Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería en Tecnologías de la Telecomunicación

Trabajo Fin de Grado

Desarrollo y validación de una plataforma de pruebas para sistema LPS basado en infrarrojo

Autor: Álvaro de la Llana Calvo
Directores: José Luis Lázaro Galilea
David Salido Monzú

2014

UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería en Tecnologías de la Telecomunicación

Trabajo Fin de Grado

Desarrollo y validación de una plataforma de pruebas para sistema LPS basado en infrarrojo

Autor: Álvaro de la Llana Calvo

Directores: José Luis Lázaro Galilea

David Salido Monzú

Tribunal:

Presidente: Alfredo Gardel Vicente

Vocal 1º: Ernesto Martín Gorostiza

Vocal 2º: José Luis Lázaro Galilea

Calificación: .....

Fecha: .....

Resumen

En el presente TFG se expone el desarrollo y validación de una plataforma de pruebas ejecutada sobre Matlab, que emularía el canal de propagación existente entre emisor y receptor para un sistema LPS basado en infrarrojo. Previo a la implementación se realiza una exposición del sistema LPS para el cual se realiza la plataforma así como el tipo de señal y fuentes de error presentes en el sistema que se desean emular. Así mismo, se realizaran una serie de pruebas que permiten demostrar la validez de la plataforma de pruebas propuesta.

Palabras clave: Sistema emulador, sistema de posicionamiento local, señal IR, procesado de señal.

Abstract

This bachelor thesis explains the development and validation of a test platform, running on Matlab, to emulate the propagation channel between an emitter and a receiver for an infrared-based local positioning system. Before the system implementation, the LPS contextualizing this work is explained, together with the signal of interest and main error sources in the system. In addition, test to validate the platform performance are carried out.

Keywords: Emulation system, local positioning system, IR signal, signal processing.

Resumen extendido

En los últimos años el problema del posicionamiento en exteriores ha sido solventando con buenos resultados por los sistemas globales de navegación por satélite (GNSS). La evolución natural es el posicionamiento en espacios interiores, donde los GNSS presenta distintas carencias como puede ser la falta de precisión debida principalmente a la atenuación y el multicamino. Por esta razón se están desarrollando alternativas denominadas sistemas de posicionamiento local (LPS), que obtienen buenos resultados en estancias interiores. Las principales tecnologías estudiadas actualmente se basan en visión computacional, radiofrecuencia, ultrasonidos, sistemas inerciales e sistemas ópticos.

En este caso, se va a centrar la atención en un LPS, basado en infrarrojos. El emisor situado a bordo de un robot móvil emite una señal IR modulada en AM a frecuencia constante. Los receptores situados en puntos conocidos sobre la célula de posicionamiento, estiman la posición del robot móvil a partir de medidas diferenciales de desfase de la señal que reciben. El principal problema de este sistema es el error por multicamino, lo que provoca que la precisión disminuya. Por tanto se está investigando en una propuesta de un nuevo sistema de medida que reduzca el efecto que produce el multicamino en la precisión. Para lo cual se enviará una señal sinusoidal modulada con un código pseudoaleatorio (PRN), con el propósito de, tras el proceso de demodulación de la señal recibida, maximizar la recuperación de potencia de la componente correspondiente al camino directo, consiguiendo así reducir en todo lo posible el efecto producido por el multicamino.

El objetivo principal de este TFG es el desarrollo de una plataforma de pruebas que permita validar en lo posible la propuesta que se encuentra en estudio. La plataforma de pruebas permitirá emular el comportamiento del canal existente entre emisor y receptor. Con esto se consigue obtener una señal 'realista' que emularía la señal que reciben los receptores del sistema real con la que poder validar el sistema de posicionamiento en estudio. La plataforma de pruebas permite emular canales con distintos efectos adversos:

- Ruido blanco gaussiano con un distintos niveles de potencia.
- Errores dinámicos debidos al movimiento del emisor y a la falta de sincronismo existente entre emisor y los receptores.

- Multicaminos a consecuencia de las reflexiones de la señal en el entorno.

El emulador está compuesto por 3 dispositivos que realizan la función de cada una de las partes del sistema LPS:

- PC: Emularía la memoria interna del emisor donde se guardaría la señal que se desea enviar. Generaría la señal deseada en Matlab y se la enviaría al AFG. Una vez que la señal pasa por el AFG y osciloscopio, el PC vuelve a recibir la señal y será el encargado de realizar un procesado de la señal recibida para ajustarla a las características deseadas.
- AFG: Se comportaría como la etapa de conversión digital-analógica dentro del emisor. Recibiría los datos del PC y los enviaría por una de sus salidas hacia el osciloscopio.
- Osciloscopio: Emularía al receptor del sistema LPS dentro del cual se desarrollaría la conversión analógico digital de los datos que recibe. También será el encargado de mandar los datos al PC.

La plataforma de pruebas se ejecuta sobre Matlab y dispone de una interfaz gráfica que facilita su uso. Desde la interfaz gráfica se pueden configurar los parámetros de cada uno de los tres posibles efectos más importantes que puede presentar un enlace óptico:

- Ruido: Permite configurar el rango de SNR que se desean emular.
- Errores dinámicos: Permite emular el error dinámico caracterizado por un retardo expresado en s/s. Este retardo relaciona los segundos que se retardaría la señal enviada frente a la ideal en un tiempo de un segundo. Este efecto se consigue aplicando a la señal ideal una modulación en fase (modulación PM).
- Multicamino: La plataforma de pruebas está configurada para emular el efecto de hasta 10 multicaminos. Cada uno de los multicaminos queda caracterizado por un valor de amplitud que se indica en tanto por uno sobre la amplitud de la señal de camino directo o linea de visión (LOS) y por un retardo en segundos sobre esa misma señal LOS.

Cada uno de estos posibles efectos adversos se puede usar por separado o combinándolos ente ellos.

Finalmente se muestra un apartado de resultados y validación donde se realizan distintas pruebas para comprobar y demostrar el correcto funcionamiento del sistema emulador.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En los últimos años, los sistemas globales de navegación por satélite (GNSS) han experimentado una enorme consolidación. Son capaces, en espacios exteriores, de obtener una posición con precisiones de pocos metros y con una cobertura inalcanzable por cualquier otro sistema de posicionamiento. Esto se consigue gracias a su infraestructura formada por una constelación de satélites dedicados. Los sistema GNSS han conseguido una gran implantación existiendo numerosas aplicaciones que hacen uso de este sistema.

La evolución natural del posicionamiento en exteriores, es el posicionamiento en espacios interiores donde GNSS presenta distintas carencias. Entre las más importantes se encuentra la falta de precisión de los sistemas de posicionamiento basados en satélites, obteniendo en el mejor de los casos una precisiones de metros, valor insuficiente para alguna de las aplicaciones en desarrollo, así como problemas de atenuación y multicamino cuando se usan estos sistemas en entornos cerrados. Por esta razón se están desarrollando alternativas denominadas sistemas de posicionamiento local (LPS), que sean capaces de trabajar en entornos interiores obteniendo altas precisiones.

Existen distintas alternativas en investigación basadas principalmente en cámaras, radiofrecuencia como puede ser GSM o wifi, ultrasonidos e infrarrojos. Cabe destacar que ninguna de las alternativas expuestas soluciona todos los problemas por lo que el uso de una u otra alternativa dependerá de los requerimientos de la aplicación.

A continuación se van a describir a grandes rasgos los diferentes sistemas de posicionamientos en interiores:

- Visión: Se basan principalmente en el reconocimiento de un punto en varias cámaras situadas en lugares conocidos de un espacio. Conocida la posición de las cámaras y donde se encuentra el punto en cada imagen se puede saber su posición. Para facilitar el reconocimiento existen sistemas que hacen uso de etiquetas, marcas que ayudan a la hora de identificar el objetivo con la cámara. Actualmente existen sistemas LPS que hacen uso de

cámaras de profundidad. La ventaja de este sistema respecto al resto es la buena precisión que se puede llegar a alcanzar. Pero en contra son sistemas que requieren una buena calibración y han de ser usados en entornos donde la iluminación sea la adecuada.

- Ultrasonidos (US): Se basan en la medida de tiempo de vuelo de una señal de ultrasonidos. La señal puede ser enviada por el objeto a posicionar o por emisores situados en lugares conocidos. En ambos casos el principio de funcionamiento es el mismo, se mide el tiempo de vuelo de la señal US, y a partir de la velocidad de propagación del sonido se puede saber la distancia a la que se encuentra el objeto. Si se utiliza más de un receptor se puede saber la posición del objetivo. La señal usada puede ser simplemente un pulso o un código que se correlará en el receptor. Son sistemas en general precisos pero no son apropiados para usos en entornos con contaminación acústica. Se ven afectados por el efecto doppler y por la variación de la velocidad de propagación del sonido debido a los cambios en la temperatura ambiente y la humedad.

- Radiofrecuencia ad-hoc: Existen principalmente dos sistemas:

  - UWB: Se usa la medida de tiempo de vuelo de un pulso o código. A partir de dicha medida se puede conocer la distancia y por tanto la posición del objetivo. Son sistemas bastante precisos pero son caros y su precisión se ve afectada por el multicamino.
  - RF ad-hoc: Estiman la posición a partir de la intensidad de señal que reciben de distintos emisores. No son en general sistemas muy precisos pero con una gran inversión se pueden conseguir buenas precisiones.

- RF reciclados: Hacen uso de infraestructuras y redes ya existentes como puede ser GSM, 4G, WLAN(wifi) para estimar la posición a partir de la intensidad de señal. Son sistemas poco precisos pero muy baratos.

- IMU (Inertial Measurement Units): Los sistemas basados en IMU se basan en el uso de acelerómetros, brújulas y magnetómetros. Conocida la posición inicial a partir de los datos de velocidad y dirección se puede estimar la posición en cada instante de tiempo. Con la información del magnetómetro se puede conocer la altura, información útil si se quiere posicionar dentro de un edifico con varias plantas. Son sistemas en general baratos y flexibles. El principal inconveniente de los sistemas basados en IMU es la poca precisión debida a los errores que se van acumulando. Requiere también sistemas para inicializarse y reinicializarse.

- Telémetros ópticos (basados en Infrarrojos (IR)). Usa señales ópticas para estimar la posición. Se habla con más detalle de estos sistemas a continuación.

Los telémetros ópticos han sido los sistemas usados tradicionalmente para la telemetría de precisión. Entre las técnicas usadas en estos sistemas LPS se encuentran:

- Interferometría: Medida de distancia ultra preciso. Requiere montaje muy estable por lo que no es válido para los sistemas de posicionamiento local usados habitualmente.
- TOF. Existen dos alternativas:
  - Medida de tiempo de vuelo de un pulso, con la que se pueden conseguir mayores alcances.
  - Medidas de fase con las que se puede conseguir precisiones más altas.

En el caso del proyecto que nos ocupa, se va a centrar la atención en un LPS (sistema de posicionamiento local), basado en infrarrojos cuya estructura general se puede ver en la figura 1.1.

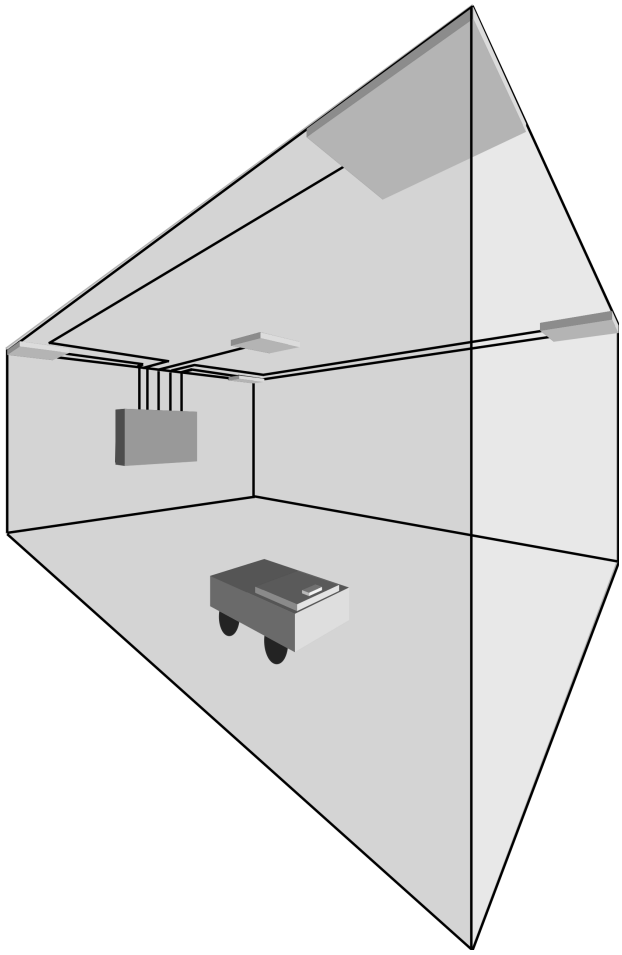

Figura 1.1: Estructura de la celda de posicionamiento del LPS basado en IR.

El emisor situado a bordo de un robot móvil emite una señal IR modulada en AM a frecuencia constante. Los receptores situados en puntos conocidos sobre la célula de posicionamiento

estiman la posición del robot móvil a partir de medidas diferenciales de desfase de la señal que reciben [1] [2].

Hasta el momento se ha demostrado la viabilidad de uso infrarrojos para conseguir, en espacios interiores, precisiones bastante altas del orden de pocos cm. El principal problema de este sistema es el error por multicamino, lo que provoca que la precisión disminuya. Se está investigando en una propuesta de un nuevo sistema de medida [3] que reduzca el efecto que produce el multicamino en la precisión. Para lo cual se enviará una señal sinusoidal modulada con un código pseudoaleatorio (PRN), con el propósito de, tras el proceso de demodulación de la señal recibida, maximizar la recuperación de potencia de la componente correspondiente al camino directo, consiguiendo así reducir en todo lo posible el efecto producido por el multicamino.

Dicha propuesta ha sido analizada teóricamente y, los resultados esperados han sido contrastados con simulaciones usando señales sintéticas. Para una validación más realista de los resultados obtenidos, previa a la implementación de un prototipo, es necesario el desarrollo de una plataforma de pruebas (set-up) que permita comprobar el efecto de trabajar con señales analógicas y un sistema emisor-receptor asíncrono.

El objetivo de la realización de este set-up de emulación es poder realizar pruebas de validación de esta propuesta, principalmente enfocadas a comprobar los resultados teóricos y simulados de precisión en presencia de ruido y errores de multicamino. El set-up deberá permitir emular la generación de señal analógica por parte del emisor y el proceso de digitalización por parte del receptor.

Por tanto el objetivo de este TFG es el desarrollo e implementación del mencionado set-up de validación del sistema LPS en estudio.

Así el objetivo general es el desarrollo y validación del set-up de emulación del enlace emisor-receptor del sistema de medida de distancia basado en infrarrojos. Se pretende generar, emitir y obtener una señal lo más parecida posible a la señal que recibirían los receptores con la que poder emular y validar el sistema LPS propuesto anteriormente.

Los objetivos específicos del desarrollo del sistema emulador o set-up son:

- Generación de señales en PC.
- Transferencia de señales y configuración desde el PC a generador de funciones (AFG).
- Emisión de las señales desde el AFG y recepción en osciloscopio.
- Transferencia de señal del osciloscopio al PC.
- Readaptación en el PC de la señal recibida del osciloscopio.

En la validación del sistema se llevará a cabo:

- Comprobación de que la señal digital obtenida al final del proceso es la esperada, teniendo en cuenta los factores por los que ha sido afectada (atenuación, errores de frecuencia, discretización, adición de ruido, . . .).

- Aplicación de la señal recibida como señal de entrada en el sistema digital de procesado de señal, que implementa la arquitectura de medida en los receptores. Esto permitirá validar los resultados obtenidos teóricamente en el análisis de la propuesta de sistema de medida con mitigación de multicamino.

Capítulo 2

Background

2.1 Sistema Propuesto

En esta sección se va a describir el sistema de posicionamiento local para el cual se va a desarrollar el set-up de validación objetivo principal de este TFG.

La estructura del sistema de posicionamiento en estudio se muestra en la figura 2.1. El sistema permite estimar la posición de un robot móvil mediante medidas diferenciales de fase de la señal recibida por los receptores.

El sistema se puede dividir en tres partes diferenciadas:

- El emisor: situado en un robot móvil emitirá una señal sinusoidal infrarroja modulada por un código PRN consiguiendo una señal de espectro ensanchado que se comportará mejor frente al ruido. El objetivo del sistema es conocer en cada instante de tiempo la posición del robot móvil.
- El enlace óptico entre emisor y receptor: hará que la señal que reciba el receptor, llamada $s(t)$ sea la suma de la señal de camino directo (LOS) más la suma de aportaciones indeseadas como pueden ser ruido blanco gaussiano, multicamino, retardo, diferencias de fase debido a la falta de sincronización, etc.
- Los receptores: situados en puntos conocidos sobre la célula de posicionamiento estimarán la posición del robot móvil a partir de medidas diferenciales de fase de la señal que reciben. Para conseguir estimar la posición del emisor la señal $s(t)$ pasará por distintas etapas que serán mencionadas posteriormente.

Las distintas etapas del sistema de posicionamiento local situadas en los receptores son:

- Coarse acquisition: Provee una estimación del retardo de la señal que recibe el receptor $s(t)$. Esta estimación se consigue mediante la correlación de la señal $s(t)$ con la señal

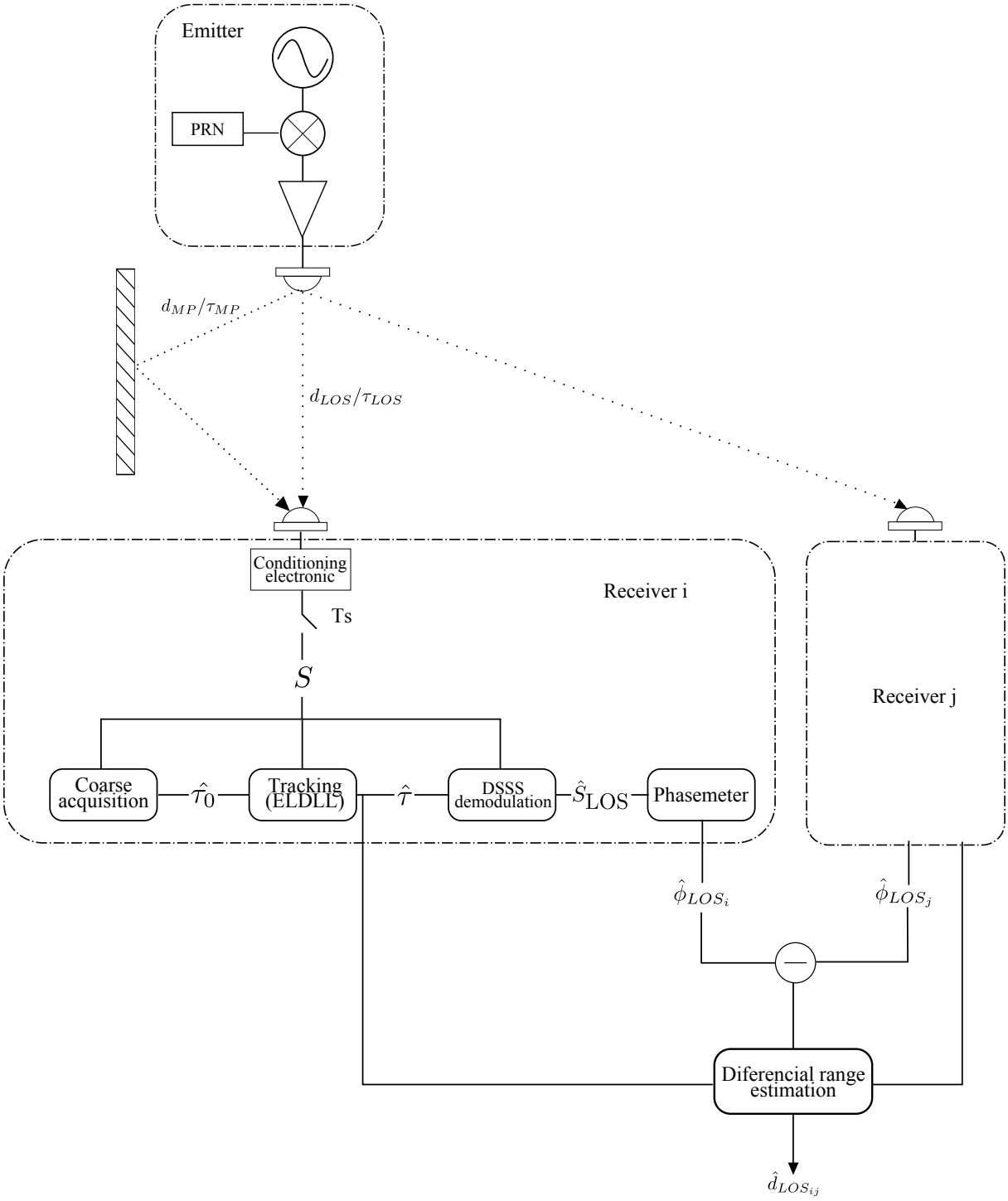

Figura 2.1: Estructura del sistema de posicionamiento.

original retardada con varios retardos entre todos los posibles. El resultado de dicha correlación es una curva que presenta un máximo cuando el retardo introducido a la señal original es el más similar al retardo de $s(t)$ y valores en torno a cero cuando los retardos introducidos a la señal original y el de $s(t)$ son distintos. Por tanto mediante esta operación es posible conseguir la estimación del retardo de $s(t)$. Esta operación es lenta por lo que la estimación del retardo ofrecido por la etapa de coarse acquisition solo se usará para inicializar el valor de $\hat{\tau}_0$ que será introducido a la etapa de tracking.

- Tracking: Provee un estimación de retardo $\hat{\tau}$ que será introducida a la etapa de DSSS demodulation. El funcionamiento de dicha etapa será explicado con más detalle en la sección 2.2.

- DSSS demodulation: Demodula la señal $s(t)$ con el código de ensanchado original $c(t)$ retardado el valor de la estimación de retardo $\hat{\tau}$ procedente de la etapa de tracking. Provee a la etapa siguiente una señal $\hat{S}_{\mathrm{LOS}}$ cuya potencia es debida principalmente al camino directo entre el emisor y el receptor y una parte muy pequeña a la potencia de los multicaminos.

- Phasemeter. Calcula la fase $\hat{\phi}_{\mathrm{LOS}}$ a partir de la señal $\hat{S}_{\mathrm{LOS}}$.

2.2 Etapa de tracking

La etapa de tracking es la encargada de proveer continuamente una estimación del retardo $\hat{\tau}$ de la señal de entrada $s(t)$. Esta estimación será usada tanto en la etapa de demodulación DSSS como en la propia etapa de tracking. En la etapa de demodulación DSSS se usará $\hat{\tau}$ para generar una señal de ensanchado con ese retardo y así conseguir que la señal demodulada tenga la máxima potencia.

El esquema general de la etapa de tracking se muestra en la figura 2.2.

Una vez que la estimación de retardo inicial $\hat{\tau}_0$ procedente de la etapa de Coarse Adquisition está disponible, se genera una secuencia de código ensanchado con ese retardo $m(t-\hat{\tau})$. Esa señal se usa para crear otras dos señales. La primera señal llamada early $m_E(t,\hat{\tau})$ será la señal $m(t-\hat{\tau})$ adelantada medio periodo de chip y la segunda señal será la señal late $m_E(t,\hat{\tau})$ retardada medio periodo de chip. Ambas señales se correlan por separado con la señal $s(t)$. El resultado de ambas correlaciones se resta obteniendo a la salida el discriminador $D_\Delta(t,\delta)$. Debido a la simetría de la correlación, si el retardo de $s(t)$ coincide con la estimación $\hat{\tau}$ el discriminador será cero. Cualquier diferencia de retardo entre $s(t)$ y $\hat{\tau}$ producirá un valor positivo o negativo del discriminador dependiendo del sentido del retardo. El discriminador $D_\Delta(t,\delta)$ será usado para calcular $\hat{\tau}$. El valor de retardo estimado será realimentado a la entrada para intentar conseguir que el discriminador sea el máximo tiempo posible 0.

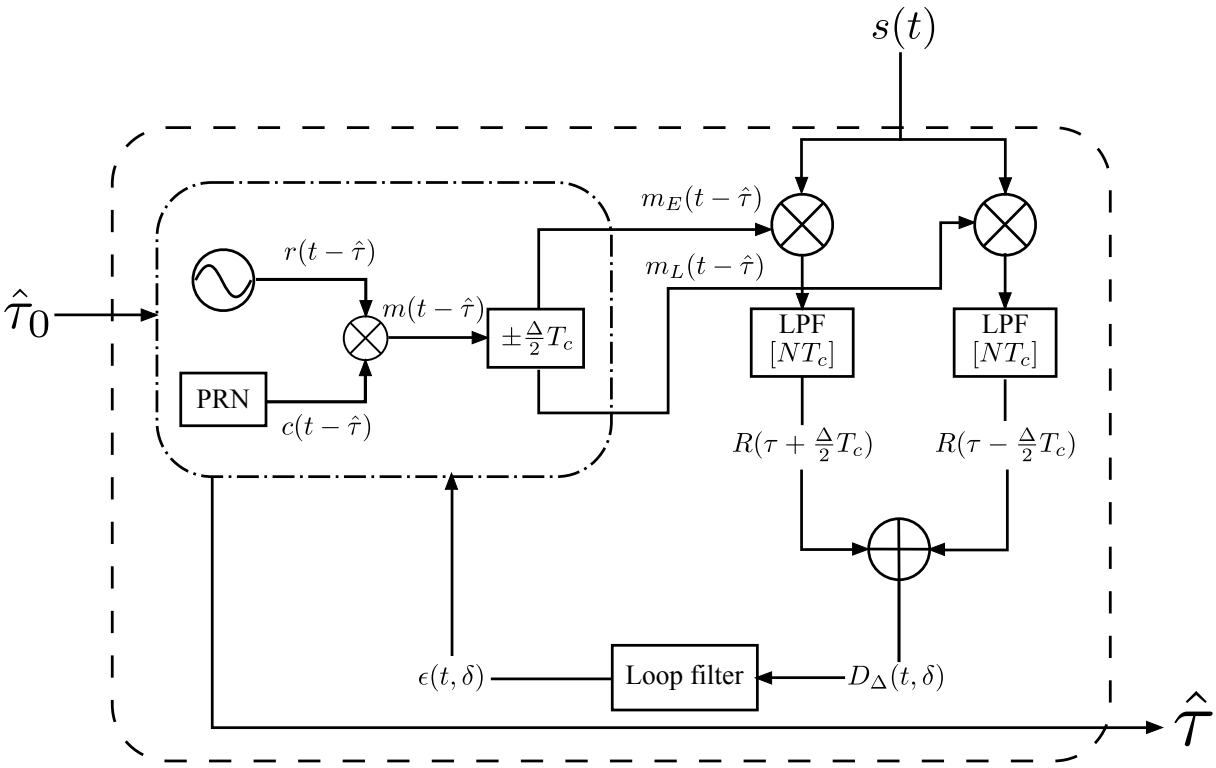

Figura 2.2: Diagrama de la etapa de tracking.

Las precisiones obtenidas en la etapa de tracking se ven afectadas por la presencia de fuentes de error, como puede ser el ruido o los errores dinámicos, de los cuales se habla con más detalle en la sección 2.4.

2.3 Estructura de la señal

2.3.1 Señal en el emisor

La señal utilizada en el sistema y que emularía la señal enviada por el emisor se trata de un tono $r(t)$ modulado con un código de espectro ensanchado $c(t)$. A continuación se describirá las componentes de la señal paso a paso.

La señal emitida $m(t)$ con una potencia P será:

$$m(t) = \sqrt{2P}r(t)c(t) \tag{2.1}$$

donde la señal $r(t)$ representa un tono con una frecuencia f_c y una fase respecto a la secuencia de ensanchado de ϕ_0:

$$r(t) = \sin(2\pi f_c t + \phi_0) \tag{2.2}$$

La secuencia de ensanchado $c(t)$ se define como:

$$c(t) = \sum_{n=0}^{\infty} p(t - nT_f) \tag{2.3}$$

donde la señal $p(t)$ es el código PRN repetido periódicamente con periodo $T_f = NT_c$, siendo N el número de chips del código PRN elegido y $T_c = 1/f_c$ el periodo de chip.

El código PRN se define como:

$$p(t) = \sum_{n=0}^{\infty} C_k \prod \left(\frac{t - kT_c}{T_c} \right) \tag{2.4}$$

donde $C_k \in \{\pm 1\}$ es el valor de cada chip y $\prod(t)$ es:

$$\begin{cases} 0 \text{ if } t < 0\,[s] \\ 1 \text{ if } 0 \leq t \leq 1\,[s] \\ 0 \text{ if } t > 1\,[s] \end{cases}$$

Los parámetros que caracterizan el código PRN así como el resto de señales del emulador son configurables. Únicamente se tiene que tener en cuenta varios parámetros que deben seguir las restricciones siguientes:

- La frecuencia de chip ha de ser igual que la frecuencia del seno. Esto es así para conseguir que cada chip module un periodo entero del seno.
- Tanto la frecuencia de chip como la frecuencia del seno deben ser iguales a la mitad del ancho de banda BW. Con esto se consigue que el filtro solo deje pasar el lóbulo principal de la densidad espectral de potencia de la señal $m(t)$ que es donde se concentrar la mayor parte de la potencia de señal.
- La fase del seno ϕ_0 tiene que ser 0 grados con respecto al código, con lo que se consigue que las transiciones de símbolo sucedan en el paso del seno por 0. Con lo cual se reduce la distorsión por la limitación de BW.

El emulador se podrá configurar con 4 frecuencias de chip distintas. Esas frecuencias son: 10MHz, 25MHz, 50MHz y 100MHz.

En el sistema real la señal $m(t)$ producto de la modulación del tono con la señal de espectro ensanchado será enviada de forma continua.

En la figura 2.3 se puede ver la representación del código utilizado:

Este código modulará un tono con la potencia requerida en cada caso, en la figura 2.4 se muestra un ejemplo de tono con potencia unidad.

La señal producto de dicha modulación $m(t)$ que emulará la señal que envía el emisor es la mostrada en la figura 2.5.

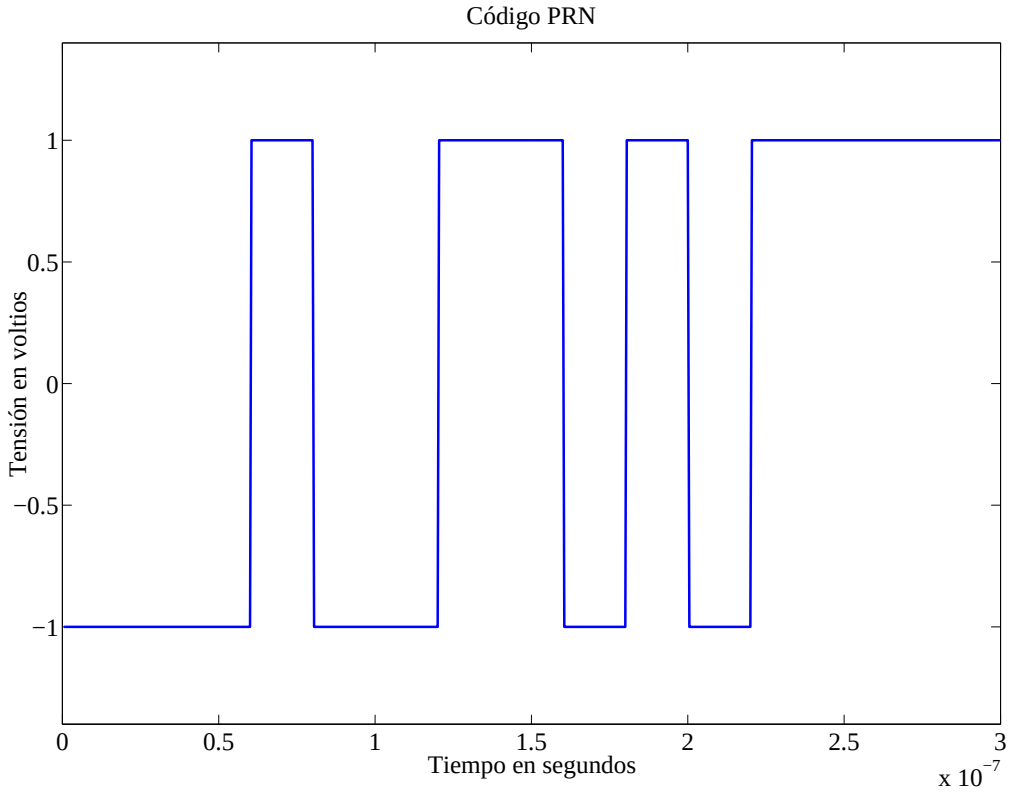

Figura 2.3: Código PRN.

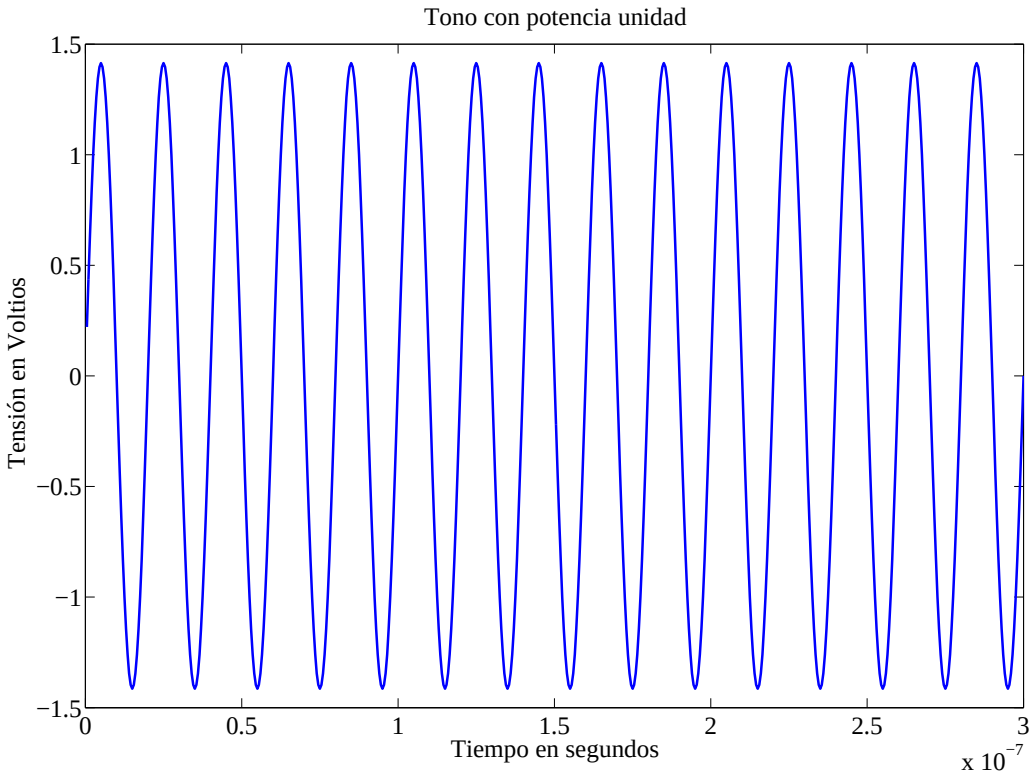

Figura 2.4: Tono con potencia unidad.

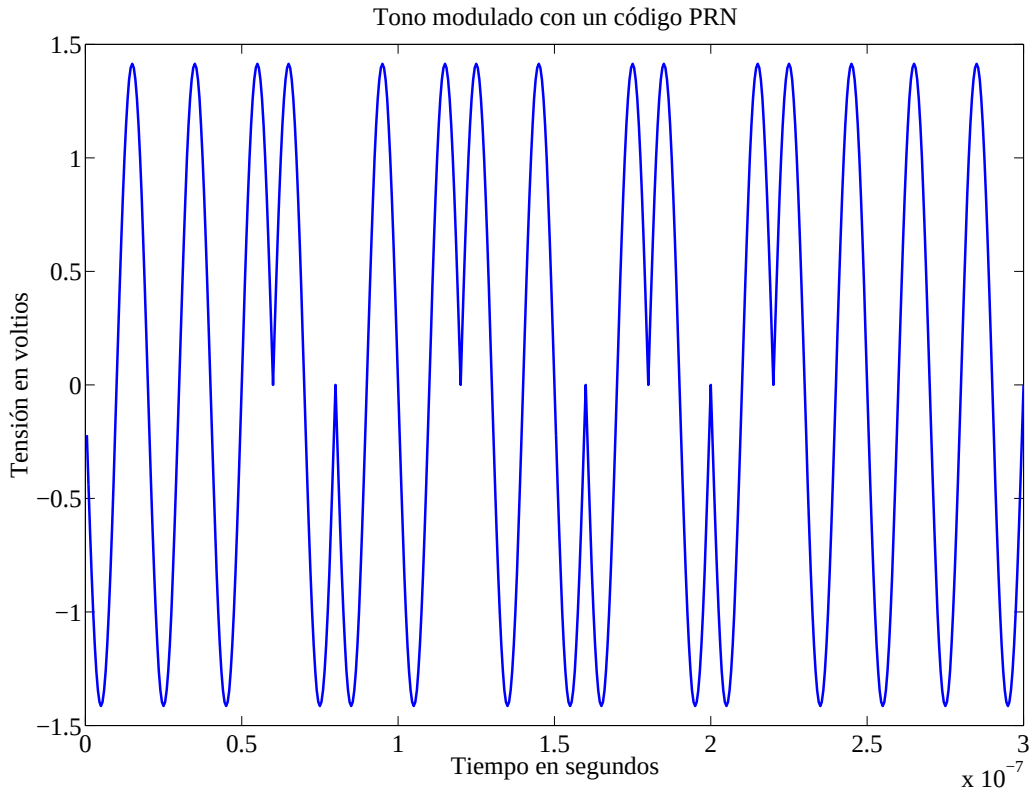

Figura 2.5: Tono modulado con código PRN.

En la figura 2.6 se pueden ver todas las señales en una misma gráfica.

2.3.2 Señal en el receptor

La señal que reciben los receptores $s(t)$ será la suma de la señal $m(t)$ enviada por el emisor más la suma de aportaciones indeseadas debidas a la presencia de un canal no ideal entre el emisor y el receptor. Las aportaciones no deseadas más importantes son: ruido banco gaussiano, errores dinámicos y multicamino.

2.3.2.1 Ruido

El ruido utilizado para emular el canal será AWGN (Additive White Gaussian Noise) limitado al ancho de banda de los receptores. Este tipo de ruido se caracteriza por ser plano en frecuencia y presentar una distribución de Gauss. El ruido queda caracterizado por una densidad espectral de potencia $N_0/2$ como se puede observar en la figura 2.7. Por tanto la varianza de ruido se calculará como la integral de la densidad espectral de potencia en el ancho de banda quedando según la expresión (2.5):

$$\sigma^2 = \frac{N_0}{2} \cdot 2BW = N_0 BW \tag{2.5}$$

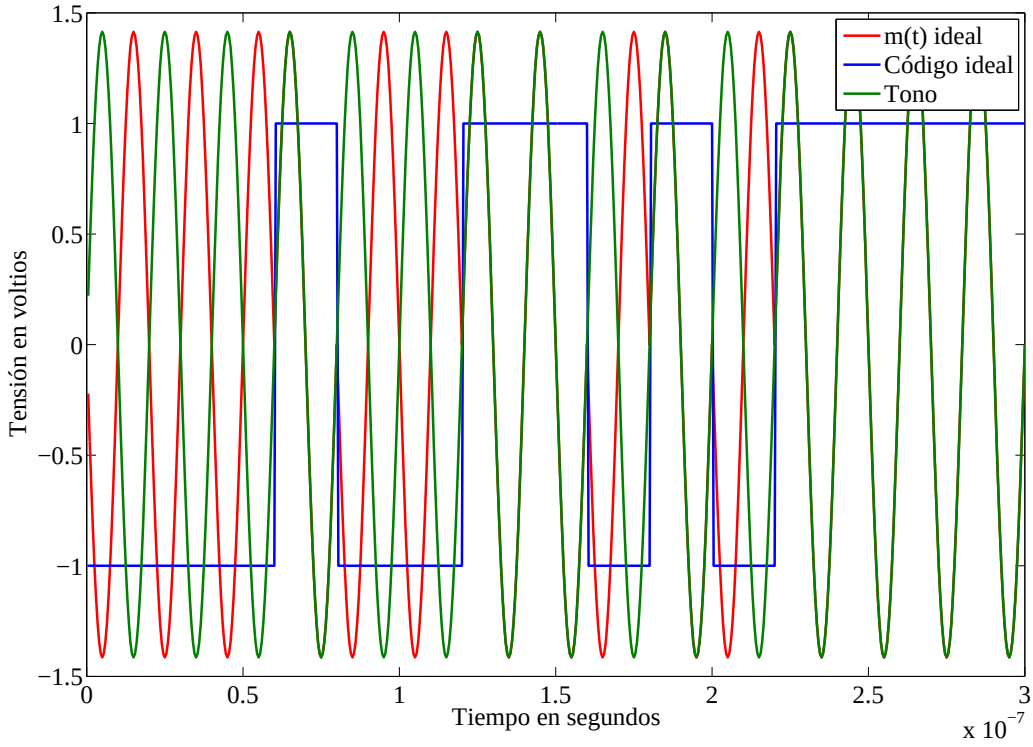

Figura 2.6: Señal $m(t)$, código ideal $c(t)$ y tono en una misma gráfica.

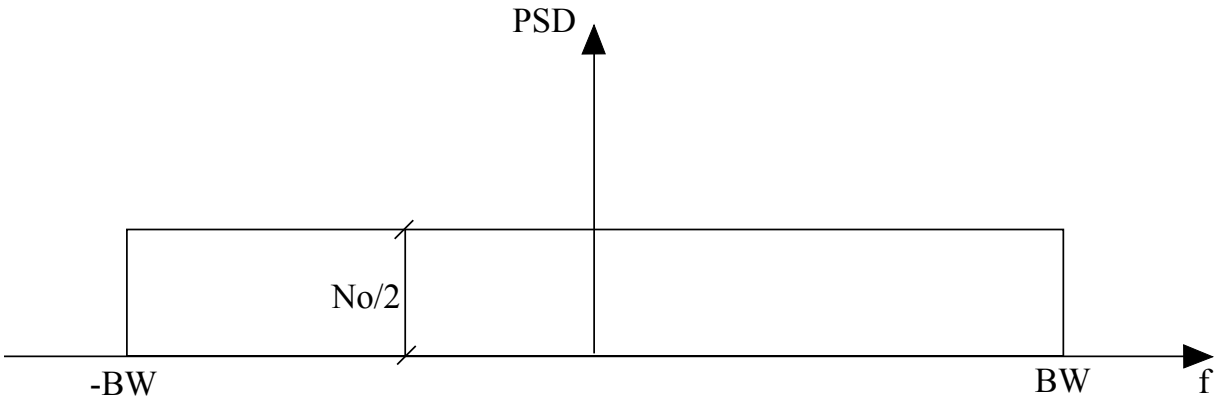

Figura 2.7: Densidad espectral de potencia del ruido.

La expresión de la señal recibida por el receptor del sistema después de pasar por dicho enlace en presencia de ruido y sin considerar multicamino quedaría según la ecuación (2.6):

$$s(t) = \sqrt{2P_r}m(t-\tau_{LOS}) + n(t) \tag{2.6}$$

donde $n(t)$ es ruido blanco gaussiano con un ancho de banda limitado por la frecuencia del filtro antialiasing.

2.3.2.2 Errores dinámicos

Los errores dinámicos son errores que afectan a la precisión de la estimación de retardo de la señal $s(t)$. Son debidos tanto a los errores de frecuencia entre emisor y receptor causados por la falta de sincronización entre ambos y al movimiento del emisor.

2.3.2.3 Multicamino

El multicamino es el fenómeno que ocurre cuando la señal del emisor llega al receptor por varios caminos distintos. Esto provoca que la señal que recibe el receptor sea un sumatorio de la señal directa (LOS) más cada una señales producidas por el multicamino cada una con un retardo y atenuación distintos.

La expresión de la señal $s(t)$ en presencia de multicamino sin tener en cuenta ningún efecto secundario es la mostrada en la ecuación 2.7.

$$s(t) = \sqrt{2P_{\text{LOS}}}r(t - t_{\text{LOS}})c(t - t_{\text{LOS}}) + \sum_{i=1}^{L} \sqrt{2P_{\text{MPi}}}r(t - t_{\text{MPi}})c(t - t_{\text{MPi}}) \tag{2.7}$$

2.3.2.4 Señal real

Por tanto la señal que recibirán los receptores será la suma de todos los efectos secundarios que introduce el canal por tanto la señal $s(t)$ tendrá la siguiente expresión:

$$s(t) = \sqrt{2P_{\text{LOS}}}r(t - t_{\text{LOS}})c(t - t_{\text{LOS}}) + \sum_{i=1}^{L} \sqrt{2P_{\text{MPi}}}r(t - t_{\text{MPi}})c(t - t_{\text{MPi}}) + n(t) \tag{2.8}$$

2.4 Fuentes de error

En esta sección se van a mostrar las expresiones teóricas que caracterizan las fuentes de error del sistema de posicionamiento local en estudio.

2.4.1 Ruido

Como ya se comentó en la sección 2.3.2.1 el ruido que afectará al sistema LPS será ruido blanco gaussiano de media nula, incorrelado y de ancho de banda plano.

La expresión de la señal recibida por el receptor del sistema después de pasar por dicho enlace en presencia de ruido y sin considerar multicamino quedaría como sigue (2.9):

$$s(t) = \sqrt{2P_r}m(t-\tau_{LOS}) + n(t)) \tag{2.9}$$

donde $n(t)$ es ruido blanco gaussiano con un ancho de banda limitado por la frecuencia del filtro antialiasing.

La expresión teórica para definir la varianza del error en segundos al cuadrado se puede expresar como 2.10:

$$\sigma_{\hat{\tau}}^2 = T_c^2 \frac{W_L}{64\text{SNR}} \tag{2.10}$$

donde T_c es el periodo de chip del código PRN utilizado, W_L es el ancho de banda en lazo cerrado del bucle de tracking y SNR es la relación señal ruido definida como:

$$\text{SNR} = \frac{P_r}{N_0/2} \tag{2.11}$$

donde $N_0/2$ es la densidad espectral de potencia del ruido, definida dicha dep como se muestra en la figura 2.7.

2.4.2 Errores dinámicos

Los errores dinámicos que afectan a la etapa de tracking del sistema son debidos principalmente al movimiento del emisor situado en un robot móvil y a los posibles errores de frecuencia entre emisor y receptor a causa de la falta de sincronización entre ambos.

La máxima variación del retardo de la señal de entrada debido al movimiento del objetivo viene dado por:

$$\Delta_\tau^{\text{objetivo}} = \frac{V_{\text{objetivo}}}{c} \tag{2.12}$$

donde c es la velocidad de propagación de la señal, que se puede aproximar a la velocidad de la luz en el vacío ($c = 3 * 10^8 m/s$). V_{objetivo} es la velocidad del objetivo expresada en m/s.

La otra fuente de error que interviene en los errores dinámicos son los errores de frecuencia entre emisor y receptor debidos a la falta de sincronización entre ambos. A partir de la frecuencia de chip f_c y la tolerancia del reloj de referencia ε_{CLK} se tiene que la máxima variación debida a los errores de frecuencia sigue la siguiente expresión:

$$\Delta_\tau^{\text{CLk}} = 1 - \frac{f_c}{fc \pm 2\frac{\varepsilon_{\text{CLK}}}{10^6} f_c} \tag{2.13}$$

Por tanto la máxima desviación debida a los errores dinámicos serán la suma de las dos aportaciones anteriores resultando la siguiente expresión

$$\Delta_\tau = \frac{V_{\text{objetivo}}}{c} + 1 - \frac{f_c}{fc \pm 2\frac{\varepsilon_{\text{CLK}}}{10^6} f_c} \tag{2.14}$$

Por tanto el error dinámico tendrá la siguiente expresión:

$$\varepsilon_{\hat{\tau}} = \frac{\Delta_\tau}{4W_L} \tag{2.15}$$

donde W_L es el ancho de banda en lazo cerrado del bucle de tracking.

2.5 Emulador

Llegados a este punto se pretende emular el enlace entre emisor y receptor para poder generar señales más realistas con las que poder validar el sistema propuesto anteriormente. Por tanto se pretende conseguir la señal $s(t)$ que se recibiría en el receptor.

El emulador, explicado con más detalle en el capitulo 3, cuenta con una etapa de conversión D/A, una simulación del enlace en el que se podrá añadir ruido blanco gaussiano con distintos niveles de SNR, errores dinámicos, multicamino, ... y posteriormente por una etapa de conversión A/D. La señal $s(t)$ obtenida después de todo este proceso será usada para validar mediante simulaciones el sistema de posicionamiento local propuesto.

Capítulo 3

Descripción y desarrollo

3.1 Descripción

El sistema emulador pretende conseguir una señal $s(t)$ lo más realista posible para lo cual se va a emular el enlace óptico que existiría entre emisor y receptor. El sistema está formado físicamente por un PC, un generador de funciones (AFG) y un osciloscopio.

En la figura 3.1 se puede observar la estructura general del sistema a desarrollar. En el PC se generará la señal deseada mediante Matlab. Se producirá un tono modulado con un código pseudoaleatorio (PRN) (la señal enviada se explica más detalladamente en la sección 3.2), al que se le podrán añadir distintos efectos secundarios como puede ser ruido aleatorio, desfase conocido y demás parámetros para conseguir simular el comportamiento de la señal en un entorno real. Esta señal digital sería la señal almacenada en la memoria local del emisor del sistema de posicionamiento que, tras un proceso de conversión digital analógico, sería enviada.

Mediante una conexión cableada PC-AFG, el PC envía al AFG la señal digital generada junto con todos los parámetros de configuración necesarios. La configuración del AFG está totalmente automatizada desde el PC. De acuerdo a los parámetros de configuración, el AFG realiza una conversión digital analógica de los datos recibidos para obtener a la salida una señal analógica con las mismas características que la señal generada en el PC. El AFG emularía al HW de generación de señal del emisor del sistema de posicionamiento.

Se conecta el AFG con el osciloscopio emulando el enlace óptico entre emisor y receptor.

El osciloscopio realiza una conversión analógica digital de los datos recibidos comportándose como la etapa de entrada del receptor del sistema de posicionamiento en estudio. El osciloscopio será previamente configurado desde el PC para determinar el comienzo de toma de muestras, la frecuencia de muestreo así como todos los parámetros necesarios para obtener los resultados deseados. El osciloscopio envía los datos digitalizados al PC según la configuración previamente establecida.

En el PC se realiza una readaptación de la señal recibida para ajustarla al formato deseado, necesario por la falta de sincronismo entre AFG y osciloscopio, así como la falta de adaptación de las frecuencias de muestreo de cada equipo y los posibles desajustes en el tamaño final de los datos. La señal recibida pueda ser utilizada en la arquitectura de medida implementada en el PC.

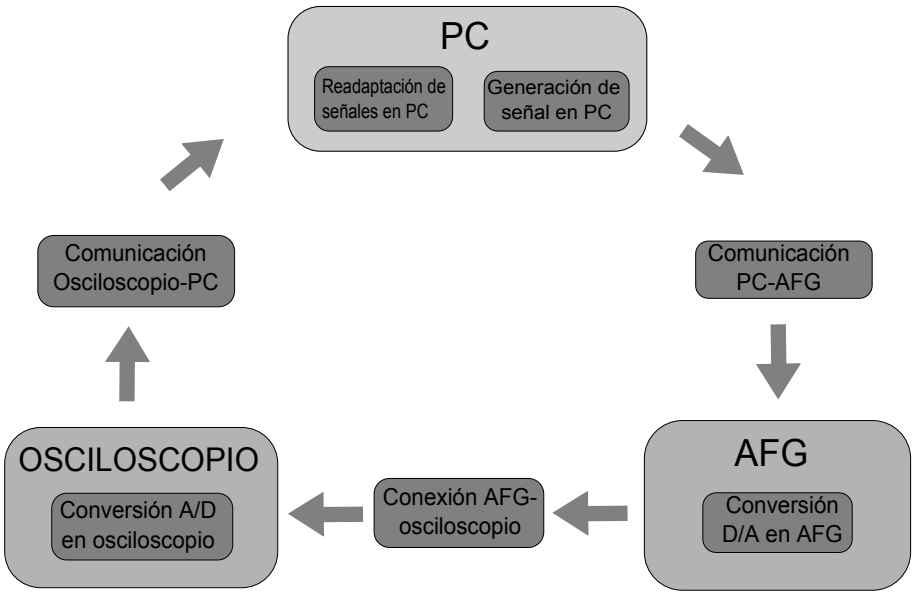

Figura 3.1: Estructura funcional del set-up de emulación del enlace emisor-receptor.

3.1.1 Equipos

El sistema como se ha comentado anteriormente está formado por un PC, un generador de funciones AFG y un osciloscopio. Las funciones de cada uno de ellos se comentará en las siguientes apartados.

3.1.1.1 PC

El PC emularía tanto a la memoria donde se almacena la señal que se quiere enviar y se manda al emisor en el sistema real, como la etapa de procesado de señal que hay después del receptor.

Por tanto el PC se encargará de generar las señales en Matlab y de enviarlas al AFG juntos con los parametros de configuración necesarios en cada caso.

La señal generada en tensión con Matlab (cada muestra se corresponde con un valor de tensión) no puede ser enviada directamente al AFG, sino que debe pasar por un proceso de codificación:

- Los datos deben estar codificados en 14 bits, o lo que es lo mismo el valor de cada punto de la señal debe estar entre 0 y $2^{14} - 1$. Posteriormente se indica el valor de tensión en

voltios pico a pico (Vpp) de la señal de salida del AFG. En este caso el valor de 0 se corresponderá con $-\text{Vpp}/2$ y $2^{14}-1$ se corresponderá con $\text{Vpp}/2$.

- Para realizar dicha codificación para cualquier señal dada en valores de tensión se han seguido los siguientes pasos:

  – Se obtiene el valor de tensión máxima de la señal que se quiere codificar para calcular el valor de la tensión pico a pico que se enviará como parámetro de configuración al AFG de la siguiente forma:

  $$\text{App} = 2\text{max}(|S|) \tag{3.1}$$

  donde S es la señal en tensión, App es la amplitud pico a pico que se envía al AFG, y max es una función que devuelve el valor máximo de los puntos que recibe como entrada. Se multiplica este valor máximo por 2 para obtener el valor pico a pico.

  – Para obtener los valores entre en 0 y $2^{14}-1$:

    ∗ Primero se divide la señal S entre el valor máximo, en este caso $\text{App}/2$, para dejar los valores de los puntos entre -1 y 1.

    $$S_{-1y1} = \left(\frac{S}{\text{App}/2}\right) \tag{3.2}$$

    ∗ Posteriormente se suma 1 para obtener los puntos entre 0 y 2.

    $$S_{0y2} = \left(\frac{S}{\text{App}/2} + 1\right) \tag{3.3}$$

    ∗ Finalmente se multiplica por $2^{13}-1$ para obtener los datos entre 0 y $2^{14}-1$. Quedando la siguiente expresión:

    $$S_c = \left(\frac{S}{\text{App}/2} + 1\right)(2^{13}-1) \tag{3.4}$$

    donde S_c es la señal codificada entre 0 y $2^{14}-1$

  – Entonces la señal de salida del AFG vendrá dada por la siguiente expresión:

  $$S_{D/A} = \frac{S_c}{2^{14}-1}\text{App} - \frac{\text{App}}{2} \tag{3.5}$$

  donde $S_{D/A}$ es la señal de salida del AFG una vez realizada la conversión digital analógica.

Una vez que la señal ha pasado por las distintas etapas de conversión D/A (AFG), canal no ideal y conversión A/D (osciloscopio) llega de nuevo al PC donde se hará un procesado para adecuar la tasa de muestreo. La señal llega procedente del osciloscopio con una tasa de muestreo

de 5GS/s, por lo que para realizar las pruebas posteriores se ha remuestreado la señal a 4 tasas de muestreo distintas:

- 5GS/s: Es la señal recibida directamente del osciloscopio.
- 1GS/s: La señal recibida del osciloscopio se pasa por una etapa de downsampling de 5.
- 10GS/s: La señal recibida de 5GS/s pasa por una etapa de upsampling de 2.
- 2GS/s: La señal de 10GS/s obtenida despues de pasar por un upsampling de 2, se la hace pasar por una etapa de downsampling de 5, con lo que finalmente se obtiene una señal de 2GS/s. Esta señal tiene la misma tasa de muestreo que la señal $m(t)$ generada en Matlab y a partir de la cual se ha obtenido $s(t)$.

3.1.1.2 AFG

El AFG es el encargado de realizar la función del emisor en el sistema LPS en estudio, realizando la conversión digital-analógico y posteriormente enviando la señal por uno de sus canales.

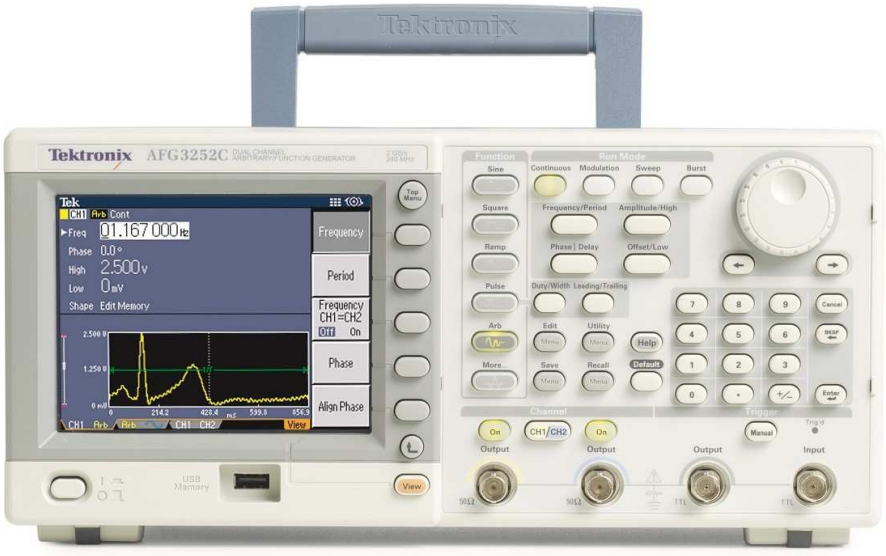

Figura 3.2: Dual channel Arbitrary Function Generator 3252 de TeKtronic.

El AFG usado es el AFG 3252 de Tektronix mostrado en la figura 3.2. Entre las numerosas características que dispone se va enumerar las que van a ser usadas en cada tipo de prueba:

- Señales ideales: Se usara la función de generación de señales arbitrarias cuyas características se pueden ver en la tabla 3.1.
- Señales con SNR: Se usará tanto la función de generación de señales arbitrarias como la función NOISE que permite generar Ruido banco gaussiano de 250 MHZ de ancho de

banda y distintos SNR. El parámetro configurable del que dispone la función NOISE es la amplitud con que se puede emular distintos SNR en función de la potencia de la señal de entrada. La correspondencia de cada valor de amplitud con un valor de SNR se puede ver en la sección 3.3.4.

- Señales con errores dinámicos: Se usará tanto la función de generación de señales arbitrarias como la modulación PM que permite emular los errores dinámicos. Los parámetros configurables de la modulación PM son: Shape, Deviation, PMfreq de los cuales se habla con más detalle en la sección 3.4.2.

3.1.1.3 Osciloscopio

El osciloscopio usado es el MSO 4104 de Tektronic (mostrado en la figura 3.3) el cual emulará al receptor del sistema de posicionamiento local. Será el encargado de realizar la conversión A/D de los datos procedentes del AFG y enviar los datos ya digitalizados a una frecuencia de muestreo de 5 GS/s al PC.

El osciloscopio permite capturar señales de hasta 100 millones de muestras, que según las especificaciones de las señales usadas equivalen a unos 2ms. Estos datos son suficientes para las pruebas que se realizarán.

El osciloscopio también dispone de una función Math que permite sumar las señales de que le llegan por dos de sus canales. Esta función será usada en las pruebas con ruido. Sumará la señal ideal que le llegue procedente del AFG con una señal de ruido AWGN que le llegue por otro canal procedente de otro canal del AFG. Esto se ha optado por realizarlo así debido a las limitaciones que presenta el AFG en las pruebas de ruido 3.3.3.

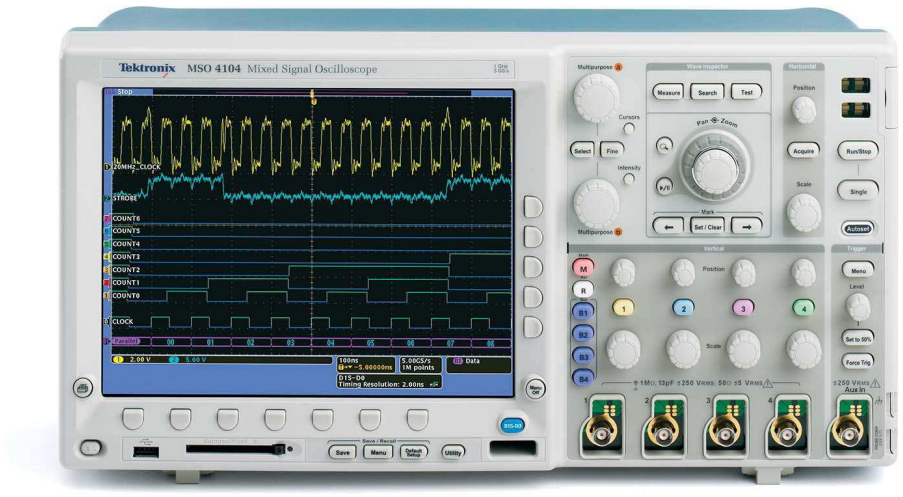

Figura 3.3: Osciloscopio MSO 4104 de TeKtronic.

3.2 Señales Ideales

Como ya se describió en la sección 2.3, la señal utilizada en el sistema y que emularía la señal enviada por el emisor se trata de un tono $r(t)$ modulado con un código de espectro ensanchado $c(t)$.

La señal emitida con una potencia P sera:

$$m(t) = \sqrt{2P}r(t)c(t) \tag{3.6}$$

La cual tendría una forma como la mostrada en la figura 3.4.

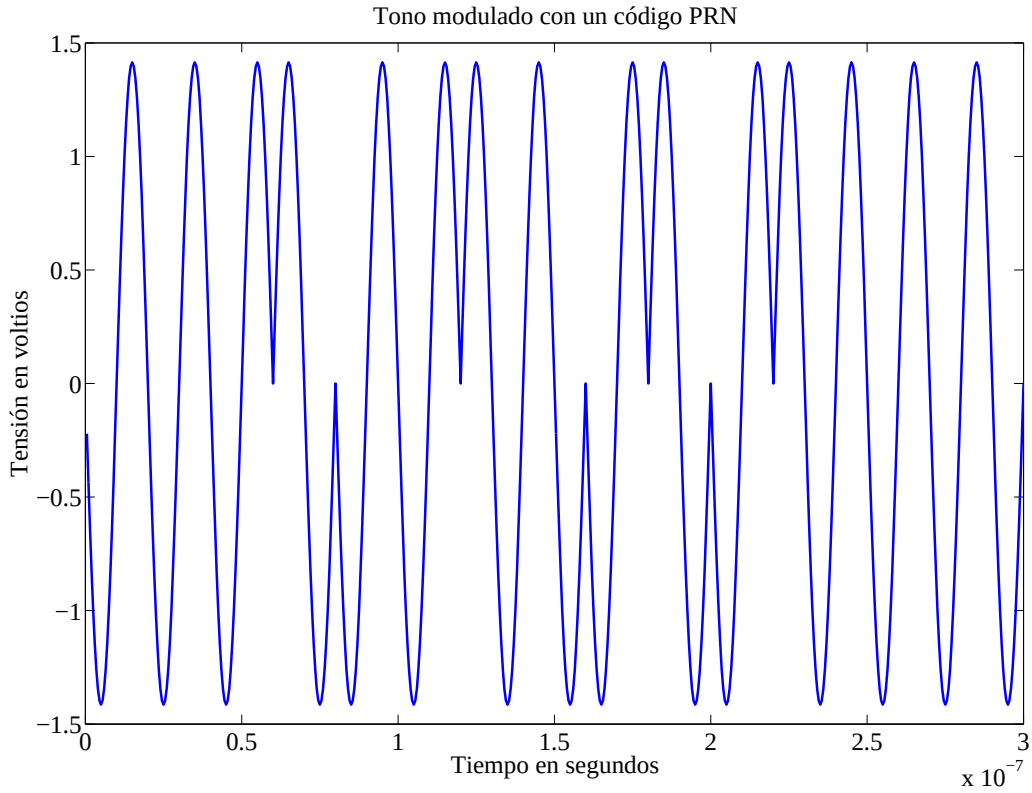

Figura 3.4: Tono modulado con código PRN, señal $m(t)$.

En el sistema real la señal $m(t)$ producto de la modulación del tono con la señal de espectro ensanchado será enviada de forma continua, por lo que en el sistema emulador lo que se hará será repetir la señal continuamente, de acuerdo con las limitaciones existentes.

Esta señal será generada en Matlab y posteriormente enviada al AFG, tras un proceso de codificación (este proceso se explica con detalle en la sección 3.1.1.1) para adaptar los datos al formato de entrada del AFG.

3.3 Señales con ruido

3.3.1 Descripción del ruido

El ruido utilizado para emular el canal será AWGN (Additive White Gaussian Noise). Este tipo de ruido se caracteriza por ser plano en frecuencia y presentar una distribución de Gauss. El ruido queda caracterizado por una densidad espectral de potencia $N_0/2$ como se puede observar en la figura 3.5. Por tanto la varianza de ruido se calculará como la integral de la densidad espectral de potencia en toda la frecuencia quedando según la expresión 3.7:

$$\sigma^2 = \frac{N_0}{2} \cdot 2BW = N_0 BW \tag{3.7}$$

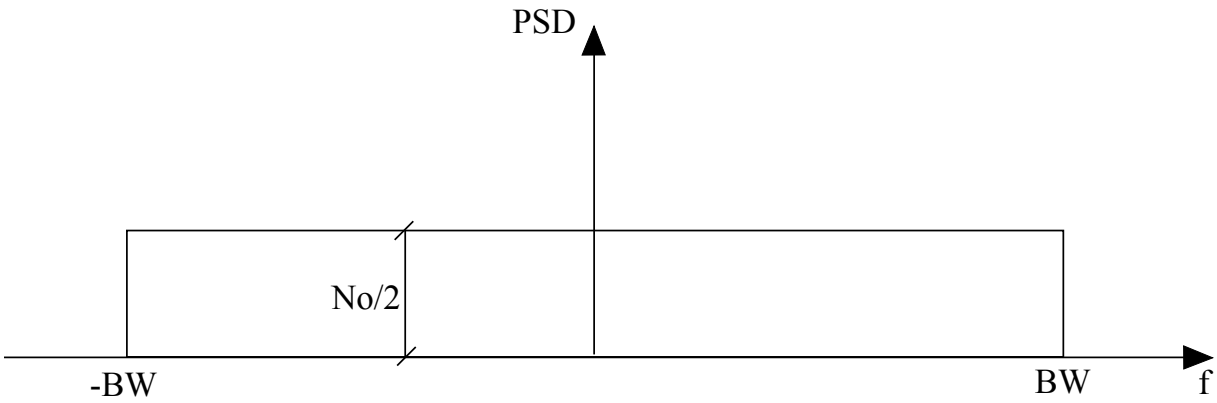

Figura 3.5: Densidad espectral de potencia del ruido.

La expresión de la señal recibida por el receptor del sistema después de pasar por dicho enlace en presencia de ruido y sin considerar multicamino quedaría según la ecuación 3.8:

$$s(t) = \sqrt{P_r} m(t - \tau_{LOS}) + n(t)) \tag{3.8}$$

donde $n(t)$ es ruido blanco gaussiano con un ancho de banda limitado por la frecuencia del filtro antialiasing.

3.3.2 Emular canal ruidoso

Para emular los efectos que provocaría un canal ruidoso sobre la señal $m(t)$, se va a proceder a sumar a la señal $m(t)$ una señal que emularía AWGN con un SNR determinado.

Existen dos opciones para emular el canal:

- Generar el ruido blanco gaussiano y sumarlo directamente a la señal $m(t)$ desde Matlab.

Tabla 3.1: Especificaciones AFG3252.

Arbitrary Waveforms	AFG 3252
Arbitrary waveforms in Burst Mode	1 mHz to 120 MHz
Effective analog bandwidth (-3dB)	225 MHz
Memory: Sample rate	2 to 16 k: 2 GS/s $>$ 16K to 128 K: 250 MS/s
Rise/Fall time	$\leq$3 ns
Jitter (RMS), typical	500 ps at 2 GS/s 4ns at 250MS/s

- Usar la función NOISE del AFG. Por tanto se podría generar la señal $m(t)$ en un canal del AFG y AWGN en el otro canal y sumarlo en el osciloscopio.

La primera opción no es viable por las limitaciones de las especificaciones del AFG (sección 3.3.3), por tanto se ha optado por usar la función NOISE del AFG.

3.3.3 Limitaciones

El módulo de generación de funciones arbitrarias de dicho AFG tiene las especificaciones [4] mostradas en la tabla 3.1.

Como se puede observar permite de 16K a 128K muestras para una frecuencia de muestreo de 250 MS/s y únicamente de 2 a 16K muestras para una frecuencia de muestreo de 2 GS/s. Dado que las señales que se van a usar tienen una frecuencia de decenas de MHz solo se puede usar para las pruebas la frecuencia de muestreo de 2 GS/s.

En el sistema emulador se pretende realizar dos tipos principales de pruebas: analizar el comportamiento del sistema LPS con señales a las que se les introduce ruido blanco gaussiano (AWGN) de varios SNR y analizar el comportamiento para del sistema ante señales con errores dinámicos.

Para el primer caso, en el que se verá el comportamiento del sistema LPS con señales en presencia de AWGN es necesario mandar un número significativo de muestras lo que implica tener que repetir la señal $m(t)$ varias veces. Existen dos formas de repetir la señal:

- Directamente desde Matlab y enviar al AFG la señal ya repetida.

- Repetir la señal que se envía al AFG desde el propio AFG. El AFG dispone de un parámetro de configuración que permite repetir la señal arbitraria, generada a partir de los datos que recibe, de forma continua.

Se va a analizar la primera opción de repetir la señal desde Matlab. A continuación se va a explicar las limitaciones existentes para el caso concreto de frecuencia de chip igual a 50 MHz, pudiéndose extrapolar dichas conclusiones para el resto de frecuencia de chip utilizadas en el emulador. Por tanto las características del código utilizado para $fc = 50$MHz son:

- Frecuencia de chip: $f_c = 50$ MHz.
- Bits de generación: 6 bits.
- Frecuencia de muestreo: 2 GHz.
- Muestras por chip: 40.
- Tamaño del código en chips: 15.
- Número de muestras del código: 600.

Teniendo en cuenta el número máximo de muestras que se pueden enviar al AFG es de 16K muestras y el tamaño de la señal que se quiere generar (600 muestras) se puede deducir que el número máximo de repeticiones de la señal $m(t)$ es 27. Esta opción tiene como principal ventaja que se puede generar un ruido perfectamente caracterizado desde Matlab y sumarselo directamente a $m(t)$ antes de enviarlo al AFG. En contra tiene el incoveniente de que la señal enviada sería demasiado corta, en tiempo sería una señal de $0{,}3\mu s$, lo que no aportaría resultados correctos ya que no existirían muchas muestras de ruido (idealmente infinitas) y la media y varianza no serían las correctas. Por tanto esta opción no sería viable.

Otra opción sería repetir la señal con ruido generada con Matlab compuesta por las 27 señales $m(t)$ más el ruido y repetirla continuamente configurando el AFG. Con esta opción se conseguiría señales más largas pero tiene un gran problema; el ruido sería el mismo en cada repetición del AFG por tanto ya no sería ruido incorrelado y los resultados no serían los esperados. Por tanto está opción tampoco sería viable.

La ultima opción es enviar la señal $m(t)$ y repetirla continuamente con el AFG. Por el otro canal que dispone el AFG se generaría AWGN mediante la función NOISE cuyos parámetros de configuración se han caracterizado en la sección 3.3.4. Ambas señales se sumarían directamente en el osciloscopio. Con esta opción se pueden conseguir señales de hasta 2ms de duración, por lo que será esta opción la usada en el emulador.

3.3.4 Caracterizar función NOISE del AFG

El Arbitrary/Function Generator AFG 3252 dispone de una función NOISE que genera ruido blanco gaussiano de distintas tensiones. Esa función será usada para añadir ruido a la señal $m(t)$ y generar la señal $s(t)$ que emularía la señal que reciben los receptor del sistemas LPS.

El único parámetro que se puede configurar en el AFG3252 de la función de ruido es su amplitud con valores entre 50 mVpp a 5Vpp. Las especificaciones y manuales del AFG3252 no indican de forma exacta a qué se corresponde esos valores de amplitud; si se corresponde a tensión pico a pico, tensión Vrms, amplitud máxima, etc.

Por tanto, se ha optado por caracterizar el ruido generado por el AFG y identificar con que valores de amplitud de la función NOISE se corresponde valores de SNR deseado.

El procedimiento seguido es el siguiente:

- Se ha enviado distintos valores de amplitud al AFG y se ha configurado la función NOISE. Los valores utilizados en voltios son:

  0,050; 0,1; 0,2; 0,5; 1; 2; 3; 4; 5

- Se han recogido las señales resultantes en el osciloscopio y guardado en Matlab.
- Para cada una de las señales recibidas se ha hecho un filtrado paso bajo a $\mathrm{BW} = 2f_c$ para quedarse solo con las componentes de ruido deseado.
- Para cada una de las señales filtradas se ha calculado su desviación estándar que se corresponderá con el valor rms de la amplitud del ruido.
- Con los valores de desviación estándar obtenidos y los valores de amplitud mandados al AFG, se ha obtenido la expresión de la recta que identifica cada valor de amplitud del AFG con un valor de tensión rms de ruido y viceversa.

Este procedimiento se ha realizado para cada una de las frecuencias de chip utilizadas en el emulador.

La ecuación que indica que valor de tensión rms de ruido se obtiene para un valor de amplitud del AFG con una precisión del 95 % es la mostrada en la ecuación 3.9.

$$\text{Para } f_c = 50\text{MHz} : V_{\text{rms}} = 0{,}07129\text{Amplitud} - 0{,}001626 \tag{3.9}$$

$$\text{Para } f_c = 10\text{MHz} : V_{\text{rms}} = 0{,}03418\text{Amplitud} - 0{,}0003805 \tag{3.10}$$

$$\text{Para } f_c = 25\text{MHz} : V_{\text{rms}} = 0{,}05206\text{Amplitud} - 0{,}001242 \tag{3.11}$$

$$\text{Para } f_c = 100\text{MHz} : V_{\text{rms}} = 0{,}09346\text{Amplitud} - 0{,}002863 \tag{3.12}$$

En la figura 3.6 se muestra la gráfica con la curva real y la identificada para la frecuencia de chip de 50 MHz, obteniéndose un resultado análogo para el resto de frecuencias de chip.

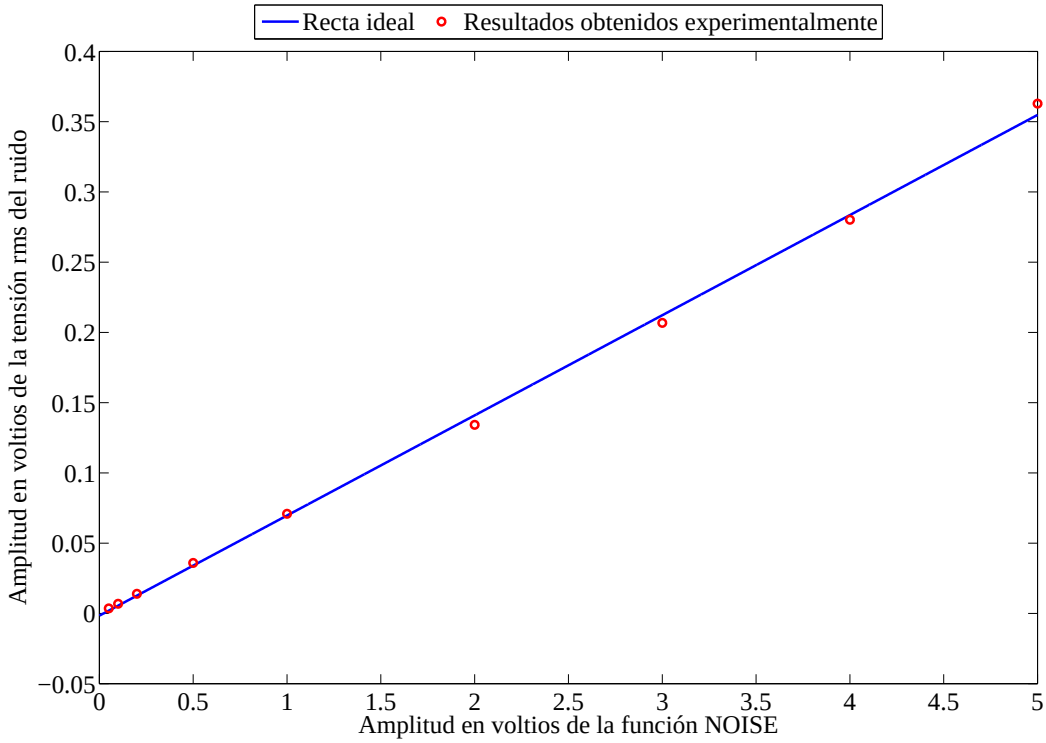

Figura 3.6: Caracterización NOISE Vrms frente a amplitud.

Por otro lado las ecuaciones que indican el valor de amplitud para un determinado valor de tensión rms con una precisión del 95% para cada frecuencia de chip son las mostradas a continuación:

$$\text{Para } f_c = 50\text{MHz} : \text{Amplitud} = 14{,}01 V_{\text{rms}} + 0{,}02483 \tag{3.13}$$

$$\text{Para } f_c = 10\text{MHz} : \text{Amplitud} = 29{,}24 V_{\text{rms}} + 0{,}01166 \tag{3.14}$$

$$\text{Para } f_c = 25\text{MHz} : \text{Amplitud} = 19{,}21 V_{\text{rms}} + 0{,}02418 \tag{3.15}$$

$$\text{Para } f_c = 100\text{MHz} : \text{Amplitud} = 10{,}69 V_{\text{rms}} + 0{,}03171 \tag{3.16}$$

En la figura 3.7 se puede ver la gráfica con la curva real y la identificada para el caso de frecuencia de chip de 50 MHz. Unicamente se ha mostrado este caso porque el comportamiento es análogo para todas las frecuencias de chip.

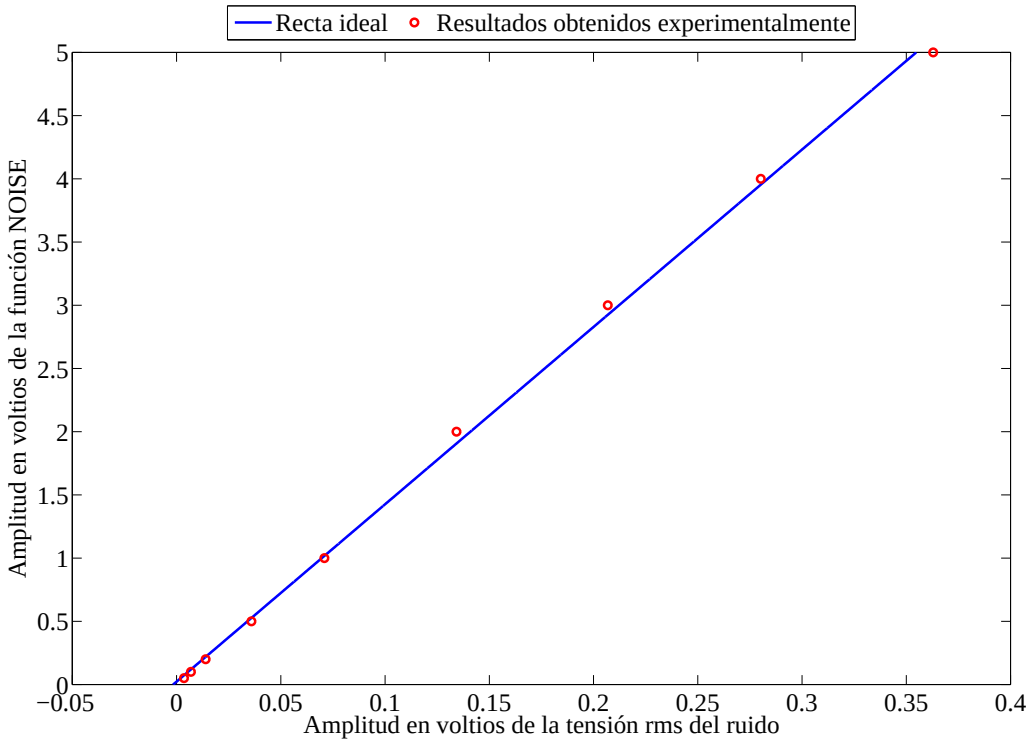

Figura 3.7: Caracterización NOISE amplitud frente a Vrms.

A partir de un valor de SNR determinado dado en dBHz se puede conseguir el valor de desviación estándar del ruido y por tanto el valor de tensión rms siguiendo los siguientes pasos:

- Se pasa el valor de SNR en dBHz a unidades naturales.

$$\text{SNR} = 10^{\frac{\text{SNR}_{\text{dBHz}}}{10}} \tag{3.17}$$

- Se calcula N_o.

$$N_o = \frac{P_s}{\text{SNR}} \tag{3.18}$$

  donde P_s es la potencia de la señal.

- Finalmente se obtiene la desviación estandar del ruido.

$$\sigma_{\text{ruido}} = \sqrt{N_o \text{BW}} \tag{3.19}$$

  donde BW es la frecuencia de corte del filtro situado a $\text{BW} = 2\,f_c$.

3.4 Errores dinámicos

3.4.1 Descripción

Los errores dinámicos que afectan al sistema LPS en estudio son debidos a dos causas fundamentales:

- El movimiento del emisor situado en un robot móvil. Esto provocará que la fase que miden los receptores aumente o disminuya según si el emisor se aleja o se acerca a los receptores.
- Los posibles errores de frecuencia entre emisor y receptor a causa de la falta de sincronización entre ambos. Esto provoca que las mediciones de fase no sean las correctas.

Las expresiones teóricas que caracterizan los errores dinámicos se mostraron en la sección 2.4.2.

3.4.2 Caracterización de los errores dinámicos

El generador AFG permite emular los errores dinámicos mediante una función que realiza una modulación en fase (PM).

Se pretende emular un desplazamiento de fase lineal caracterizado por una pendiente expresada en unidades de segundos/segundos o microsegundos/segundos. Esto significa que por cada segundo de la señal $m(t)$ la señal $s(t)$ tendrá un desfase de x microsegundos.

Para emular los errores dinámicos la función de modulación en fase del AFG cuenta con varios parámetros importantes:

- Shape: Indica la forma de onda que se usará para modular en fase. En este caso como lo que se pretende es un desplazamiento lineal de la fase la forma de onda requerida es una onda triangular.
- Deviation: Indica la amplitud la forma de onda triangular. Su valor va desde -180 a 180 grados.
- PMfreq: Indica la frecuencia de la forma de onda triangular que se usará para modular.

En la figura 3.8 se muestra de forma gráfica lo que representa los parámetros Deviation y PMfreq para el caso de una forma de onda (Shape) triangular.

Existen por tanto dos parámetros configurables para conseguir la pendiente de retardo,llamada k y representada en s/s, deseada en cada caso. Se ha optado por fijar deviation en 180^o y modificar el valor de PMfreq en cada caso.

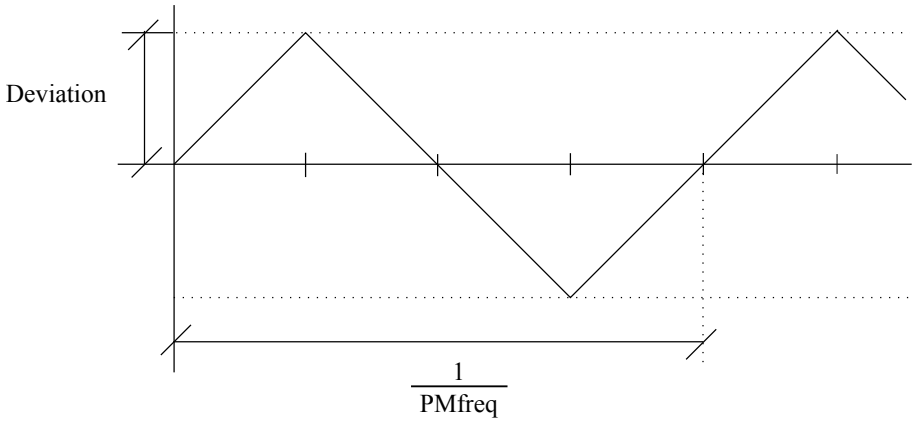

Figura 3.8: Parámetros modulación PM.

La pendiente k expresada en $^\circ/s$ viene dada por la expresión:

$$k = \frac{\text{Deviation}}{\frac{1}{4\text{PMfreq}}} \left[\frac{^\circ}{s}\right] \tag{3.20}$$

Para poder indicar k en segundos/segundos se multiplica por el siguiente factor expresado en $s/^\circ$:

$$\frac{1}{f360} \left[\frac{s}{^\circ}\right] \tag{3.21}$$

Ya que el periodo de la señal $T = 1/f$ se corresponde con 360°.

La expresión final que indica el valor de PMfreq para un valor determinado de k es la mostrada en 3.22.

$$\text{PM}_{\text{freq}} = \frac{kf360}{4\text{deviation}} \tag{3.22}$$

donde f es la frecuencia de la señal que se quiere modular, k es la pendiente de retardo en s/s, deviation vale 180° y el resultado de PM_{freq} está expresado en Hz.

El parámetro de configuración del AFG PMfreq puede tener valores comprendidos entre 2mHz y 50kHz, por tanto se podrán generar errores dinámicos con pendiente k comprendidos entre los 80 ps/s y los 2 ms/s, rango suficiente para las pruebas realizadas con el emulador.

3.5 Multicamino

El multicamino es el fenómeno que ocurre cuando la señal del emisor llega al receptor por varios caminos distintos. Esto provoca que la señal que recibe el receptor sea un sumatorio de la señal directa (LOS) más cada una señales producidas por el multicamino cada una con un retardo y atenuación distintos.

3.5.1 Descripción

La expresión de la señal $s(t)$ en presencia de multicamino sin tener en cuenta ningún efecto secundario es la mostrada en la ecuación 3.23.

$$s(t) = \sqrt{2P_{\text{LOS}}}r(t - t_{\text{LOS}})c(t - t_{\text{LOS}}) + \sum_{i=1}^{L} \sqrt{2P_{\text{MPi}}}r(t - t_{\text{MPi}})c(t - t_{\text{MPi}}) \tag{3.23}$$

La implementación del multicamino se realizará en Matlab, realizando un sumatorio de todos los multicaminos cada uno de ellos con su potencia y retardo y se sumarán a la señal $s(t)$ ideal. La señal compuesta será mandada al AFG.

3.6 Problemas en la realización del banco del pruebas

3.6.1 Limitación del generador de funciones

Como ya se comentó en la sección 3.3.3 el generador de funciones AFG3252 presenta una serie de limitaciones que han dificultado el proceso de implementación de la plataforma de pruebas o emulador.

3.6.2 Limite de muestras insuficiente

Como se observa en la tabla 3.1 y ya se mencionó en la sección 3.3.3 las 16K muestras de entrada que permitía el AFG para una frecuencia de muestreo de 2GS/s eran insuficientes para las pruebas con ruido. Finalmente se optó por enviar al AFG una señal corta y repetirla continuamente en el AFG y sumarle en el osciloscopio ruido procedente de otro canal del AFG. No es la opción que más rango de valores de SNR permita pero es la única que era viable. A continuación se van a explicar con detalle las pruebas y problemas encontrados hasta llegar a la solución que finalmente se ha usado.

Se pretendía conseguir una secuencia larga de muestras que representase un tiempo lo suficientemente alto como para que las pruebas de ruido fuesen consistentes. Como ya se comentó en la sección 3.3.3 no es posible generar la señal en Matlab, sumarle el ruido y enviarselo al AFG por la limitación de muestras, ni repetir una secuencia corta de muestras con ruido generado desde Matlab porque el ruido ya no sería incorrelado y los resultados no serían los deseados. Como la mejor forma de añadir ruido a la señal $m(t)$ sería desde Matlab ya que está perfectamente caracterizado y permitía un amplio rango de SNR, se siguió intentando utilizar esta opción.

Se pensó en generar señales lo más largas posibles dentro de la limitación de 16K muestras a las que se sumaría directamente el ruido desde Matlab. Se repetiría el proceso de generación de señal y suma de ruido un número suficiente de veces. Esto daría lugar a una serie de fragmentos de señal que se unirían para generar una secuencia larga. El proceso sería el siguiente:

- Se generaría la señal $m(t)$.
- Se generaría una señal de ruido blanco gaussiano $n(t)$. En cada repetición la señal $n(t)$ es distinta por lo que se soluciona el problema de las muestras correladas.
- Se sumarían ambas señales formando la señal $s(t) = m(t) + n(t)$.
- Se enviaría la señal $s(t)$ al AFG.
- Se guardaría en el PC la señal recibida del osciloscopio.
- La señal del osciloscopio guardada a 5GS/s y se remuestrea para adaptarla a 2GS/s, frecuencia de muestreo a la que se genera las señales en Matlab. Otra opción usada fue unir la secuencia con la tasa de 5GS/s y luego la secuencia entera remuestrearla a 2GS/s.

La señal recibida remuestreada a 2GS/s se uniría a la anterior. Así se pueden conseguir secuencias lo suficientemente largas para realizar las pruebas de ruido. El problema reside en el proceso de unión de una secuencia con otra.

Se realizarón algoritmos que permitieran unir con la mayor precisión una secuencia detrás de la otra, pero la máxima precisión que se pudo conseguir fue de menos de una muestra a 5GS/s. La dificultad del proceso de unión era debida al error introducido por el disparo tanto del AFG como del osciloscopio. En cuanto a esto se probaron varias opciones:

- Unir la salida de disparo del AFG al a la entrada de disparo del osciloscopio. Cuando una señal salía por un canal del AFG se ponía a nivel alto la señal de disparo del AFG, por lo que idealmente el trigger de entrada del osciloscopio debería capturar la señal siempre en la misma posición, lo cual no era siempre así.
- Conectar una salida del AFG (que llevaría la señal $s(t)$) a un canal de entrada del osciloscopio y una señal cuadrada configurada para que empiece en un instante conocido después de la señal $s(t)$ a otro canal del osciloscopio. Se configuraría el osciloscopio para que el trigger de entrada se realizase con esta última señal. Al igual que antes no funcionaba como debería.
- Configurar el trigger para que se activase a partir de un nivel de la señal $s(t)$. Esta opción solo sería válida para pruebas con un nivel de SNR muy alto por lo que no es viable.

Como ya se comentó ningún sistema funcionaba a la perfección por lo que aún haciendo uso de algoritmos que intentaban mitigar en todo lo posible los errores del disparo solo se consiguió una precisión de menos de una muestra a una frecuencia de muestreo de 5GS/s. En la figura 3.9 se muestra la señal unida en el tiempo en una intersección de dos secuencias y en la figura 3.10 la fase de la señal después de realizar una demodulación I/Q.

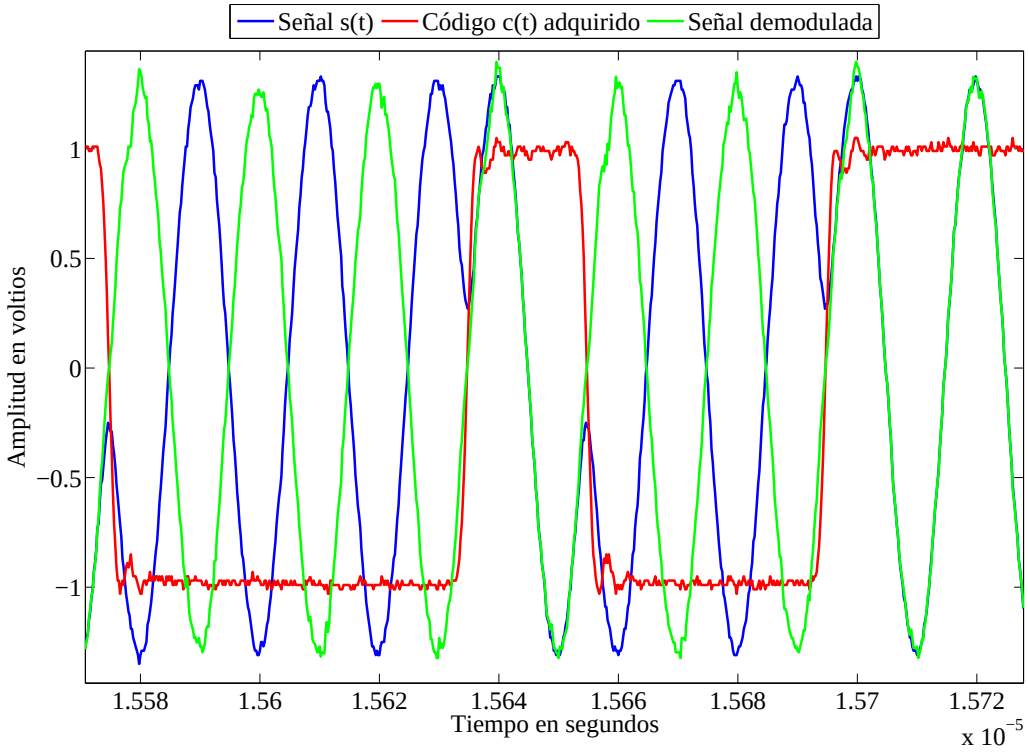

Figura 3.9: Señales $s(t)$, $c(t)$ y demodulada en una unión de dos secuencias.

En la figura 3.9 se muestra en color azul la señal $s(t)$ adquirida, en rojo el código c(t) adquirido y en verde la señal producto de la demolación realizada. Para llevar a cabo la demodulación se ha multiplicado la señal $s(t)$ por $c(t)$ consiguiendo un tono de una frecuencia de 50MHz (ya que se ha elegido 50 MHz para la frecuencia de chip). En el instante de tiempo $1{,}565 \cdot 10^{-5}$ s se produce un cambio de una secuencia a otra que a simple vista en la señal en el tiempo no se puede apreciar. Como las diferencias de fase entre secuencias son muy pequeña se va a realizar una demodulación I/Q de la señal demodulada (verde) y se va a observar el comportamiento de la fase. En la figura 3.10 se muestra el comportamiento de la fase de la señal demodulada a lo largo del tiempo. Se puede observar como en las uniones de una secuencia con otra hay un 'salto' de fase de aproximadamente 0.0065 radianes. En la figura 3.11 se muestra el mismo resultado que la figura 3.10 pero ampliado. Como se puede observar a parte de aparecer saltos también existe una cierta pendiente debida a los errores de frecuencia existentes entre los relojes de los dispositivos.

Debido a estos 'saltos' de fase, aún viendo que son mínimos se tuvo que descartar esta

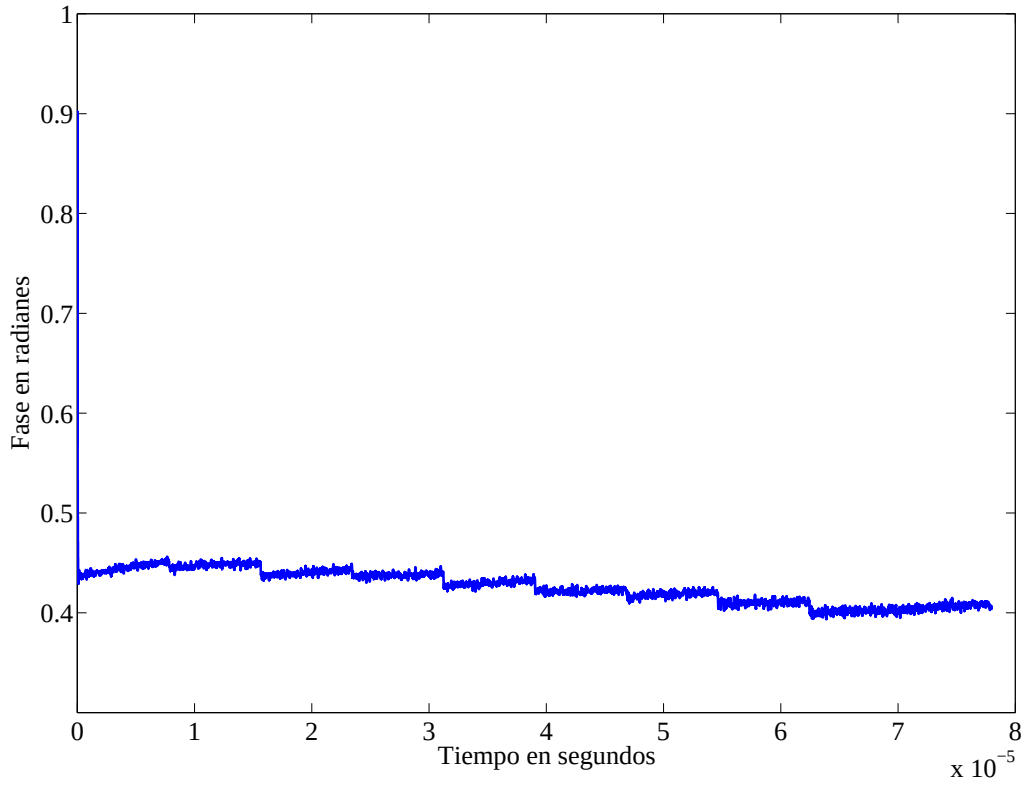

Figura 3.10: Fase de la señal demodulada en el tiempo.

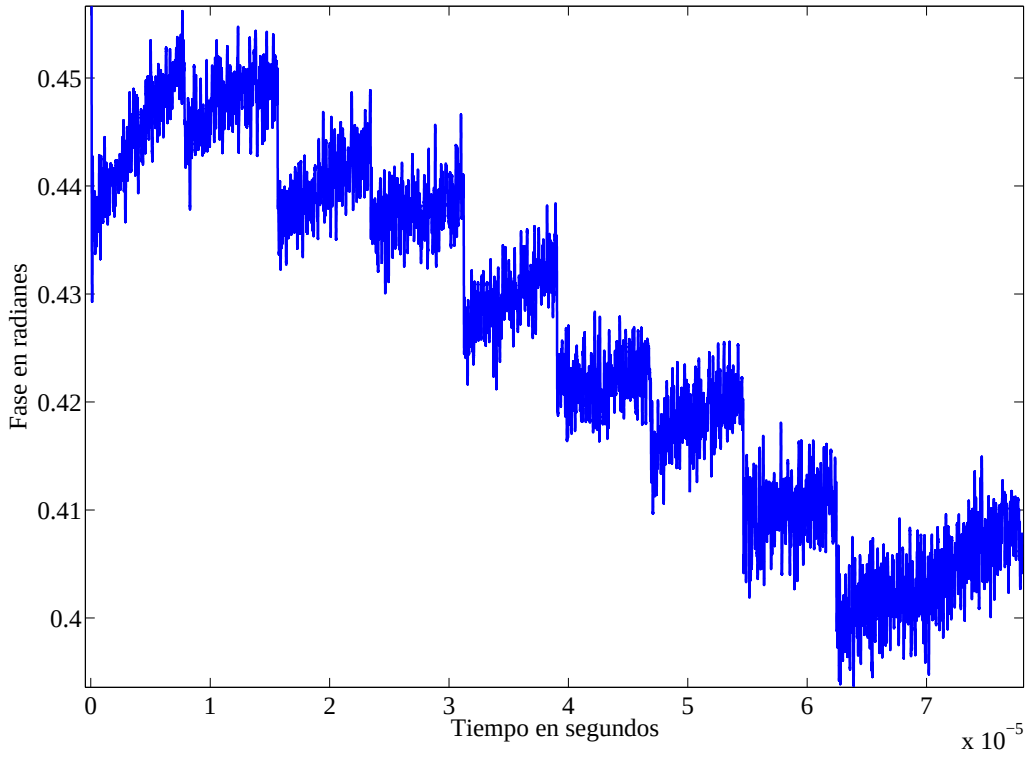

Figura 3.11: Fase de la señal demodulada en el tiempo ampliada.

opción por los posibles problemas que podrían causar en la simulaciones posteriores que se realizasen con esos datos.

3.6.3 Limite de amplitud entre 50 mVpp y 5 Vpp

El generador de funciones solo permite sacar señales por sus salidas con una amplitud máxima de 5 Vpp. Esto genera diversos problemas en las pruebas de ruido. Como se ha comentado anteriormente la relación señal ruido caracterizada por el SNR se consigue en el emulador variando la relación de potencia entre la señal $s(t)$ que se envía por uno de los canales del AFG y la potencia de ruido que se envía por otro canal. En el caso del emulador la relación de potencia se lleva a cabo con una relación entre las amplitudes de señal y ruido. Esto provoca que exista un límite superior e inferior de relación señal ruido. Si el SNR es muy alto la relación entre la amplitud de señal y la amplitud de ruido será muy alta, por tanto la amplitud de señal será mayor que 5 Vpp y la amplitud de ruido menor que los 50 mVpp. Lo contrario sucedería para SNR muy bajos.

Para poder maximizar la resolución de los datos, en el sistema emulador se ha optado por fijar la amplitud de ruido y ajustar la amplitud de la señal en función del SNR deseado. Esto permite conseguir SNR entre 60 y 90 dBHz.

Cuando no se introducen multicaminos la solución de fijar un valor para tensión de ruido es viable, ya que tras un proceso de pruebas se determinó cual era el valor adecuado para la tensión de ruido que permitiera variando la amplitud de la señal conseguir SNR entre 60 y 90 dBHz.

Al introducir multicaminos esta solución ya no es viable porque la potencia de la señal enviada al AFG es la suma de la señal de LOS y los multicaminos por lo que el rango de SNR dependería del número de MP configurados. Ante este problema se ha diseñado un sistema que configura un valor de ruido y prueba que, con el número de MP establecidos, se puede configurar el valor de amplitud de señal para el SNR deseado. Si el programa detecta que la amplitud de señal es menor o mayor que los limites del AFG se reduce la amplitud de la señal de ruido y se vuelve a probar. Y así hasta poder conseguir una amplitud de ruido que permite configurar una amplitud de señal para emular un SNR concreto.

Capítulo 4

Interfaz gráfica y función de comunicación con los dispositivos

4.1 Interfaz gráfica

4.1.1 Descripción

En esta sección se va describir la interfaz gráfica del sistema emulador descrito en este TFG. En primer lugar se mostrará el aspecto físico de la interfaz gráfica y posteriormente se continuará con la descripción de las partes y su funcionamiento. Cabe destacar que la introdución de los valores en los cuadros de texto se ha de realizar según el formato de entrada que utiliza Matlab.

La interfaz gráfica del emulador se puede ver en la figura 4.1. Esta compuesta por 6 grupos de elementos diferenciados, mostrados en la figura 4.2:

- Configurador del canal (Rojo): En este apartado se configura las condiciones y las características del canal que se desea emular.
- Selección de la frecuencia de chip (Verde): Se puede elegir entre cuatro posibles frecuencias de chip: 10 MHz, 25 MHz, 50 MHz, 100 MHz.
- Nombre del fichero en el que se guardarán las variables (Azul oscuro).
- Representación de señales (Amarillo).
- Paneles de error e información (Rosa).
- Botón de ejecución (Azul claro).

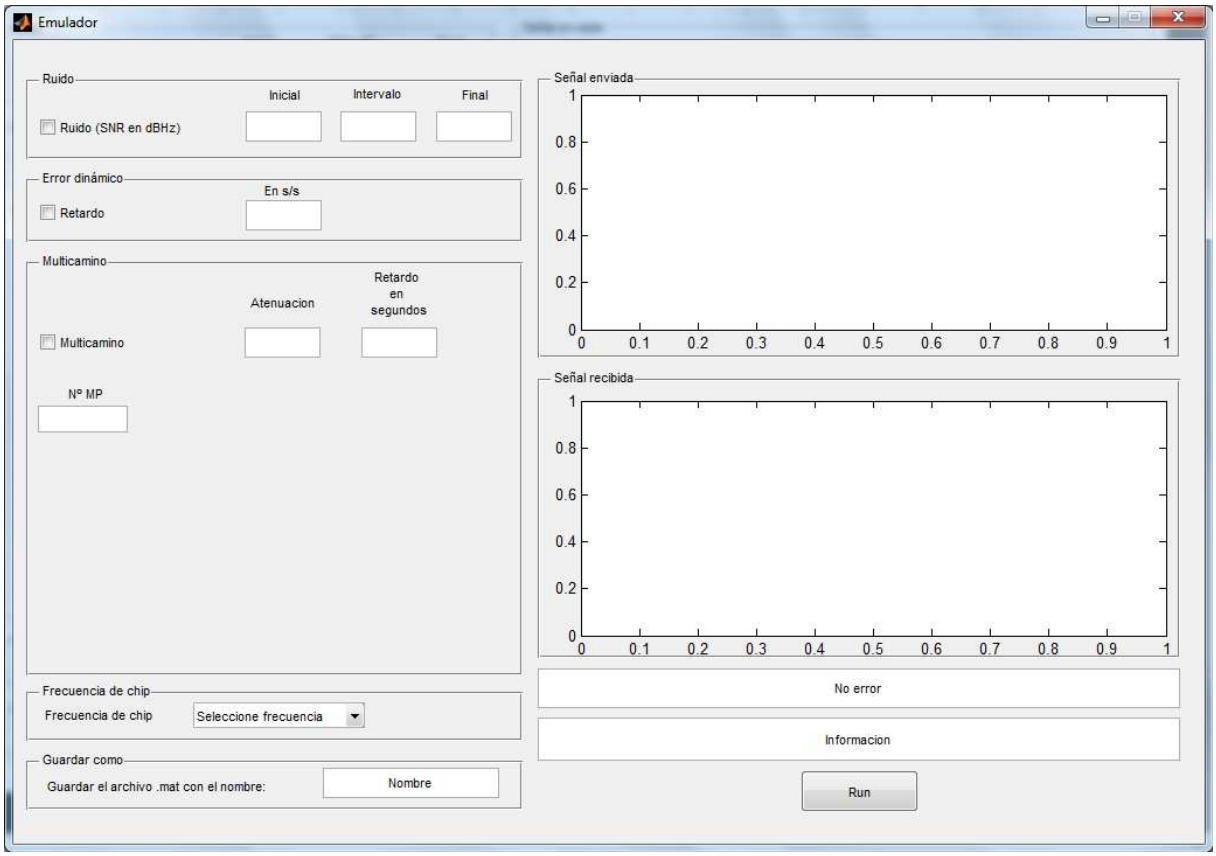

Figura 4.1: Interfaz gráfica del emulador.

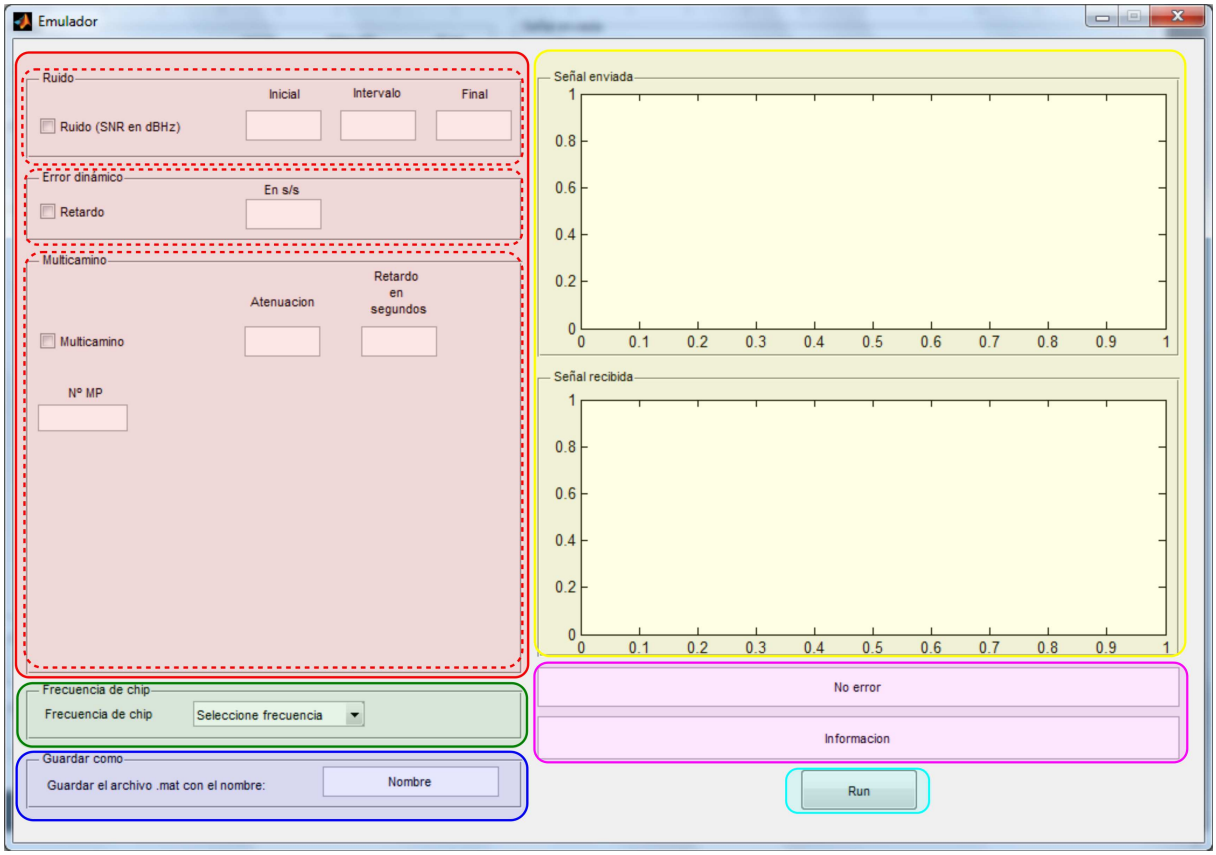

Figura 4.2: Distintas partes que componen la interfaz gráfica del emulador.

4.1.2 Descripción de las distintas partes que forman la interfaz gráfica del emulador

4.1.2.1 Configurador del canal

En esta sección se explicará las distintas partes que componen la parte de configurador de canal (color rojo en la figura 4.2) de la interfaz gráfica del emulador. Esta compuesto por 3 partes diferenciadas:

- Ruido.
- Error dinámico.
- Multicamino.

Este apartado del emulador es el encargado de configurar las características del canal que se desea emular.

Ruido

En este apartado se puede introducir el rango de valores de SNR expresados en dBHz que se quieren emular. Está compuesto por 3 cuadros de texto donde se puede introducir los valores de SNR deseados. Esos cuadros de texto son:

- Inicial: Se introduce el valor de SNR en dBHz inicial.
- Intervalo: Se introduce el valor de SNR en dBHz que marcará los saltos ente un valor de SNR y el siguiente.
- Fin: Se introduce el valor de SNR en dBHz final.

Para activar esta característica en el emulador se debe marcar con un tick el cuadrado situado a la izquierda de 'Ruido(SNR en dBHz)'. Se pueden introducir valores en los cuadros de texto explicados anteriormente pero solo se tendrán en cuenta si existe un tick en el cuadro mencionado más arriba. En la figura 4.3 se puede ver un ejemplo en el que se muestra a la izquierda cuando el ruido está activado (tick en el cuadrado) y a la derecha cuando está desactivada esta opción aún existiendo valores en los cuadros de texto.

Figura 4.3: Ruido activado y desactivado.

El apartado de ruido del emulador está configurado para insertar unos valores de SNR en los cuadros de texto iniciales cuando se hace tick en el cuadro. Estos valores son los mostrados en la figura 4.3 y se pueden cambiar por cualquier otro valor en cualquier momento.

Dado que existen limitaciones debidas a las características del AFG los valores admitidos de SNR están comprendidos ente 60 dBHz y 90 dBHz.

Error dinámico

En este apartado se puede configurar el valor de retardo que emularía los errores dinámicos. El valor se introduce en segundos/segundos (s/s). Al igual que en el caso anterior para activar esta opción es necesario marcar con el tick el cuadro. También se introduce un valor de retado inicial al marcar ese cuadro con un valor de 1e6 que representa un $1\mu s/s$.

Los valores de retardos que se pueden introducir están limitados por las características del AFG y por la frecuencia de chip seleccionada. Los valores aceptados para cada frecuencia son los siguientes:

- 10 MHz: Entre 4e-10 s y 0.01 s.
- 25 MHz: Entre 1.6e-10 s y 0.04 s.
- 50 MHz: Entre 8e-11 s y 0.002 s.
- 100 MHz: Entre 4e-11 s y 1e-3 s.

Multicamino

En este apartado se configuran los distintos multicaminos que se pretenden emular. El número de multicaminos está limitado ente 0 y 10 multicaminos.

Existen tres cuadros de texto en este apartado:

- Atenuación: En este cuadro de texto se introducirá el valor en tanto por uno de la amplitud de cada multicamino respecto a la amplitud de la señal original. Por ejemplo un valor de 0.5 simbolizaría que ese multicamino presenta una amplitud del 50 % respecto a la amplitud de linea directa. Aparecerán tantos cuadros de texto como multicaminos se introduzcan en el cuadro de texto 'Nº MP'.
- Retardo en segundos: Se introducirá el valor en segundos del retardo que se quiere que presente el multicamino con respecto a la señal de linea directa. Aparecerán tantos cuadros de texto como multicaminos se introduzcan en el cuadro de texto 'Nº MP'.
- Nº MP: Se introduce el número de multicaminos que se desean emular. Aparecerán tantos cuadros de texto de 'Atenuación' y 'Retardo en segundos' como número de multicaminos se haya introducido en este cuadro de texto.

Al igual que el resto de parámetros del emulador, para que se activen es necesario marcar con un tick el cuadro correspondiente. Por defecto cuando se marca el tick aparece 1 multicamino con un valor en 'Atenuación' de 0.5 y un retardo de 2e-7. Valores que se pueden modificar en cualquier momento.

En la figura 4.4 se muestra una captura de pantalla del emulador cuando se introduce un 10 en el cuadro de texto 'Nº MP'. Como se puede ver aparecen 10 cuadros de texto para introducir la atenuación y retardo de multicamino. Por defecto aparecen con el valor 0 que se puede cambiar por el valor deseado.

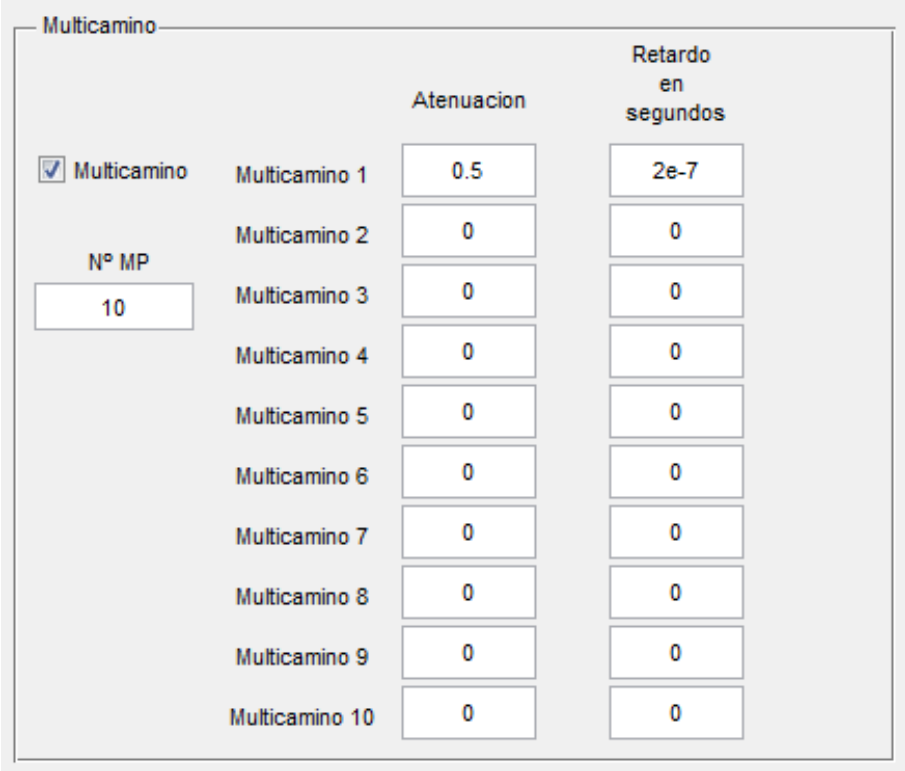

Figura 4.4: Captura del emulador para 10 multicaminos.

4.1.2.2 Selección de frecuencia de chip

En este apartado (verde en la figura 4.2) se configura la frecuencia de chip con la que se quiere configurar el emulador. Se seleccionará la frecuencia a partir de una lista despegable. Existen 5 opciones para elegir diferentes:

- Seleccione frecuencia: Es la opción por defecto. Si no se cambia de opción al ejecutarse el emulador la frecuencia de chip se configura automáticamente para $f_c = 50$MHz.
- 10 MHz: Configura la frecuencia de chip a 10 MHz.
- 25 MHz: Configura la frecuencia de chip a 25 MHz.

- 50 MHz: Configura la frecuencia de chip a 50 MHz.
- 100 MHz: Configura la frecuencia de chip a 100 MHz.

En la figura 4.5 se muestra las distintas opción de la lista desplegable.

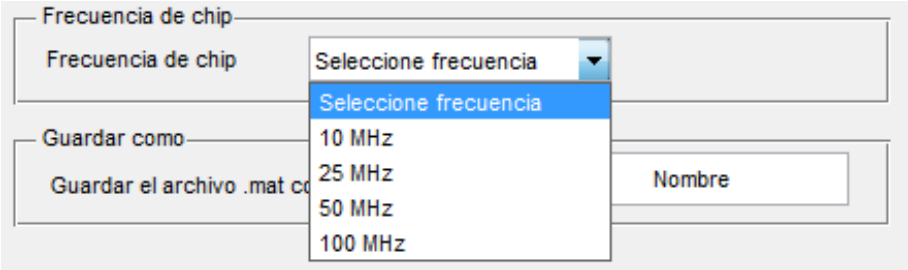

Figura 4.5: Selección de frecuencia de chip.

4.1.2.3 Nombre del fichero en el que se guardarán las variables

En este apartado (azul en la figura 4.2) se puede configurar el fichero .mat que contendrá todas las variables necesarias. El nombre deseado para el fichero se deberá introducir en el cuadro de texto sin necesidad de añadir la extensión .mat. Si no se modifica el valor existente en el cuadro de texto el nombre del fichero donde se guardarán las variables será Nombre.mat.

Las distintas variables que se guardarán en el fichero serán:

- 'v_osc_1_1G','v_osc_1_2G','v_osc_1_5G','v_osc_1_10G'. Variables que contienen la señal recibida del osciloscopio cada una remuestreada a una frecuencia de muestreo de 1 GS/s, 2 GS/s, 5 GS/s y 10 GS/s. Los valores de amplitud y por tanto potencia de estas señales se pueden deducir de la variable 'A_señal'. Se usan estos valores de potencia para aprovechar al máximo las características del AFG. Cada variable se trata de una matriz en la que cada fila corresponde con un valor de SNR determinado.
- 't_osc_1_5G'. Tiempo devuelto por el osciloscopio a una tasa de muestreo de 5GS/s.
- 'v_1G','v_10G','v_5G','v_2G'. Variables que contienen la señal recibida del osciloscopio cada una remuestreada a una frecuencia de muestreo de 1 GS/s, 2 GS/s, 5GS/s y 10GS/s. Estas señales han sido normalizadas para conseguir la señal que se obtendría si la potencia de la señal $m(t)$ fuese la unidad. Cada variable se trata de una matriz en la que cada fila corresponde con un valor de SNR determinado.
- 'SNR_dB'. Vector que contiene los valores de SNR en dBHz utilizados.
- 'A_senal'. Vector que contiene las amplitudes que se han introducido a las señales $m(t)$ para cada SNR.

- 'Vrms_ruido'. Vector que contiene las amplitudes que se introducen al AFG para emular cada SNR.

4.1.2.4 Representación de señales

En este apartado (amarillo en la figura 4.2) se muestra la señal enviada al emulador en la parte superior y la señal recibida del emulador en la parte inferior. Como se pueden hacer barridos de SNR la señal mostrada en la señal recibida será la correspondiente al último SNR. La señal enviada va combiado según el SNR que se este emulando en cada instante.

El eje y de ambas figuras está expresado en voltios y el eje x en tiempo en segundos.

4.1.2.5 Paneles de error e información

En este apartado (rosa en la figura 4.2) se muestran los mensajes de error y evolución del emulador.

Los mensajes de error posibles dentro del panel de error (panel superior) son:

- 'No error'. No existe ningún error.
- 'La amplitud de la señal de salida no es correcta ya que se ha saturado el AFG. Imposible arreglar'. Se ha introducido un valor de SNR, una frecuencia de chip o un número tal de multicamino que debido a las limitaciones del AFG hacen que no sea posible emular ese canal.
- 'La amplitud de la señal de salida es inferior al mínimo por lo que la señal no existe o no se puede representar. Se finaliza el programa.'. Se han introducido unos parámetros al emulador que hacen que la señal enviada al AFG sea nula o con una amplitud menor que el límite inferior del AFG.

Los mensajes de información del panel de información (panel inferior) son:

- 'Emulando...'. Aparece cuando se pulsa el botón 'Run'.
- 'Procesando...'. Se muestra cuando el emulador empieza a procesar los datos recibidos del osciloscopio.
- 'Final. Puede introducir nuevos valores.'. Este mensaje aparece cuando se finaliza la ejecución del emulador. La finalización puede ser debida al llegar al final de la correcta ejecución del emulador o debido a la presencia de un error que impide continuar al programa.

4.1.2.6 Botón de ejecución

Este botón (azul claro en la figura 4.2) inicia la ejecución del emulador. Solo se debe pulsar el botón 'Run' una vez introducidos los datos iniciales deseados o después de que se muestre el mensaje 'Final. Puede introducir nuevos valores.'.

4.2 Funciones de comunicación con los dispositivos

A continuación se van a explicar las funciones encargadas de la comunicación desde el PC al AFG y desde el PC al osciloscopio.

4.2.1 AFG3000S

AFG3000S.m es la función encargada de la comunicación entre el PC y el AFG. Permite enviar señales arbitrarias desde Matlab a la memoria interna del AFG así como configurar todos los parámetros necesarios en el sistema emulador propuesto. Se va a explicar con detalle cada una de las partes que componen la función AFG3000S.m.

En el fragmento de código 4.1 se puede ver la declaración de la función así como una explicación de las variables de entrada de dicha función.

Listado 4.1: AFG3000S-Declaración de la función.

```
function AFG3000S( wave, freq, App,salida, modo, parametro, wave2, freq2
    , App2,salida2, modo2, parametro2,soloCH1 )
%
%
%   function AFG3000S( wave, freq, App,salida, modo, parametro, wave2,
    freq2, App2,salida2, modo2, parametro2,soloCH1 )
%
%   -> wave es la señal que se quiere enviar al AFG, el maximo de
    muestras es
%   131072. La señal debe estar codificada de 0 a 2^14.
%
%   -> freq es la frecuencia de toda la señal, es la inversa del tiempo
    que
%   dura la señal entera.
%
%   -> App es la amplitud pico a pico de la señal. Debe ser un valor de
    50mVpp
%   a 5Vpp.
%
%   -> salida es el tipo de salida que se quiere.
```

```
%   'mem' saca por la salida los datos de la memoria.
%   'sin' saca por la salida un seno.
%   'noise' saca por la salida ruido.
%
%   -> modo indica el modo de la señal de salida
%   'Burst' se establece el modo Burst
%   'PM' la señal se modula en fase
%   Si se quiere modo continuo se pone manualmente en el AFG y en la
%   variable modo se pone cualquier caracter que no sea ni 'Burst' ni '
    PM'
%
%   -> parametro es un valor que depende del modo en el que se este:
%   Si modo=='Burst' indica el numero de ciclos que se envia en cada
    ráfaga.
%   Si modo=='PM' indica el valor de PMfreq calculado como parametro=(k*
    freq*360)/(180*4)
%
%   -> soloCH1 si se pone a 1 solo se configura el canal 1 y se pone
    otra
%   cosa se configura el canal 1 y el 2.
```

En el fragmento de código 4.2 se muestra la creación de los objetos de dispositivos necesarios para la comunicación entre PC y AFG así como el establecimiento de la conexión.

Para el proceso de creación de los objetos de dispositivos se usan las funciones de la toolbox de Matlab Instrument Toolbox, así como el software Tekvisa suministrado por el fabricante del dispositivo Tektronic.

El primer paso es la creación de la variable 'afg' con la función 'instrfind' a la que se introducen entre otros parámetros de entrada el tipo de conexión, en este caso 'visa-usb' y la dirección USB del dispositivo, en este caso 'USB0::0x0699::0x0345::C010029::0::INSTR'. Una vez creada la variable 'afg' se crea el objeto de dispositivo USB 'deviceObj' con ayuda de la función 'icdevice' a la que se le introduce como parámetros de entrada la variable 'afg' y 'tek_afg3000.mdd'.'tek_afg3000.mdd' es una librería para Matlab que incluye algunas de las funciones más importantes de la serie de generadores de funciones AFG 3000, con las cuales se puede configurar el AFG sin necesidad de conocer a fondo la sintaxis propia con la que se configuran estos dispositivos. Cabe destacar que ha sido necesario en la realización de este emulador el uso tanto de las funciones de la librería 'tek_afg3000.mdd' como la configuración de algunos parámetros a través la sintaxis propia del dispositivo.

Los dos comandos siguientes a la creación de objetos de dispositivos establecen el tamaño del buffer tanto de entrada como de salida.

Posteriormente se lleva a cabo la conexión entre el PC y el AFG con la función connect(deviceObj).

Para evitar posibles errores en cada conexión PC-AFG se borran los buffers y se realizar un 'clear' del dispositivo.

Listado 4.2: AFG3000S-Creación de los objetos de dispositivos.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%            Canal 1                %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Comunicación con el instrumento

afg = instrfind('Type', 'visa-usb', 'RsrcName', 'USB0::0x0699::0x0345::
   C010029::0::INSTR', 'Tag', '');

% Crear un objeto VISA-USB si este no existe, sino usa el objeto que ha
% encontrado

if isempty(afg)
    afg = visa('tek', 'USB0::0x0699::0x0345::C010029::0::INSTR');
else
    fclose(afg);
    afg = afg(1);
end

% Crear objeto de dispositivo.
deviceObj = icdevice('tek_afg3000.mdd', afg);

% Configurar InputBufferSize a 5000000 valor por defecto 512
set(afg,'InputBufferSize',5000000);

% Configurar OutputBufferSize a 500000000 valor por defecto 512
set(afg,'OutputBufferSize',500000000);

% Conectar el objeto de dispositivo al hardware.
connect(deviceObj);

% Limpiar buffer
clrdevice(afg);

% Reset
% fwrite(afg, '*rst;');

% Clear
fwrite(afg, '*cls;');
```

En el listado 4.3 se muestra como se configura la salida en función del parámetro 'salida'. Los fragmentos de código mostrados a continuación son usados para configurar el canal 1. Para

la configuración del canal 2 se procederá de la misma manera cambiando un (1) por un (2) en determinadas funciones.

Listado 4.3: AFG3000S-Selección de la forma de onda a la salida del AFG.

```
%% Canal 1
if(strcmp(salida,'mem'))
    % Preprocesado
    % Codifica la variable 'wave' al formato de datos binario que
       entiende
    % el AFG
    binblock = zeros(2 * length(wave), 1);
    binblock(2:2:end) = bitand(wave, 255);
    binblock(1:2:end) = bitshift(wave, -8);
    binblock = binblock';

    % Se construye el bloque de cabecera en binario
    bytes = num2str(length(binblock));
    header = ['#' num2str(length(bytes)) bytes];

    % Se limpia la memoria editable y se configuran las muestras
    fwrite(afg, ':trace:define ememory, 131072;'); %Max 131072

    % Se envia los datos a la memoria editable
    fwrite(afg, [':trace ememory,' header binblock ';'], 'uint8');

    % Se copian los datos a la memoria interna USER1
    fwrite(afg, 'DATA:COPY USER1,EMEMory');
end

% Configurar el tipo de salida. Memoria interna, Ruido o Seno.
if(strcmp(salida,'mem'))
    set(deviceObj.Waveform(1), 'Shape', 'User1');
end

if(strcmp(salida,'noise'))
    set(deviceObj.Waveform(1), 'Shape', 'pseudo-random noise');
end

if(strcmp(salida,'sin'))
    set(deviceObj.Waveform(1), 'Shape', 'sin');
end

% Activar la salida del canal 1
fwrite(afg, ':output1 on;');
```

Si lo que se quiere es enviar una función arbitraria al AFG la variable 'salida' tendrá el valor

'mem'. En este caso la señal de entrada 'wave' deberá ser una señal entre 0 y $2^{14} - 1$ como ya se explicó en la sección 3.1.1.1. Una vez se tiene la variable 'wave' es necesario codificarla de nuevo al formato de entrada del AFG3000S. Una vez enviado en dicho formato la señal será guardada en la memoria de usuario número 1 'USER1'.

Según el valor de la variable 'salida' la forma de onda a la salida del AFG puede ser la señal guardada en memoria (salida=mem), ruido blanco gaussiano (salida=noise) o un seno (salida=sin).

Finalmente se activa la salida del canal 1.

En el fragmento 4.4 se configura tanto la frecuencia como la amplitud de la señal de salida del AFG.

Listado 4.4: AFG3000S-Configuración de la frecuencia y la amplitud de la señal.

```
% Configurar correctamente la frecuencia. Solo se hará para los casos de
    en
% los que la salida sea la memoria interna o el seno.
% Si la salida es la memoria interna la frecuencia será la inversa del
% tiempo que duren todas las muestras introducidas.

if(strcmp(salida,'mem') ||strcmp(salida,'sin')  )
    set(deviceObj.Frequency(1),'Frequency',freq); %freq=1/t_PRN
end

% Configurar correctamente la amplitud de la señal.

set(deviceObj.Voltage(1),'Amplitude',App);

% Leer el valor que se ha guardado en 'Amplitude' internamente del AFG.
% Para comprobar que siempre se configura correctamente la amplitud
App_gen1=get(deviceObj.Voltage(1), 'Amplitude')
%Se bloqueará el programa hasta que la amplitud sea la correcta.

while(round(App_gen1*1000)<0.99*round(App*1000))||(round(App_gen1*1000)
   >1.02*round(App*1000))
    set(deviceObj.Voltage(1),'Amplitude',App);
    App_gen1=get(deviceObj.Voltage(1), 'Amplitude')
    disp('La amplitud guardada en AFG no es igual a la deseada')
end
```

Como puede observarse la frecuencia solo se configura en los casos en los que la variable 'salida' toma el valor de 'mem' o 'sin', ya que la función NOISE como es lógico no permite cambiar la frecuencia de la señal. En el caso que se desee configurar la salida en modo función arbitraria (salida=mem) el parámetro Frequency establece la frecuencia de la señal completa,

por tanto, para configurar correctamente este parámetro lo más sencillo es introducir la inversa del tiempo total que se quiere que ocupe toda la señal enviada al AFG.

El parámetro de configuración Amplitude estable la amplitud en tensión pico a pico de la señal de salida. Se observó que la configuración de dicho parámetro presentaba algunos problemas ya que no siempre tomaba el valor de amplitud que se indicaba en la función set(deviceObj.Voltage(1),'Amplitude',App). Por tanto se optó por escribir en el dispositivo el valor deseado de tensión pico a pico y posteriormente leer del dispositivo el valor de dicho parámetro. Este proceso se repetiría hasta que ambos valores fuesen los deseados.

El fragmento de código presentado en 4.5 se muestra la configuración del modo de salida de la señal y posteriormente la desconexión del dispositivo.

Listado 4.5: AFG3000S-Configuración del modo y desconexión con el dispositivo.

```
% Configura el modo Burst

if strcmp(modo, 'Burst')
set(deviceObj.Burstmode(1),'Enabled','on');
set(deviceObj.Burstmode(1),'Cycles',parametro); % parametro es en este
    caso el número de ciclos
% set(deviceObj.Burstmode(1),'DelayTime',3e-3);
end

% Configura el modo de modulacion de fase (PM)
if strcmp(modo,'PM')
    set(deviceObj.Pm(1), 'Deviation', 'max');
    set(deviceObj.Pm(1), 'Frequency', parametro); %parametro=(k*freq
        *360)/(180*4)
    set(deviceObj.Pm(1), 'Function', 'triangle');
    set(deviceObj.Pm(1), 'Source', 'internal');
    set(deviceObj.Pm(1), 'State', 'on');
end

% Configura el modo continuo deshabilitando el resto

if strcmp(modo, 'C')
    set(deviceObj.Burstmode(1),'Enabled','off');
    set(deviceObj.Pm(1), 'State', 'off');
end


%% Desconectar

% Borra el buffer
clrdevice(afg);
```

```
% Desconectar correctamente con el dispositivo
fclose(afg);
delete(afg);
clear afg;

% Si solo se quiere configurar el canal 1
if soloCH1==1
    return;
end
```

Como se puede apreciar se configura un modo u otro dependiendo del valor que tome la variable de entrada 'modo'. Si su valor es 'Brust' configurará el AFG en modo ráfaga para que a su salida la señal se repita un número de ciclos determinado por 'parametro'. Si lo que se quieren realizar son pruebas de errores dinámicos se debe configurar el modo PM o modulación en fase. Se configuran los parámetros de este modo según lo mencionado en la sección 3.4.2. Si finalmente se quiere que la señal de salida se repita continuamente se usará el modo continuo.

Para la correcta desconexión del PC con el dispositivo AFG lo primero que se realiza es la limpieza del buffer con la función clrdevice(afg). Posteriormente y para finalizar se cierra y eliminada el objeto de dispositivo USB creado inicialmente.

Para la configuración del canal 2 del AFG se siguen los mismos pasos que para la configuración del canal 1. Existe la posibilidad de configurar únicamente el canal 1 del AFG lo cual se consigue escribiendo un 1 en la variable de entrada 'soloCH1'.

4.2.2 OSC

La función osc.m es la encargada de la comunicación PC-osciloscopio. A partir de unas variables de entrada de entrada devuelve el valor en tensión de todos los puntos que componen la forma de onda de la señal que en ese instante se muestra en el osciloscopio.

En el fragmento de código 4.6 se muestra la declaración de la función osc.m así como una breve explicación de las variables de entrada de dicha función.

Listado 4.6: OSC-Declaración de la función.

```
function [ v,t,vunit,tunit] = osc( points, CH, App, autoset,tiempo_pausa
    )
%
%   function [ v,t,vunit,tunit] = osc( points, CH, autoset)
%
%   -> points son los puntos que se quieren coger
%
%   -> CH canal del que se quiere guardar las muestras,
%   1-->channel1
```

```
%   2-->channel2
%   3-->math
%
%   -> App Amplitud de la señal que se enviará al Osciloscopio en Vpp se
%   usará para ajustar adecuadamente la escala. Esta variable se puede
    usar
%   para ajustar la escala que se desea insertando un valor en App que
    se
%   corresponda con la escala deseada. Las expresiónes son:
%
%                       escala=(App/2)/4;
%                       App=8*escala;
%
%   -> autoset, si se quiere que se haga autoset se pone un 1 si no se
    pone
%   un 0
%
%   -> tiempo_pausa se realiza una pausa con el tiempo insertado en este
%   parametro en segundos.
%
```

La función osc.m devuelve las variables:

- v: Valor de voltaje en voltios de cada muestra.
- t: Tiempo en segundos de cada muestra.
- vunit: Unidades en las que se expresa la tensión.
- tunit: Unidades en las que se expresa el tiempo.

Las muestras que devuelve el osciloscopio son tomadas con una tasa de muestreo de 5GS/s y el número de muestras se configura con el parámetro 'points'.

Se puede obtener la señal del osciloscopio tanto del canal 1 como del canal 2 así como de la math que representa la suma de las señales del canal 1 y 2. Esta función math es usada en las pruebas con ruido blanco gaussiano. La variable de entrada que se encarga de elegir de que canal se quiere leer es 'CH', siendo 1 para el canal1, 2 para el canal 2 y 3 para la función math.

El resto de variables de entrada se definen a grandes rasgos en el fragmento 4.6.

A continuación se va mostrar la parte de la función osc.m encargada de la configuración de la lectura de los datos del canal 1. Posteriormente se mencionarán las pequeñas diferencias existente para configurar el resto de canales.

En el fragmento 4.7 se muestra el proceso de creación de los objetos de dispositivos así como el establecimiento de la conexión PC-osciloscopio.

Listado 4.7: OSC-Creación de los objetos de dispositvos y establecimiento de la conexión PC-Osciloscopio.

```
% Crear un VISA-USB object.
interfaceObj = instrfind('Type', 'visa-usb', 'RsrcName', 'USB0::0x0699
    ::0x0401::C020170::0::INSTR', 'Tag', '');

% Crear un objeto VISA-USB si este no existe, sino usa el objeto que ha
% encontrado

if isempty(interfaceObj)
        interfaceObj = visa('TEK', 'USB0::0x0699::0x0401::C020170::0::
            INSTR');
else
        fclose(interfaceObj);
        interfaceObj = interfaceObj(1);
end

% Crear un objeto de dispositivo.
deviceObj = icdevice('MSO4104_mod.mdd', interfaceObj);

% Configurar InputBufferSize

set(interfaceObj, 'InputBufferSize', 50000000);

% Configurar OutputBufferSize

set(interfaceObj, 'OutputBufferSize', 500000000);

% Conectar el objeto de dispositivo al hardware.
connect(deviceObj);
```

Al igual que en el caso del AFG, el primer paso es la creación de la variable interfaceObj con la función instrfind a la que se introducen entre otros parámetros de entrada el tipo de conexión, en este caso visa-usb y la dirección usb del dispositivo, en este caso 'USB0::0x0699::0x0401::C020170::0::INSTR'. Una vez creada la variable 'interfaceObj' se crea el objeto de dispositivo usb 'deviceObj' con ayuda de la función icdevice a la que se le introduce como parámetros de entrada la variable 'interfaceObj' y 'MSO4104_mod.mdd'.'MSO4104_mod.mdd' al igual que 'tek_afg3000.mdd' es una librería para Matlab que incluye algunas de las funciones más importantes de la serie de osciloscopios MSO4104, con las cuales se puede configurar el osciloscopio sin necesidad de conocer a fondo la sintaxis propia con la que se configuran estos dispositivos. Cabe destacar que ha sido necesario en la realización de este emulador la modificación de la libreria 'MSO4104.mdd' suministrada por Matlab para conseguir que la nueva librería 'MSO4104_mod.mdd' tuviera

las funcionalidades necesarias. Al igual que en el caso del AFG se ha hecho uso tanto de las funciones de la libreria 'MSO4104_mod.mdd' como de la sintaxis propia del dispositivo para configurar algunos parámetros.

Los dos comandos siguientes establecen el tamaño del buffer tanto de entrada como de salida.

Posteriormente se lleva a cabo la conexión entre el PC y el osciloscopio con la función connect(deviceObj).

Para evitar posibles errores en cada conexión PC-osciloscopio se borran los buffers.

Finalmente se establece la conexión con el comando connect(deviceObj).

En el fragmento de código 4.8 se muestra el proceso para configurar el número de muestras de la señal, la escala de amplitudes así como la posibilidad de realizar un autoset. Después se muestra la función encargada de guardar en las variables v, t, vunit, tunit los datos procedentes de la señal que se encuentra presente en el osciloscopio.

Listado 4.8: OSC-Configuración de los puntos de la señal la escala de amplitudes la posibilidad de realizar un autoset así como el envío de la señal del osciloscopio al PC.

```
% Configurar el número de muestras que se van a adquirir
set(deviceObj.Waveform(1), 'EndingPoint', points); %Max es 10000000.0

%Ajuste de escala
escala_num_1=(App/2)/4;
escala_str_1=['ch1:scale ' num2str(escala_num_1)];
fprintf(interfaceObj,escala_str_1);

% Autoset
if autoset==1
        invoke(deviceObj, 'autoset');
end

% Limpiar buffer
clrdevice(interfaceObj);

% Configure property value(s).
% set(deviceObj.Acquisition(1), 'Control', 'single');

% Se configura el tiempo de pausa
pause(tiempo_pausa);

% Se guardan en las variables v,t,vunit,tunit los datos adquiridos
% del osciloscopio
groupObj = get(deviceObj, 'Waveform');
groupObj = groupObj(1);
```

```
[v,t,vunit,tunit] = invoke(groupObj, 'readwaveform', 'channel1');
```

Finalmente se procede a realizar la desconexión entre el PC y el osciloscopio. Las lineas de código utilizadas se muestran en el fragmento de código 4.9. Se puede apreciar como primero se limpia el buffer y posteriormente se procede al cierre y eliminación del objeto de dispositivo usb.

Listado 4.9: OSC-Desconexión PC-osciloscopio.

```
% Limpiar buffer
clrdevice(interfaceObj);

% Desconectar
fclose(interfaceObj);
delete(interfaceObj);
clear interfaceObj;
```

Para el canal 2 y la función math el procedimiento es análogo a excepción de la función de lectura de la señal del osciloscopio que para el canal 2 y la función math quedará como se muestra en el listado 4.10.

Listado 4.10: OSC-Canal 2 y Math.

```

% Canal 2
groupObj = get(deviceObj, 'Waveform');
groupObj = groupObj(1);
[v,t,vunit,tunit] = invoke(groupObj, 'readwaveform', 'channel2');


% Math
groupObj = get(deviceObj, 'Waveform');
groupObj = groupObj(1);
[v,t,vunit,tunit] = invoke(groupObj, 'readwaveform', 'math');
```

4.2.3 Inicialización de los dispositivos

En esta sección se van a mostrar las dos funciones encargadas de inicializar el generador de funciones y el osciloscopio.

4.2.4 Inicialización del AFG

La función encargada de inicializar el generador de funciones o AFG se llama inicializar_AFG3000S.m y se muestra en el fragmento de código 4.11.

Listado 4.11: Función de inicialización del AFG 'inicializar_AFG3000S.m'.

```
function inicializar_AFG3000S


%% Comunicación con el instrumento

afg = instrfind('Type', 'visa-usb', 'RsrcName', 'USB0::0x0699::0x0345::
    C010029::0::INSTR', 'Tag', '');

% Crear un objeto VISA-USB si este no existe, sino usa el objeto que ha
% encontrado

if isempty(afg)
    afg = visa('tek', 'USB0::0x0699::0x0345::C010029::0::INSTR');
else
    fclose(afg);
    afg = afg(1);
end

% Crear objeto de dispositivo.
deviceObj = icdevice('tek_afg3000.mdd', afg);

% Configurar InputBufferSize a 5000000 valor por defecto 512
set(afg,'InputBufferSize',5000000);

% Configurar OutputBufferSize a 500000000 valor por defecto 512
set(afg,'OutputBufferSize',500000000);

% Conectar el objeto de dispositivo al hardware.
connect(deviceObj);

%Limpiar buffer
clrdevice(afg);

% Reset
fwrite(afg, '*rst;');

% Clear
fwrite(afg, '*cls;');


%% Desconectar
% Borra el buffer
clrdevice(afg);

% Desconectar correctamente con el dispositivo
```

```
fclose(afg);
delete(afg);
clear afg;

end
```

Como se puede observar todas las lineas de código ya han sido explicadas en la sección 4.2.1 a excepción del reset. El reset permite llevar al AFG a un modo de reset o inicial. Esta inicialización se lleva a cabo cada vez que pulsamos el boton 'Run' de la interfaz gráfica, con ello se consigue evitar problemas debidos a errores internos o situaciones de bloqueo.

4.2.5 Inicialización del osciloscopio

La función encargada de inicializar el osciloscopio se llama inicializar_osc.m y se muestra en el fragmento de código 4.12.

Listado 4.12: Función de inicialización del osciloscopio 'inicializar_osc.m'.

```
function inicializar_osc
%

% Crear un VISA-USB object.
interfaceObj = instrfind('Type', 'visa-usb', 'RsrcName', 'USB0::0x0699
    ::0x0401::C020170::0::INSTR', 'Tag', '');

% Crear un objeto VISA-USB si este no existe, sino usa el objeto que ha
% encontrado
if isempty(interfaceObj)
    interfaceObj = visa('TEK', 'USB0::0x0699::0x0401::C020170::0::INSTR'
        );
else
    fclose(interfaceObj);
    interfaceObj = interfaceObj(1);
end

% Crear un objeto de dispositivo.
deviceObj = icdevice('MSO4104_mod.mdd', interfaceObj);

% Configurar InputBufferSize
set(interfaceObj, 'InputBufferSize', 50000000);

% Configurar OutputBufferSize
set(interfaceObj, 'OutputBufferSize', 500000000);

% Conectar el objeto de dispositivo al hardware.
```

```
connect(deviceObj);

% Configura el máxima de muestras que puede adquirir en 1M. El máximo
% podría ser 10M.
fprintf(interfaceObj,'HORIZONTAL:RECORDLENGTH 1000000');


% Impedancia a 50 Ohm
fprintf(interfaceObj,'CH1:TERMINATION 50.0E+0');
fprintf(interfaceObj,'CH2:TERMINATION 50.0E+0');

% Usar todo el BW disponible
set(deviceObj.Channel(1), 'BandwidthLimit', 'full');
set(deviceObj.Channel(2), 'BandwidthLimit', 'full');

% Configurar el parametro 'Couplin' en dc
set(deviceObj.Channel(1), 'Coupling', 'dc');
set(deviceObj.Channel(2), 'Coupling', 'dc');

% Establecer la ganancia en 1
set(deviceObj.Channel(1), 'Probegain', 1.0);
set(deviceObj.Channel(2), 'Probegain', 1.0);

% Habilitar el canal 1 y el canal 2
set(deviceObj.Channel(1), 'State', 'on');
set(deviceObj.Channel(2), 'State', 'on');

% Configurar la función Math como suma del ch1 y el ch2
set(deviceObj, 'Math', 'ch1 + ch2');

% Limpiar buffer
clrdevice(interfaceObj);

% Desconectar
fclose(interfaceObj);
delete(interfaceObj);
clear interfaceObj;


end
```

Al igual que en el caso del AFG se inicializa el osciloscopio para conseguir configurarlo correctamente. El osciloscopio usado guarda la última configuración, por tanto si se usase el mismo osciloscopio para distintas aplicaciones abría que configurar algunos parámetros manualmente cada que se quisiera usar el sistema emulador. Con la función inicializar_osc.m se consigue configurar el osciloscopio de manera que el sistema emulador o plataforma de pruebas

funcione correctamente.

En el mismo código se comenta la función de cada una de las lineas de código pero a continuación se van enumerar los distintos parámetros que se configuran:

- Número máximo de muestras que puede adquirir. Se configura en 1M muestras porque es un valor óptimo para las pruebas realizadas.
- Impedancia de entrada. Se estable en 50 Ohm ya que es la impedancia de salida del AFG. Así se consigue adaptación de impedancias.
- Se usa todo el BW disponible.
- Se establece el acoplamiento en continua (dc).
- Se configura la ganancia en la unidad.
- Se activan los canales 1 y 2. Por el canal 1 generalmente irá la señal $s(t)$ y por el canal 2 la señal de ruido $n(t)$.
- Se configura la función Math como la suma de los canales 1 y 2.

Capítulo 5

Resultados

5.1 Introducción

En este capitulo se va comprobar el correcto funcionamiento del sistema emulador. Se van a generar ejemplos en Matlab para cada uno de los posibles tipos de señales que se usarán en el sistema emulador. Estas señales se harán pasar por el sistema emulador obteniendo una señal $s(t)$ que será comparada con la que idealmente se tendría que obtener. Posteriormente se mostrarán unos resultados de aplicación, es decir, se usarán las señales generadas con el emulador para realizar simulaciones sobre una de las etapas del sistema de posicionamiento local en estudio.

5.2 Resultados de validación de señales ideales

En primer lugar se va a pasar por el emulador señales a las que no se les añadirá ninguna aportación de ruido o errores dinámicos. La señal generada en Matlab pasará por etapas de conversión D/A, un canal sin ruido ni errores dinámicos, una conversión A/D y un postprocesado para adecuar la tasa de muestreo.

En la figura 5.1 se muestra el resultado de enviar un seno por sistema emulador.

Por otra parte se va a enviar la señal $m(t)$. Una vez recibida la señal en el PC se va a comparar visualmente con la señal $m(t)$ generada directamente con Matlab. El resultado es el mostrado en la figura 5.2. Para poder caracterizar mejor los errores en la señal recibida se va a proceder a realizar la correlación de dicha señal con $m(t)$ ideal. El resultado de realizar la correlación de un número de muestras tal que se correspondan a dos señales $m(t)$ consecutivas se muestra en la figura 5.3. Se ha realizado la correlación de dos señales $m(t)$ consecutivas para que se pueda observar mejor el comportamiento de la correlación.

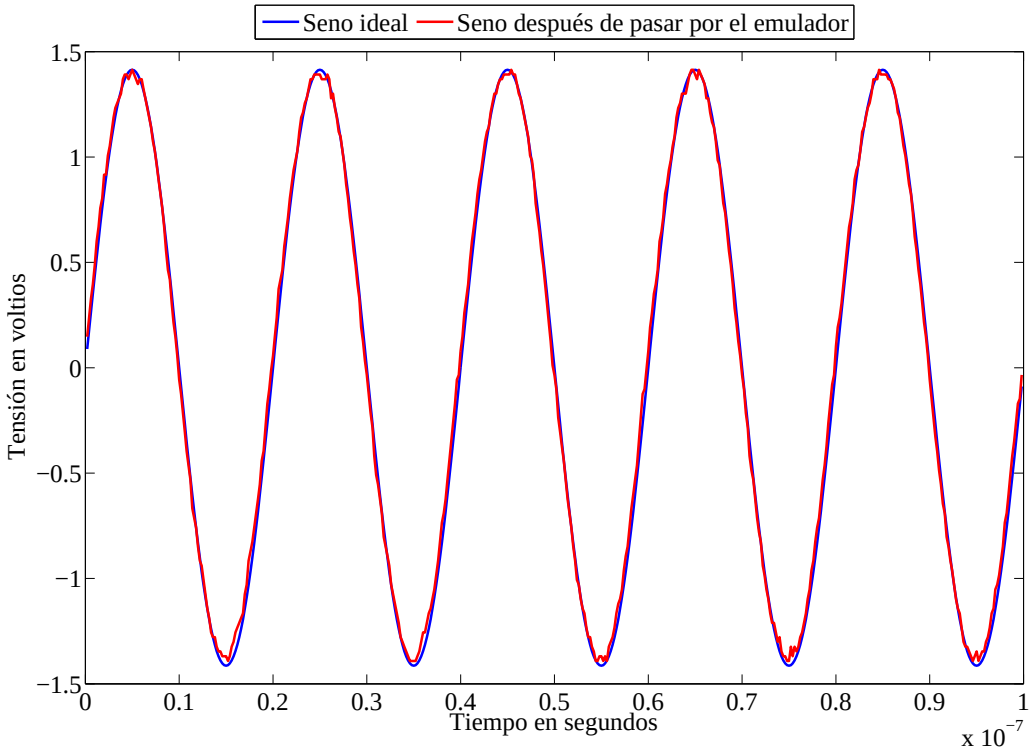

Figura 5.1: Seno ideal generado en Matlab y después de pasar por el emulador.

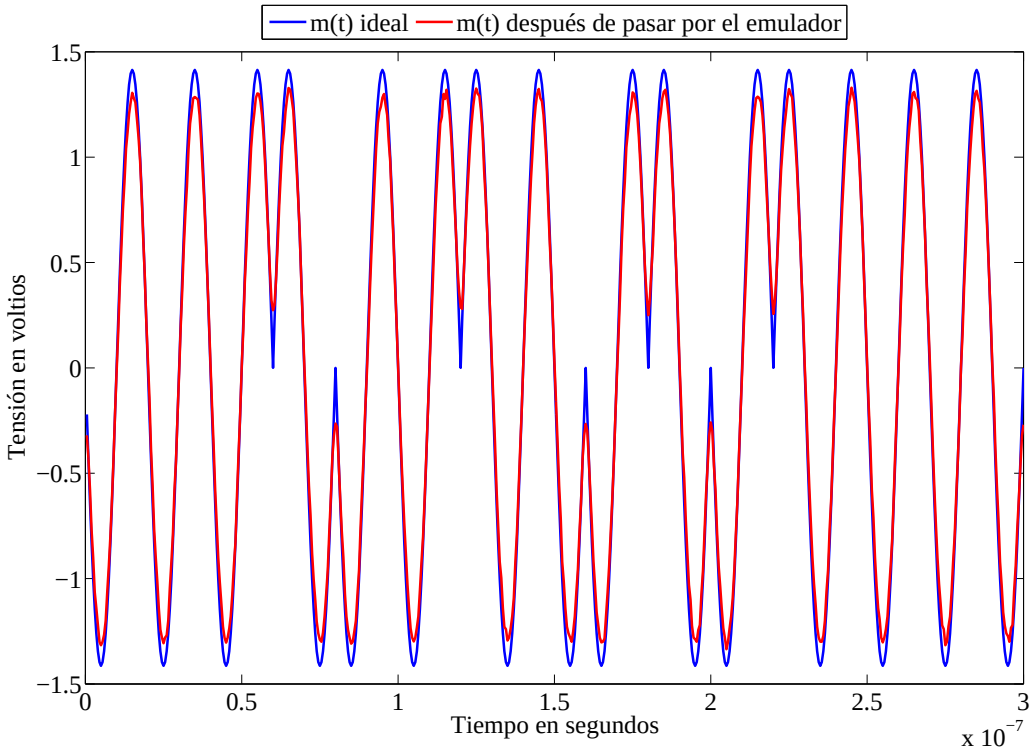

Figura 5.2: $m(t)$ ideal generado en Matlab y después de pasar por el emulador.

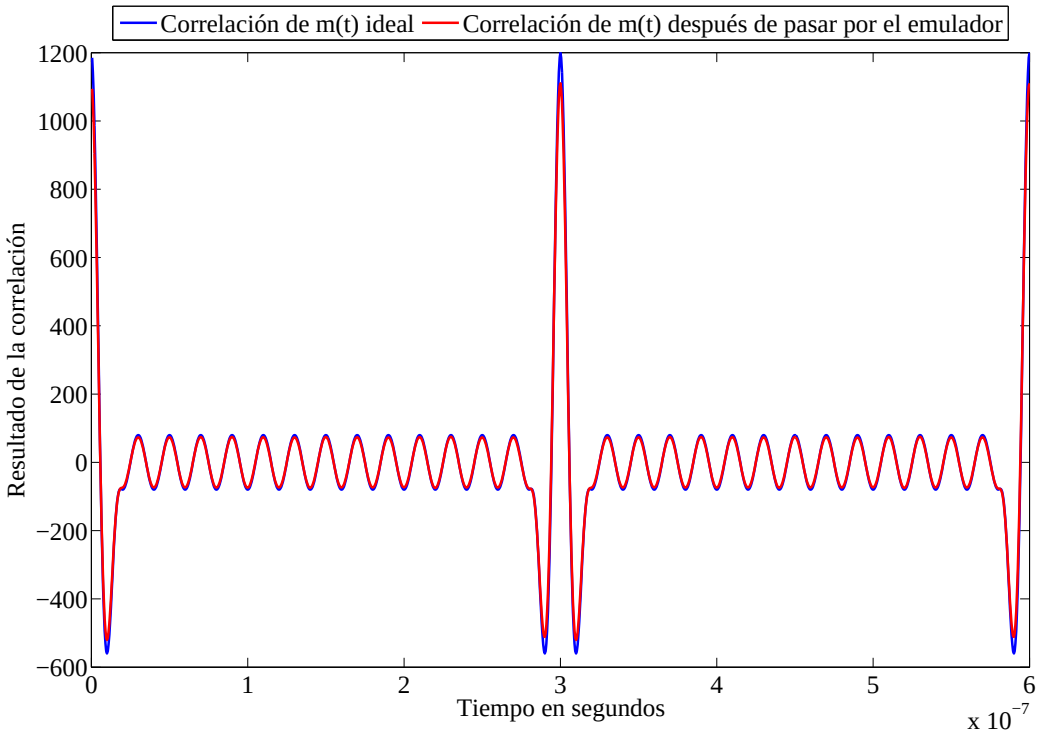

Figura 5.3: Resultado de la correlación de $m(t)$ ideal y de $m(t)$ después de pasar por el emulador.

En la figura 5.4 se muestra el código ideal y después de pasar por el emulador. Al igual que en el caso anterior se realizará la correlación de la señal ideal con la señal recibida. El resultado es el mostrado en la figura 5.5.

5.3 Resultados de validación de señales con ruido

En esta sección se va a validar la fiabilidad del apartado de ruido blanco gaussiano del emulador. Se configurará el emular a un determinado SNR y se observará después que la señal recibida presenta ese nivel SNR. Se va a mostrar a continuación la validación únicamente para el caso en el que la frecuencia de chip es 50 MHz, ya que el procedimiento para el resto de frecuencia se realizaría de forma análoga.

Para la validación de la generación de ruido blanco gaussiano a partir de un determinado valor de SNR expresado en dBHz se va a generar un tono a 50 MHz y se va hacer pasar por el emulador. Posteriormente se va comprobar si la relación señal ruido de la señal recibida es la esperada.

Los pasos a seguir son los siguientes:

- Se genera un tono de 50 MHz con una potencia de señal que permita al AFG generar una

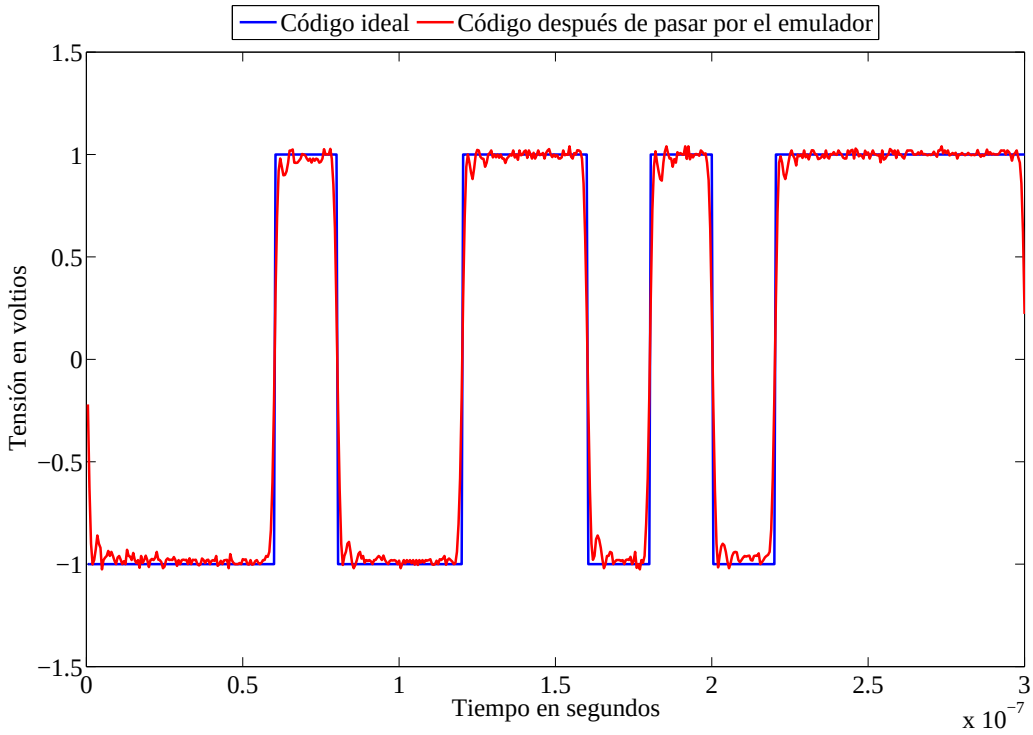

Figura 5.4: Código ideal generado en Matlab y después de pasar por el emulador.

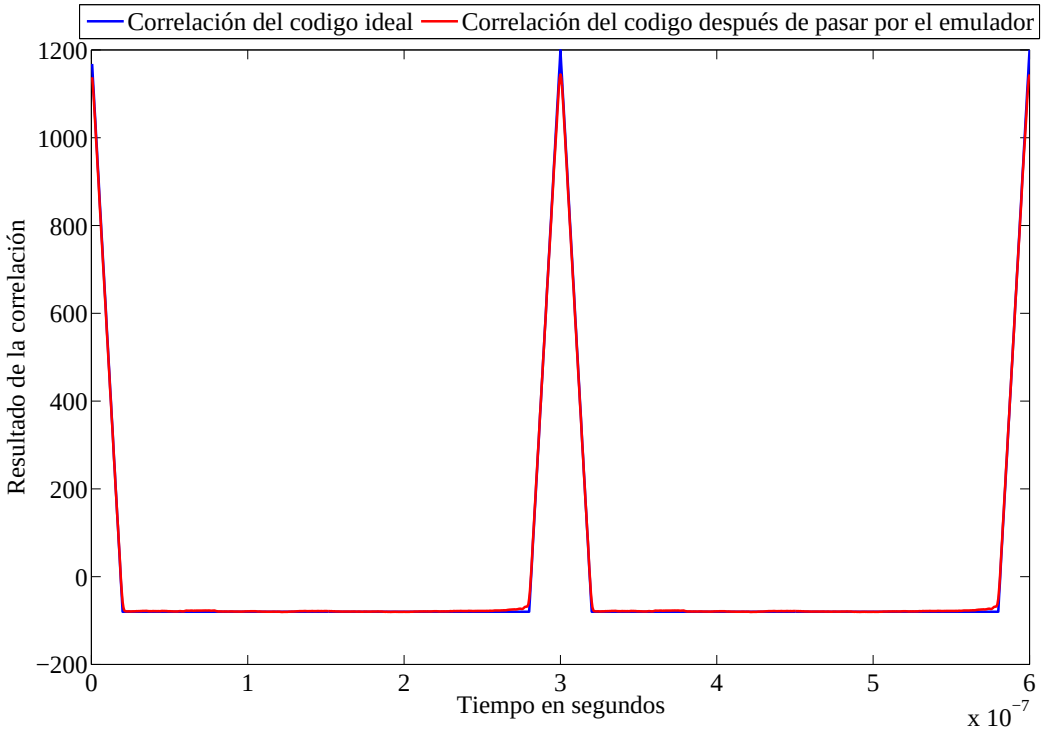

Figura 5.5: Resultado de la correlación del Código ideal generado en Matlab y del código después de pasar por el emulador.

señal de ruido sin saturarlo. El valor numérico de la potencia de señal no es relevante en este caso.

- Se indica un nivel de relación señal ruido en dBHz.
- A partir del dato de potencia de señal y SNR se calculan los valores de tensión que serán enviados al AFG.
- Se envían los parámetros de configuración al AFG y posteriormente se guardan en Matlab después de pasar por el osciloscopio.
- La señal que se recibe en el tiempo es un tono al que se le ha sumado ruido blanco gaussiano. Y en la frecuencia es una componente frecuencial muy elevada a 50 MHz y una serie de armónicos.
- Para saber cual es el valor de SNR obtenido lo que se hará será:
  - Pasar la señal recibida por un filtro banda eliminada para eliminar la componente de 50 MHz.
  - Posteriormente pasar dicha señal filtrada por un segundo filtro ahora paso bajo con frecuencia de corte 100 MHz para quedarse solo con el ruido que pasaría al sistema (el sistema filtra la señal a 100 MHz).
  - Se elimina la posible componente continua existente.
  - Se calcula la desviación estándar de la señal resultante después de pasar por ambos filtros.
  - A partir de ese valor de desviación estándar se calcula N_0 según la siguiente expresión:

    $$N_0 = \frac{\sigma^2_{\text{ruido}}}{\text{BW}} \tag{5.1}$$

    Siendo BW el ancho de banda del filtro en este caso 100 MHz.
  - Finalmente se calcula el SNR según la siguiente expresión:

    $$\text{SNR} = 10\log(\frac{P_s}{N_o}) \tag{5.2}$$

    donde P_s es la potencia de la señal enviada.

Se va a mostrar el procedimiento descrito arriba para en un ejemplo en el que se usará un tono de 50 MHz, SNR = 70 dBHz y potencia de señal 10 mW.

La señal después de pasar por el emular es la mostrada en la figura 5.6. Se puede observar como existe una componente frecuencial de alto valor para la frecuencia de 50 MHz correspondiente al tono. En la representación ampliada se muestra con mayor detalle la densidad de

potencia del ruido. Se observa como es un ruido plano en frecuencia hasta aproximadamente los 250 MHz donde empieza a decaer debido a que el ancho de banda del AFG es 250 MHz.

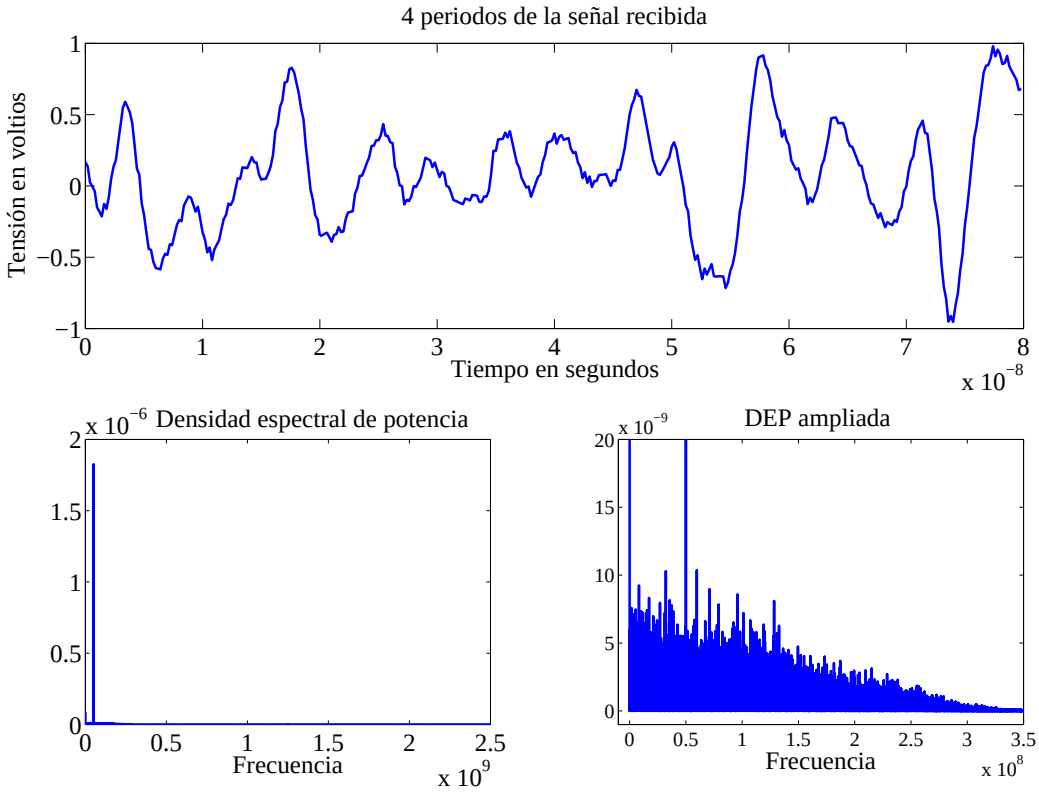

Figura 5.6: Representación en el tiempo y en la frecuencia de la señal recibida.

Después de pasar por el filtro banda eliminada para eliminar la componente del tono a 50 MHz, el filtro paso bajo para quedarse solo con las componentes de ruido deseadas y eliminar la componente continua, la señal resultante en el dominio de la frecuencia es la mostrada en 5.7. Se puede observar como el ruido está afectado por el filtro paso bajo de orden 1 que simula la limitación de banda del receptor (filtro antialiasing) hasta los 100 MHz excepto en torno a los 50 MHz que es nulo debido al filtro banda eliminada.

Una vez obtenida la señal filtrada mediante el proceso descrito anteriormente se obtienen los siguientes datos:

- Desviación estándar de ruido = 310 mV.
- $N_o = 9{,}6096 \cdot 10^{-10}$
- SNR=70.1729 dBHz

Como se puede observar el valor de SNR obtenido y el configurado solo se diferencian en un 0,26 % por lo que el emulador proporciona datos fiables.

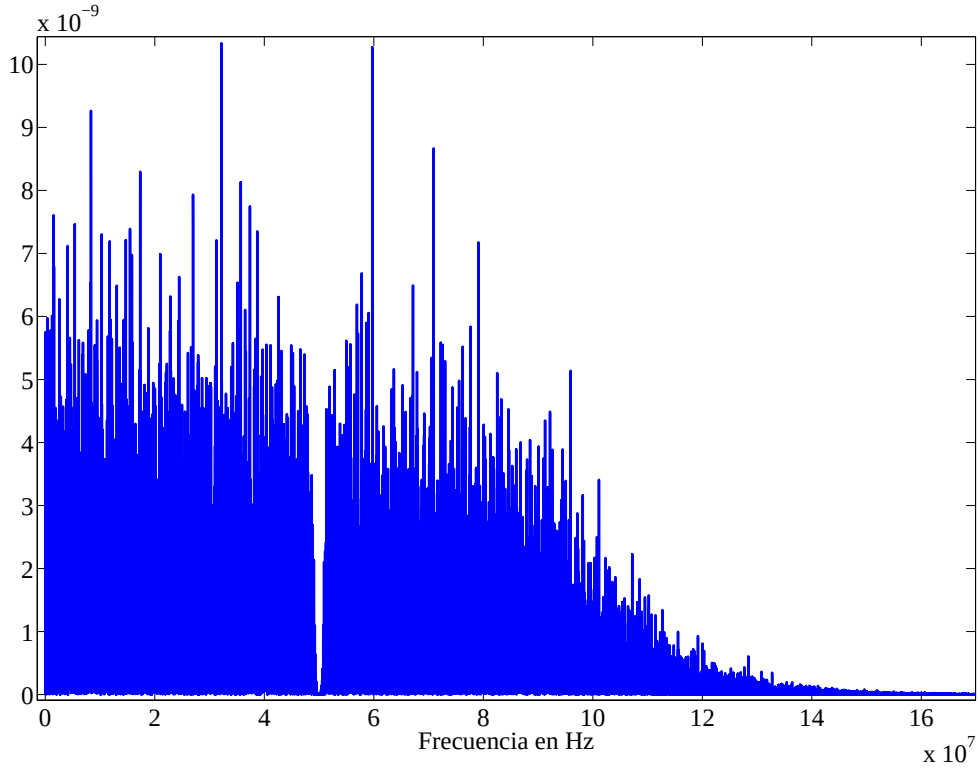

Figura 5.7: Densidad espectral de potencia de la señal después de pasar por un filtro banda eliminada y filtro paso bajo.

Siguiendo el mismo procedimiento explicado anteriormente se han realizado pruebas para SNR comprendidos entre 55 y 90 dBHz consiguiendo los resultados mostrados en la figura 5.8. Como se puede observar los valores obtenidos tienen una precisión mayor del 99 %. Dado que se ha comprobado que el funcionamiento para el resto de frecuencias de chip aceptadas por el emulador es similar, se puede concluir que el emulador reproduce fielmente las características de un canal ruidoso para cada valor de SNR determinado.

5.4 Resultados de validación de señales con errores dinámicos

Para la validación de los errores dinámicos en el emulador se va generar un seno en Matlab, el cual se hará pasar por el emulador indicando un error dinámico de $k = 1\mu s/s$ sin ruido. Una vez recibida la señal del osciloscopio, se medirá el desfase mediante una demodulación I/Q. El resultado de la demodulación I/Q indica el desfase de la señal recibida para cada instante de tiempo. Este desfase tendrá componentes de alta frecuencia (concretamente dos veces la frecuencia de la señal) y componentes de baja frecuencia que son las que indican el desfase introducido mediante la modulación PM. Las componentes de alta frecuencia serán eliminadas

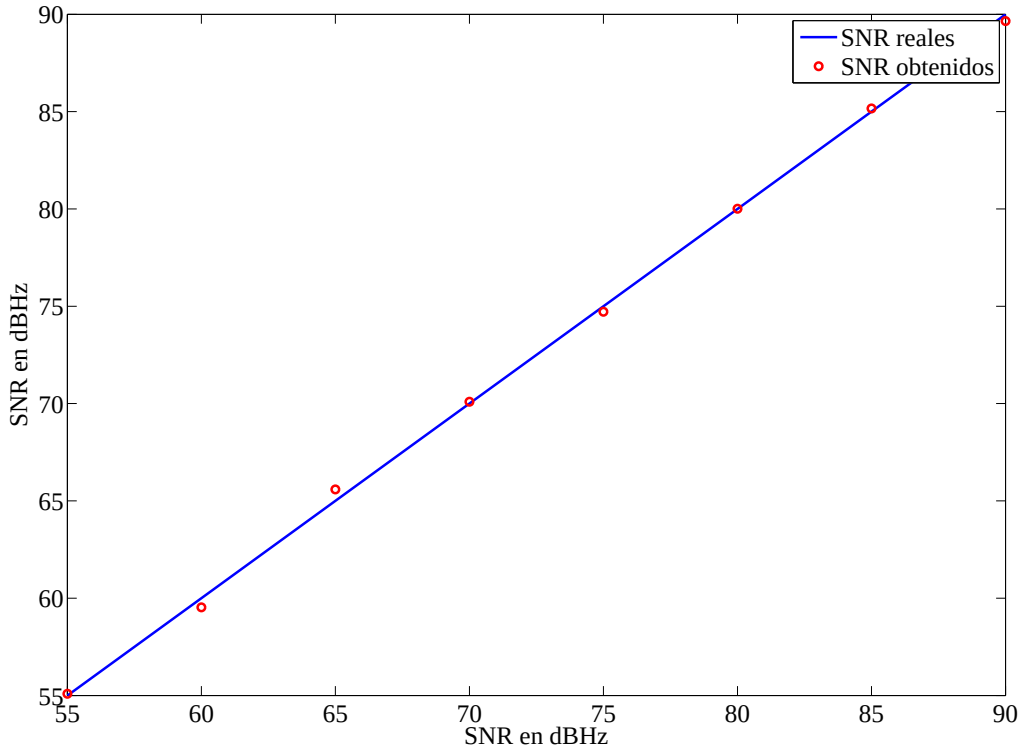

Figura 5.8: Relación señal a ruido real y esperada.

mediante un filtro paso bajo quedándose solo la información deseada.

El desfase de la señal recibida después de pasar por el demodulador I/Q es el mostrado en la figura 5.9. Una vez calculado el desfase se va a comprobar que realmente el retardo es el deseado. Para ello se siguen los siguientes pasos:

- Se mide la pendiente del retardo de la figura 5.9 resultado de la demodulación I/Q obteniendo un valor de 315,2617 rad/s.

- Para comprobar si este valor dado en rad/s se corresponde con el $1\mu s/s$ se multiplica por un factor que muestra el resultado en s/s. Dicho factor se corresponde a la siguiente expresión $2\pi f \left[\frac{\text{radianes}}{\text{segundo}}\right]$ siendo f la frecuencia de la señal, en este caso 50 MHz. Por tanto para esta señal en concreto el factor sería $\pi 10^8 \left[\frac{\text{radianes}}{\text{segundo}}\right]$.

- Entonces el valor de retardo será: $315{,}2617/\pi 10^8 = 1{,}0035 \cdot 10^{-}6 \left[\frac{s}{s}\right]$.

La figura 5.10 muestra el retardo expresados en segundos en función del tiempo también en segundos.

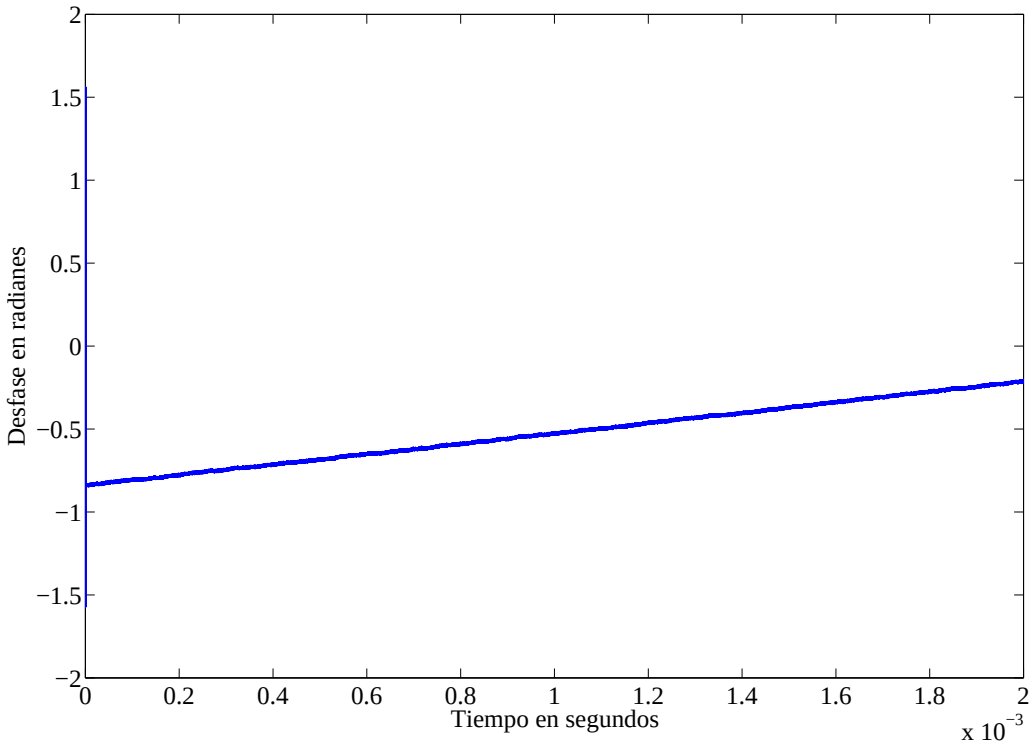

Figura 5.9: Desfase en radianes de la señal $s(t)$ recibida.

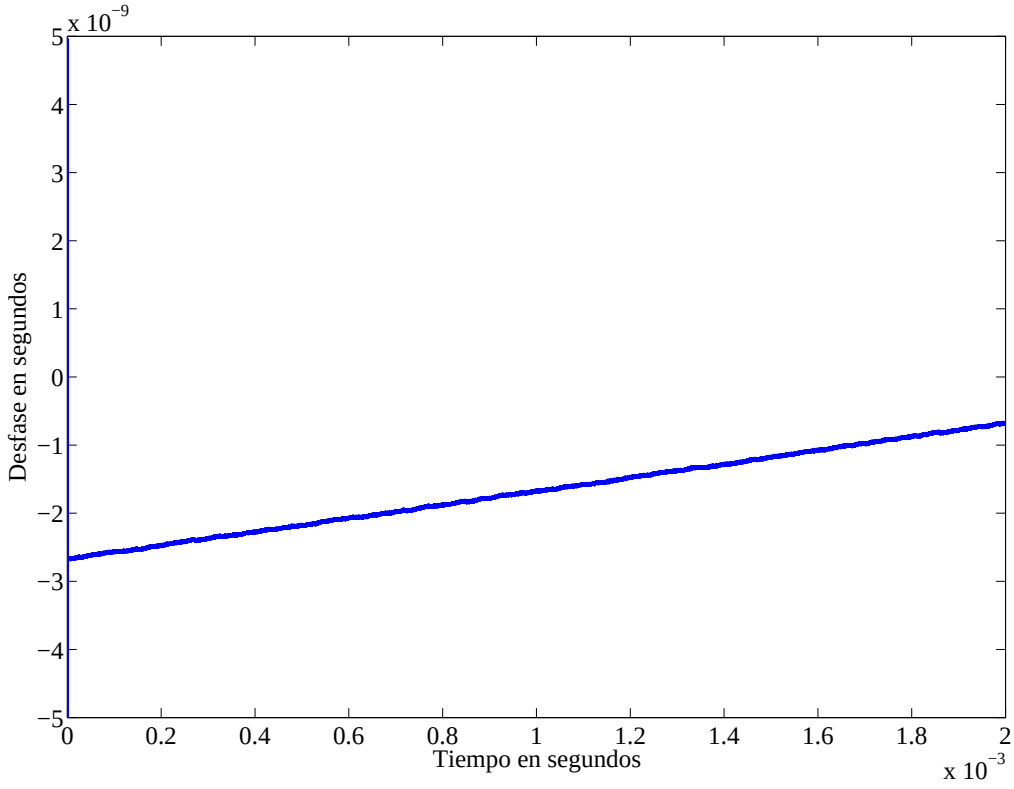

Figura 5.10: Desfase en segundos de la señal $s(t)$ recibida.

5.5 Resultados de validación de multicamino

Para la validación del emulador en cuanto al multicamino se refiere se van a generar distintos multicaminos o MP (Multipath) que serán sumados a la señal directa (LOS) y se va a observar el resultado obtenido después de pasar por el emulador.

Se va a configurar el emulador para conseguir multicaminos con la misma potencia que la señal directa y con distintos retardos. Concretamente se va a realizar cuatro pruebas distintas con cuatro retardos correspondientes a $T/4$, $T/2$, $3T/4$ y T, siendo T el periodo de la señal $m(t)$. Se han elegido estos valores ya que al ser fracciones del periodo total de la señal se puede saber a priori el resultado aproximado que se debe obtener. En la figura 5.11 se puede ver el resultado de las cuatro pruebas en las que la señal $s(t)$ es la suma del camino directo y un MP con la misma potencia que él pero cada vez con un retardo distinto. Se ha usado la señal $s(t)$ con una potencia unidad correspondiente a una amplitud de pico de $\sqrt{2}$V.

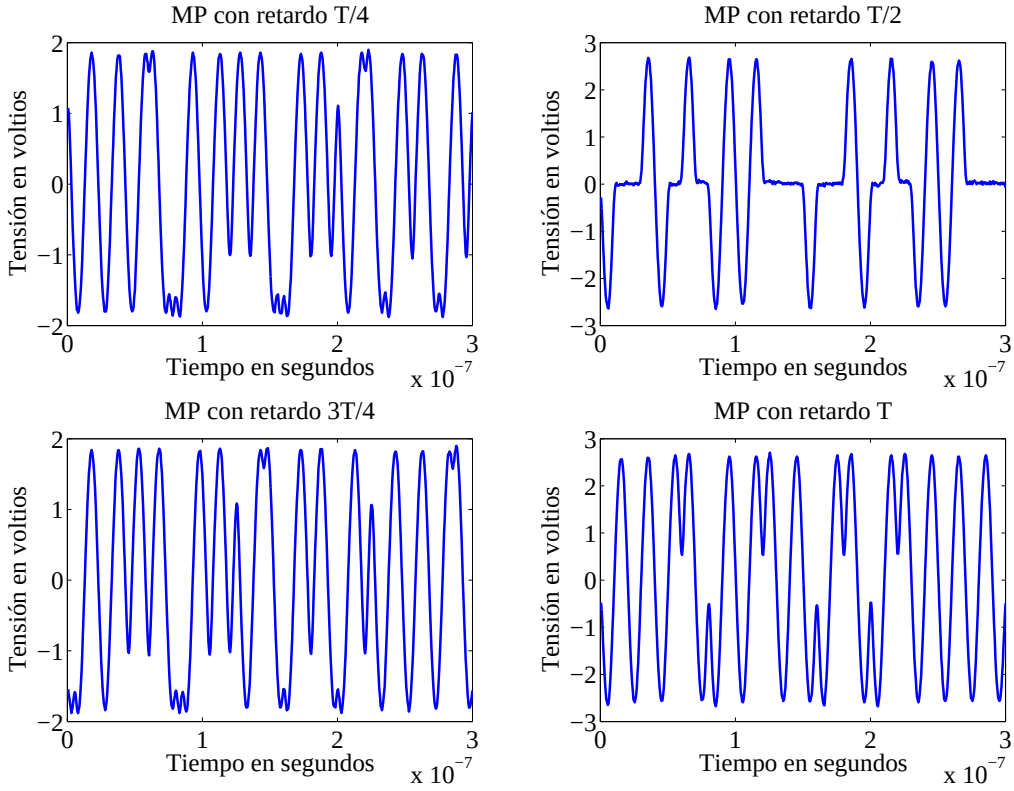

Figura 5.11: Señal $s(t)$ con MP con distintos retardos y potencia idéntica al LOS.

Se observa cómo en la prueba correspondiente al multicamino con retardo igual al periodo de la señal $m(t)$ se obtiene una señal con la misma forma que la señal $m(t)$ pero con el doble de potencia. Esto es debido a que el camino directo y el multicamino se suman en fase.

En el caso en el que el multicamino tiene un retardo de $T/2$ se observa cómo hay partes de la señal que suman en fase por tanto la amplitud es de $2\sqrt{2}V$ y otras en las que se suman en contrafase con la que la amplitud se hace 0V.

Los casos en los que los multicaminos son $T/4$ y $3T/4$ se observa como la señal resultante ya no se parece tanto a $m(t)$ si no que ahora es la suma de la señal $m(t)$ con ella misma adelantada y atrasada $T/4$.

5.6 Resultados de aplicación

En esta sección se va a mostrar el resultado de pruebas realizadas en la etapa de tracking con señales generadas a partir del emulador. Se trata de un ejemplo de como se usaría el emulador, y se utiliza también para validar mejor los resultados mostrados anteriormente. El emulador valdría para probar el sistema LPS completo, pero en este caso se ha elegido solo la etapa de tracking.

La señal $s(t)$ generada en el emulador se ha introducido a un modelo de simulación de Matlab de la etapa de tracking.

5.6.1 Ruido

Como ya se había detallado en la sección 2.4.1, la expresión teórica para definir la varianza del error en segundos al cuadrado se puede expresar como 5.3:

$$\sigma_{\hat{\tau}}^2 = T_c^2 \frac{W_L}{64\text{SNR}} \tag{5.3}$$

donde T_c es el periodo de chip del código PRN utilizado, W_L es el ancho de banda equivalente del bucle de tracking y SNR es la relación señal ruido.

En la figura 5.12 se muestra los resultados de la desviación típica, raíz cuadrada de la varianza, del ruido a partir de señales con distintos SNR y distintas frecuencias de corte del filtro. En trazo continuo se muestra el resultado teórico esperado y con círculos los resultados obtenidos aplicando a la etapa de tracking la señal obtenida del emulador.

5.6.2 Errores dinámicos

La expresión teórica del error dinámico como ya se detallo en la sección 2.4.2 tiene la forma:

$$\varepsilon_{\hat{\tau}} = \frac{\Delta_\tau}{4W_L} \tag{5.4}$$

En la figura 5.13 se muestra el error dinámico frente al ancho de banda del filtro. En trazo continuo se muestra el error dinámico teórico y con círculos, los resultados obtenidos con la señal generada en el emulador.

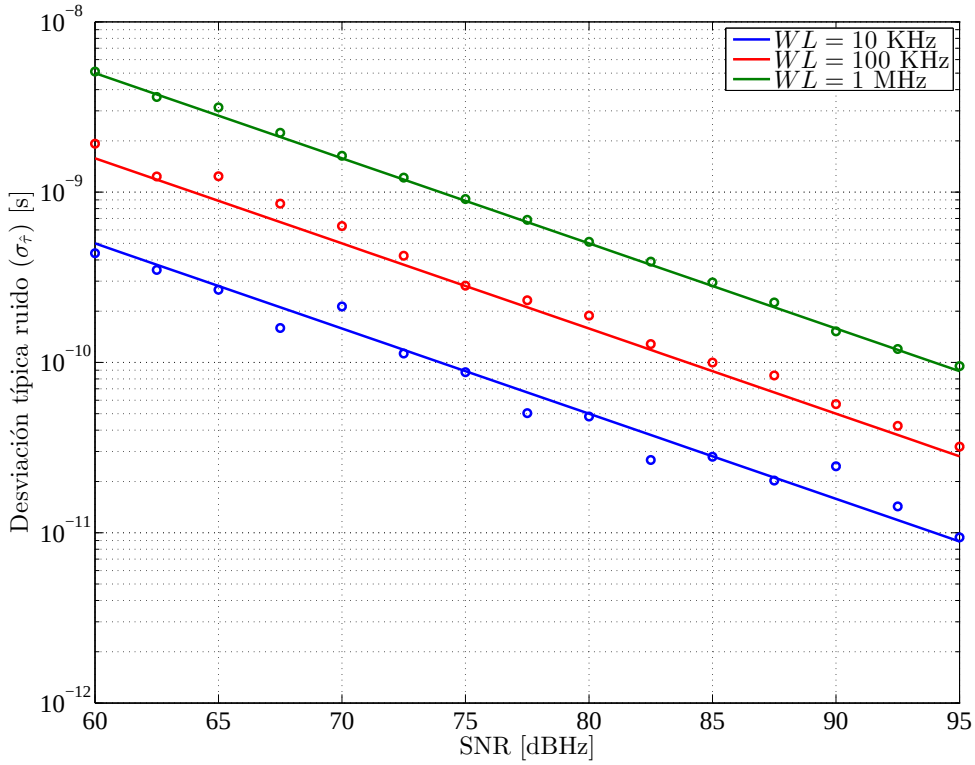

Figura 5.12: Resultados de la desviación típica del error de la etapa de tracking.

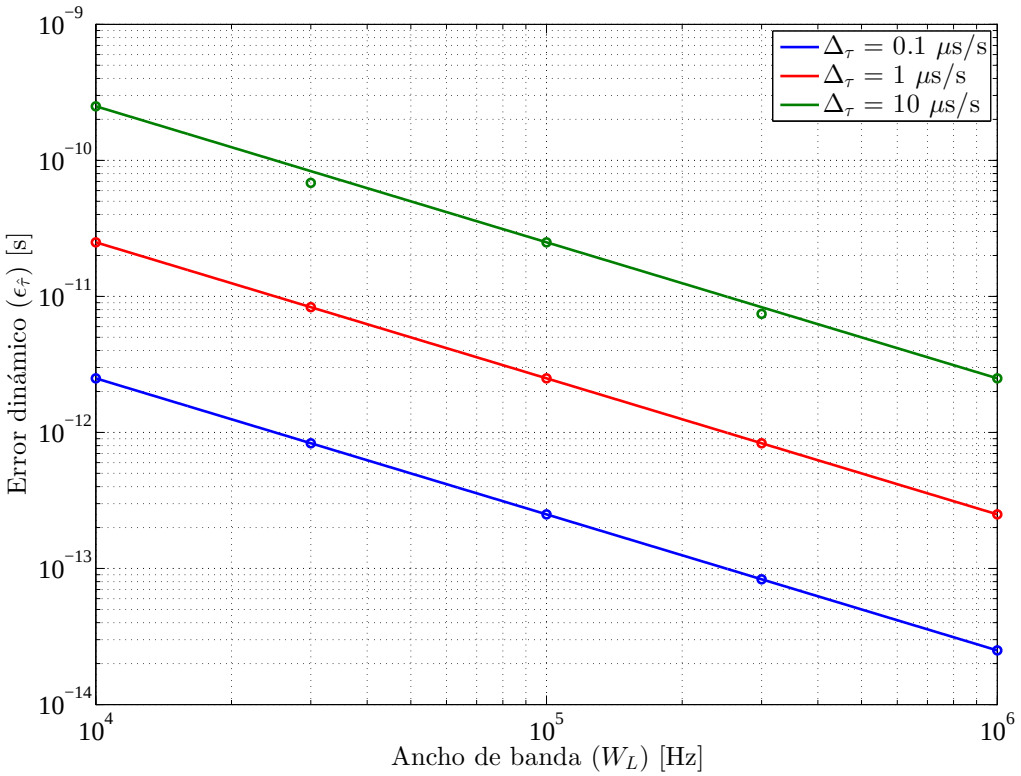

Figura 5.13: Error dinámico de la etapa de tracking.

Capítulo 6

Conclusiones y líneas futuras

6.1 Conclusiones

En el presente TFG se ha conseguido cumplir los objetivos generales y específicos planteados al inicio del documento.

Se ha desarrollado una plataforma de pruebas o emulador que consigue emular el canal existente entre emisor y receptor del sistema LPS basado en infrarrojos explicado en la sección 2.1. El emulador ofrece una señal 'realista' con la que poder validar dicha propuesta de sistema LPS. Se ha validado la plataforma de pruebas consiguiendo el resultado esperado en cada una de las pruebas que han sido realizadas.

Para la realización de la plataforma de pruebas se ha tenido que caracterizar y usar de forma adecuada los parámetros de configuración tanto del generador de funciones (AFG) como del osciloscopio para poder conseguir que funcionasen de la forma esperada.

Para facilitar la usabilidad del emulador se ha diseñado una interfaz gráfica con la que poder configurar los parámetros del canal de forma más sencilla e intuitiva. Con esto se consigue que todo el proceso de comunicación y configuración entre PC-AFG y PC-Osciloscopio sea transparente para el usuario.

Finalmente se ha realizado un ejemplo del uso principal para el cual ha sido desarrollada esta herramienta, se ha realizado una simulación de una de las etapas del sistema LPS en estudio con la señal generada en el emulador. Concretamente se han realizado simulaciones en Matlab de la etapa de tracking del sistema LPS con señales generadas en la plataforma de pruebas, consiguiendo los resultados esperados tanto para las pruebas de ruido y como para las pruebas en las que se emulaban errores dinámicos.

6.2 Líneas futuras

Una vez desarrollada la implementación de la plataforma de pruebas, se pueden especificar una serie de líneas de crecimiento que permitan evolucionar el proyecto. Se pueden destacar las siguientes propuestas de trabajo futuro:

- En cuanto a la mejora de la funcionalidad de la interfaz gráfica:
  - Dar la posibilidad de modificar todos los parámetros de configuración de la señal.
  - Permitir el uso de cualquier frecuencia de chip.
  - Dar la posibilidad de elegir entre otros tipos de señales o incluso poder cargar una señal generada anterioromente y guardada en un fichero .mat o variable de Matlab.
  - Poder elegir las variables que se desean guardan dentro de todas las que se generan.
  - Permitir visualizar en las gráficas las señales que se elijan previamente.
- En cuanto a la implementación de la plataforma de pruebas.
  - Usar otros tipos de códigos como puede ser gold, kasami, etc. Y dar la posibilidad de elegir el deseado desde la interfaz gráfica.
  - Cambiar los equipos y utilizar otros con mejores características para poder representar un mayor rango de SNR.
  - Añadir más posibles efectos adversos que emularía con mayor precisión a un enlace entre receptor y emisor más realista. Como podría ser añadir errores dinámicos no lineales.
  - Usar un emisor y receptor de infrarrojos físico que conectase el generador de funciones y el osciloscopio, en lugar de la conexión cableada existente. Este caso permitiría un mayor rango de pruebas posibles.

Capítulo 7

Pliego de condiciones

7.1 Requisitos de hardware

A continuación se presentan los requisitos hardware que son necesarios para la realización de este proyecto:

- PC.
  - Procesador: Intel Core 2 Duo E8400 3,00 GHz 64 bits.
  - RAM: 4 GB.
  - HD: 230 GB.
- Osciloscopio digital con FFT.
  - Frecuencia de muestreo: 5 GS/s.
  - Ancho de banda: 1 GHz.
- Generador de funciones arbitrarias (AFG).
  - Frecuencia de conversión: 2 GS/s.
  - Frecuencia máxima de salida (sinusoidal): 240 MHz.
- 2 cables usb tipo AB.
- 2 cables con conector BNC a ambos extremos.

7.2 Requisitos de software

A continuación se presentan los requisitos hardware que han sido necesarios para la realización de este proyecto.

- Windows 7 Enterprise 64 bits.
- MATLAB R2011b.
- TekVISA [5].
- Driver 'tek_afg3000.mdd'.
- Driver 'MSO4104.mdd'.
- Microsoft Office 2010.
- Miktex.
- TeXnicXenter.
- Inkscape.

Capítulo 8

Presupuesto

8.1 Costes por materiales

El coste por materiales queda expuesto en la tabla 8.1 siendo el total 20 €.

Tabla 8.1: Costes por materiales.

Componente	Unidades	Precio/Unidad(€)	Total(€)
Cable USB tipo AB	2	5	10
Cable conector BNC-BNC	2	5	10

8.2 Costes por equipos

El coste por equipos se desglosa en la tabla 8.2 siendo el total 1605 €.

Tabla 8.2: Costes por equipos.

Equipo	Precio (€)	Duración	Uso	Total (€)
PC Intel Core 2 Duo 3 GHz 4 GB RAM	800	4 años	6 meses	100
Osciloscopio Tektronic MSO 4104	20000	10 años	6 meses	1000
Generador de funciones Tek AFG 3252	10100	10 años	6 meses	505

8.3 Costes por recursos software

El coste por recursos software se puede ver en la tabla 8.3 siendo el total 3272,5 €.

Tabla 8.3: Costes por recursos software.

Equipo	Precio (€)	Duración	Uso	Total (€)
Microsoft Windows 7 Enterprise	135	3 años	6 meses	22.5
MATLAB R2011b	6000	1 año	6 meses	3000
Microsoft Office 2010	500	1 año	6 meses	250

8.4 Costes por tiempo empleado

El coste por tiempo empleado se muestra en la tabla 8.4 siendo el total 45900 €.

Tabla 8.4: Costes por tiempo empleado.

Función	Horas	€/Hora	Total (€)
Ingeniería	900	50	45000
Mecanografía	90	10	900

8.5 Coste total del presupuesto de ejecución material

El coste total del presupuesto de ejecución material se puede ver en la tabla 8.5 siendo el total 50797,5 €.

Tabla 8.5: Coste total del presupuesto de ejecución material.

Costes por materiales	20 €
Costes por equipos	1605 €
Costes por recursos software	3272,5 €
Costes por tiempo empleado	45900 €
Total	50979,5 €

8.6 Gastos generales y beneficio industrial

Gastos obligados por disponer de una instalación donde llevar a cabo el proyecto más el beneficio industrial, supone un recargo del 25 % sobre el presupuesto de ejecución material. En la tabla 8.6 se muestra el valor total que asciende a 12699,375 €.

Tabla 8.6: Gastos generales y beneficio industrial.

Gastos generales y beneficio industrial	12699,375 €

8.7 Presupuesto de ejecución por contrata

En este apartado se incluye el presupuesto de ejecución material, los gastos generales y el beneficio industrial, los resultados son los mostrados en la tabla 8.7, siendo el total de 63678,875 €.

Tabla 8.7: Presupuesto de ejecución por contrata.

Presupuesto de ejecución material	50979,5 €
Gastos generales y beneficio industrial	12699,375 €

8.8 Importe total

El importe total será el coste de ejecución más el 21 % de I.V.A., siendo el total de 77051,44 €.

Tabla 8.8: Importe total.

Importe total	77051,44 €

Bibliografía

[1] E. M. Gorostiza, J. L. L. Galilea, F. J. M. Meca, D. S. Monzú, F. E. Zapata, and L. P. Puerto., "Infrared sensor system for mobile-robot positioning in intelligent spaces." *Sensors*, Mayo 2011.

[2] E. M. Gorostiza, "Sistema de posicionamiento local para localización absoluta de robots móviles en espacios inteligentes mediante infrarrojos," Ph.D. dissertation, 2011.

[3] D. Salido-Monzú, E. Martín-Gorostiza, A. Wieser, J. L. Lázaro-Galilea, and F. Domingo-Pérez, "Multipath mitigation for a phase-based infrared ranging system applied to indoor positioning," in *International Conference on Indoor Positioning and Indoor Navigation*, 28-31st Octubre 2013.

[4] *Arbitrary Function Generators AFG3000C Series Datasheet*.

[5] *Tektronix Tekvisa Programmer Manual*.

[6] *Tektronix AFG3000 Series Arbitrary/Function Generators Quick Start User Manual*.

[7] *Tektronix AFG3000 Series Arbitrary/Function Generators Programmer Manual*.

[8] *MSO4000 and DPO4000 Series Digital Phosphor Oscilloscopes Programmer Manual*.

[9] *MSO4000B and DPO4000B Series Digital Phosphor Oscilloscopes User Manual*.

Apéndice A

Manual de usuario

A.1 Manual

A.1.1 Conexionado

A.1.1.1 Conexionado PC-AFG

La conexión entre el PC y el AFG se realizará según se muestra en la figura A.2 mediante un cable USB tipo AB como el mostrado en la figura A.1.

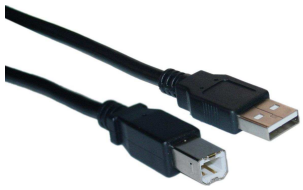

Figura A.1: Cable USB usado en la conexión de los dispositivos con el PC.

A.1.1.2 Conexionado PC-Osciloscopio

La conexión entre el PC y el osciloscopio se realizará según se muestra en la figura A.3 mediante un cable USB tipo AB como el mostrado en la figura A.1.

A.1.1.3 Conexionado AFG-Osciloscopio

La conexión entre el AFG y el osciloscopio se realizará según se muestra en la figura A.3 mediante dos cables con conexión BNC como el mostrado en la figura A.5. Por el cable representado como (1) que une el canal 1 del AFG con el canal 1 del osciloscopio irá la señal propiamente dicha y por el cable representado como (2) irá la señal de ruido blanco gaussiano usada en las pruebas de ruido.

Figura A.2: Conexionado del PC con el AFG.

Figura A.3: Conexionado del PC con el osciloscopio.

Figura A.4: Conexionado del AFG con el osciloscopio.

Figura A.5: Cable con conector BNC usado en la conexión del AFG con el osciloscopio.

A.1.2 Software

A.1.2.1 Instalación TekVISA

Para la comunicación del PC con el AFG y con el osciloscopio es necesario instalar en el PC la interfaz TekVISA que se puede descargar desde la pagina de Tektronic siguiente http://www.tek.com/oscilloscope/tds7054-software/tekvisa-connectivity-software-v400.

A.1.2.2 Driver de Matlab

Para la realización del emulador se ha hecho uso de dos drivers 'tek_afg3000.mdd' y 'MSO4104.mdd', ambos suministrados por mathworks en las siguientes páginas web http://www.mathworks.com/matlabcentral/fileexchange/24088-tektronix-afg3000-series-matlab-driver-and-example y http://www.mathworks.com/matlabcentral/fileexchange/21780-tektronix-mso4104-oscilloscope-driver. Ambos proporcionan funciones con las que poder configurar los dispositivos sin tener un conocimiento exhaustivo de los comandos propios tanto del AFG como del osciloscopio.

El driver 'MSO4104.mdd' ha sido modificado según las necesidades y usado en el proyecto con el nombre 'MSO4104_mod.mdd'.

Para poder usar la plataforma de pruebas los drivers 'MSO4104_mod.mdd' y 'tek_afg3000.mdd' deben estar en el directorio que se encuentre el fichero del sistema emulador.

A.1.2.3 Direcciones USB dispositivos

Para realizar la conexión con el dispositivo es necesario conocer la dirección física usb tanto del AFG como del osciloscopio. Se va a explicar a continuación el método para obtener dicha dirección.

Para obtener la dirección USB-ID del AFG:

- Se pulsa el botón Utility Menu (1) mostrado en la figura A.6.
- La dirección USB ID se mostrará en la pantalla principal del AFG junto con las distintas direcciones y parámetros de conexión que permite el AFG.

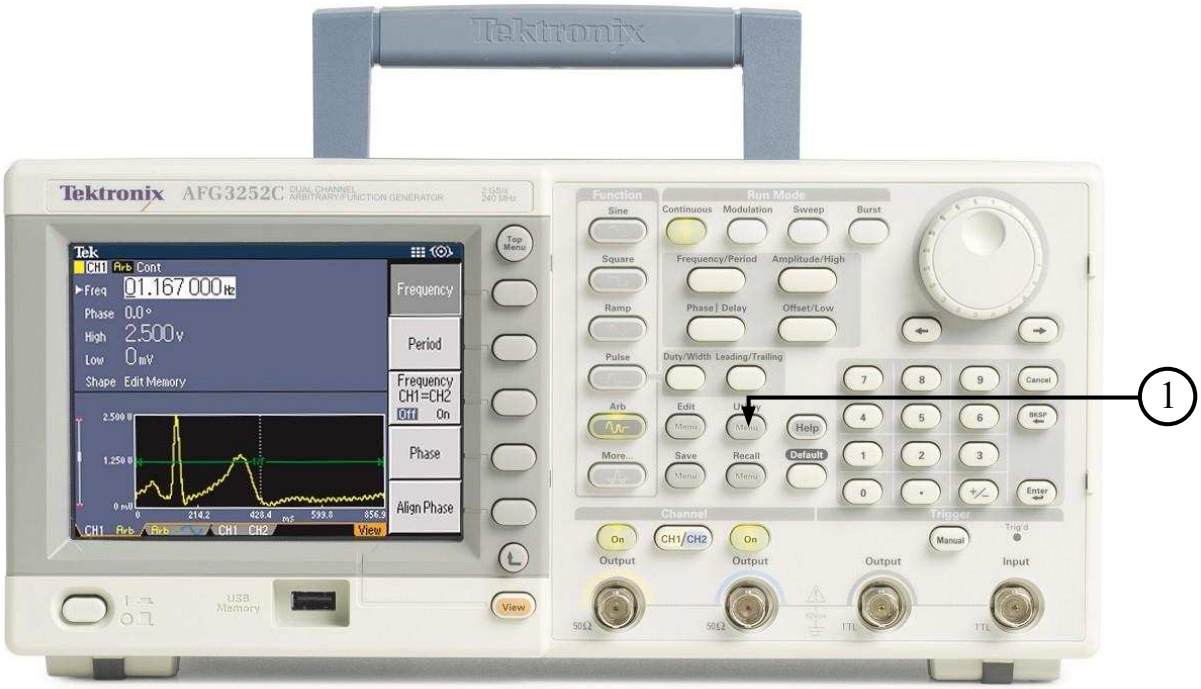

Figura A.6: Botones del AFG 3252 para la obtención del USB-ID.

Para obtener la dirección USB-ID del osciloscopio, figura A.7:

- Pulsar el botón Utility (1).
- Pulsar el botón (2) 'Utility Page' y seleccionar con la rueda (a) el valor de I/O.
- Una vez hecho esto pulsar el botón (3) 'USB Computer'

- Aparecerá un nuevo menú en la parte derecha del osciloscopio donde habrá que pulsar el botón (4) 'USBTMC Configuration'
- Aparece en la pantalla principal la dirección USB-ID compuesta por tres parámetros que son: Vendor ID, Model ID y Serial Number.

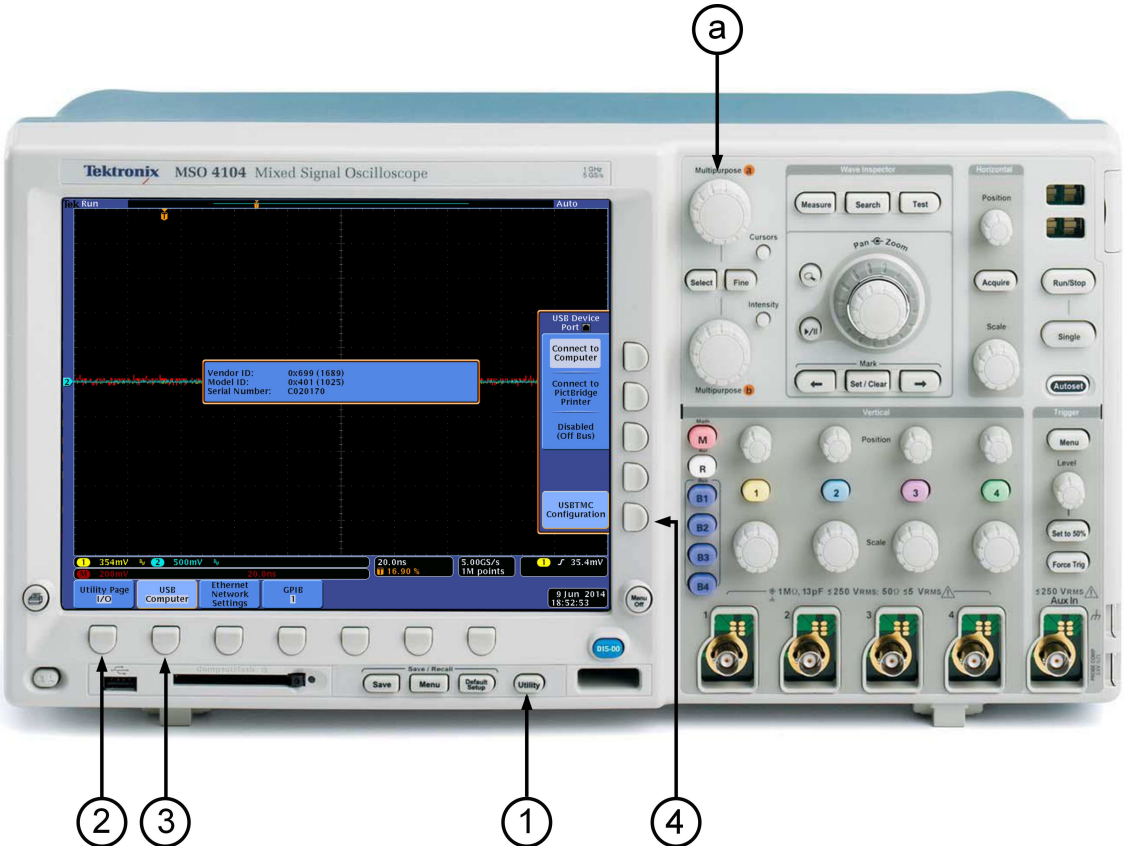

Figura A.7: Botones del osciloscopio MSO4104 para la obtención del USB-ID.

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR