunos de ellos se van a mencionar a continuación.

- Limitador con transistores: usado para demodular señales FM en sistemas analógicos. El transistor recorta la señal cuando supera un margen dinámico. Para eliminar los armónicos que no interesen se emplea un conjunto LC que filtra la banda FM.
- La señal se alimenta con un PLL (phase locked loop) y la señal de error resultante de la combinación de la original y el PLL se utiliza como señal demodulada.

- Discriminador Foster-Seeley: Formado por un filtro electrónico que disminuye la amplitud de algunas frecuencias en relación con otras. Seguido del filtro se encuentra un sistema de demodulación AM. Hay una variante de este método que se llama "detector de envolvente" que consiste en quedarse con la envolvente de la señal, recuperando así la señal original.
- Los métodos anteriores tratan señales de manera analógica, pero existe la posibilidad de demodular una señal en frecuencia usando un procesador digital de señal. Un ejemplo sería procesar la señal con el método de la "arcotangente" que calcula la frecuencia instantánea de una señal compleja (I y Q) a partir de la derivada de la fase instantánea. Otro menos común, que es el que se va a estudiar en este documento, es el uso del algoritmo de "CORDIC" que es parecido al de la "arcotangente" pero con operaciones más sencillas y es menos común su uso.

3.2 USO DE DSP EN SISTEMAS DE RADIOAFICIONADO

A lo largo de los últimos años, se han desarrollado dos tipos de DSP: los de punto fijo y los de punto flotante. Los procesadores de punto fijo representan los datos en un punto fijo (p.e 16 bit) y contienen arquitecturas muy rápidas y baratas, aunque su precisión es media y disponen de un margen dinámico reducido. Por el contrario, los procesadores de punto flotante representan los números en forma de notación científica y se usan para trabajar con números muy grandes. Si es cierto, que su velocidad disminuye con respecto a los de punto fijo y consumen algo más, pero lo compensan aumentando la precisión y el margen dinámico. A la hora de usar uno de estos dos hay que estudiar bien los requisitos de la aplicación.

La evolución de los microprocesadores ha potenciado decisivamente las comunicaciones digitales, haciéndolas asequibles tanto a la industria como a los particulares. Ello, unido a la irrupción del ordenador personal, ha permitido a los radioaficionados experimentar una serie de modalidades de comunicaciones digitales algunas de las cuales se han popularizado y extendido notablemente entre el colectivo durante los dos últimos decenios. La apertura al mercado de tecnologías como el DSP (Procesador Digital de Señal) no ha

hecho más que potenciar las posibilidades existentes y ampliarnos los horizontes para la experimentación debido a las múltiples ventajas que este implementa. Los sistemas de procesamiento de señal por software son muy baratos y ofrecen servicios baratos consumiendo pocos recursos.

Muchas de las radios portátiles de hoy en día incluyen este tipo de dispositivos debido a su bajo consumo y a sus bajos precios. Un ejemplo de este tipo de dispositivos sería el modelo "TECSUN PL-505" que se trata de un receptor de FM estéreo con funciones de demodulación por DSP. El ejemplo mencionado y casi todos los dispositivos encontrados en el mercado con este tipo de microprocesadores utilizan como método de demodulación un PLL digital y son usados para rangos de frecuencia FM de 88 a 108 MHz.

Como se ha mencionado en el apartado anterior y usando un punto de apoyo en este mismo, se va a probar "CORDIC" con un DSP para ver si los resultados y tiempos de ejecución son buenos y si es un tipo de algoritmo que podría sustituir a los PLL en este tipo de sistemas.

Capítulo 4. DEFINICIÓN DEL TRABAJO

4.1 JUSTIFICACIÓN

En esta sección se detallan los motivos principales del desarrollo de este proyecto.

4.1.1 CAMBIO A DIGITAL

Como se ha mencionado anteriormente, AMSAT es una organización que lleva poniendo satélites en órbita desde hace mucho tiempo, pero todos ellos cuentan con tecnologías analógicas a la hora de realizar el procesamiento de señales como pueda ser el caso de la demodulación en FM. Puede ser de buena práctica la sustitución de algunas de estas prácticas por procesos digitales que añaden las siguientes ventajas:

4.1.1.1 Sistemas fáciles de diseñar

En la mayoría de los casos de uso, suele ser mucho más fácil el diseño de sistemas digitales que de sistemas analógicos. Mientras que los sistemas analógicos requieren de la elección del tipo de materiales además de los valores de estos para que el sistema funcione en cohesión, los sistemas digitales únicamente requieren elegir un tipo de hardware y configurar todos estos parámetros en el software, lo que suele ser menos tedioso y más cómodo.

4.1.1.2 Fácil de almacenar y operar

Los sistemas analógicos suelen requerir que se respeten ciertos criterios de calidad que afectan a la transmisión y recepción de una señal. Estos criterios de calidad se ajustan y se ven condicionados por parámetros como la relación señal ruido. Este tipo de parámetros se puede ver modificado por numerosos factores como atenuación en cables, distorsiones por aire o reflexiones haciendo que la calidad pueda a llegar a disminuir hasta tal punto en el que la señal no sea idéntica a la deseada.

Por el contrario, los sistemas digitales permiten fácilmente almacenar las señales y operar con ellas haciendo posible la recuperación de las señales originales de manera más fácil. Esta capacidad de almacenamiento permite también una gran facilidad de gestión y mantenimiento de la información.

4.1.1.3 Factores económicos en radioaficionados

Los sistemas digitales se aprovechan del uso del software para ahorrar costes en mantenimiento. Esto no pasa en los sistemas analógicos, ya que, en la mayoría de los casos, no es casi ni posible este mantenimiento (no puedes sustituir piezas en el espacio si no las tienes).

Por otro lado, muchos microprocesadores de procesamiento digital están hechos con ciertos materiales resistentes a la radiación, lo que hace que no haya que utilizar materiales más caros y específicos como en los sistemas analógicos.

4.1.2 CORDIC

CORDIC es un algoritmo que realiza aproximaciones a ciertas operaciones trigonométricas de manera mucho más sencilla y con operaciones de suma, resta y producto. Al no usar la división, ahorra mucho tiempo de procesamiento lo que puede ser clave a la hora de diseñar un sistema receptor de audio para radioaficionados donde no se disponen de procesadores con relojes suficientemente rápidos. Por ello, se va a comparar en Matlab su eficacia en cuanto a calidad respecto de otros algoritmos como el de la “arcotangente” y, posteriormente, se verá si merece la pena comparando la relación calidad/tiempo.

4.2 OBJETIVOS

El objetivo principal del proyecto será la programación de un método de demodulación digital usando algoritmos como el de la arcotangente o el método CORDIC para su implantación en la placa DSP de la gama STM32FXXX que se dispone actualmente. Hay que destacar, que este tipo de placas contienen un tipo de microprocesadores que reducen

tanto el coste como la energía usada en el procesamiento, lo que es primordial en los sistemas de satélites, sobre todo en los sistemas de radioaficionado. Para conseguir lo mencionado, se recibirá una señal analógica, a la que previamente se le añadirá una componente de offset (las conversiones ADC no suelen tomar valores negativos), y se realizará una conversión a digital con un ADC. Esta señal será procesada mediante uno de los algoritmos mencionados anteriormente eligiendo cual es más eficaz y proporciona una mejor curva SNR out/in. Finalmente, se usará un conversor DAC para recuperar la señal analógica de audio y se comprobará si se escucha adecuadamente.

La placa DSP que se utiliza no tiene mucha memoria RAM por lo que interesa estudiar como programar de la manera más eficaz para que no haya desbordamiento de memoria. Para ello, se estudiará si es posible procesar la señal utilizando únicamente números con coma flotante o si es necesario usar números con formato Q15/Q31 además de intentar usar el menor número de muestras almacenadas. Esto permitirá también una reducción considerable en los tiempos de procesamiento.

Sería interesante también, estudiar la posibilidad de realizar un procesado en bloques en el que cada bloque realiza una acción, reduciendo así la posibilidad de solape entre las distintas muestras que van llegando al sistema en tiempo real. Por ejemplo, se podría llenar medio buffer del ADC y, mientras se llena la otra mitad, procesar la señal y realizar la conversión digital-analógica.

El objetivo a largo plazo (uno o dos años), es implementar el software diseñado en uno de los satélites que se lanzarán al espacio. Esto solo podrá pasar si el diseño es eficaz y las señales recibidas de audio en tiempo real son demoduladas de manera correcta para un rango completo de frecuencias de 300 a 3000 Hz.

4.3 METODOLOGÍA

En primer lugar, es necesario tener claro los numerosos conceptos del ámbito del procesamiento de señales (muestreo, FFT, diezmado, filtros FIR, etc.). Por ello, el proyecto comienza estudiando todos los conceptos que se aplican durante la vida de este.

En segundo lugar, se instalará la herramienta STM32CubeIDE que se trata de un entorno para la programación de las distintas placas de la gama STM32xxx. Al principio, se realizarán algunas pruebas de programas sencillos como el parpadeo de algunos leds, uso de la UART, ADC, DAC, etc. Gracias a que se ha aprendido a configurar la placa en este punto, se podrá dejar el ADC configurado y realizar pruebas en el laboratorio para calcular la frecuencia de muestreo deseada.

A continuación, se procederá a instalar las librerías de DSP y realizar algunas pruebas. Se adjunta un link en el que se detallan los pasos para realizar este proceso, ya que es obligatorio tener una configuración en el proyecto para el uso de estas librerías:

https://www.youtube.com/watch?v=vCcALaGNlyw&ab_channel=YetAnotherElectronicsChannel

Lo siguiente, será realiza pruebas en Matlab para observar los resultados que se pueden esperar. Para ello, se incluirá también un programa que realice la modulación FM de una señal. Aquí se elegirá el tipo de demodulador que se usa y, una vez concluido este apartado, se traducirá el código a lenguaje C para programar la placa STM32xxxx y realizar pruebas con una muestra fija. Una vez observado que con la muestra fija el funcionamiento es adecuado se configurará el conversor digital/analógico y se realizarán pruebas en tiempo real del procesado de señales.

4.4 PLANIFICACIÓN

En la figura 1, se muestra un diagrama de Gantt de la planificación inicial del proyecto.

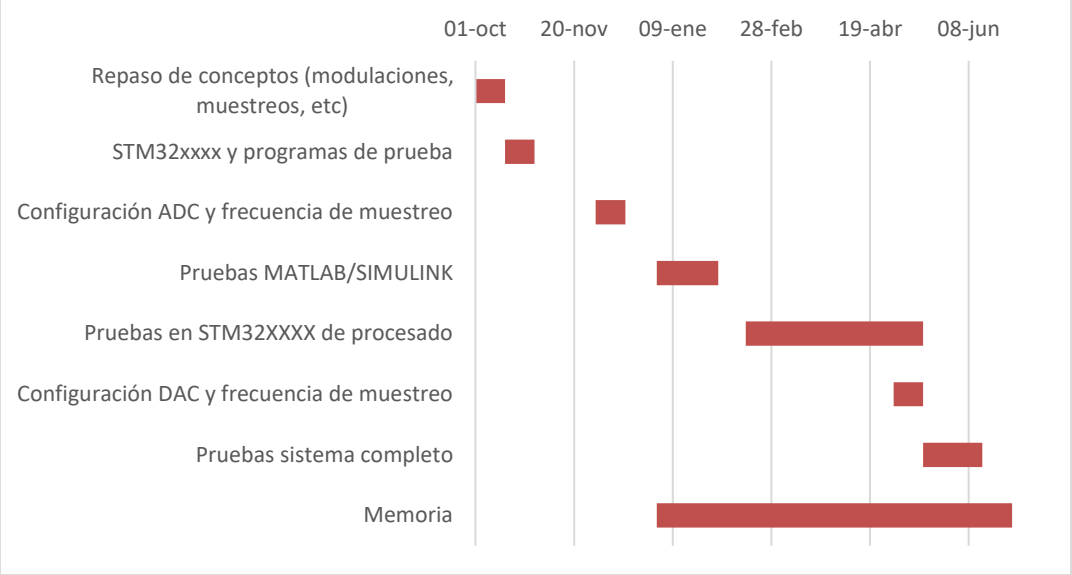


Figura 1. Planificación del proyecto

Como se ha explicado en el punto 4.3, el primer paso será el de repaso de conceptos de procesamiento de señal. Aquí, se estudiarán los distintos métodos de demodulación de señales en frecuencia, teorema de submuestreo y sobremuestreo, filtros FIR, diezmadores, etc.

Una vez se tienen claros los conceptos, se realizarán pruebas sencillas en el entorno STM32CubeIDE y con la placa NUCLEO-L476RG. Se aprovecharán las pruebas con el ADC para elegir su configuración además de la frecuencia de muestreo deseada para el proceso. Se ha de remarcar que elegir bien la frecuencia de muestreo es la parte más importante del proyecto porque los parámetros de los bloques que vienen a continuación del ADC dependen de la banda de Nyquist en la que quede nuestra señal.

El siguiente paso será realizar pruebas en Matlab. Se realizarán pruebas tanto de una señal modulada generada manualmente con Matlab como de una señal recibida de un generador

de señal. El demodular a partir de la señal manual nos permitirá conocer como funcionan los distintos tipos de demodulación en frecuencia y ver si tiene sentido implementar un demodulador basado en CORDIC. Con la señal del generador, se probarán los algoritmos de CORDIC y de la arcotangente para realizar una comparación. Así, veremos si se pierde mucha calidad de señal cuando usamos CORDIC.

Posteriormente, se traducirá todo el código a C para implementarlo en el proyecto. A su vez, se copiarán las tablas de los osciladores y los coeficientes de los filtros en el proyecto para no perder tiempo de procesamiento en generarlas por software. Cuando se tenga el programa en C, se realizarán pruebas en tiempo real del procesado tomando algunas muestras y viendo la salida en Matlab.

Por último, se configurará el DAC y se realizarán pruebas del sistema completo. Es importante este paso comprobar que los tiempos de los distintos bloques cuadren de manera:

$$\frac{Tadc}{2} \geq Tproc \qquad (1)$$

$$\frac{Tadc}{2} \geq Tdac \qquad (2)$$

Capítulo 5. RECEPTOR DE AUDIO

5.1 DESCRIPCIÓN GENERAL

La Figura 2 representa un esquema general del receptor de audio diseñado durante la vida de este proyecto.

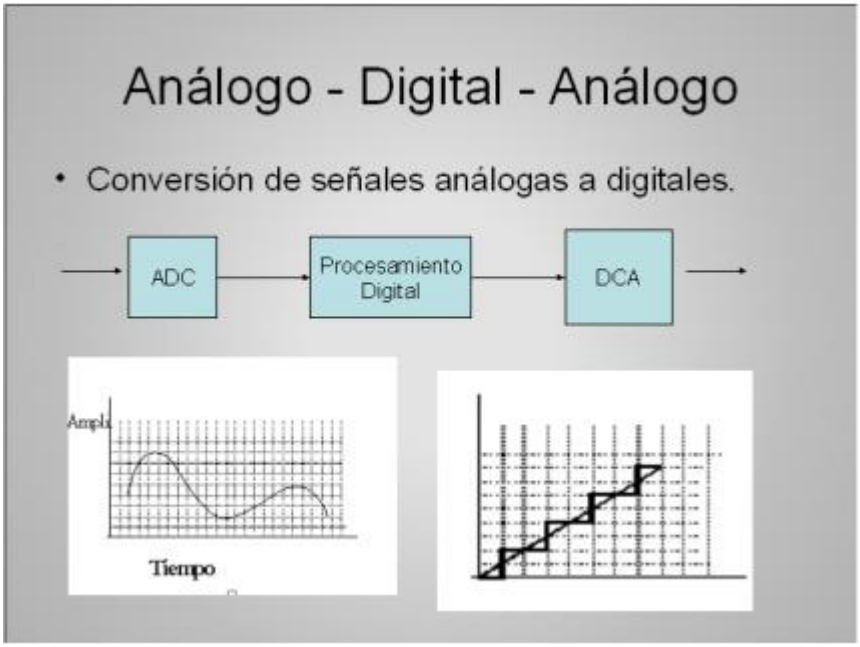


Figura 2. Esquema general

Primero, se estudiará el conversor analógico-digital. Aquí, es muy importante una selección correcta de la frecuencia de muestreo además de la indicación de los niveles de referencia de la cuantificación y del valor medio, ya que no recibe valores negativos (formato entero sin signo).

El segundo bloque es el del procesamiento digital, en el que se incluirán y configurarán los distintos elementos que intervienen en todo el procesado de la señal. Se hará uso de dos

osciladores, dos diezmadores, un bloque que recuperará la moduladora con CORDIC (probando con diferenciación de fase o sin ella) y un filtro paso-banda que eliminará las frecuencias que quedan fuera de la banda de 300-3000 Hz.

El último bloque es un conversor digital-analógico usado para recuperar la señal de forma analógica. Aquí, se configurará un "timer" que activará el DAC según la frecuencia de muestreo resultante del bloque de procesado.

Tanto en el ADC como el DAC, se usa DMA (Direct Memory Access) que permite a ambos subsistemas acceder a la memoria de datos, tanto leer como escribir, sin necesidad de usar la CPU. Al no estar sometida a continuas interrupciones, se permite el uso de la CPU para otras tareas, disminuyendo el tiempo de ejecución de los distintos procesos del sistema.

5.2 CONVERSOR ANALÓGICO-DIGITAL

En este apartado, se han seleccionado el parámetro de la frecuencia de muestreo y el uso de submuestreo frente a sobremuestreo. Además, para la correcta recepción de la señal se añade un valor de nivel medio a la señal a la entrada del conversor.

Se ha de mencionar, que la primera opción manejable para la frecuencia de la portadora de FM fue a 45 MHz para entrar con una señal más rápida. La otra opción era entrar a 455 kHz que es más lento, pero puede reducir muchos problemas que se estudian.

5.2.1 PROCESO DE MUESTREO DE UNA SEÑAL ANALÓGICA

5.2.1.1 Oversampling o sobremuestreo

Según la teoría de Nyquist, las señales tienen que ser muestreadas usando una frecuencia de muestreo que sea el doble o mayor que la máxima frecuencia de nuestra señal.

$$Fs \geq 2f \qquad (3)$$

Para nuestro ejemplo, en el que las frecuencias de las portadoras a seleccionar son de 455 kHz y 45 MHz respectivamente, se estudia la viabilidad para un ancho de banda máximo de 15 kHz:

	Caso 1 (455 kHz)	Caso 2 (45000 kHz)
Frecuencia máxima	470 kHz	45015 kHz
Fs mínima	940 kHz	90030 kHz

Tabla 1. Frecuencias de muestreo con oversampling

Para el caso 2, en el que la frecuencia de la portadora estará a 45 MHz, obtenemos una frecuencia de muestreo mínima de 90 MHz aproximadamente. La placa que se usa tiene un procesador que funciona como máximo a 80 MHz por lo que sería inviable elegir una frecuencia adecuada según este criterio.

Por el contrario, el caso 1 sería más que viable pudiendo elegir una frecuencia de muestreo, por ejemplo, de 1 MHz. Sin embargo, el ADC solo es capaz de almacenar 2048 muestras debido a las limitaciones de memoria.

$$t = \frac{1}{Fs} * 2048 \qquad (4)$$

Si para la ecuación 4, que muestra el tiempo del buffer almacenado por el ADC para el número máximo de muestras, se elige la **mínima frecuencia** de muestreo, se obtendría el **tiempo máximo** que ocuparía el buffer para este caso. Aplicando esto, obtenemos un tiempo máximo de 2,18 ms para un buffer entero. Como se verá en otro apartado, se procesa medio buffer, teniendo 1,14 ms de muestras del ADC, lo que hace inviable la selección de esta frecuencia de muestreo porque no se cumpliría la ecuación 1.

5.2.1.2 Undersampling o submuestreo

Si en vez de muestrear dos veces por encima de la máxima frecuencia, muestreamos por debajo, estaremos realizando un submuestreo o "undersampling". Esta técnica es conocida también por los nombres de muestreo en la banda de paso, muestreo de armónicos o muestreo super-Nyquist. El teorema de muestreo de Nyquist-Shannon ofrece una versión modificada del teorema de muestreo de Nyquist y dice que basta con muestrear la señal con una frecuencia superior o igual al doble de **ancho de banda** de la señal y no al de la frecuencia máxima. Es muy útil a la hora de trabajar con señales de espectro estrecho en las que el ancho de banda es mucho menor que la máxima frecuencia. Además, al muestrear más lento, podremos almacenar más tiempo para un número menor de muestras.

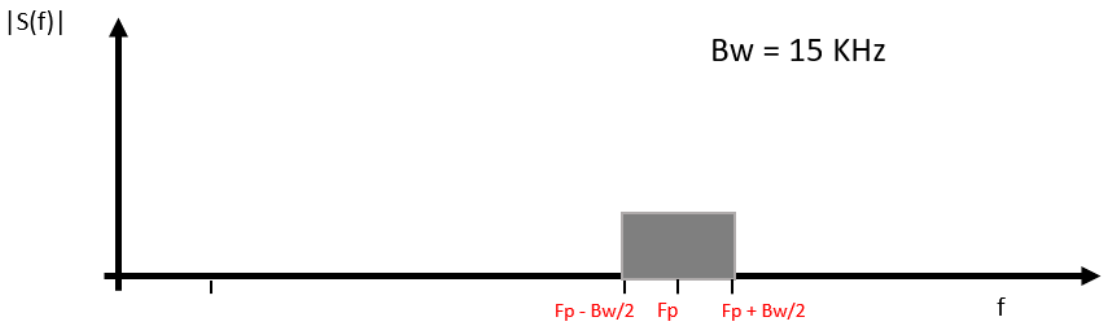


Figura 3. Espectro de la señal continua

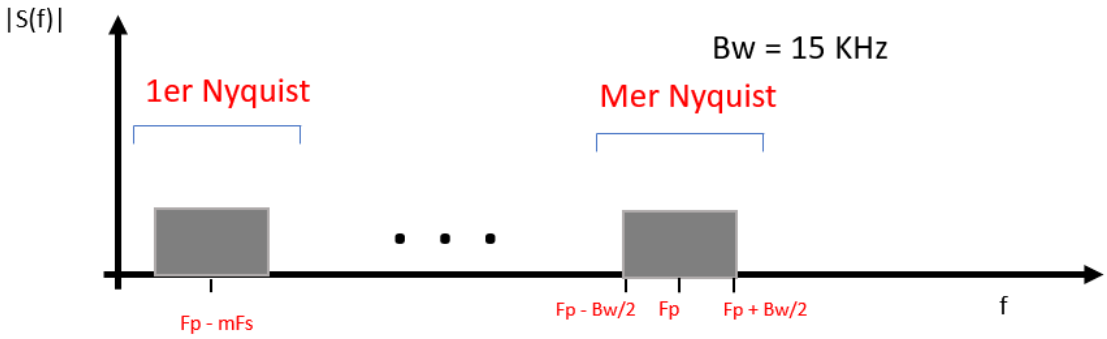


Figura 4. Espectro de la señal submuestreada

Como podemos observar la señal esta submuestreada, es decir, las bandas de Nyquist aparecen por debajo de la frecuencia portadora.

Es necesario imponer y cumplir unas condiciones para evitar que se produzcan efectos de alliasing entre las distintas bandas de Nyquist.

$$Fp - mFs + \frac{BW}{2} < Fp - (m-1)Fs - \frac{BW}{2} \qquad (5)$$

Si resolvemos la ecuación anterior obtenemos:

$$Fs > BW \qquad (6)$$

Para asegurarnos de que se cumple siempre se impone la condición del doble.

Ahora bien, la señal que recibe el receptor tiene como máximo ancho de banda 15 kHz. Si aplicamos este criterio se tendría que muestrear a una frecuencia mínima de 30 kHz. Sustituyendo en la ecuación 4 la frecuencia de muestreo mínima, obtenemos un tiempo total de buffer de 68 ms aproximadamente, lo que es más que suficiente para que cuadren los tiempos entre el procesado, el ADC y el DAC.

5.2.2 SELECCIÓN DE LA FRECUENCIA DE MUESTREO

Una vez estudiados los distintos tipos de muestreo, se elige la opción del submuestreo porque cuadra más para el sistema que se quiere diseñar. Ahora bien, será necesario escoger una frecuencia de muestreo suficientemente rápida y que deje margen para el tiempo de procesado sea menor que el tiempo que se tarda en rellenar medio buffer en el conversor analógico-digital. Además, se necesita que por lo menos la primera banda de Nyquist aparezca por debajo de la mitad de la frecuencia de muestreo para que se sea capaz de ver donde cae la frecuencia de la portadora y poder diseñar de manera más cómoda el resto de las componentes del receptor. Por ejemplo, no se puede elegir la frecuencia de los osciladores sin saber el dato de la frecuencia de la portadora en la banda de Nyquist elegida.

Para la selección de la frecuencia de muestreo se elige usar el reloj de 80 MHz, que es la máxima frecuencia disponible y el procesado será más rápido, y se configura la frecuencia de muestreo variando los distintos parámetros que ofrece la configuración del conversor.

A continuación, se inserta una imagen que muestran los parámetros del ADC y se explican los más relevantes usados para seleccionar la frecuencia de muestreo.

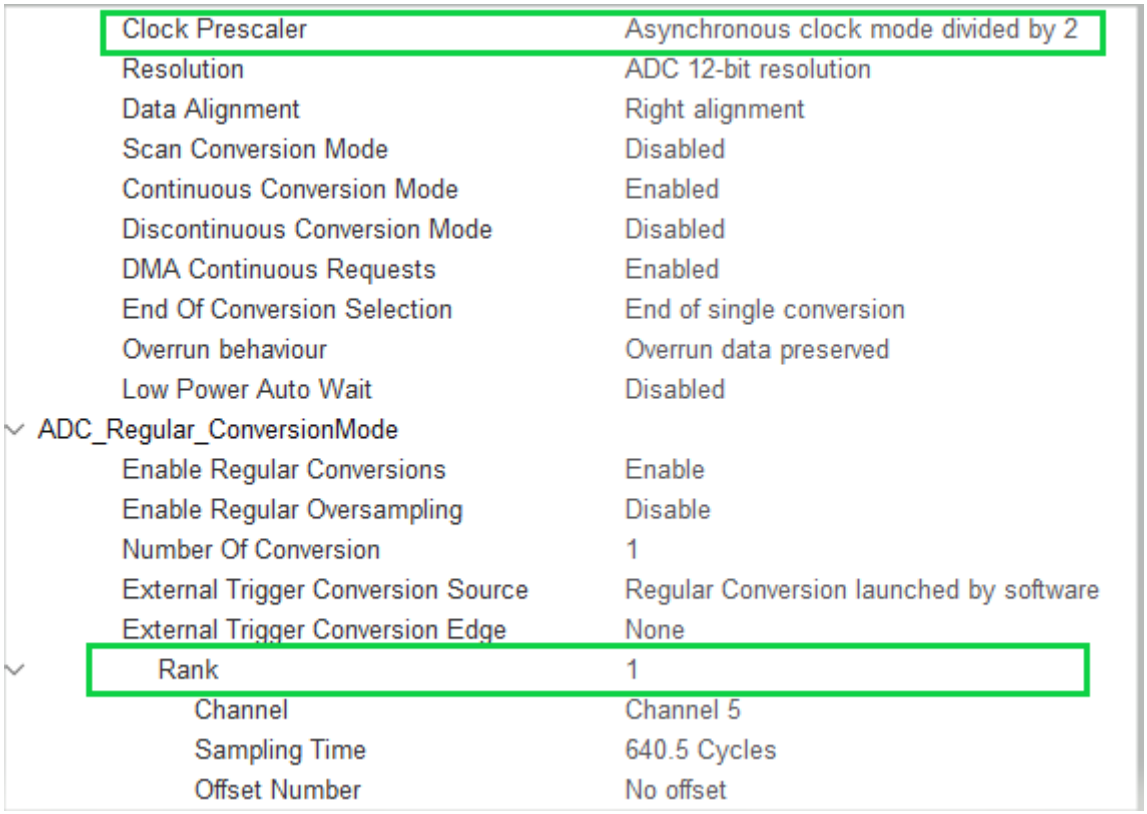


Figura 5. Parámetros relevantes del conversor analógico-digital para la selección de la frecuencia de muestreo

- Clock prescaler: Divide la frecuencia del reloj por un número, normalmente entero o potencia de 2, para conseguir ajustar un timer más lento que el que se tiene. En el aso de la Figura 5 se divide el reloj por 2 lo que equivaldría a usar un reloj de 40 Mhz ($\frac{80}{2}$ MHz).
- Rank: Parecido al clock prescaler, se usa para configurar la frecuencia de muestreo. El parámetro sampling time se usa para seleccionar los ciclos a los que se quiere guardar una muestra. A más ciclos, más lento será el muestreo. Al igual que antes, se podría sacar la frecuencia de muestreo dividiendo el reloj configurado por esta cantidad. Una peculiaridad que se expone en la documentación es que el reloj tarda

en arrancar 12.5 ciclos, por lo que habría que sumarle esta cantidad al sampling time.

Ahora ya se saben los parámetros que se van a seleccionar para elegir la frecuencia de muestreo por lo que se estudia cual será una configuración decente de estos parámetros para una frecuencia de portadora FM de 455 kHz.

CLOCK PRESCALER = 1			
RANK	**Fs (kHz)**	**Banda Nyquist (kHz)**	**N**
2,5	5.333,33	455	0
6,5	4.210,53	455	0
12,5	3.200	455	0
24,5	2.162,16	455	0
47,5	1.333,33	455	0
92,5	761,9	455	0
247,5	307,69	147,3	1
640,5	122,51	87,47	3

Tabla 2. Estudio de la frecuencia de muestreo para un prescaler de 1

CLOCK PRESCALER = 2

RANK	Fs (kHz)	Banda Nyquist (kHz)	N
2,5	2.666,66	455	0
6,5	2.105,26	455	0
12,5	1.600	455	0
24,5	1.081,08	455	0
47,5	667	455	0
92,5	380,95	74,05	1
247,5	153,85	147,3	2
640,5	61,26	26,21	3

Tabla 3. Estudio de la frecuencia de muestreo para un prescaler de 2

Ahora, se estudia cuáles de estas combinaciones son compatibles con las especificaciones detalladas a lo largo del apartado 5.2. Se había especificado que se ha de elegir submuestreo, por lo que se eliminan todas las frecuencias de muestreo cuyo valor este por encima de 455 kHz (N = 0). También, se pueden descartar aquellas cuya banda de Nyquist seleccionada quede por encima de la mitad de la frecuencia de muestreo.

Con todas las restantes, se procede a estudiar los tiempos en los que se rellena medio buffer (1024 muestras) para que los elementos del sistema receptor funcionen en sincronía.

ID	Fs (kHz)	Tiempo (ms)

1	307,69	3,33
2	380,95	2,69
3	61,26	16,72

Tabla 4. Tiempos de muestreo de medio buffer

Como se verá en otro apartado, solamente la combinación 3 va a cumplir los requisitos de tiempo del sistema por lo que se elige esta frecuencia de muestreo para diseñar el resto de las componentes.

5.2.3 RESOLUCIÓN DMA

El ADC guarda los valores en formato uin12_t. Esto quiere decir que usa 12 bits para la conversión dividiendo la escala de referencia en 4095 valores con un valor máximo de voltaje de 3,3 V. Como el ADC se configura por DMA o acceso a memoria directo, es necesario seleccionar la resolución del DMA.

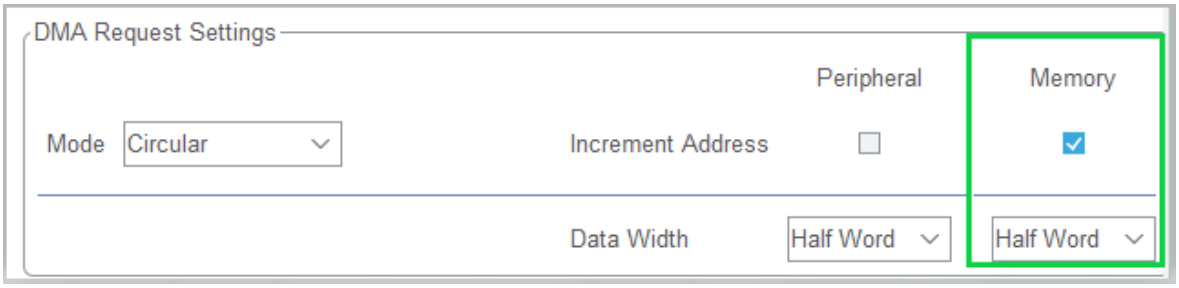


Figura 6. Resolución DMA

Teniendo en cuenta que el nivel de referencia es de 3,3 V se procede a calcular los distintos niveles de resolución y elegir el formato en el que se van a almacenar las muestras según la siguiente fórmula:

$$Resolución = \frac{3.3}{2^{n}-1} \qquad (7)$$

Siendo n el número de bits a usar:

- Para un byte (8 bits), se obtiene una resolución de 12,9 mV.
- Para media palabra (16 bits), se obtiene una resolución de 0.05 mV.
- Para una palabra (32 bits), se obtiene una resolución de 0,77 nV.

Utilizar una resolución de 32 bits puede no ser eficiente debido la gran cantidad de memoria utilizada. Tampoco se puede coger 8 bits porque, como se ha comentado, el ADC convertirá los valores a 12 bits y dejaríamos fuera demasiados valores. Por ello, se escoge utilizar media palabra.

5.2.4 CONFIGURACIONES ADICIONALES

- **Resolución ADC**: se usa una resolución de 12 bits.
- **Alineación de datos**: se configura la entrada de datos de izquierda a derecha (Figuras 8 y 9).
- **Conversión continua activada**: esto evita que se requiera de activación por software cada vez que se quiera escribir con el ADC. Se configura para que vuelva a escribir cuando se llene el buffer (tiempo real).
- **DMA continuous request**: se usa el DMA en modo circular (Figuras 8 y 9) para que se realicen las conversiones de 12 a 16 bits constantemente y no se pare cuando acabe una conversión única.

Figura 7. DMA -Buffer vacío

Figura 8. DMA -Buffer lleno

5.2.5 NIVEL MEDIO

Como se ha remarcado en el subapartado 5.2.3, el ADC va a trabajar con formatos uint12_t. Esto quiere decir que convertirá los valores analógicos a digital en formato entero **sin signo** con un nivel de referencia de 3,3 V y en 12 bits, lo que equivale a decir que el 0 tomará el valor "0000000000" (0) y el 3,3 tomará el valor "111111111111" (4095).

Sabiendo esto, es lógico pensar que la señal a la entrada no puede tomar valores negativos por lo que habrá que diseñar un circuito previo que añada un nivel medio a la señal para que esta únicamente tenga valores positivos. A esto, hay que sumarle que parece que existe un amplificador operacional a la entrada del ADC que aumenta el valor de la señal recibida.

Como en las especificaciones no se comenta este aspecto (o no se ha encontrado), se decide realizar una simple prueba con una señal de 550 mV de valor medio y 950 mVp-p.

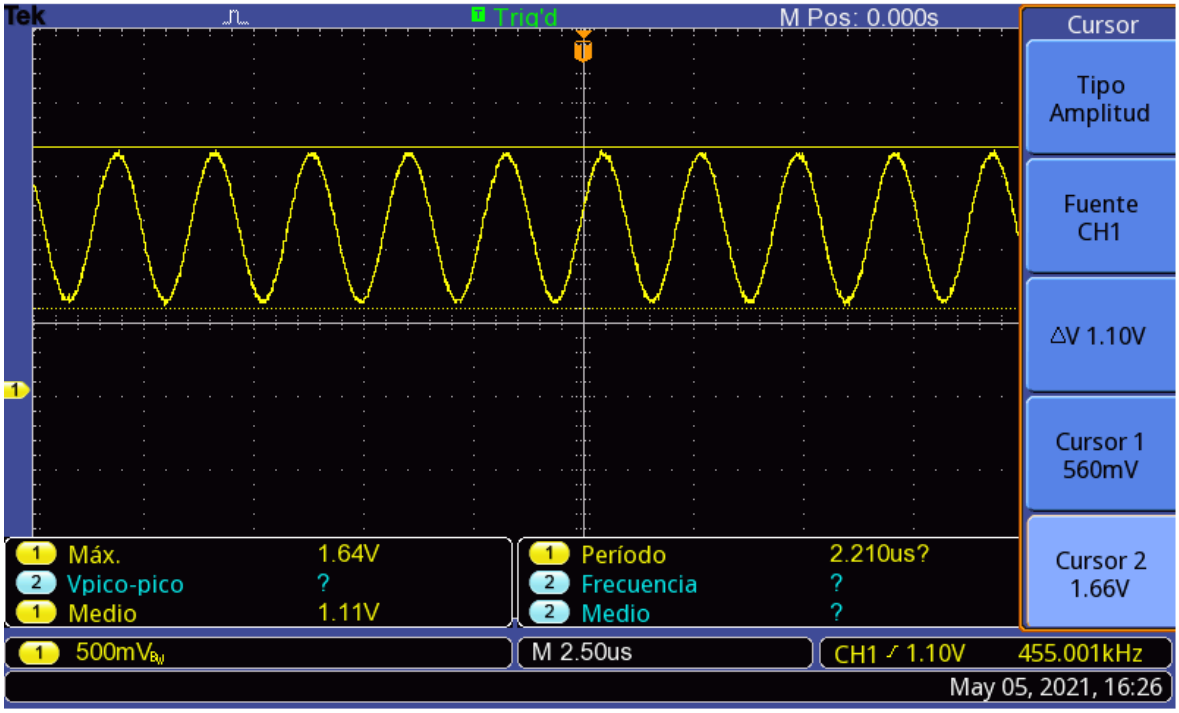


Figura 9. Prueba entrada

Ahora, el valor de voltaje pico-pico es de unos 1,1 V, obteniendo una ganancia 1,16 V/V. Esto no debería pasar, al estar aumentando el nivel de señal a la entrada del ADC parece ser que está saliendo corriente de la placa.

En este proyecto no se diseñan ni se especifican los valores que debe tener la señal a la entrada del ADC y simplemente se realizan pruebas con señales que cumplan los requisitos. Por esto, es importante realizar un estudio previo de los niveles de la señal y de las resistencias de entrada que tienen que ser bajas antes de conectar el sistema receptor.

5.2.6 TIEMPO DEL ADC

En este apartado simplemente se comprueba que el tiempo que tarda el ADC en rellenar un buffer (o medio) sea el que se ha especificado. Para ello, se activa un pin en la placa cuando se rellene medio buffer y se desactiva cuando el buffer esté completo.

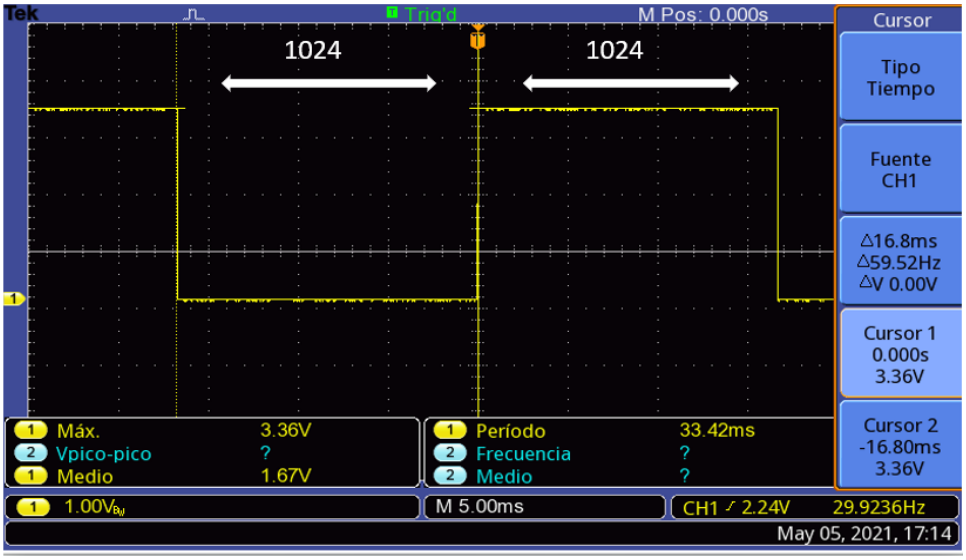


Figura 10. Tiempo ADC

Se observa que el periodo de la señal es de 33,42 ms. Medio buffer correspondería con unos 16,71 ms lo que equivale a una frecuencia de muestreo de 61280 Hz. Se trata de una frecuencia de muestreo casi idéntica a la que se había calculado en apartados anteriores peor ahora la primera banda de Nyquist queda entorno a los 26000 Hz y no 26210 Hz lo que puede suponer una diferencia importante a la hora de diseñar el resto de las componentes.

5.3 BLOQUE DE PROCESAMIENTO

En este apartado se configuran todos los elementos que forman parte del procesado digital para dejar la señal preparada para su conversión a analógico con el DAC.

A continuación, se muestra un esquema general de las componentes diseñadas dentro de este bloque.

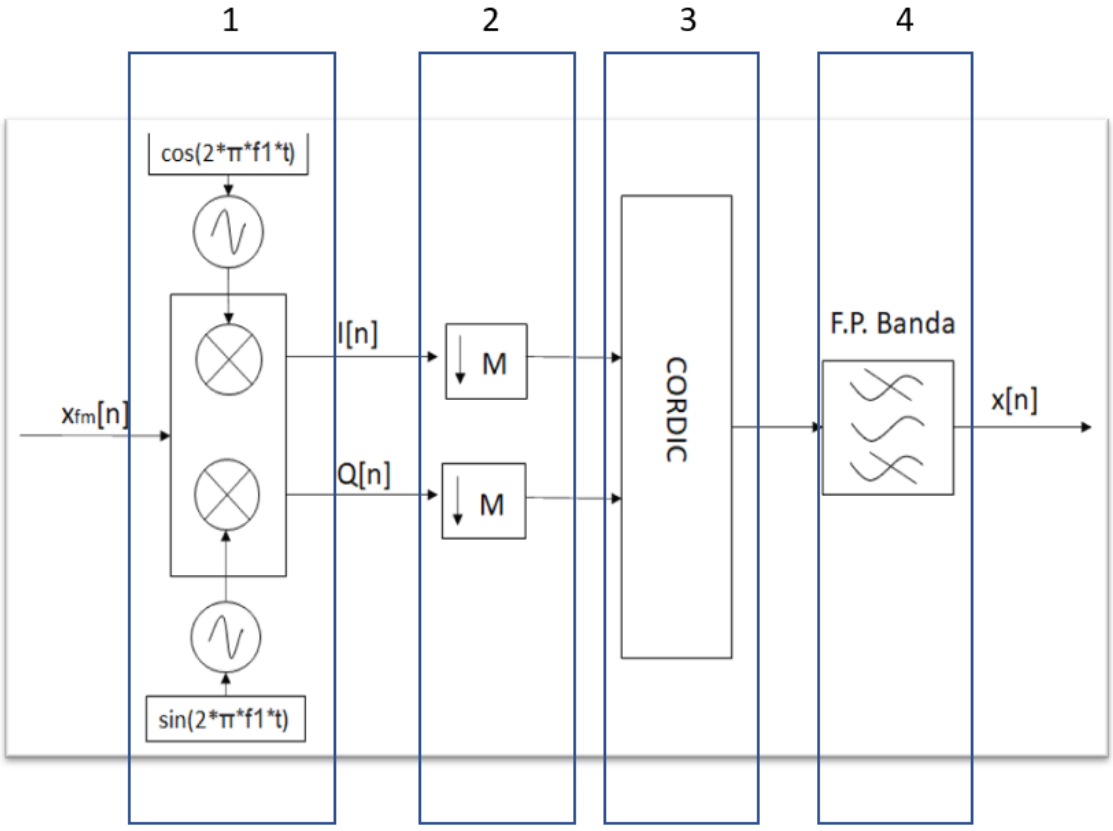


Figura 11. Esquema del bloque de procesado

Definimos xfm[n] como la señal muestreada por el ADC y x[n] la señal digital que pasará al DAC.

Hay que mencionar que se han hecho todas las pruebas con Matlab y tanto con una señal modulada a mano como con una señal modulada y muestreada por el ADC. La señal que se le pasó al ADC es una señal modulada por un generador de señal con frecuencia de 2500 Hz, portadora de 455 kHz y desviación en frecuencia de 5000 Hz.

5.3.1 PRUEBAS CON SEÑAL MODULADA MANUAL

Se comienza realizando unas pruebas en Matlab con una señal simple con los datos de la Tabla 5.

Moduladora (Hz)	Portadora (Hz)	beta	Fs (kHz)
2000	20000	1	200

Tabla 5. Datos señal simple manual

Obteniendo las siguientes señales:

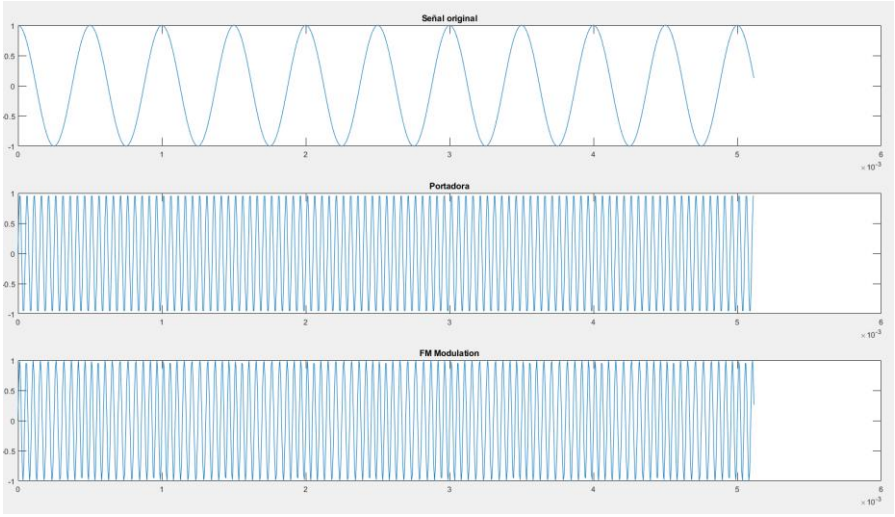


Figura 12. Original, portadora y modulada prueba simple

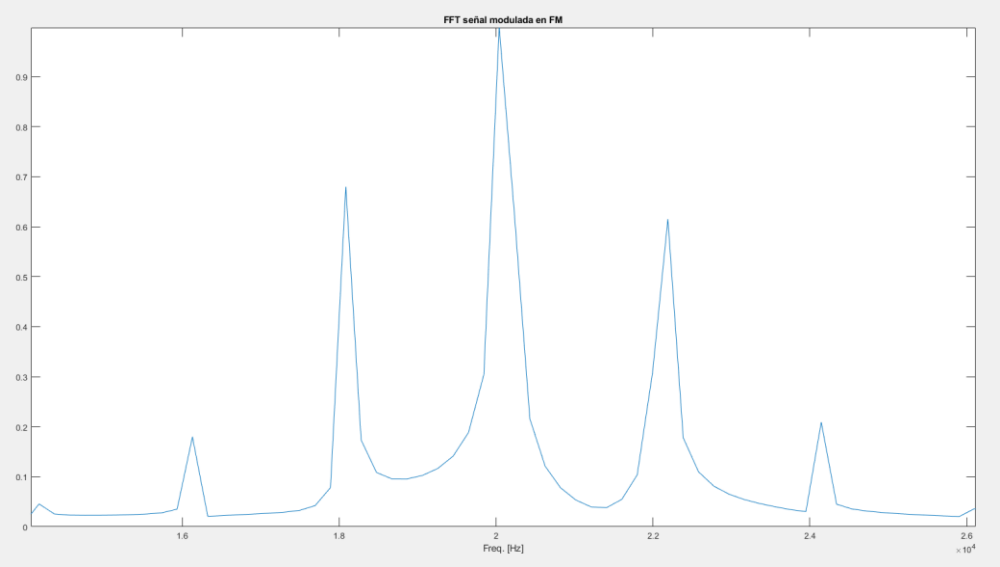


Figura 13. Original, portadora y modulada prueba simple en frecuencia

Vemos como el tono de la portadora aparece a los 20 kHz y las componentes moduladas aparecen cada 2000 Hz.

Con esta señal generada manualmente procedemos a realizar la demodulación tanto con CORDIC como con la arcotangente para ver las diferencias en los resultados.

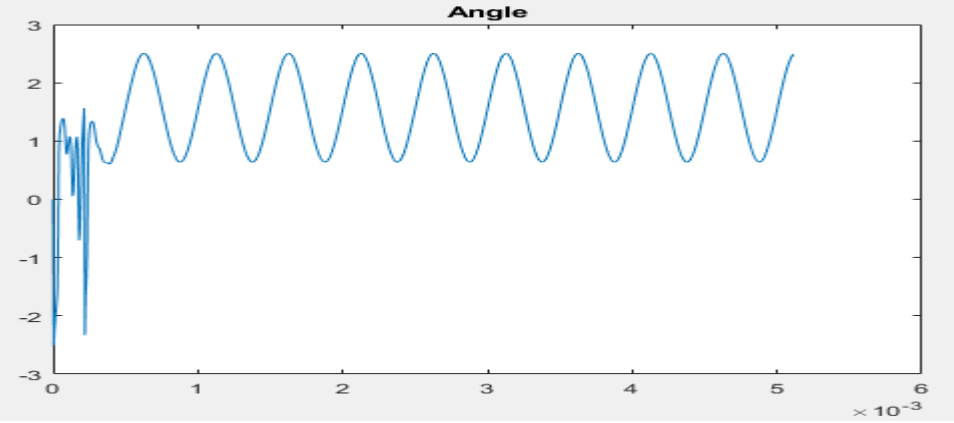


Figura 14. Demodulada con arcotangente para pruebas manuales

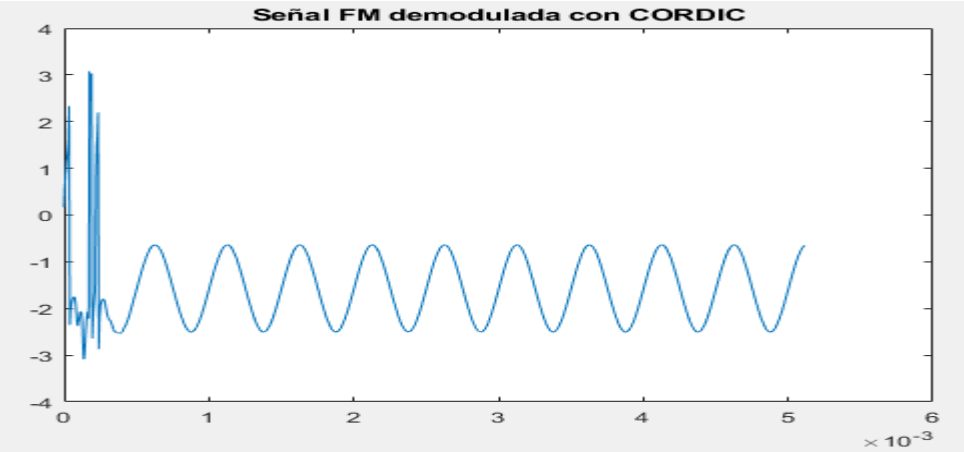


Figura 15. Demodulada con CORDIC para pruebas manuales

Las señales no son idénticas pero lo importante es que tengan la misma frecuencia que es la frecuencia de la moduladora inicial.

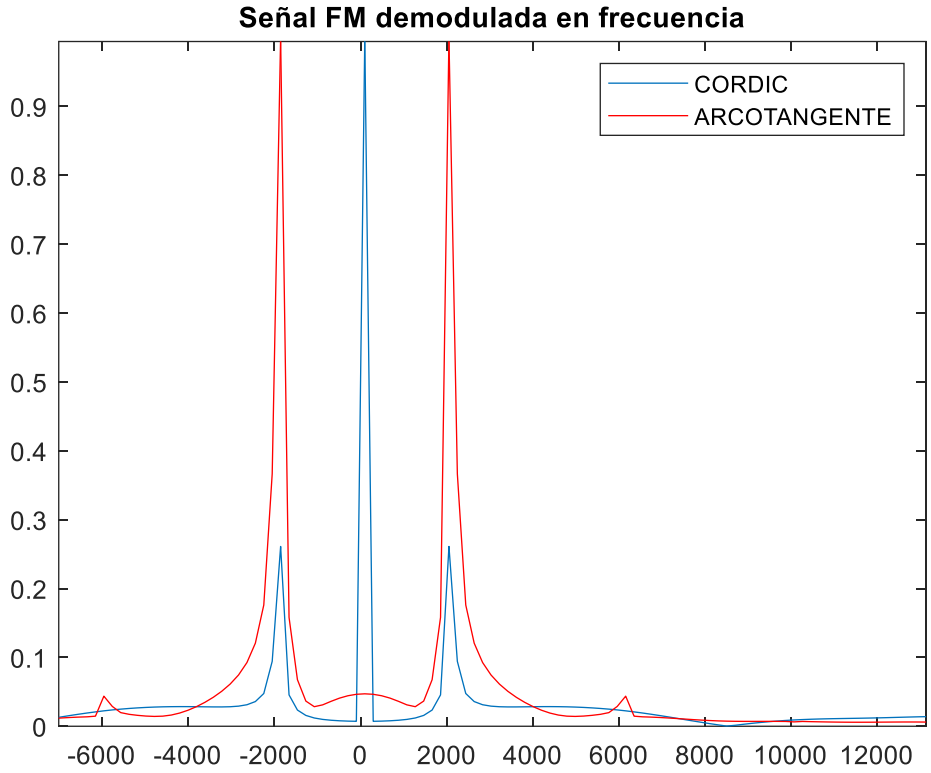


Figura 16. Demodulada en frecuencia para pruebas manuales

Efectivamente, la frecuencia de ambas señales es la misma.

5.3.2 PASO A BANDA BASE

Se va a realizar la etapa 1 de la Figura 11. Se trata de generar dos señales a las que llamaremos I y Q o lo que equivale a decir que se tiene la señal en cuadratura. Estas

señales resultarán del producto de la señal que recibimos del conversor analógico-digital y dos osciladores que están desfasados entre ellos 90 grados.

Antes del producto es necesario ver donde cae la portadora para la muestra recogida.

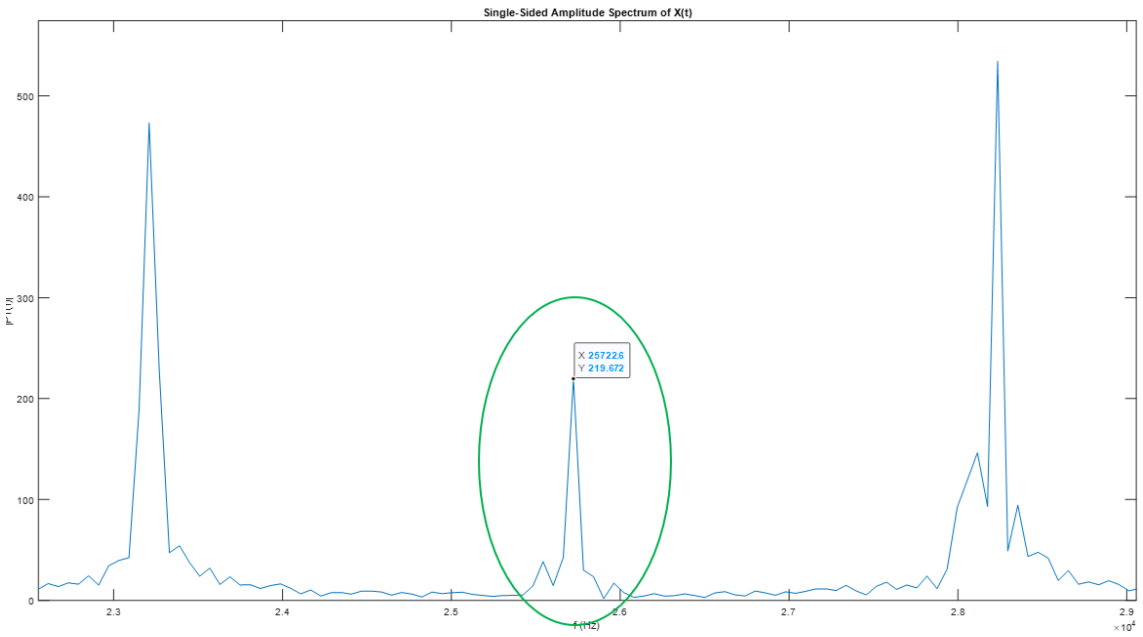


Figura 17. Transformada de Fourier de muestras del ADC

En la Figura 17 se ve como la portadora está entorno a los 25740 Hz. Esta va a ser la frecuencia que queremos eliminar por lo que será la frecuencia del tono de los osciladores.

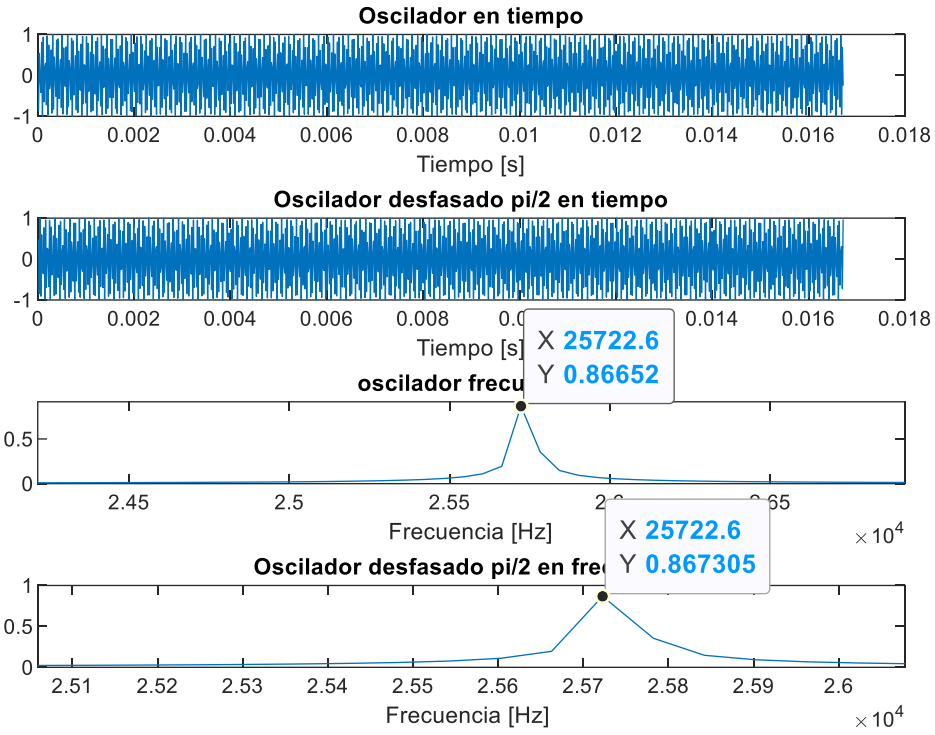


Figura 18. Osciladores en tiempo y frecuencia

Las imágenes de la Figura 18 representan las señales de los osciladores, tanto en tiempo como en frecuencia, desfasadas 90 º para el mismo número de muestras que la señal recogida por el ADC. Se evita tener que generar estos valores cada vez que se ejecute un procesado y se guardan los valores de los osciladores en la escala de tiempo en una tabla de datos dentro del programa.

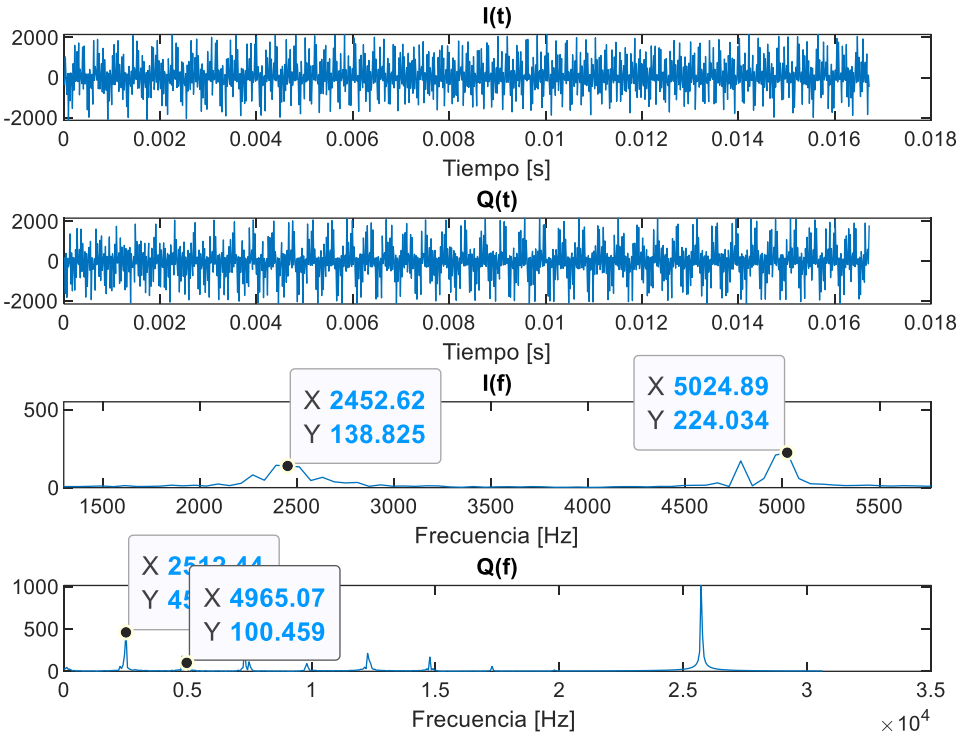


Figura 19. Señal en cuadratura

Ahora las primeras componentes de la señal FM están entorno a los 2500 Hz. Aparecen más componentes porque no se ha realizado un filtrado paso-banda previo para quedarse con la banda de Nyquist deseada. Esto se hace, para reducir al máximo el tiempo de procesado en la placa ya que los filtros tardan mucho debido a sus numerosas operaciones.

5.3.3 DIEZMADO

En la muestra que se ha usado para realizar las pruebas, se tiene una frecuencia de muestreo de unos 61300 Hz. La señal a la salida del sistema receptor va a tener como

máximo una frecuencia de 3000 Hz. Por ello, ya que los distintos alias de la señal están muy separados, puede aparecer ruido entre las componentes. Para reducir este efecto, es necesario disminuir la frecuencia de muestreo eliminando cierta cantidad de muestras:

$$fs' \geq 2fm \qquad (8)$$

Para reducir la frecuencia de muestreo se utilizan dos diezmadores para las señales I e Q. Al diezmar nos quedamos con una muestra de cada M y desechamos el resto. Si la frecuencia de muestreo resultante está muy próxima a 2fm (6000 Hz) va a resultar más complejo el diseño del filtro a la salida del procesado y el número de coeficientes va a ser menor reduciendo también la efectividad. Además, la calidad de la salida de la señal en el DAC será peor ya que se utilizarían menos muestras por periodo en la representación de la señal. Se decide realizar un diezmado con un factor M equivalente a 4.

$$fs' = \frac{fs}{M} \sim 15300\ Hz \qquad (9)$$

Las placas de la familia ARM disponen de unas librerías para el procesado digital de señales. En estas librerías se puede encontrar distintas funciones de diezmado. Todas ellas requieren de un filtro FIR previo a la etapa de diezmado. Por ello, se diseñan con Matlab los coeficientes de este filtro mediante la función designMultiRateFir(1,4). Está función diseñará un filtro con un factor de interpolado (aumento frecuencia de muestreo) de 1 y un factor de diezmado de 4.

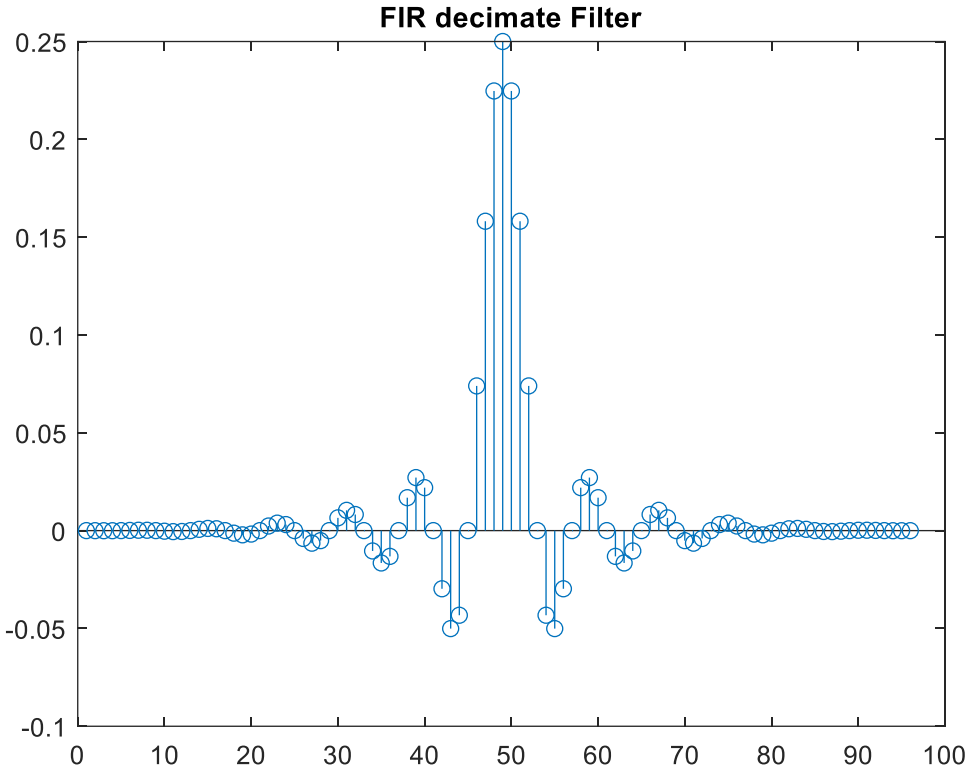


Figura 20. FIR Decimate Filter

Se observan los resultados para M igual a 4 y M igual a 8 para ver si el diezmado casi al límite de la condición de la ecuación 8 puede influir. Analizando las Figuras 21 y 22 se aprecia como para M igual a 4 aparece la segunda componente de la señal, pero está suficientemente separada para que se pueda eliminar con un filtrado paso bajo. Se decide implementar el diezmado con M igual a 4.

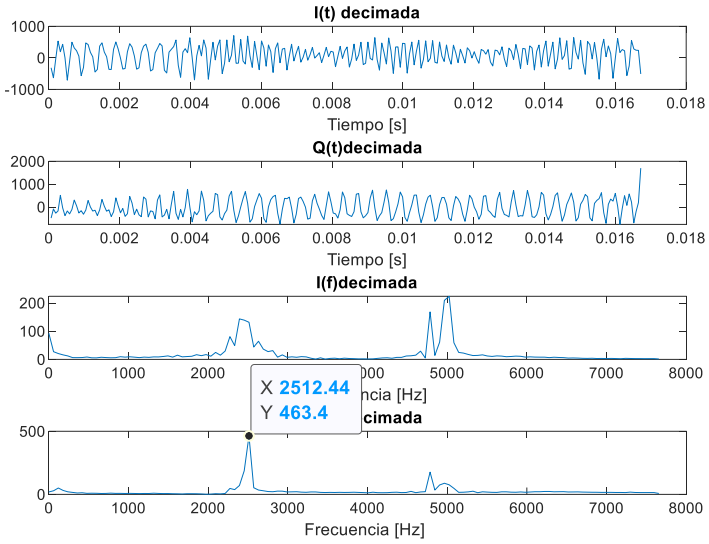


Figura 21. I y Q decimadas con M igual a 4

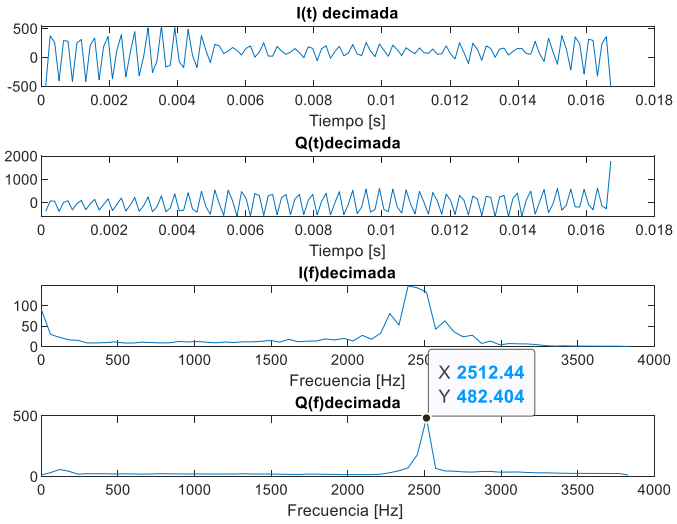


Figura 22. I y Q decimadas con M igual a 8

5.3.4 RECUPERACIÓN DE SEÑAL MEDIANTE CORDIC

La tercera etapa de procesado es la de la demodulación en frecuencia. Se ha decidido usar CORDIC porque ofrece un resultado parecido al de la arcotangente usando operaciones más sencillas y que consumen menos tiempo de computación.

El algoritmo CORDIC son las siglas de CoOrdinate Rotation Digital Computer. Este algoritmo es uno de los mútiples algoritmos usados para la demodulación en frecuencia y traduce un punto a lo largo un círculo unitario para implementar varias funciones trigonométricas y parabólicas que corresponden al mapeo entre coordenadas polares y rectangulares. Tiene dos modos de funcionamiento, pero en este proyecto solo se ha utilizado y programado el modo de vectorización. Este modo recibe las coordenadas en cuadratura de la señal y un número de iteraciones. Halla la diferencia de fase que hay entre las dos componentes rotando el ángulo y limitándolo entre -pi/2 y pi/2 para que no haya saltos bruscos en los resultados hasta que se completen todas las iteraciones.

Se analizan los resultados de este algoritmo en el capítulo 6.

5.3.5 DISEÑO DE FILTRO FIR

Como se ve en las transformadas de Fourier de la Figura 21, la señal resultante de aplicar el algoritmo de CORDIC contiene más componentes que la deseadas. Las señales moduladoras que se van a transmitir tienen como máximo una frecuencia de 3000 Hz y

utilizan las frecuencias bajas para fines que están fuera del rango de este proyecto. Por ello, es necesario pasar la señal por un filtro FIR paso banda que tenga una frecuencia de corte inferior de unos 300 Hz y una frecuencia de corte superior de 3000 Hz. Además, la frecuencia de muestreo de este filtro será la resultante del bloque diezmador del apartado 5.3.3 (15300 Hz).

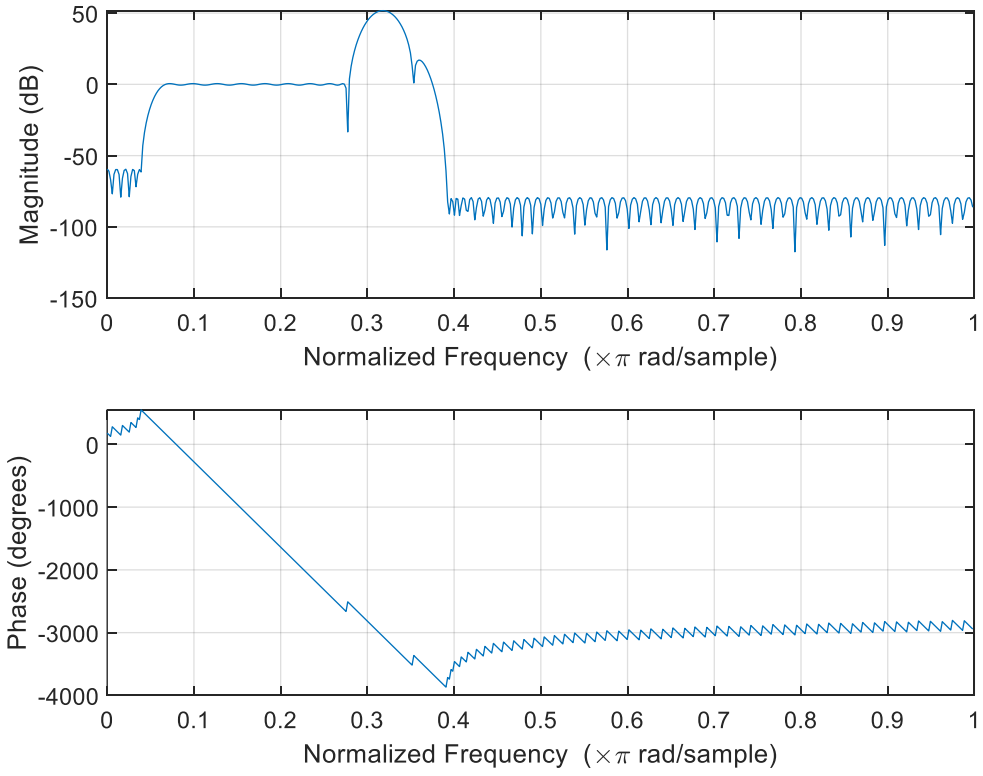


Figura 23. Respuesta en frecuencia filtro

5.4 CONVERSOR DIGITAL-ANALÓGICO

El último bloque del sistema consiste en el diseño de un conversor digital-analógico que reproduzca la señal a la salida de la placa. Esto es necesario si se quiere conectar a la salida

un altavoz para escuchar el audio o realizar algún tipo de filtrado analógico que podría ser necesario.

Para este apartado se ha utilizado DMA y un "timer" que activará la conversión según una frecuencia. Esta frecuencia se configurará para que sea la misma que la frecuencia de muestreo resultante en el bloque de procesamiento (entorno a los 15300 Hz). Para la configuración de esta frecuencia se procede a la modifación de dos parámetros que van a dividir a la frecuencia del reloj según la siguiente fórmula:

$$fst = \frac{Fclk}{(1+PSC)(1+ARR)} \qquad (10)$$

Donde fst es la frecuencia del timer, Fclk la frecuencia del reloj (80 MHz), PSC es el prescaler o la división del reloj y ARR es el Auto Reload Register o contador de periodo.

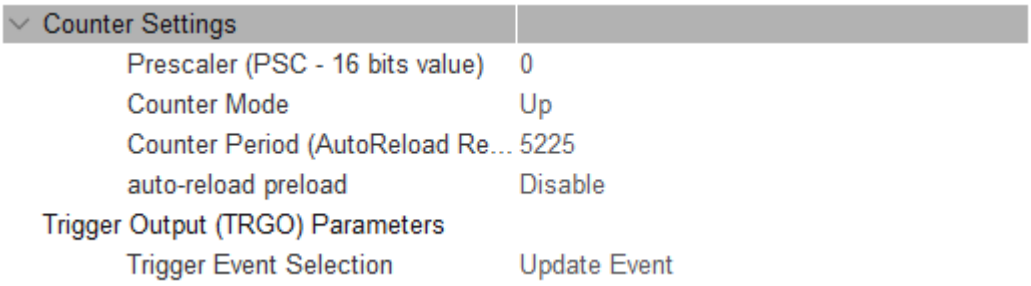


Figura 24. Configuración Timer 6

Se ha optado por poner el PSC a 0 y el ARR a 5225 lo que equivale a una frecuencia del timer de 15308 Hz.

Capítulo 6. ANÁLISIS DE RESULTADOS

En este capítulo se analiza la efectividad del algoritmo de CORDIC a la salida del bloque de procesado, el funcionamiento del DAC y se analizan los tiempos de ejecución de los bloques.

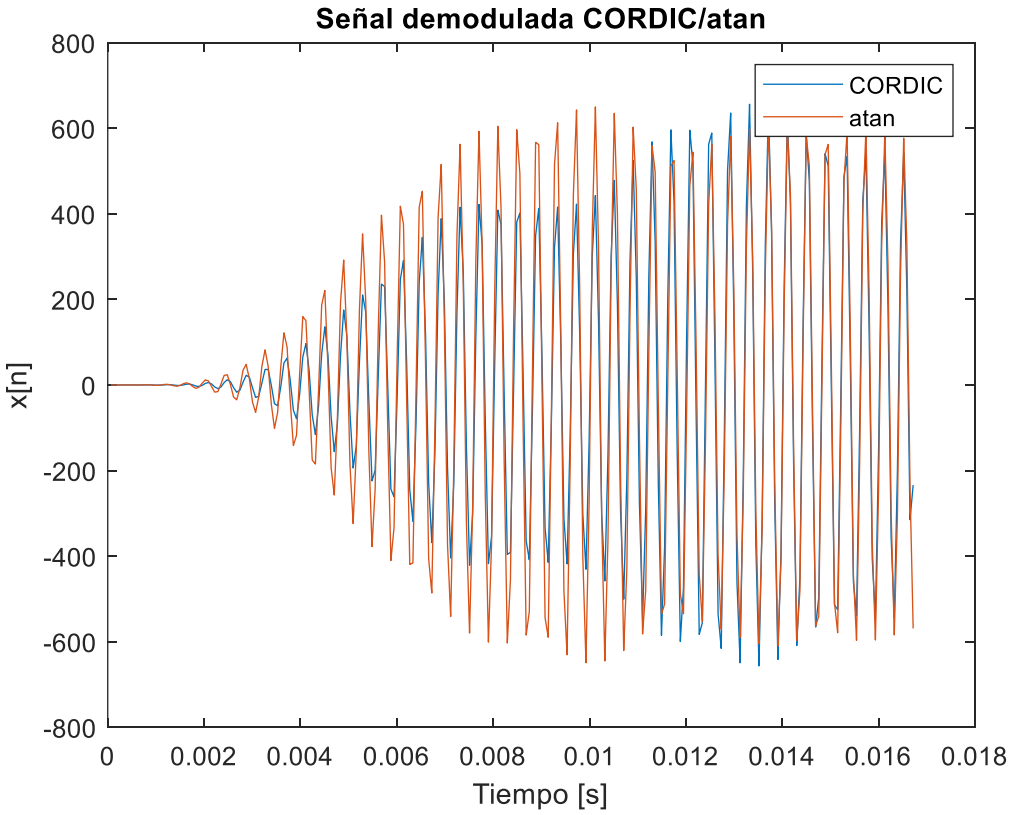


Figura 25. CORDIC y atan en tiempo

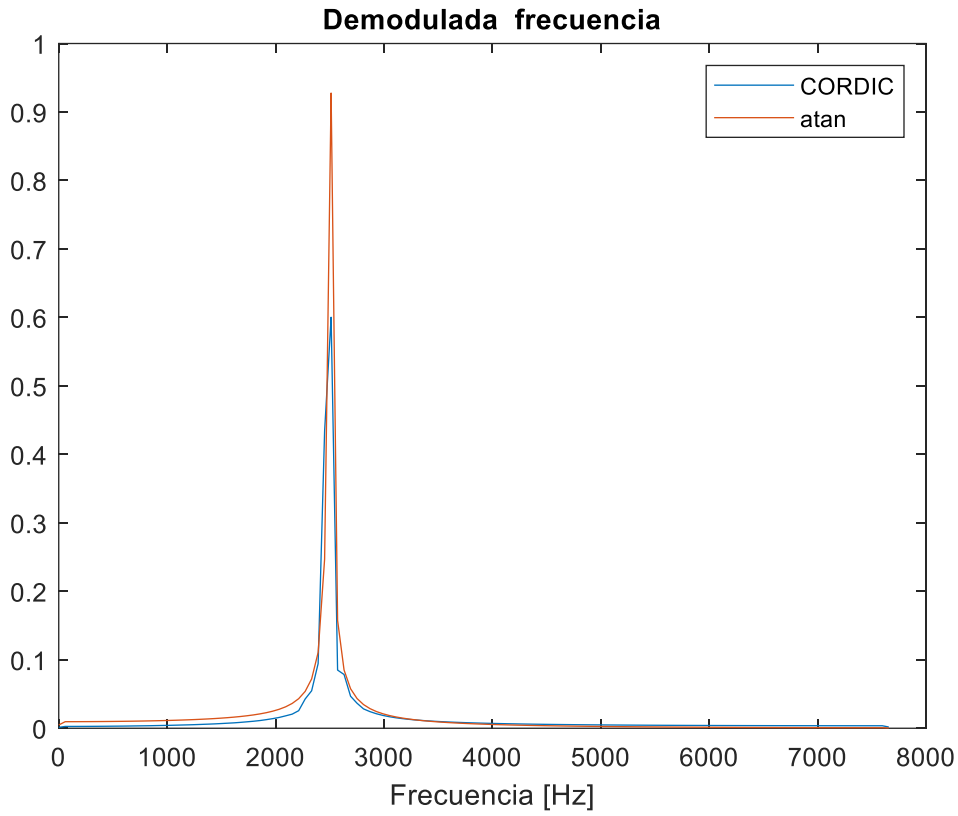


Figura 26. CORDIC y atan en frecuencia

Se puede observar como los algoritmos de CORDIC y de la arcotangente producen resultados similares, con la misma frecuencia y variando solo en la amplitud de la señal. Si es cierto, que ninguno de los dos algoritmos produce el tono puro de 2500 Hz en la escala de tiempos que es el que se buscaba en esta prueba. Esto puede ser debido a que el submuestreo hace que haya ruido de cuantificación en el resultado final.

Otra cosa que se observa es que al principio la señal siempre está a cero y esto es debido a que, al estar limitado el sistema en memoria, se han de tomar pocas muestras y la parte en de "arranque" del filtro siempre estará a cero. Se pierden unos 5 ms de señal lo que es un problema ya que la señal ocupa 16 ms y perder casi un tercio no se debería permitir.

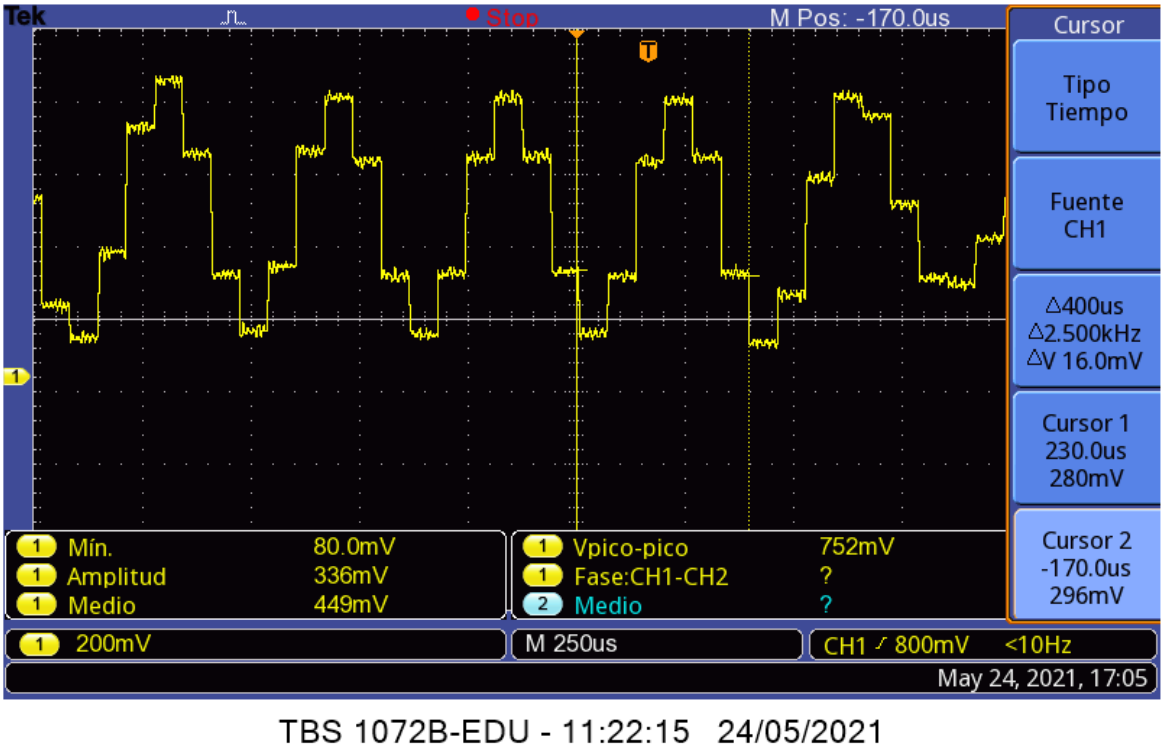


Figura 27. Salida DAC de tono a 2500 Hz

La Figura 27 muestra la señal a la salida del conversor digital-analógico. Se ve claramente una señal periódica de 2500 Hz de frecuencia. Esta señal no es continua debido a que no se utilizan suficientes muestras por periodo para representarla. Al final de la imagen, se observa un pequeña alliasing por parte de la señal que se corresponde con el siguiente buffer procesado producido por no procesar periodos o semiperiodos completos.

Por último, vamos a ver los tiempos de los distintos bloques del sistema receptor. La Figura 28 representa los tiempos en sincronía de todos los bloques: ADC, DAC y

procesado. El ADC comienza activado y va rellenando un buffer de 2048 muestras. Cada vez que se rellenan 1024 muestras, se activa y se realiza el procesado de estas muestras. La primera activación del DAC solo ocurre una vez se ha completado el procesado y, luego, se deja activado puesto que el sistema está en sincronía.

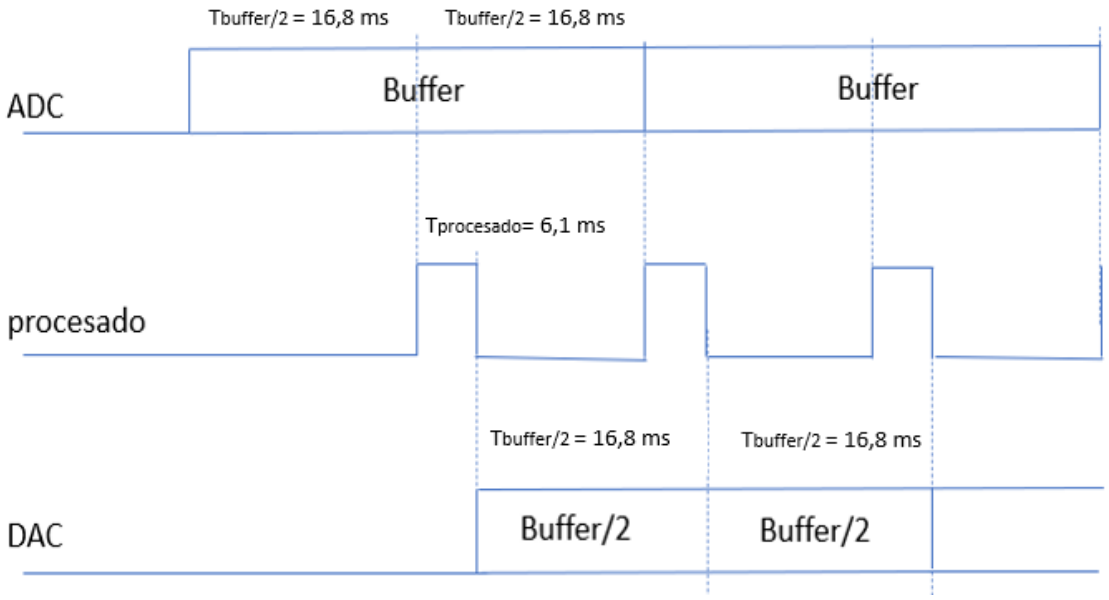


Figura 28. Sincronía bloques

Capítulo 7. CONCLUSIONES Y TRABAJOS FUTUROS

Durante la vida de este proyecto se ha diseñado un sistema receptor de audio dividido en tres bloques: ADC, procesamiento y DAC. Para el apartado del ADC se ha ajustado la frecuencia de muestreo para conseguir cumplir con los requerimientos del sistema de sincronía de tiempos y de submuestreo. En el procesamiento se ha logrado diseñar una cadena de bloques formado por dos osciladores, dos diezmadores, CORDIC y un filtro paso-banda que logre el procesado de la señal para conseguir recuperar la señal original modulada en FM. Por último, se ha configurado un timer para activar el DAC y realizar la conversión a analógico de la señal procesada. Todos estos bloques se han programado usando lenguaje en C en la placa NUCLEO L476RG con bajo consumo y con funciones de DSP.

Si es cierto, que los resultados producidos por CORDIC y comparados con el algoritmo de la arcotangente no son malos, pero dejan un poco insatisfechos. Esto se debe a que hay variaciones de amplitud en la señal recuperada y no funcione tan bien para frecuencias próximas a cero. Por ello, se recomienda en un futuro la prueba de otro tipo de sistemas como pueda ser el uso de un DPLL (digital phase locked loop).

Respecto a la implantación del sistema en cualquier satélite, se recomienda un estudio previo exhaustivo de los niveles de señal que pueda recibir la placa, teniendo en cuenta los niveles de referencia del sistema y las condiciones que hacen que cambien. También, será necesario incorporar un filtrado analógico a la salida del DAC para eliminar las componentes que queden por encima de los 3000 Hz. Como recomendación, si se quiere más calidad de la señal a la salida, usar una placa con mucha más memoria RAM y almacenar más muestras en el buffer del ADC para no perder tanta información debido al filtrado. Por último, se recomienda el uso de un reloj de 80 MHz o, en el caso de que se use otra placa de la gama con mayor o menor frecuencia de reloj, rehacer el estudio de las

frecuencias del ADC y cambiar las tablas de datos correspondientes a los osciladores y a los coeficientes de los filtros.

Capítulo 8. BIBLIOGRAFÍA

[1] DSP. Recuperado de https://www.electronicasi.com/wp-content/uploads/2013/04/dspElectronica-avanzada.pdf

[2] Edo, C. A. M. (s. f.). SDR - Radio definida por software. aeitm. https://www.aeitm.es/hemeroteca/tribuna-del-asociado/96-tribuna-del-asociado/238-sdr-radio-definida-por-software

[3] AMSAT. (2021). AMSAT – The Radio Amateur Satellite Corporation. Recuperado de https://www.amsat.org/.

[4] Pablo Turmero, Monografias.com. (1999). Ventajas de lo digital frente a lo analógico - Monografias.com. Monografias. https://www.monografias.com/trabajos102/presentacion-ventajas-lo-digital-frente-lo-analogico/presentacion-ventajas-lo-digital-frente-lo-analogico.shtml

[5] A. (s. f.). Modulación en frecuencia. scribbr. https://www.ecured.cu/Modulaci%C3%B3n_de_frecuencia

[6] Staff, E. (2011, 11 enero). DSP Tricks: Frequency demodulation algorithms. Embedded.com. https://www.embedded.com/dsp-tricks-frequency-demodulation-algorithms/

[7] STM32 32-bit Arm Cortex MCUs. (s. f.). STMicroelectronics. https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html

[8] Arm Ltd. (s. f.). Cortex-M4 –. Arm Developer. https://developer.arm.com/ip-products/processors/cortex-m/cortex-m4

[9] Gamez, J. L. (s. f.). *DIGITAL VS ANALÓGICO y Portabilidad vs mobilidad (1)*. Scribd. https://es.scribd.com/document/161042229/DIGITAL-VS-ANALOGICO-y-Portabilidad-vs-mobilidad-1

[10] Colaboradores de Wikipedia. (2021, 24 mayo). *Acceso directo a memoria*. Wikipedia, la enciclopedia libre.

https://es.wikipedia.org/wiki/Acceso_directo_a_memoria#:%7E:text=El%20acceso%20directo%20a%20memoria,de%20procesamiento%20(CPU)%20principal

[11] Sunny, S. (2018, 30 julio). *CORDIC Based FM Demodulator for Digital Telecommand Receiver*. IJERT. https://www.ijert.org/cordic-based-fm-demodulator-for-digital-telecommand-receiver

ANEXO A

CORDIC

```
#include "main.h"


#include <math.h>
#include <string.h>
#include <stdio.h>

#include "cordic2.h"
#include <stdio.h>




void vectorization_mode_2(float32_t * x, float32_t * y,uint32_t
n,uint32_t size, float32_t *angle)
{

        // Initialize angle to 0
        float32_t z1,z2,z3,z4,z5 ,x_rot,y_rot;
        float32_t K1=1,K2=1,K3=1,K4=1,K5=1 ;
        float32_t x_next=0,y_next=0;
        int blkCnt,nCnt;
        float32_t ciclo = 2*PI;
        float32_t ref = 4095/3.3;
        float32_t pi_medios = PI/2;
        #ifndef ARM_MATH_CM0_FAMILY

        float32_t d =0;

        float32_t inI1,inI2,inI3,inI4,inI5;  /* TEMPORARY INPUT OF I*/
        float32_t inQ1,inQ2,inQ3,inQ4,inQ5;   /* TEMPORARY INPUT OF Q*/

        blkCnt = size >> 2u;
        nCnt = 0u;
```

```
        while(blkCnt > 0u)
        {
                inI1 = *x;
                inI2 = *(x+1);
                inI3 = *(x+2);
                inI4 = *(x+3);
                inI5 = *(x+4);
                inQ1 = *y;
                inQ2 = *(y+1);
                inQ3 = *(y+2);
                inQ4 = *(y+3);
                inQ5 = *(y+4);
                z1=0;
                z2=0;
                z3=0;
                z4 =0;
                z5 = 0;

                /*Process first 4 values of I and Q*/

                //First
                if (inQ1 < 0)
                {
                        d = 1;
                }else
                {
                        d = -1;
                }

                x_rot = -1*d*inQ1;
                y_rot = d * inI1;
                z1 = z1 + d*pi_medios;
                inI1 = x_rot;
                inQ1 = y_rot;

                //Second
                if (inQ2 < 0)
                {
                        d = 1;
                }else
                {
                        d = -1;
```

```
}
x_rot = -d*inQ2;
y_rot = d * inI2;
z2 = z2+ d*pi_medios;
inI2 = x_rot;
inQ2 = y_rot;

//Third
if (inQ3 < 0)
{
        d = 1;
}else
{
        d = -1;
}
x_rot = -d*inQ3;
y_rot = d * inI3;
z3 = z3 + d*pi_medios;
inI3 = x_rot;
inQ3 = y_rot;

//Fourth
if (inQ4 < 0)
{
        d = 1;
}else
{
        d = -1;
}
x_rot = -d*inQ4;
y_rot = d * inI4;
z4 = z4 + d*pi_medios;
inI4 = x_rot;
inQ4 = y_rot;
if (inQ5 < 0)
{
        d = 1;
}else
{
        d = -1;
}
x_rot = -d*inQ5;
```

```
                y_rot = d * inI5;
                z5 = z5 + d*pi_medios;
                inI5 = x_rot;
                inQ5 = y_rot;

                while(nCnt <n)
                {
                        // First
                        if(inQ1<0)
                        {
                                d=1;
                        }else{
                                d=-1;
                        }
                        x_next =  inI1-  inQ1*d*powf(2,-nCnt);
                        y_next =  inQ1+ inI1*d*powf(2,-nCnt);
                        z1 =  z1 -   d * atanf(2^-nCnt);
                        inI1 = x_next;
                        inQ1 = y_next;
                        K1 = (float32_t) K1 * 1/sqrt(1 +powf(2,-2*nCnt));


                        //Second
                        if(inQ2<0)
                        {
                                d=1;
                        }else{
                                d=-1;
                        }
                        x_next = (float32_t) inI2-inQ2*d*powf(2,-nCnt);
                        y_next = (float32_t) inQ2+inI2*d*powf(2,-nCnt);
                        z2 =  z2 -  ((float32_t) d * atanf(powf(2,-
nCnt)));
                        inI2 = x_next;
                        inQ2 = y_next;
                        K2 = (float32_t) K2 * 1/sqrt(1 +powf(2,-2*nCnt));


                        //Third
                        if(inQ3<0)
                        {
                                d=1;
```

```
                }else{
                        d=-1;
                }
                x_next =  inI3-inQ3*d*powf(2,-nCnt);
                y_next = (float32_t) inQ3+inI3*d*powf(2,-nCnt);
                z3 =  z3 -  ((float32_t) d * atanf(2^-nCnt));
                inI3 = x_next;
                inQ3 = y_next;
                K3 = (float32_t) K3 * 1/sqrt(1 +powf(2,-2*nCnt));


                //Fourth
                if(inQ4<0)
                {
                        d=1;
                }else{
                        d=-1;
                }
                x_next = (float32_t) inI4-inQ4*d*powf(2,-nCnt);
                y_next = (float32_t) inQ4+inI4*d*powf(2,-nCnt);
                z4 =  z4 -  ((float32_t) d * atanf(2^-nCnt));
                inI4 = x_next;
                inQ4 = y_next;
                K4 = (float32_t) K4 * 1/sqrt(1 +powf(2,-2*nCnt));
                //Fifth
                if(inQ5<0)
                {
                        d=1;
                }else{
                        d=-1;
                }
                x_next = (float32_t) inI5-inQ5*d*powf(2,-nCnt);
                y_next = (float32_t) inQ5+inI5*d*powf(2,-nCnt);
                z5 =  z5 -  ((float32_t) d * atanf(2^-nCnt));
                inI5 = x_next;
                inQ5 = y_next;
                K5 = (float32_t) K5 * 1/sqrt(1 +powf(2,-2*nCnt));



                nCnt++;
        }
```

```
                //Finish iterations for 4 positions


                //Store results
                //First
                *x = (float32_t) K1 * inI1;
                *y = (float32_t) K1 * inQ1;


                //Second
                *(x+1) = (float32_t) K2 * inI2;
                *(y+1) = (float32_t) K2 * inQ2;


                //Third
                *(x+2) = (float32_t) K3 * inI3;
                *(y+2) = (float32_t) K3 * inQ3;

                //Second
                *(x+3) = (float32_t) K3 * inI4;
                *(y+3) = (float32_t) K3 * inQ4;



                *angle = (float32_t) (z1)*ref/ciclo;
                *(angle+1) = (float32_t) (z2)*ref/ciclo;
                *(angle+2) = (float32_t) (z3)*ref/ciclo;
                *(angle+3) = (float32_t) (z4)*ref/ciclo;

                /* Change directions of memory*/
                x = x + 4;
                y =y+4;
                angle = angle + 4;
                blkCnt--;
        }

        /*Check if not multiple of 4 */
        blkCnt = size % 0x4u;

#else

  /* Run the belonCnt code for Cortex-M0 */
```

```
  /* Initialize blkCnt nCntith number of samples */
  blkCnt = size;
  nCnt = 0u;

#endif /* #ifndef ARM_MATH_CM0_FAMILY */

/*If not multiple of 4 we compute next values 1 by 1*/

    while(blkCnt >0u)
    {
        if (*y < 0)
        {
                        d = 1;
        }else
                {
                d = -1;
        }
        x_rot = -d* (*y);
                y_rot = d * (*x);
                z1 = z1 + d*PI/2;
                *x = x_rot;
                *y = y_rot;
                nCnt = 0;
                while(nCnt<n)
                {
                        if(*x<0)
                        {
                                d=1;
                        }else{
                                d=-1;
                        }
                        x_next = (float32_t) *x-(*y)*d*powf(2,-nCnt);
                        y_next = (float32_t) inQ1+inI1*d*powf(2,-nCnt);
                        z1 =  z1 -  ((float32_t) d * atanf(2^-nCnt));
                        *x = x_next;
                        *y = y_next;
                        K1 = (float32_t) K1 * 1/sqrt(1 +powf(2,-2*nCnt));
                        nCnt++;
                }
                *x = (float32_t) (K1 * (*x));
                *x = *(x+1);
```

```
                *y = (float32_t) (K1 * *y);
                *y = *(y+1);
                *angle++ = (float32_t) z1*ref/ciclo;
    }
}

/*
 * cordic2.c
 *
 *  Created on: 25 abr. 2021
 *      Author: amart
 */
```

WHILE(1)

```
while (1)
  {

          if(flag_procesado==1)
          {
                  //Procesamos
//                //Multiplicamos por oscilador
                  arm_mult_f32(&lectura[0], &oscilador[0], &I[0], N/2);
                  arm_mult_f32(&lectura[0], &oscilador_pi_2[0], &Q[0],
N/2);



                  arm_fir_decimate_f32(&decimate_4,&I[0], &U[0], N/2);
                  arm_fir_decimate_f32(&decimate_4,&Q[0], &V[0], N/2);
```

```
//
//                //Demodulamos

                vectorization_mode_2(&U[0], &V[0], 8, N/8,  &angle[0]);

//                Filtramos
                arm_fir_f32(&fir_settings_passband, &angle[0], &angle[0],
N/8);

//                Preparamos para DAC

                arm_min_f32(&angle[0], N/8, &valor_min, &indice_min);
                valor_min = -1*valor_min + 120;
                arm_offset_f32(&angle[0], valor_min, &angle[0], N/8);

                for(int i=0;i<N/8;i++)
                {
                        angle_int[i] = (uint32_t) angle[i];
                }
//                //Conversor DAC
                  flag_procesado =0;
                  flag_dac =1;

          }


  }
```

RELOJES

```
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  RCC_PeriphCLKInitTypeDef PeriphClkInit = {0};
```

```
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLM = 1;
  RCC_OscInitStruct.PLL.PLLN = 10;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV7;
  RCC_OscInitStruct.PLL.PLLQ = RCC_PLLQ_DIV2;
  RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV2;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }

  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_4) != HAL_OK)
  {
    Error_Handler();
  }
  PeriphClkInit.PeriphClockSelection =
RCC_PERIPHCLK_USART2|RCC_PERIPHCLK_ADC;
  PeriphClkInit.Usart2ClockSelection = RCC_USART2CLKSOURCE_PCLK1;
  PeriphClkInit.AdcClockSelection = RCC_ADCCLKSOURCE_SYSCLK;
  if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
  {
    Error_Handler();
  }

  if (HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1) !=
HAL_OK)
  {
    Error_Handler();
  }
}
```

ADC

```
static void MX_ADC1_Init(void)
{



  ADC_MultiModeTypeDef multimode = {0};
  ADC_ChannelConfTypeDef sConfig = {0};

  hadc1.Instance = ADC1;
  hadc1.Init.ClockPrescaler = ADC_CLOCK_ASYNC_DIV2;
  hadc1.Init.Resolution = ADC_RESOLUTION_12B;
  hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc1.Init.ScanConvMode = ADC_SCAN_DISABLE;
  hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
  hadc1.Init.LowPowerAutoWait = DISABLE;
  hadc1.Init.ContinuousConvMode = ENABLE;
  hadc1.Init.NbrOfConversion = 1;
  hadc1.Init.DiscontinuousConvMode = DISABLE;
  hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
  hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
  hadc1.Init.DMAContinuousRequests = ENABLE;
  hadc1.Init.Overrun = ADC_OVR_DATA_PRESERVED;
  hadc1.Init.OversamplingMode = DISABLE;
  if (HAL_ADC_Init(&hadc1) != HAL_OK)
  {
    Error_Handler();
  }

  multimode.Mode = ADC_MODE_INDEPENDENT;
  if (HAL_ADCEx_MultiModeConfigChannel(&hadc1, &multimode) != HAL_OK)
  {
    Error_Handler();
  }

  sConfig.Channel = ADC_CHANNEL_5;
  sConfig.Rank = ADC_REGULAR_RANK_1;
  sConfig.SamplingTime = ADC_SAMPLETIME_640CYCLES_5;
  sConfig.SingleDiff = ADC_SINGLE_ENDED;
  sConfig.OffsetNumber = ADC_OFFSET_NONE;
  sConfig.Offset = 0;
```

```
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }

}
```

DAC

```
static void MX_DAC1_Init(void)
{

  DAC_ChannelConfTypeDef sConfig = {0};


  hdac1.Instance = DAC1;
  if (HAL_DAC_Init(&hdac1) != HAL_OK)
  {
    Error_Handler();
  }
  /** DAC channel OUT1 config
  */
  sConfig.DAC_SampleAndHold = DAC_SAMPLEANDHOLD_DISABLE;
  sConfig.DAC_Trigger = DAC_TRIGGER_T6_TRGO;
  sConfig.DAC_OutputBuffer = DAC_OUTPUTBUFFER_ENABLE;
  sConfig.DAC_ConnectOnChipPeripheral = DAC_CHIPCONNECT_DISABLE;
  sConfig.DAC_UserTrimming = DAC_TRIMMING_FACTORY;
  if (HAL_DAC_ConfigChannel(&hdac1, &sConfig, DAC_CHANNEL_1) != HAL_OK)
  {
    Error_Handler();
  }

}
```

ANEXO B

Este anexo hace referencia a la relación con los Objetivos de Desarrollo Sostenible que involucran a este trabajo.

Este trabajo esta asociado con el ODS número 7 que se corresponde con el de “energía asequible y no contaminante”. No utiliza energías no contaminantes como tal, pero propone un sistema de un satélite con mucho menos consumo de energía lo que contribuye a que tanto países desarrollados como los que no lo son puedan acceder a esta tecnología.

Por otro lado, contribuye a la “educación de calidad” que se corresponde con el ODS número 4. La lectura de este documento puede servir de apoyo a la hora de llevar a cabo los estudios ya que es una aplicación de muchos conceptos importantes estudiados durante la etapa universitaria de muchas carreras.