industriales
etsii

Escuela Técnica Superior de Ingeniería Industrial

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería Industrial

Diseño de un Analizador Vectorial de Redes de bajo coste para la caracterización de medios dieléctricos

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Autor: **Javier Mariano Garrido López**
Director: Manuel Jiménez Buendía
Codirector: Ana Belén Toledo Moreo
Juan Domingo González Teruel

Cartagena, Julio de 2021

Universidad Politécnica de Cartagena

Índice general

Índice de figuras

Índice de tablas

Lista de Símbolos

a_n	Onda de potencia incidente	$\sqrt{W}$
b_n	Onda de potencia reflejada	$\sqrt{W}$
$\mathbf{B}$	Inducción magnética	T
c	Velocidad de la luz en el vacío	m/s
c_1, c_2, c_3	Constantes para el cálculo de la permitividad a partir del coeficiente de reflexión	-
C	Capacidad	F
$\mathbf{D}$	Desplazamiento eléctrico	C/m^2
$\mathbf{E}$	Campo eléctrico	V/m
E_D	Término de error en la directividad directa	-
E'_D	Término de error en la directividad inversa	-
E_X	Término de error en el aislamiento directo	-
E'_X	Término de error en el aislamiento inverso	-
E_S	Término de error en la adaptación de la fuente directo	-
E'_S	Término de error en la adaptación de la fuente inverso	-
E_L	Término de error en la adaptación de la carga directo	-
E'_L	Término de error en la adaptación de la carga inverso	-
E_{RT}	Término de error en las mediciones de reflexión directas	-
E'_{RT}	Término de error en las mediciones de reflexión inversas	-
E_{TT}	Término de error en las mediciones de transmisión directas	-
E'_{TT}	Término de error en las mediciones de transmisión inversas	-
f	Frecuencia	Hz

f_r	Frecuencia de relajación	Hz
$\mathbf{F}$	Fuerza	N
$\mathbf{H}$	Excitación magnética	A/m
I	Corriente eléctrica	A
j	Unidad imaginaria	-
$\mathbf{J}$	Densidad de corriente eléctrica	A/m^2
L	Autoinducción	H
n	Índice de refracción	-
$\mathbf{P}$	Vector de polarización	C/m^2
q	Carga eléctrica	C
r	Resistencia normalizada	Ω
R	Resistencia	Ω
RL	Pérdida de retorno (return loss)	dB
$RMSE$	Error cuadrático medio	-
ROE	Razón de onda estacionaria	-
S_{11}	Coeficiente de reflexión del puerto de entrada	-
$S_{11}^{L,i}$	Coeficiente de reflexión del líquido medido	-
S_{11}^{O}	Coeficiente de reflexión del aire	-
S_{11}^{W}	Coeficiente de reflexión del agua destilada	-
S_{12}	Coeficiente de transmisión inverso	-
S_{21}	Coeficiente de transmisión directo	-
S_{22}	Coeficiente de reflexión del puerto de salida	-
$S_{\mathrm{xx}a}$	Parámetro S real	-
$S_{\mathrm{xx}m}$	Parámetro S medido	-
$\tan\delta$	Tangente de pérdidas	-
T	Coeficiente de transmisión	-
v	Velocidad	m/s
V	Tensión	V
V_i	Onda de tensión incidente	V
V_n^+	Onda de tensión incidente	V
V_n^-	Onda de tensión reflejada	V
V_r	Onda de tensión reflejada	V
V_t	Onda de tensión transmitida	V
x	Reactancia normalizada	Ω
X	Reactancia	Ω
X_C	Reactancia capacitiva	Ω
X_L	Reactancia inductiva	Ω
z	Impedancia normalizada	Ω
Z	Impedancia	Ω

Z_0	Impedancia característica de la línea de transmisión	Ω
Z_L	Impedancia de la carga conectada a la línea de transmisión	Ω
β	Parámetro de caracterización de la distribución de relajación	-
Γ	Coeficiente de reflexión	-
Γ_i	Parte imaginaria del coeficiente de reflexión	-
Γ_r	Parte real del coeficiente de reflexión	-
ε	Permitividad eléctrica	C^2/Nm^2
ε'	Parte real de la permitividad compleja	C^2/Nm^2
ε''	Parte imaginaria de la permitividad compleja	C^2/Nm^2
ε_0	Permitividad eléctrica del vacío	C^2/Nm^2
ε_h	Permitividad de alta frecuencia de la relajación de la frecuencia más baja	-
ε_r	Permitividad relativa	-
$\varepsilon_r^\star$	Permitividad estimada	-
$\varepsilon_{r,L,i}$	Permitividad relativa del líquido medido	-
$\varepsilon_{r,O}$	Permitividad relativa del aire	-
$\varepsilon_{r,W}$	Permitividad relativa del agua destilada	-
ε_s	Permitividad estática	-
ε_∞	Permitividad a alta frecuencia	-
θ_i	Ángulo incidente	rad
θ_r	Ángulo reflejado	rad
θ_t	Ángulo transmitido o refractado	rad
λ	Longitud de onda	m
μ	Permeabilidad magnética	Tm/A
μ'	Parte real de la permeabilidad compleja	Tm/A
μ''	Parte imaginaria de la permeabilidad compleja	Tm/A
ρ	Densidad de carga eléctrica	C/m^3
σ	Conductividad eléctrica	S/m
σ'	Parte real de la conductividad compleja	S/m
σ''	Parte imaginaria de la conductividad compleja	S/m
τ	Tiempo de relajación	s
χ_e	Susceptibilidad eléctrica	-
ω	Frecuencia angular	rad/s
∇	Operador nabla	-

Resumen

El presente proyecto tiene su origen en la necesidad de poder caracterizar los medios dieléctricos de una forma precisa, más económica y más rápida. La medida de la permitividad es muy importante en aplicaciones científicas, industriales y médicas y, por tanto, es fundamental conocer su valor en los principales materiales dieléctricos. Para ello, el trabajo se centra en la realización del diseño de un Analizador de redes vectorial (VNA, *Vector Network Analyzer*) de tamaño reducido, portátil y de bajo coste que ofrezca una buena calidad y precisión de medida y una velocidad rápida para la medición y la calibración en la caracterización de medios dieléctricos.

El VNA diseñado dispone de un microcontrolador para gestionar todo el sistema, realizar el tratamiento de la señal y realizar la adquisición de los datos, además de otros componentes como osciladores, generador de señales a diferentes frecuencias, mezcladores de frecuencia o códec de audio, entre otros. Se ha realizado la implementación de un solo puerto de medida para realizar las medidas de reflexión. Además, se ha implementado un sistema de comunicación SDI-12 para la adquisición de datos mediante un datalogger.

En primer lugar, se ha realizado un estudio previo sobre el funcionamiento de un VNA. Seguidamente, se ha desarrollado el diseño electrónico de un prototipo de VNA usando herramientas de diseño de PCB. Después, se ha procedido a la adquisición de todos los componentes necesarios y a la fabricación del prototipo de VNA, así como a la programación del microcontrolador en lenguaje C, que se encargará de gestionar el sistema. Finalmente, se han realizado los experimentos y ensayos con el prototipo funcional para validar el funcionamiento del mismo.

Abstract

This research results from the need to characterize dielectric media in a precise, cheaper and faster way. The measurement of permittivity is very important in scientific, industrial and medical applications, being essential to know its value in the main dielectric materials. Thus, the project is focused on the design of a small, portable and low-cost Vector Network Analyzer (VNA) that offers good quality and precision of measurement and a fast speed for measurement and calibration in the characterization of dielectric media.

The designed VNA has a microcontroller to manage the entire system and perform the signal processing and data acquisition, in addition to other components, such as oscillators, signal generator at different frequencies, frequency mixers or audio codec, among others. A single measurement port has been implemented to perform reflection measurements. Moreover, an SDI-12 communication system has been implemented for data acquisition through a datalogger.

First of all, a preliminary study has been carried out on the functioning of a VNA. Next, the electronic design of a VNA prototype has been developed using PCB design tools. Then, all the necessary components have been acquired and the VNA prototype was assembled, as well as the programming of the microcontroller in C language, which will be in charge of managing the system. Finally, experiments and tests with the functional prototype have been carried out to validate its operation.

Capítulo 1

Introducción

Este proyecto se enmarca en el ámbito de la aplicación de la tecnología de la radiofrecuencia y las microondas para la determinación de las propiedades dieléctricas de los materiales con un bajo coste y con la posibilidad de realizarlo en campo.

La caracterización de medios dieléctricos se ha desarrollado de una forma muy amplia en los últimos tiempos. Para investigaciones de las características macroscópicas de los materiales, recientemente, ha habido un interés creciente en la determinación de las propiedades dieléctricas de los materiales en el rango de la radiofrecuencia y las microondas. Esto es debido a la importancia de esas propiedades para construir componentes electrónicos de alta frecuencia, las propiedades de los materiales superconductores, la calidad del sustrato de las placas de circuito impreso, la eficiencia de los materiales de absorción de microondas y el rendimiento de los diseños de antenas dieléctricas. Por tanto, debido a su amplio uso, es importante conocer las propiedades dieléctricas de los materiales y, para ello, es necesario su medida.

Existe una gran variedad de métodos para determinar las propiedades dieléctricas de los materiales. En función del equipo utilizado y de los principios de medida, se encuentran los métodos no resonantes y los métodos resonantes. Las consideraciones para la elección del método concreto para medir las propiedades dieléctricas de los materiales dependen de factores como el grosor necesario del material, el tamaño de la guía de ondas, las limitaciones de frecuencia y la precisión de las medidas. Para realizar las mediciones, habitualmente se emplea un analizador de redes vectorial como fuente, que

permite obtener datos acerca de la reflexión y la transmisión de la medida. Sin embargo, el coste de analizadores que realicen medidas de calidad, así como sus grandes dimensiones, hace que su uso no esté muy difundido y se limite a entornos de investigación o laboratorios. Por tanto, es primordial realizar estudios y desarrollos de sistemas que presenten un coste y tamaño reducidos que permitan que estos instrumentos sean más accesibles.

En el presente trabajo, se ha realizado el diseño y fabricación de un analizador de redes vectorial de bajo coste y dimensiones reducidas para obtener el valor de la permitividad de materiales dieléctricos a partir de la medida del coeficiente de reflexión.

La primera parte del proyecto se centra en el estudio y diseño del mismo. Posteriormente, se procede a la adquisición de los componentes y al montaje del prototipo, así como a la programación del microcontrolador. Finalmente, se realiza la calibración del analizador y se muestran los resultados obtenidos para cinco líquidos distintos y el aire tras la realización de los ensayos con el prototipo.

1.1. Objetivos

El objetivo principal del presente proyecto consiste en realizar el diseño y la fabricación de un analizador de redes vectorial o VNA de tamaño reducido, portátil y de bajo coste que ofrezca una buena calidad y precisión de medida y una velocidad rápida para la medición y la calibración. El VNA diseñado se utilizará para la caracterización de medios dieléctricos, como líquidos, suelos o tejidos biológicos.

Dentro de esta línea, el presente Trabajo Fin de Grado tiene los siguientes objetivos particulares:

- Realizar un estudio previo de los principios teóricos de funcionamiento de un analizador de redes vectorial. También se realizará un estudio sobre la arquitectura de los analizadores de redes vectoriales.
- Desarrollar el diseño electrónico de un prototipo de analizador de redes vectorial de bajo coste utilizando herramientas de diseño de PCB y programas de simulación. Este prototipo contendrá un microcontrolador

para la gestión del sistema, tratamiento de señal y adquisición de datos, y componentes adicionales como osciladores, generador de señales a diferentes frecuencias, mezcladores de frecuencia o códec de audio, entre otros. Se podrá realizar la implementación de dos puertos de medida, aunque se considera suficiente la implementación de un único puerto para realizar las medidas de reflexión. Se programará el microcontrolador usando el entorno y lenguaje de programación adecuados para la gestión del sistema, tratamiento de señal y adquisición de datos. El VNA diseñado será pequeño para facilitar su uso en campo.

- Realizar el diseño y fabricación de un datalogger basado en una ESP32 que registre los datos medidos por el VNA a través de comunicación UART.
- Implementar un sistema de comunicación SDI-12 para la adquisición de datos mediante un datalogger SDI-12.
- Adquirir todos los componentes necesarios para el analizador de redes vectorial, y realizar la fabricación del prototipo.
- Realizar experimentos y ensayos con el prototipo funcional para validar el funcionamiento del mismo para su uso en la medición de la permitividad.

Capítulo 2

Estado del arte

En este capítulo, se ha realizado una revisión bibliográfica previa para conocer el estado actual y los últimos avances, técnicas y desarrollos llevados a cabo sobre la materia estudiada para situar el problema planteado.

El estudio comienza profundizando acerca de los conocimientos de las líneas de transmisión, para conocer las bases que permiten entender los resultados obtenidos por el analizador de redes vectorial. Después, se realiza el análisis del origen y el funcionamiento de un VNA, incluyendo la corrección de errores, su calibración, sus ventajas e inconvenientes y los últimos progresos en su área. Finalmente, se sigue con el estudio sobre los materiales dieléctricos y los métodos que existen para caracterizarlos, con el objetivo de comprender los conceptos básicos en los que se basa el proyecto.

2.1. Líneas de transmisión

La transmisión de señales electromagnéticas puede ser radiada, si se realiza a través del aire o el vacío, o guiada, si se realiza a través de un cable que confina las ondas. Una línea de transmisión es el medio por el que se propaga la señal electromagnética, desde el generador a la carga. Un ejemplo de línea de transmisión muy utilizada es el cable coaxial. En baja frecuencia, la línea de transmisión se modela como un cable ideal, y se pueden aplicar las Leyes de Kirchhoff sin ningún problema. Sin embargo, en alta frecuencia, se

producen fenómenos ondulatorios (reflexión, desfase), y no se podrán aplicar las Leyes de Kirchhoff. Además, trabajar con tensiones y corrientes resulta difícil. En este caso, se utilizan los parámetros S o parámetros de dispersión y la Carta de Smith.

Uno de los conceptos más importantes en el análisis de las redes de alta frecuencia es la propagación de las ondas incidente, reflejada y transmitida a lo largo de las líneas de transmisión. Se puede hacer una analogía de la propagación de estas ondas con la luz y una lente, donde parte de la onda de luz incidente sobre la lente se refleja desde su superficie y la mayor parte de la luz continúa a través de la lente. Este concepto es válido para las señales de radiofrecuencia y los dispositivos y redes eléctricas, donde parte de la onda de radiofrecuencia incidente sobre el dispositivo se refleja y la mayor parte de la señal se transmite a través del dispositivo. El análisis de redes se centra en la medida precisa de las relaciones entre la señal reflejada y la señal transmitida con la señal incidente.

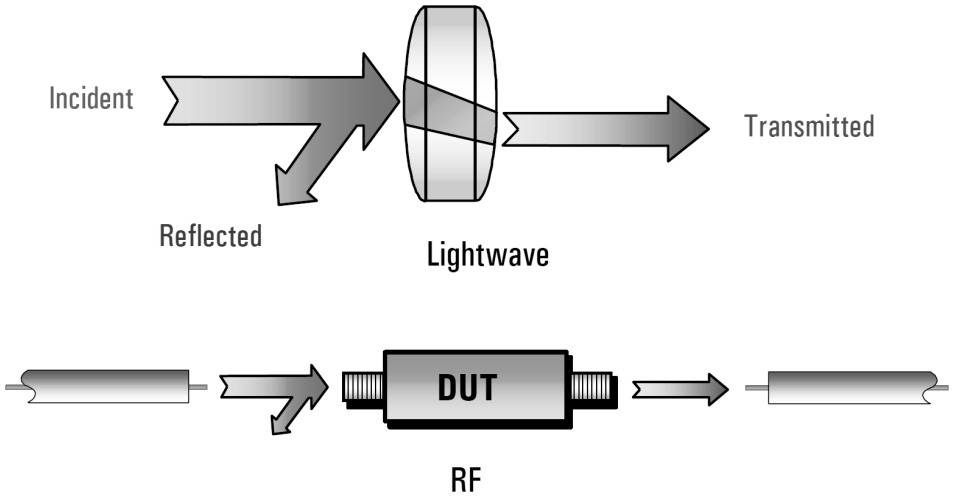


Figura 2.1: Propagación de las ondas incidente, reflejada y transmitida. Analogía con la luz [15]. DUT (*Device Under Test*) es el dispositivo bajo prueba.

2.1.1. Ondas electromagnéticas

Las ondas electromagnéticas son ondas transversales que se forman cuando un campo eléctrico entra en contacto con un campo magnético y, por tanto, están compuestas por la oscilación simultánea de un campo eléctrico con un campo magnético. El campo eléctrico y el campo magnético de una onda electromagnética son perpendiculares entre sí, y también son perpendiculares a la dirección de propagación de la onda electromagnética.

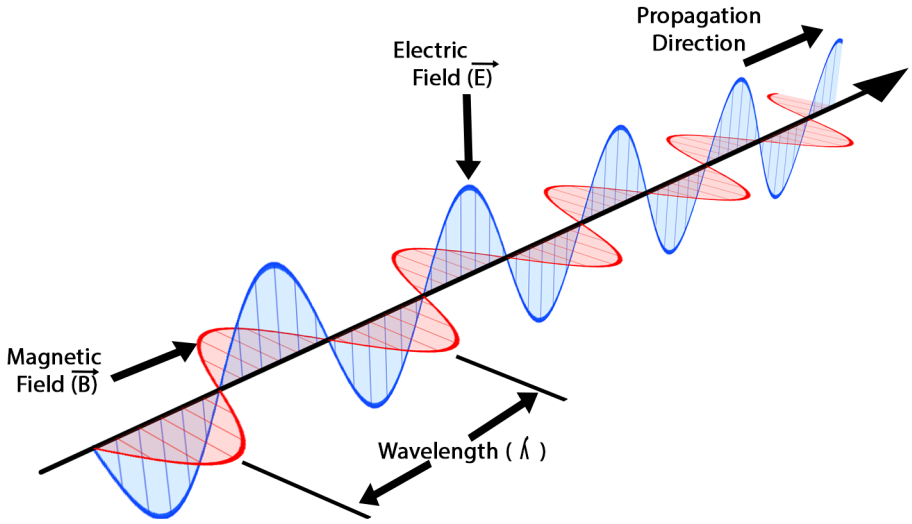


Figura 2.2: Onda electromagnética.

Las ondas electromagnéticas se propagan en el vacío a la velocidad de la luz (aproximadamente, $c = 3 \times 10^8\ m/s$). Una onda electromagnética puede propagarse en un medio material o en el vacío, es decir, no necesitan un medio material para propagarse. Podemos clasificar las ondas electromagnéticas según su frecuencia de oscilación, lo que se conoce como espectro electromagnético. Este incluye ondas de radio, microondas, infrarrojo, luz visible, ultravioleta, rayos X y rayos gamma, de menor a mayor frecuencia. La longitud de onda λ y la frecuencia f de las ondas electromagnéticas se relacionan con su velocidad v mediante la expresión:

$$v = \lambda f \tag{2.1}$$

El comportamiento de las ondas electromagnéticas está descrito por las ecuaciones de Maxwell. Además, las ondas electromagnéticas pueden experimentar los fenómenos de reflexión, refracción, difracción e interferencia.

2.1.1.1. Ecuaciones de Maxwell

Las ecuaciones de Maxwell son un conjunto de cuatro ecuaciones que describen por completo el electromagnetismo, junto con la ecuación de la Fuerza de Lorentz, que relaciona la fuerza $\mathbf{F}$ que experimenta una carga q que se mueve con una velocidad $\mathbf{v}$ en un campo eléctrico $\mathbf{E}$ y un campo magnético $\mathbf{B}$:

$$\mathbf{F} = q\mathbf{E} + q\mathbf{v} \times \mathbf{B} \tag{2.2}$$

En conjunto, las cuatro ecuaciones de Maxwell son la Ley de Gauss, la Ley de Gauss para el magnetismo, la Ley de Faraday y la Ley de Ampère generalizada.

- **Ley de Gauss**

La Ley de Gauss describe la relación entre un campo eléctrico estático y las cargas eléctricas que lo causan. El campo eléctrico apunta hacia el exterior de las cargas positivas y hacia el interior de las cargas negativas, y el flujo neto del campo eléctrico a través de cualquier superficie cerrada es proporcional a la carga encerrada por la superficie:

$$\oint_S \mathbf{D} \cdot d\mathbf{S} = q \tag{2.3}$$

$$\nabla \cdot \mathbf{D} = \rho \tag{2.4}$$

La ecuación (2.3) es la forma integral y la ecuación (2.4) es la forma diferencial. Además, ρ es la densidad de carga.

- **Ley de Gauss para el magnetismo**

La Ley de Gauss para el magnetismo establece que no existen los monopolos magnéticos. El campo magnético es generado por un dipolo magnético, y el flujo neto del campo magnético a través de cualquier superficie cerrada es cero:

$$\oint_S \mathbf{B} \cdot d\mathbf{S} = 0 \tag{2.5}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{2.6}$$

La ecuación (2.5) es la forma integral y la ecuación (2.6) es la forma diferencial.

- **Ley de Faraday**

La Ley de Faraday establece que un campo magnético variable induce un campo eléctrico. El trabajo por unidad de carga requerido para mover una carga alrededor de un lazo cerrado es igual a la variación del flujo magnético a través de la superficie cerrada:

$$\oint_C \mathbf{E} \cdot d\boldsymbol{\ell} = -\frac{d}{dt} \int_S \mathbf{B} \cdot d\mathbf{S} \tag{2.7}$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \tag{2.8}$$

La ecuación (2.7) es la forma integral y la ecuación (2.8) es la forma diferencial.

- **Ley de Ampère generalizada**

La Ley de Ampère generalizada establece que el campo magnético puede ser generado de dos formas: por una corriente eléctrica y por un campo eléctrico variable. El campo magnético inducido alrededor de un lazo cerrado es proporcional a la corriente eléctrica y a la variación del flujo eléctrico a través de la superficie cerrada:

$$\oint_C \mathbf{H} \cdot d\boldsymbol{\ell} = \int_S \mathbf{J} \cdot d\mathbf{S} + \frac{d}{dt} \int_S \mathbf{D} \cdot d\mathbf{S} \tag{2.9}$$

$$\nabla \times \mathbf{H} = \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t} \tag{2.10}$$

La ecuación (2.9) es la forma integral y la ecuación (2.10) es la forma diferencial.

2.1.1.2. Propiedades ópticas de las ondas de radio

Las ondas electromagnéticas, como ya hemos mencionado, pueden experimentar los fenómenos de reflexión, refracción, difracción e interferencia. Las ondas de radiofrecuencia y las microondas son tipos de ondas electromagnéticas, por lo que también experimentarán estos efectos. Las ondas de radio cubren el intervalo de frecuencias de 10 kHz a 300 MHz, y las microondas se encuentran en el rango de frecuencias de 300 MHz a 300 GHz.

- **Reflexión**

El fenómeno de reflexión de una onda electromagnética es el cambio de dirección de la onda al entrar en contacto con una superficie que separa dos medios distintos, volviendo al medio desde donde incidió la onda en la superficie.

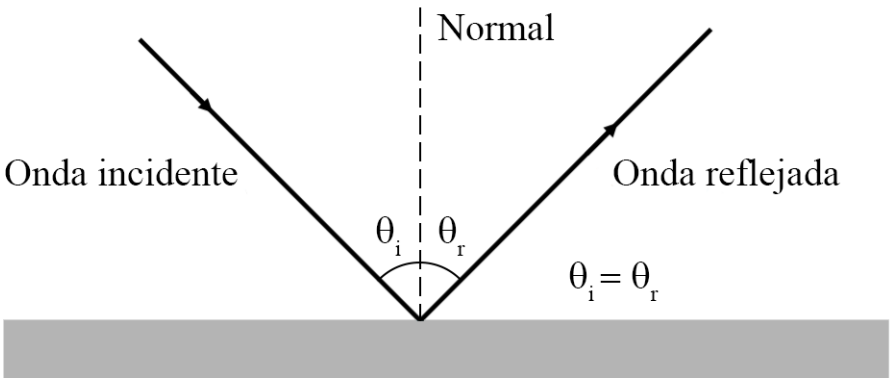


Figura 2.3: Fenómeno de reflexión de una onda electromagnética.

La velocidad de la onda incidente y reflejada es la misma, ya que están en el mismo medio. Por tanto, el ángulo de incidencia es igual al ángulo de reflexión: $\theta_i = \theta_r$.

Por otra parte, la intensidad de la onda de tensión reflejada (V_r) es menor a la intensidad de la onda de tensión incidente (V_i). El coeficiente de reflexión Γ expresa la relación entre estas magnitudes, indicando la amplitud relativa y el desfase entre ambas, de la forma:

$$\Gamma = \frac{V_r}{V_i} = \frac{\hat{V}_r e^{j\theta_r}}{\hat{V}_i e^{j\theta_i}} = \frac{\hat{V}_r}{\hat{V}_i} e^{j(\theta_r - \theta_i)} \tag{2.11}$$

Para un conductor perfecto, se cumple que $|\Gamma| = 1$ (reflexión total). Si no se produce ninguna reflexión, entonces $|\Gamma| = 0$. El coeficiente de reflexión Γ se utiliza en el estudio de las líneas de transmisión.

- **Refracción**

El fenómeno de refracción de una onda electromagnética es el cambio de dirección y velocidad que experimenta la onda al pasar de un medio a otro, con distintos índices de refracción.

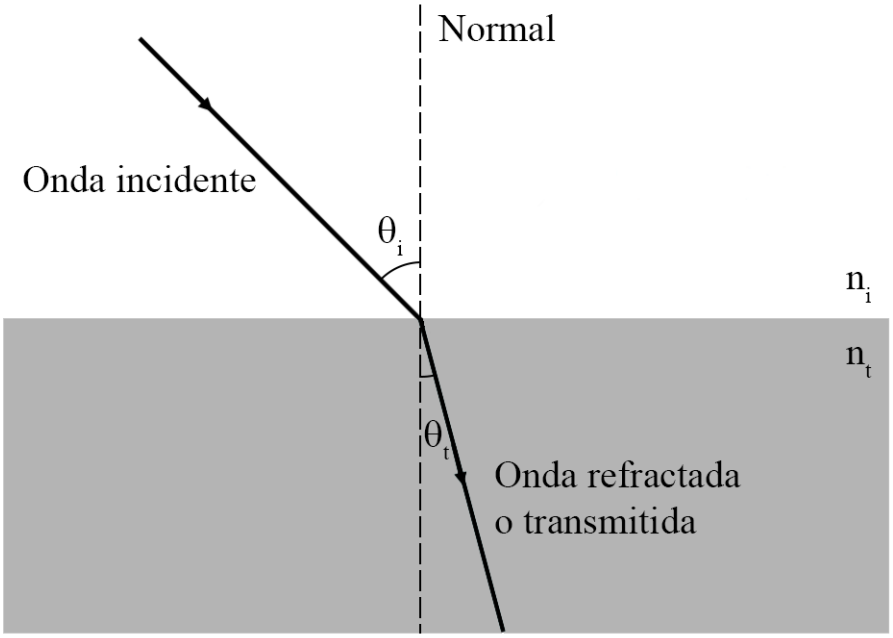


Figura 2.4: Fenómeno de refracción de una onda electromagnética.

El índice de refracción de un medio es la relación entre la velocidad de la luz en el vacío (c) y la velocidad de propagación de la onda electromagnética en el medio (v):

$$n = \frac{c}{v} \tag{2.12}$$

En la refracción, el ángulo de incidencia θ_i y el índice de refracción del primer medio n_i están relacionados con el ángulo de refracción θ_t y el índice de refracción del segundo medio n_t mediante la Ley de Snell:

$$n_i \sin\theta_i = n_t \sin\theta_t \tag{2.13}$$

En una línea de transmisión, la refracción hace referencia a la transmisión de la onda electromagnética. El coeficiente de transmisión o refracción T expresa la relación entre la intensidad de la onda de tensión transmitida (V_t) y la intensidad de la onda de tensión incidente (V_i), y también se puede calcular conociendo los parámetros de la línea de transmisión (impedancia característica Z_0 y carga Z_L) o el coeficiente de reflexión Γ:

$$T = \frac{V_t}{V_i} = \frac{2Z_L}{Z_L + Z_0} = 1 + \Gamma \tag{2.14}$$

- **Difracción**

La difracción de una onda electromagnética consiste en la desviación de la onda cuando se encuentra con un obstáculo o una rendija, los cuales se convierten en una fuente secundaria de la onda.

De acuerdo con el principio de Huygens, cada punto de un frente de onda se convierte en una fuente secundaria de ondas, que emite nuevas ondas, y el nuevo frente de onda es la envolvente de las ondas secundarias emitidas. Cuando una onda incide sobre un obstáculo, el obstáculo se convierte en una fuente secundaria de ondas, emitiendo nuevas ondas denominadas ondas difractadas, es decir, la onda se propaga como si el obstáculo se convirtiera en un nuevo centro emisor de ondas.

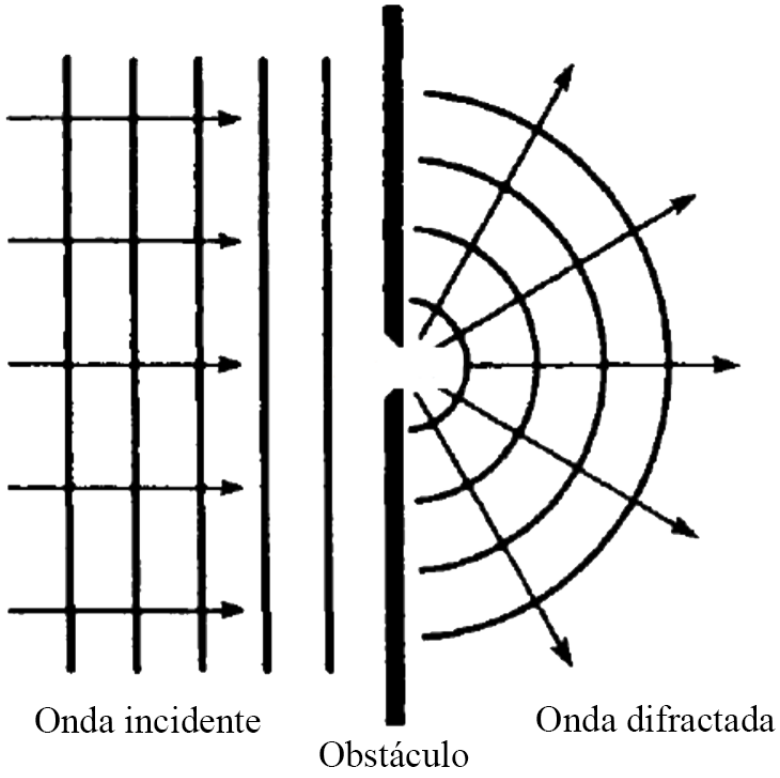


Figura 2.5: Fenómeno de difracción de una onda electromagnética.

- **Interferencia**

El fenómeno de interferencia ocurre cuando dos o más ondas se superponen formando una onda resultante, según el Principio de Superposición.

Por tanto, la interferencia de ondas de radio consiste en la combinación de la onda de radio original con distintas ondas electromagnéticas de forma que el funcionamiento del sistema se degrada, ya que la onda resultante es distinta a la onda de radio original. Dependiendo de la fase existente entre las ondas, la interferencia podrá ser constructiva o destructiva, dependiendo de si sus amplitudes se suman o se restan.

2.1.2. Impedancia

La impedancia es una medida de la oposición que presenta un circuito al paso de una corriente cuando se aplica una tensión. Es una generalización del concepto de resistencia a los circuitos de corriente alterna, y posee tanto magnitud como fase.

El símbolo para la impedancia es Z, y se define como la relación entre los fasores de tensión (V) e intensidad (I) de un circuito:

$$Z = \frac{V}{I} \tag{2.15}$$

La impedancia, en su forma general, presenta magnitud y fase o, de forma equivalente, parte real e imaginaria:

$$Z = |Z|\, e^{j\varphi} = R + jX \tag{2.16}$$

Donde:

$$\varphi = \arctan\left(\frac{X}{R}\right) \tag{2.17}$$

$$|Z| = \sqrt{R^2 + X^2} \tag{2.18}$$

La parte real de la impedancia se denomina resistencia (R). La resistencia es la responsable de la disipación de potencia y del cambio de la amplitud en las señales. Cuando la impedancia es resistiva pura ($Z = R$), no hay desfase entre la tensión y la corriente.

La parte imaginaria de la impedancia se denomina reactancia (X). La reactancia induce un desfase entre la tensión y la corriente y puede ser capacitiva o inductiva. Una reactancia capacitiva es debida a los condensadores o capacidades del circuito. Si el condensador tiene una capacidad C, la frecuencia de las señales de tensión e intensidad es f y se define la frecuencia angular como $\omega = 2\pi f$, la reactancia capacitiva se calcula como:

$$X_C = \frac{1}{j\omega C} = -\frac{j}{\omega C} \tag{2.19}$$

Por otra parte, una reactancia inductiva es debida a las bobinas o autoinducciones del circuito. Si la bobina tiene una autoinducción L, la reactancia inductiva se calcula como:

$$X_L = j\omega L \tag{2.20}$$

La reactancia total será la suma de la reactancia capacitiva y la reactancia inductiva del circuito:

$$X_T = X_L + X_C = j\omega L + \frac{1}{j\omega C} \tag{2.21}$$

De forma que:

$$X = \mathrm{Im}(X_T) = \omega L - \frac{1}{\omega C} \tag{2.22}$$

Por tanto, de forma genérica, se puede expresar la impedancia como:

$$Z = R + jX = R + j\omega L + \frac{1}{j\omega C} = R + j\left(\omega L - \frac{1}{\omega C}\right) \tag{2.23}$$

2.1.3. Parámetros S

Para caracterizar un sistema de RF o una línea de transmisión se utilizan los parámetros S, aunque existen otros parámetros. Para ello, se debe distinguir entre la onda incidente, que se propaga desde el analizador de redes hacia el dispositivo bajo prueba (DUT, *Device Under Test*), y la onda reflejada, que se propaga en dirección contraria, es decir, desde el DUT hasta el analizador de redes.

Se definen las ondas de potencia o cantidades de onda como cantidades que se miden en unidades de $\sqrt{W}$ ($M^{1/2}LT^{-3/2}$). La onda de potencia incidente (entrante) es a_n y la onda de potencia reflejada (saliente) es b_n. Por su definición, la potencia incidente en el dispositivo será $|a_n|^2$ y la potencia reflejada por el dispositivo será $|b_n|^2$, donde n indica el puerto. Las ondas de potencia son versiones normalizadas de las ondas de tensión incidente y reflejada V_n^+ y V_n^-, respectivamente. Estas ondas de tensión están relacionadas con la impedancia característica del sistema Z_0 de la forma:

$$a_n = \frac{V_n^+}{\sqrt{Z_0}} \tag{2.24}$$

$$b_n = \frac{V_n^-}{\sqrt{Z_0}} \tag{2.25}$$

Los parámetros S se definen como los cocientes de las distintas ondas de potencia. En una red de dos puertos, se encuentran los parámetros S_{11}, S_{12}, S_{21}, S_{22}. Dichos parámetros relacionan matricialmente las ondas de potencia incidentes con las reflejadas, de la siguiente forma:

$$\begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \tag{2.26}$$

También, expandiendo las matrices en ecuaciones:

$$b_1 = S_{11}a_1 + S_{12}a_2 \tag{2.27}$$

$$b_2 = S_{21}a_1 + S_{22}a_2 \tag{2.28}$$

Si se considera una onda de potencia incidente en el puerto 1 (a_1), pueden resultar ondas existentes tanto del puerto 1 (b_1) como del puerto 2 (b_2).

No obstante, si el puerto 2 está terminado en una carga idéntica a la impedancia característica del sistema (Z_0), entonces, debido al Teorema de la máxima transferencia de potencia, b_2 será absorbida totalmente haciendo $a_2 = 0$. Por lo tanto:

$$S_{11} = \frac{b_1}{a_1} \tag{2.29}$$

$$S_{21} = \frac{b_2}{a_1} \tag{2.30}$$

De forma análoga, si el puerto 1 está terminado en la impedancia característica del sistema (Z_0), entonces $a_1 = 0$, y se puede obtener:

$$S_{12} = \frac{b_1}{a_2} \tag{2.31}$$

$$S_{22} = \frac{b_2}{a_2} \tag{2.32}$$

Se definen estos parámetros S como:

- S_{11}: coeficiente de reflexión a la entrada.
- S_{22}: coeficiente de reflexión a la salida.
- S_{12}: coeficiente o ganancia de transmisión inverso.
- S_{21}: coeficiente o ganancia de transmisión directo.

Los parámetros S son números complejos. De forma genérica, se pueden tener redes de N puertos, donde la matriz de parámetros S será de tamaño $N \times N$:

$$\begin{pmatrix} b_1 \\ \vdots \\ b_N \end{pmatrix} = \begin{pmatrix} S_{11} & \cdots & S_{1N} \\ \vdots & \ddots & \vdots \\ S_{N1} & \cdots & S_{NN} \end{pmatrix} \begin{pmatrix} a_1 \\ \vdots \\ a_N \end{pmatrix} \tag{2.33}$$

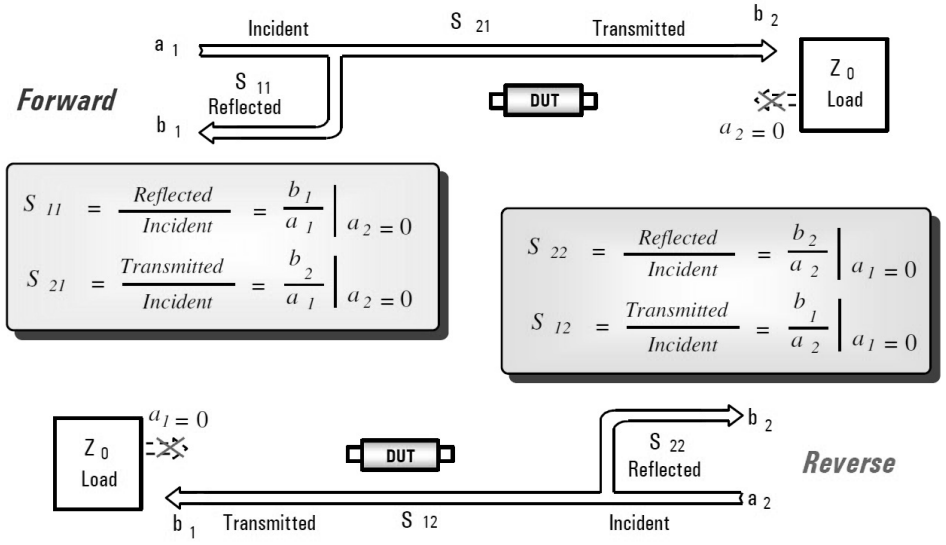


Figura 2.6: Parámetros S en una red de dos puertos [15].

2.1.4. Carta de Smith

La Carta de Smith es un instrumento gráfico de cálculo o nomograma empleado en la ingeniería de la radiofrecuencia para resolver problemas de líneas de transmisión y adaptación de impedancias. Es un diagrama que muestra cómo varía la impedancia compleja de una línea de transmisión a lo largo de su longitud. También permite relacionar un coeficiente de reflexión complejo con una impedancia compleja. En la Carta de Smith, se puede medir una impedancia normalizada z. Si la impedancia de la línea de transmisión es Z_L y su impedancia característica es Z_0, la impedancia normalizada se define como:

$$z = \frac{Z_L}{Z_0} \tag{2.34}$$

Esto se hace así ya que el valor de la impedancia característica de la línea de transmisión Z_0 no es universal, tomando habitualmente el valor de $Z_0 = 50\ \Omega$, pero puede valer 75 Ω para aplicaciones de televisión o 100 Ω para sistemas diferenciales. Por tanto, el valor de la impedancia de la línea de transmisión (Z_L) se calcula usando la impedancia característica (Z_0) de la misma:

$$Z_L = zZ_0 = R + jX \tag{2.35}$$

Si la impedancia Z_L no coincide con la impedancia característica del sistema Z_0, parte de la potencia de la fuente es reflejada hacia atrás en una onda de potencia que viaja en dirección contraria. La potencia reflejada se caracteriza por el coeficiente de reflexión Γ:

$$\Gamma = \frac{Z_L - Z_0}{Z_L + Z_0} = \frac{z-1}{z+1} \tag{2.36}$$

Y de forma equivalente:

$$z = \frac{1+\Gamma}{1-\Gamma} \tag{2.37}$$

La impedancia normalizada z y el coeficiente de reflexión Γ son números complejos, que se escriben genéricamente de la forma:

$$z = r + jx \tag{2.38}$$

$$\Gamma = \Gamma_r + j\Gamma_i \tag{2.39}$$

La equivalencia entre el coeficiente de reflexión y la impedancia normalizada, por tanto, es la siguiente, teniendo en cuenta la ecuación (2.36):

$$\Gamma = \Gamma_r + j\Gamma_i = \frac{r^2 + x^2 - 1}{(r+1)^2 + x^2} + j\frac{2x}{(r+1)^2 + x^2} \tag{2.40}$$

Y teniendo en cuenta la ecuación (2.37):

$$z = r + jx = \frac{1 - \Gamma_r^2 - \Gamma_i^2}{(1-\Gamma_r)^2 + \Gamma_i^2} + j\frac{2\Gamma_i}{(1-\Gamma_r)^2 + \Gamma_i^2} \tag{2.41}$$

La Carta de Smith se genera usando las ecuaciones (2.40) y (2.41) para representar los contornos de la impedancia z en el plano Γ. Completando los cuadrados en las ecuaciones anteriores, se obtienen las siguientes ecuaciones correspondientes a círculos:

$$\left(\Gamma_r - \frac{r}{1+r}\right)^2 + \Gamma_i^2 = \left(\frac{1}{1+r}\right)^2 \tag{2.42}$$

$$(\Gamma_r - 1)^2 + \left(\Gamma_i - \frac{1}{x}\right)^2 = \left(\frac{1}{x}\right)^2 \tag{2.43}$$

Por tanto, para representar una impedancia compleja en la Carta de Smith, el diagrama muestra:

- Circunferencias de resistencia constante r, correspondientes a la ecuación (2.42).
- Circunferencias de reactancia inductiva o capacitiva constante x, correspondientes a la ecuación (2.43).

Es decir, hay una relación directa entre la z de la Carta de Smith y el coeficiente de reflexión Γ de la línea de transmisión.

Gráficamente, en la Carta de Smith, $|\Gamma| = 1$ es la circunferencia externa de la carta, y nos indica que hay una desadaptación total (se refleja todo). Por otra parte, $|\Gamma| = 0$ es el centro de la carta, y nos indica que hay una adaptación total (no se refleja nada). Para un circuito abierto, se tienen los valores de $z = \infty$ y $\Gamma = 1$ y, para un cortocircuito, se tienen los valores de $z = 0$ y $\Gamma = -1$. Para una adaptación total, como se ha mencionado, los valores son de $z = 1$ y $\Gamma = 0$. Se puede visualizar la Carta de Smith en la siguiente figura:

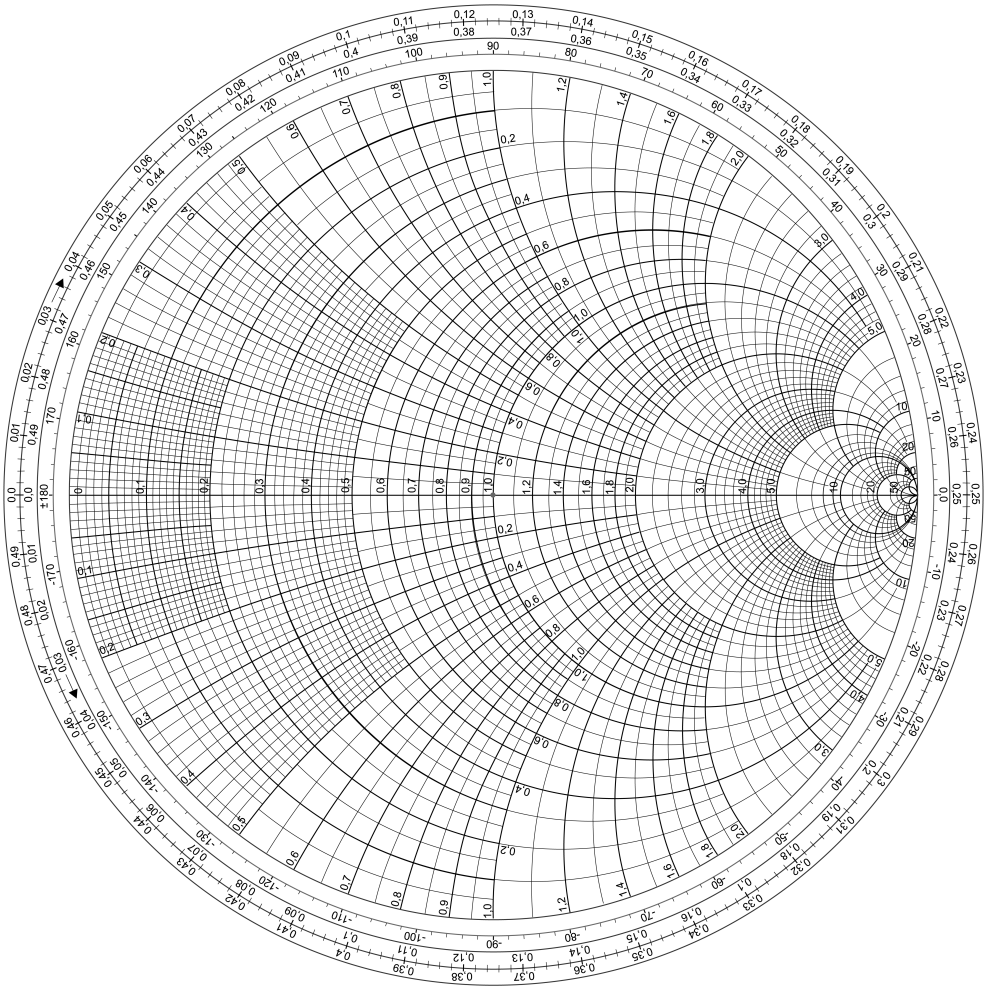


Figura 2.7: Carta de Smith [13].

2.1.5. Parámetros de una línea de transmisión

Entre los parámetros más importantes de una línea de transmisión, se encuentran los siguientes:

- **Coeficiente de reflexión**

Como se ha visto, el coeficiente de reflexión hace referencia a S_{11} y S_{22}. Su valor ideal se suele representar con la letra Γ, y es un número complejo. Puede ser calculado a partir de la impedancia compleja de la línea de transmisión, tal y como se ha introducido en el apartado anterior:

$$\Gamma = \frac{Z_L - Z_0}{Z_L + Z_0} = \frac{z-1}{z+1} \tag{2.44}$$

Se puede obtener el valor de la impedancia de la línea Z_L a partir del coeficiente de reflexión Γ:

$$Z_L = Z_0 \frac{1+\Gamma}{1-\Gamma} \tag{2.45}$$

- **Razón de onda estacionaria**

La razón de onda estacionaria, *ROE* o *SWR*, es una medida de la adaptación de la impedancia de la carga con la impedancia característica de la línea de transmisión, y se calcula como:

$$ROE = \frac{1+|\Gamma|}{1-|\Gamma|} \tag{2.46}$$

Se puede destacar que, en este caso, el valor de la razón de onda estacionaria no es un número complejo.

- **Pérdida de retorno (*return loss*)**

La pérdida de retorno o *return loss* (RL) es una medida de la potencia de la señal reflejada por una discontinuidad en la línea de transmisión, que es debida a una desadaptación entre la impedancia de la carga conectada a la línea y la impedancia característica de la misma. Se expresa en decibelios (dB), y se calcula como:

$$RL = -20 \log |\Gamma| \tag{2.47}$$

- **Coeficiente de transmisión**

Tal y como se ha mencionado, el coeficiente de transmisión hace referencia a S_{21} y S_{12}, y describe la relación entre la amplitud de la onda transmitida y la amplitud de la onda incidente en una discontinuidad en la línea de transmisión. Su valor ideal se suele representar también con la letra T, y es un número complejo. Puede ser calculado mediante:

$$T = \frac{2Z_L}{Z_L + Z_0} = 1 + \Gamma \tag{2.48}$$

2.2. Analizador de redes vectorial (VNA)

Un analizador de redes vectorial o VNA es un instrumento que permite analizar las propiedades de las redes eléctricas, especialmente los parámetros de dispersión o parámetros S, a altas frecuencias, que dan información acerca de la reflexión y la transmisión de las señales eléctricas. Ofrece una medida de la magnitud y fase (vector) de los parámetros de dispersión haciendo un barrido en frecuencia. Los VNA se suelen utilizar para probar componentes o dispositivos y verificar las simulaciones de los diseños con el fin de comprobar que los sistemas y sus componentes funcionan correctamente. Frecuentemente, los analizadores de redes se emplean para caracterizar redes de dos puertos, como amplificadores y filtros, pero se pueden usar en redes con cualquier número de puertos. También, a partir de los datos de los parámetros de dispersión, se pueden determinar las propiedades dieléctricas del medio de transmisión bajo estudio.

Por otra parte, las medidas vectoriales de un VNA pueden ser transformadas al dominio del tiempo, lo cual otorga más posibilidades a la hora de interpretar y procesar datos. Además, entre las razones por las que se prueban los dispositivos con analizadores de redes, se encuentran que: los dispositivos se pueden usar como componentes de otros sistemas de radiofrecuencia mayores, por lo que es necesario conocer si el componente funciona correctamente para un funcionamiento preciso del sistema; también garantiza que no haya distorsión en una señal de comunicación, teniendo en cuenta parámetros lineales y no lineales de la señal; y asegura una buena absorción y aprovechamiento de la potencia.

2.2.1. Origen y evolución del VNA

Los analizadores de redes de radiofrecuencia (RF) y microondas han permitido la evolución de los componentes de alta frecuencia y su diseño, y viceversa. La capacidad de medir las propiedades esenciales de circuitos y dispositivos (transmisión, reflexión e impedancia) permite a los ingenieros optimizar el rendimiento de sus componentes, como amplificadores, convertidores de frecuencia, separadores de señal y dispositivos de filtrado [31].

En la década de 1960, la tecnología de semiconductores estaba comenzando a afianzarse. Los muestreadores basados en diodos semiconductores se convirtieron en los componentes fundamentales de la instrumentación. Estos se utilizaron para muestrear formas de onda y permitir que se realizasen mediciones de amplitud y fase relativas en las señales. Además, las fuentes de señal basadas en osciladores de onda regresiva (BWO) permitieron tomar medidas en un amplio rango de frecuencias. De aquí, surgieron los primeros analizadores de redes vectoriales, y fue como consecuencia de la necesidad de realizar medidas precisas en el rango de la radiofrecuencia y las microondas. Un ejemplo de analizador capaz de realizar mediciones de barrido de fase y amplitud fue el analizador de redes de RF Hewlett-Packard 8407, que permitió la comparación de la amplitud y fase de dos formas de onda de hasta 110 MHz y, en 1967, HP presentó el analizador de redes 8410, que amplió la capacidad de barrido a 12 GHz, y fue un sistema de sobremesa basado en múltiples cajas integradas, como se puede observar en la Figura 2.8.

Figura 2.8: Analizador de redes de sobremesa HP 8410 [31].

Como consecuencia del desarrollo de los analizadores de redes, se desarrollaron los cables coaxiales de precisión como estándares de referencia para impedancias, usados para transportar señales eléctricas a altas frecuencias, empleando conductores de muy alta conductividad y un dieléctrico. También se desarrollaron nuevas técnicas para garantizar la efectividad de las medidas realizadas con los analizadores de redes vectoriales.

Al mismo tiempo, el concepto de parámetros S estaba comenzando a desarrollarse. Esto permitió situar la transmisión, la reflexión y la impedancia en una única representación bidimensional que podía medirse y visualizarse rápidamente. Esta revolución en el diseño de alta frecuencia permitió a los ingenieros utilizar los nuevos semiconductores de alta frecuencia que se estaban desarrollando.

En 1970, se estaban desarrollando ordenadores que podían expandir las capacidades de los instrumentos, y esto permitió la creación nuevos analizadores de redes que aportaron matemáticas de corrección de errores, mediciones por pulsos y otras capacidades. En la década de 1980, la unión de fuentes de banda ancha, muestreadores mejorados y microprocesadores más rápidos dio lugar al desarrollo de analizadores de redes vectoriales más avanzados, que permitieron establecer el estándar de metrología para mediciones de microondas e introdujeron muchas mejoras en el diseño de componentes. Además, permitían la corrección completa de errores y eran de menor tamaño. En la década de 1990, hubo un gran auge en el desarrollo de dispositivos inalámbricos. Este fue el primer mercado de consumo de alta frecuencia y el analizador de redes, que era una herramienta de I + D, se convirtió en un dispositivo de fabricación convencional, y la capacidad de proporcionar mediciones rápidas se volvió muy importante. En la década del 2000, el nivel de integración de los dispositivos de RF y microondas comenzó a expandirse rápidamente, y los analizadores de redes evolucionaron de instrumentos de dos puertos a instrumentos de cuatro puertos para aplicaciones de producción en masa, y también se desarrollaron analizadores de red con capacidad de hasta 67 GHz.

Figura 2.9: Analizador de redes vectorial de cuatro puertos [31].

Más tarde, se introdujeron analizadores de redes vectoriales que extendían el número de puertos que podían medir (sistemas de N puertos). Estos sistemas utilizan conmutadores y acopladores internos para integrar el equipo de prueba con el analizador, lo que ofrece un rendimiento que es comparable al de los sistemas de dos o cuatro puertos. Se desarrollaron analizadores de redes de 8, 12, 16 y 32 puertos, para determinar más características de las líneas de transmisión. También, se desarrollaron nuevas técnicas para acortar el proceso de calibración de los analizadores de redes sin comprometer la calidad de la medición.

Desde 2010, surge la necesidad de analizadores de redes portátiles de alto rendimiento, ya que cada vez más técnicos e ingenieros necesitan realizar mediciones precisas en condiciones adversas. Se han creado analizadores con este formato, que permiten realizar pruebas de cables y antenas, análisis de redes vectoriales, análisis de espectro, mediciones de potencia, análisis de interferencias y mediciones de voltaje vectorial.

Figura 2.10: Analizador de redes vectorial portátil.

2.2.1.1. VNA en la actualidad

Actualmente, las necesidades están cambiando para los VNA. Las tendencias principales son las necesidades de probar dispositivos multipuerto en mucho menos tiempo sin sacrificar la precisión, probar varios dispositivos en una sola estación de prueba y reducir el tamaño de las estaciones de prueba. Además, las pruebas deben poder abordar la creciente complejidad de las obleas de silicio, los dispositivos inalámbricos, los sistemas de radar avanzados y más dispositivos, que continúan incorporando más capacidad en menos espacio.

En conclusión, se puede observar que los avances y evolución de los analizadores de redes vectoriales y de los componentes de alta frecuencia son complementarios, permitiendo acelerar el progreso tecnológico. Principalmente, un VNA es un instrumento utilizado para determinar las características fundamentales de las redes eléctricas y circuitos de radiofrecuencia. Sin embargo, existen otros campos donde los VNA pueden ser utilizados, como caracterizar propiedades de los materiales a distintas escalas, lo cual lo hará una herramienta esencial durante muchos años.

2.2.1.2. VNA para medios dieléctricos

Partiendo de estas ideas, con este Trabajo Fin de Grado se pretende diseñar y construir un VNA de tamaño reducido, portátil y de bajo coste que ofrezca una buena calidad y precisión de medida, así como una velocidad rápida para la medición y la calibración. En concreto, este VNA se utilizará para la caracterización de medios dieléctricos, midiendo materiales como líquidos o el suelo, aunque sus aplicaciones pueden ser múltiples, como realizar la medida de cualquier línea de transmisión o red eléctrica, caracterizar un tejido para la detección de cáncer, etc.

Existen proyectos orientados a realizar analizadores de redes vectoriales con este formato. Un proyecto enfocado a esto tiene el nombre de NanoVNA, desarrollado por Tomohiro Takahashi [42], que se puede ver en la Figura 2.11, y que se ha tomado como referencia para la realización de este proyecto. Se trata de un analizador de redes vectorial de mano de pequeño tamaño, que posee batería, una pantalla LCD táctil y permite realizar el barrido de frecuencias hasta los 1,5 GHz, entre otras cosas.

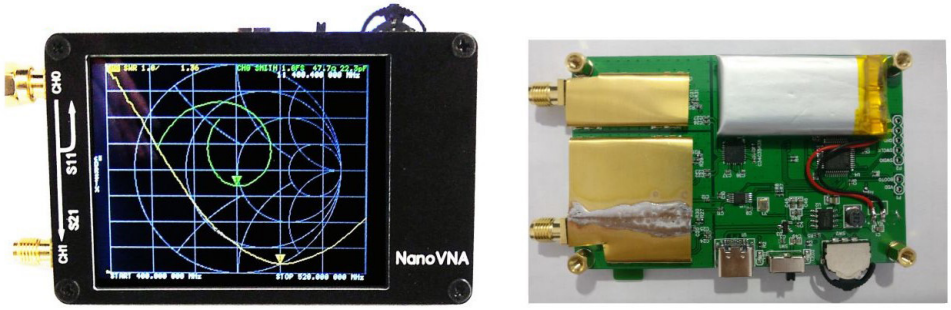


Figura 2.11: NanoVNA [42].

2.2.2. Tipos de analizadores de redes

Existen dos tipos de analizadores de redes que se pueden utilizar para analizar redes de radiofrecuencia. Estos tipos de analizadores de redes son diferentes, pero ambos pueden medir los parámetros de los componentes y dispositivos de RF de diferentes formas:

- **Analizador de Redes Vectorial (VNA)**. Mide propiedades de amplitud y fase. En concreto, mide la diferencia en amplitud y fase entre la señal de prueba y la respuesta del dispositivo. Se trata de los analizadores que poseen un precio más elevado, y también son complejos de diseñar y construir.
- **Analizador de Redes Escalar (SNA)**. Mide propiedades de amplitud, únicamente. Específicamente, mide la diferencia en amplitud entre la señal de prueba y la respuesta del dispositivo. Este tipo de analizadores son más simples y tienen un coste menor al de los analizadores de red vectoriales.

Los dos tipos de analizadores de redes son diferentes en su composición y en la forma en que pueden realizar mediciones. El analizador de redes escalar es el menos caro, pero también proporciona la menor cantidad de información. El analizador de redes vectorial proporciona mucha más información, pero también son considerablemente más caros.

Los modelos de analizadores de redes más comunes son los de dos puertos, pero también existen modelos de cuatro puertos y modelos que poseen otras mejoras como pantalla táctil, la posibilidad de conectar un ratón o teclado, plataforma en base Windows, etc. En el presente proyecto, tal y como se ha mencionado, se centrará la atención en los analizadores de redes vectoriales o VNA.

2.2.3. Funcionamiento del VNA

Un analizador de redes vectorial es un instrumento empleado para medir y analizar circuitos de radiofrecuencia. Los VNA son empleados para medir todo tipo de dispositivos, activos o pasivos, desde filtros sencillos hasta módulos complejos usados en comunicaciones de satélites. Para todo esto, emplea técnicas de barrido en frecuencia.

Para obtener las características de los dispositivos bajo prueba, el analizador de redes vectorial emite una señal senoidal de prueba a una frecuencia determinada que es aplicada al dispositivo, el cual proporciona como respuesta otra señal senoidal. El analizador mide la diferencia de amplitud y de fase entre ambas señales.

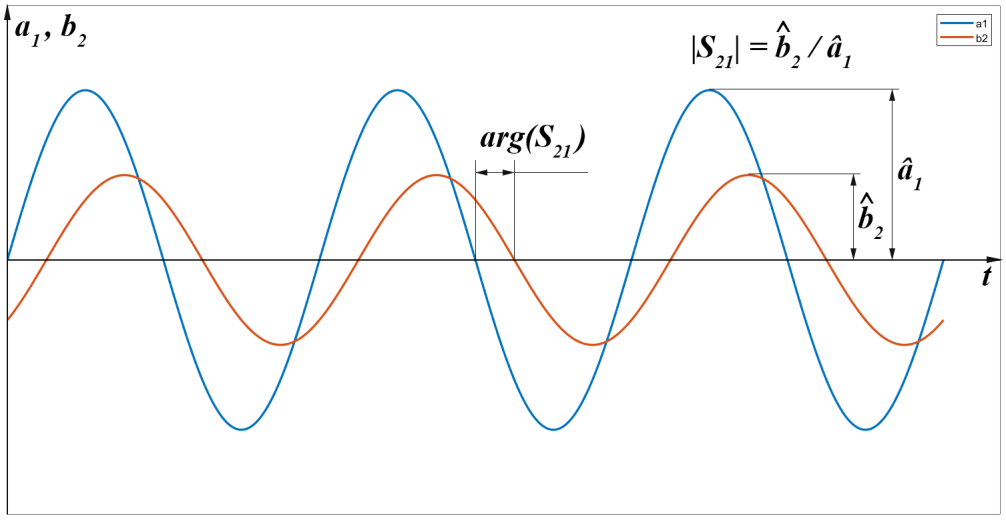


Figura 2.12: Diferencia entre fase y amplitud de la señal de prueba y la señal de respuesta del dispositivo.

Un VNA, tal y como se ha mencionado, es un instrumento capaz de analizar las propiedades de las redes eléctricas a altas frecuencias, especialmente los parámetros S. Mide la magnitud y la fase del vector que forman los parámetros S haciendo un barrido en frecuencia.

2.2.3.1. Arquitectura de un VNA

La arquitectura básica de un VNA incluye un generador de señales, separadores de señal, receptor de la señal y procesador y pantalla, tal y como se puede ver en la Figura 2.13.

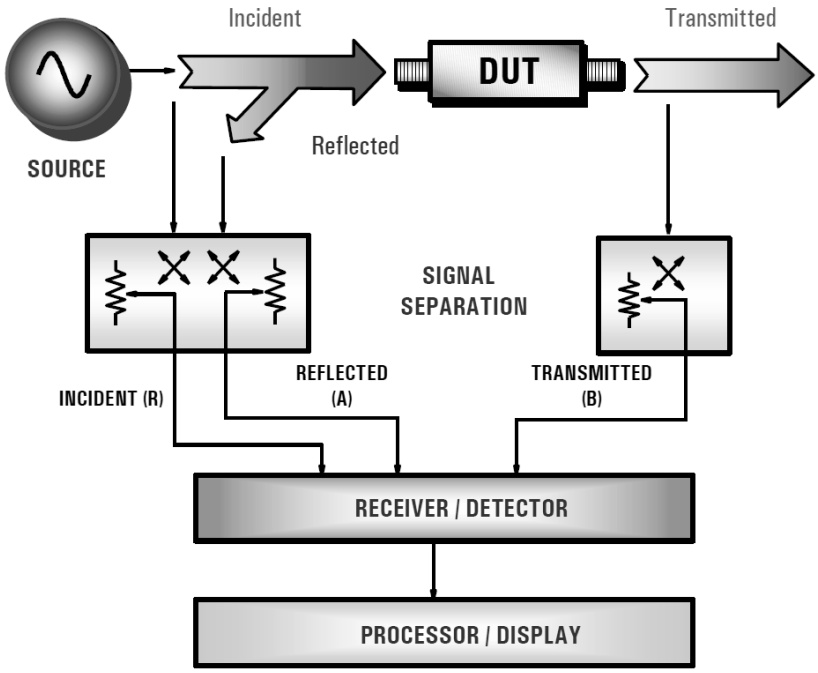


Figura 2.13: Arquitectura de un VNA [15].

- **Generador de señales**

El generador de señal es la fuente se encarga de proporcionar la señal de prueba que alimentará al DUT, y es capaz de realizar un barrido de frecuencias en el rango de la radiofrecuencia y las microondas.

- **Separadores de señal**

El separador de señal se encarga de obtener la medida de la señal incidente como referencia. Esta función se puede realizar con divisores de tensión o atenuadores en general, normalmente resistivos, pero tienen la desventaja de que pueden tener pérdidas. También se puede realizar esta función con acopladores direccionales, que tienen pocas pérdidas y un buen aislamiento, y se suelen utilizar en altas frecuencias.

El separador de señal también es el responsable de separar la onda incidente de la onda reflejada a la entrada del dispositivo bajo prueba. Se puede efectuar esta función utilizando acopladores direccionales, con bajas pérdidas y alto aislamiento de la reflexión, aunque también se pueden utilizar puentes de resistencias para el puerto en cuestión, que trabajan en continua y tienen pérdidas.

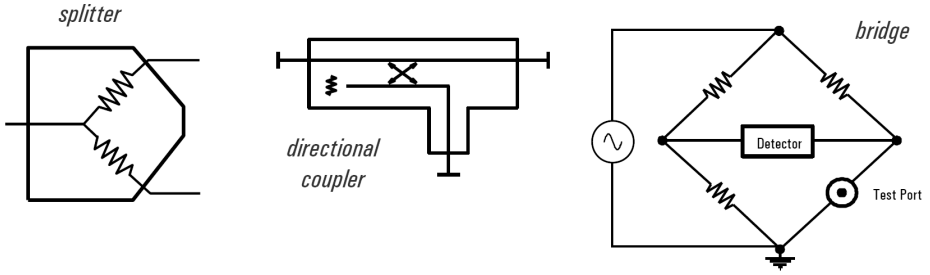


Figura 2.14: Separadores de señal [15].

- **Receptor de la señal**

Los receptores realizan las mediciones. Existen dos formas principales para proporcionar una señal de detección en el analizador de redes: mediante un diodo receptor o mediante un receptor sintonizado.

Los diodos detectores convierten la señal de radiofrecuencia a un nivel proporcional de continua. Hacen una detección únicamente escalar, por lo que se pierde la información de fase.

El receptor sintonizado está constituido por un mezclador de frecuencias, que recibe la señal de RF y una señal de un oscilador local, y un filtro. La salida contiene la información de la señal de entrada trasladada a la frecuencia intermedia. Con esto, obtenemos información tanto de la magnitud como de la fase de la señal. El analizador de red utiliza un conversor analógico/digital y un procesador de señal digital para obtener el valor del módulo y la fase de la señal intermedia.

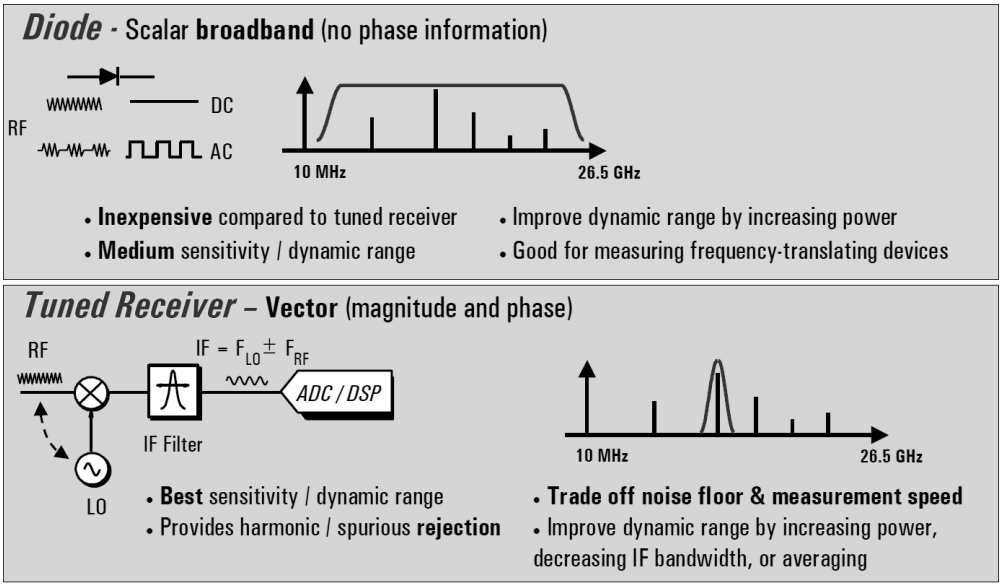


Figura 2.15: Receptores de señal [15].

- **Procesador y pantalla**

A partir de la señal procesada del receptor, es necesario mostrar dicha señal en un formato que se pueda interpretar. Realizando un procesamiento adecuado de la señal, se formatean los datos de reflexión y transmisión para permitir que la información se interprete lo más fácilmente posible. La mayoría de los analizadores de redes incluyen representaciones según barridos lineales y logarítmicos, diagramas polares o Carta de Smith.

Figura 2.16: VNA con pantalla e interfaz de usuario.

2.2.3.2. Medida de parámetros S con un VNA

El esquema básico de un VNA de dos puertos, que mide los parámetros de dispersión S_{11}, S_{12}, S_{21}, S_{22}, es el siguiente:

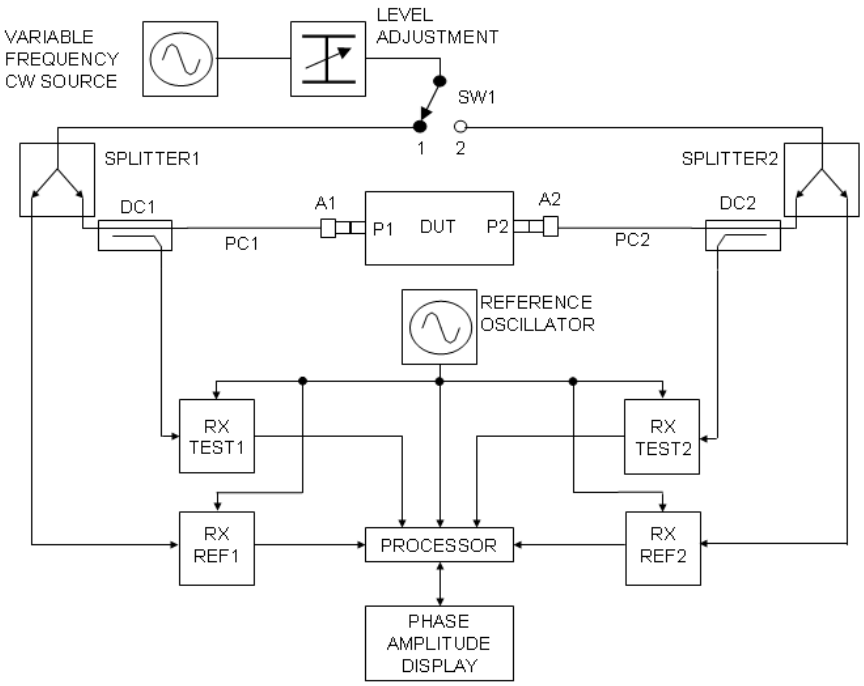


Figura 2.17: Medida de parámetros S y partes básicas de un VNA.

La posición del interruptor SW1 establece la dirección en la que la señal de prueba pasará a través del dispositivo bajo prueba. Con SW1 en la posición indicada (posición 1), la señal de prueba será incidente sobre el puerto P1, y el VNA medirá los parámetros S_{11} y S_{21}. Al cambiar de posición, la señal de prueba incidirá sobre el puerto P2, y medirá S_{22} y S_{12}.

Suponemos que SW1 se encuentra en la posición 1. El generador de barrido genera señales de prueba que varían entre dos frecuencias programadas. Posee un atenuador variable para ajustar su nivel de potencia. La señal de prueba incidente llega al separador de señal, que manda la señal, por una parte, al receptor de referencia 1 (RX REF1), que mide la potencia de la onda incidente y, por otra parte, al puerto P1 a través del acoplador direccional DC1. El acoplador direccional DC1 mide la potencia de la onda reflejada desde P1, y se la envía al receptor de prueba 1 (RX TEST1). De la misma forma, la señal transmitida a través del dispositivo y que sale del puerto P2 pasa a través del acoplador direccional DC2, que mide la potencia de dicha onda transmitida y se la envía al receptor de prueba 2 (RX TEST2). Todos los receptores comparten el mismo oscilador de referencia, y son capaces de medir la amplitud y la fase de la señal a cada frecuencia.

Con SW1 en la posición 1, tendremos S_{11} y S_{21}. De forma análoga, con SW1 en la posición 2, tendremos S_{22} y S_{12}, ya que las señales de prueba son aplicadas al puerto P2, de forma que la potencia de la onda incidente la mide el receptor de referencia 2 (RX REF2) la potencia de la onda reflejada la mide el receptor de prueba 2 (RX TEST2) a través del acoplador direccional DC2 y la potencia de la onda transmitida la mide el receptor de prueba 1 (RX TEST1) a través del acoplador direccional DC1.

Todas las señales de salida de los receptores se envían a un procesador que se encarga de realizar el procesamiento matemático y que nos devuelve el resultado de la magnitud y fase de los parámetros S correspondientes. Finalmente, muestra en pantalla los parámetros y el formato deseados.

La conexión de los puertos de prueba del VNA con los dos puertos (P1 y P2) del dispositivo bajo prueba (DUT) debe hacerse utilizando adaptadores de conector adecuados (A1 y A2) y cables de precisión (PC1 y PC2).

2.2.4. Corrección de errores

Existen tres fuentes de errores de medida básicas: errores sistemáticos, errores aleatorios y errores de deriva.

Los errores sistemáticos se pueden eliminar tras la calibración del instrumento, ya que son causados por imperfecciones en el analizador y son repetibles, predecibles e invariantes en el tiempo.

Los errores aleatorios no se pueden eliminar con la calibración del instrumento, ya que cambian con el tiempo de forma aleatoria y son impredecibles. El ruido del instrumento, como el ruido de fase de la fuente generadora o el ruido del muestreador y el ruido de alta frecuencia son las principales fuentes de error aleatorio.

Los errores de deriva son causados por el instrumento después de realizar la calibración. Son causados principalmente por la variación de temperatura y se pueden eliminar realizando más calibraciones.

2.2.4.1. Errores sistemáticos

Los principales errores sistemáticos asociados a las medidas de redes son los siguientes:

- Errores relacionados con la fuga de la señal: directividad y diafonía (crosstalk).
- Errores relacionados con la reflexión de la señal: adaptación de fuente y carga.
- Errores relacionados con la respuesta en frecuencia de los receptores: mediciones de reflexión y transmisión.

El modelo de errores completo para dos puertos incluye los seis términos mencionados para el sentido directo y para el sentido inverso, para un total de doce términos de error.

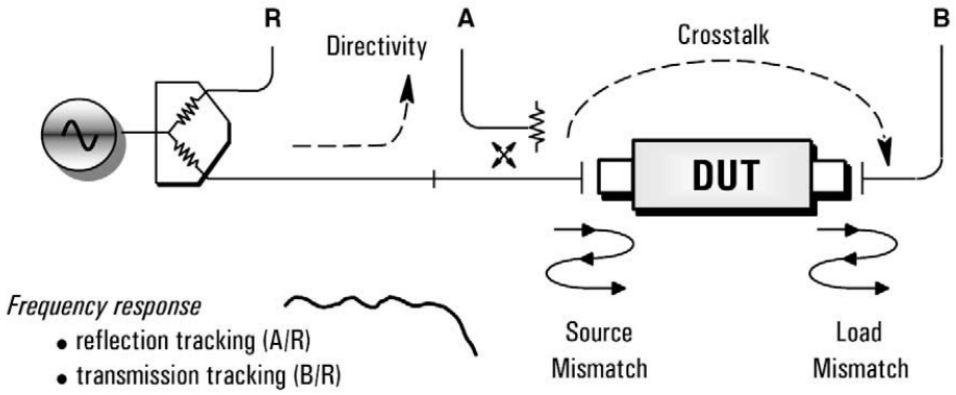


Figura 2.18: Términos de error [15].

Los dos tipos principales de corrección de errores que se puede realizar son corrección de respuesta (normalización) y corrección vectorial. La corrección de respuesta es simple de realizar, pero solo corrige algunos de los doce posibles términos de error sistemático (las mediciones de reflexión y transmisión). Por tanto, la precisión de medida es inferior comparada con la corrección vectorial. La calibración de respuesta es, básicamente, una medida normalizada donde se almacena una traza de referencia en memoria y los datos de medición posteriores se dividen por esta traza de referencia.

La corrección vectorial de errores requiere un analizador que pueda medir magnitud y fase, es decir, un analizador de redes vectorial. También necesita medidas de más estándares de calibración. Este método tiene en cuenta todos los términos principales de error sistemático y ofrece medidas muy precisas.

2.2.4.2. Corrección vectorial de errores

La corrección vectorial de errores es el proceso de caracterizar los términos de error sistemático midiendo estándares de calibración conocidos y, luego, eliminar los efectos de estos errores de medidas posteriores.

La calibración de un puerto se utiliza para medidas de reflexión y puede medir y eliminar tres términos de error sistemático: directividad, adaptación de fuente y mediciones de reflexión. La calibración completa de dos puertos se utiliza tanto para medidas de reflexión como para medidas de transmisión,

y los doce términos de error sistemático son medidos y eliminados. Por tanto, la calibración de dos puertos requiere doce medidas de cuatro estándares conocidos, *SHORT-OPEN-LOAD-THROUGH: SOLT* (cortocircuito, circuito abierto, carga, a través), y algunos estándares se miden varias veces, por ejemplo, el estándar *THROUGH* (o *THRU*) normalmente se mide cuatro veces.

La corrección de errores de dos puertos es la forma más precisa de corrección de errores, ya que tiene en cuenta todas las fuentes principales de error sistemático. El modelo de error para un dispositivo de dos puertos se muestra en la siguiente figura. Además, tras la figura, se muestran las ecuaciones para obtener los parámetros S reales ($S_{\mathrm{xx}a}$) del dispositivo a partir de los parámetros S medidos ($S_{\mathrm{xx}m}$), una vez que se han determinado los términos de error sistemático. Cada parámetro S real es función de los cuatro parámetros S medidos.

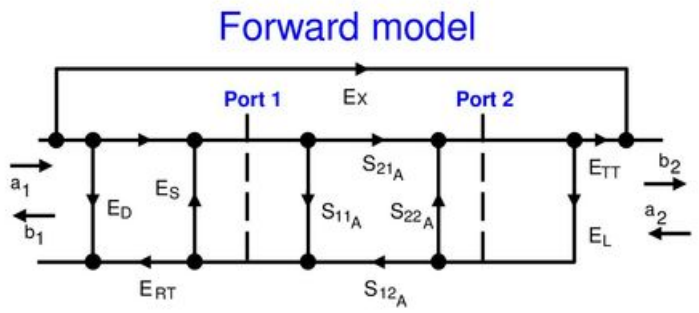


Figura 2.19: Modelo de error directo para un dispositivo de dos puertos [15].

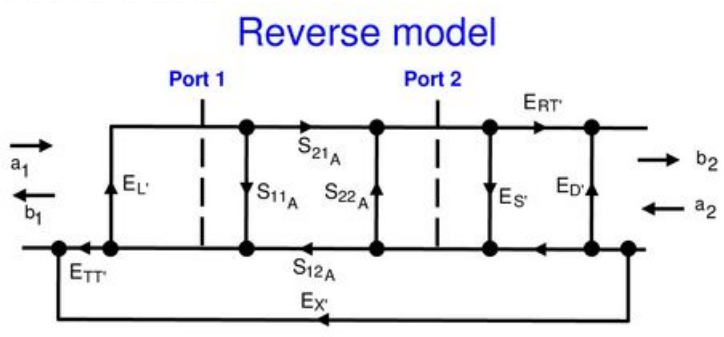


Figura 2.20: Modelo de error inverso para un dispositivo de dos puertos [15].

$$S_{11a} = \frac{\frac{S_{11m}-E_D}{E_{RT}}\left(1+\frac{S_{22m}-E'_D}{E'_{RT}}E'_S\right) - E_L\frac{S_{21m}-E_X}{E_{TT}}\frac{S_{12m}-E'_X}{E'_{TT}}}{\left(1+\frac{S_{11m}-E_D}{E_{RT}}E_S\right)\left(1+\frac{S_{22m}-E'_D}{E'_{RT}}E'_S\right) - E'_L E_L\frac{S_{21m}-E_X}{E_{TT}}\frac{S_{12m}-E'_X}{E'_{TT}}} \tag{2.49}$$

$$S_{21a} = \frac{\frac{S_{21m}-E_X}{E_{TT}}\left(1+\frac{S_{22m}-E'_D}{E'_{RT}}\left(E'_S-E_L\right)\right)}{\left(1+\frac{S_{11m}-E_D}{E_{RT}}E_S\right)\left(1+\frac{S_{22m}-E'_D}{E'_{RT}}E'_S\right) - E'_L E_L\frac{S_{21m}-E_X}{E_{TT}}\frac{S_{12m}-E'_X}{E'_{TT}}} \tag{2.50}$$

$$S_{12a} = \frac{\frac{S_{12m}-E'_X}{E'_{TT}}\left(1+\frac{S_{11m}-E_D}{E_{RT}}\left(E_S-E'_L\right)\right)}{\left(1+\frac{S_{11m}-E_D}{E_{RT}}E_S\right)\left(1+\frac{S_{22m}-E'_D}{E'_{RT}}E'_S\right) - E'_L E_L\frac{S_{21m}-E_X}{E_{TT}}\frac{S_{12m}-E'_X}{E'_{TT}}} \tag{2.51}$$

$$S_{22a} = \frac{\frac{S_{22m}-E'_D}{E'_{RT}}\left(1+\frac{S_{11m}-E_D}{E_{RT}}E_S\right) - E'_L\frac{S_{21m}-E_X}{E_{TT}}\frac{S_{12m}-E'_X}{E'_{TT}}}{\left(1+\frac{S_{11m}-E_D}{E_{RT}}E_S\right)\left(1+\frac{S_{22m}-E'_D}{E'_{RT}}E'_S\right) - E'_L E_L\frac{S_{21m}-E_X}{E_{TT}}\frac{S_{12m}-E'_X}{E'_{TT}}} \tag{2.52}$$

Los términos de error relacionados con la fuga de señal son E_D, E'_D (directividad) y E_X, E'_X (aislamiento). Los errores relacionados con la reflexión son E_S, E'_S (adaptación de fuente) y E_L, E'_L (adaptación de carga). Por último, los errores relacionados con la respuesta en frecuencia de los receptores son E_{RT}, E'_{RT} (mediciones de reflexión) y E_{TT}, E'_{TT} (mediciones de transmisión).

2.2.5. Calibración y proceso de medida del VNA

Para medir los parámetros S de una línea de transmisión con un VNA, primero se deben configurar las frecuencias entre las que hará el barrido en frecuencia. También, se debe configurar la potencia y el número de puntos. Después, se calibra el VNA, incluyendo todos sus puertos. La calibración de un analizador de redes es un proceso de alta precisión, donde se deben tener en cuenta tanto la impedancia en la que se está operando como las condiciones en las que está operando el equipo.

El estándar de calibración, como se ha mencionado en el apartado anterior, usa cuatro dispositivos de prueba, que se denominan *OPEN* (red abierta), *SHORT* (red en cortocircuito) y *LOAD* (red con carga) para calibrar la reflexión y *THRU* (red conectada) para calibrar la transmisión, que deben ser conectados a los puertos del analizador para que este pueda comparar y establecer la diferencia entre los diferentes modos. Los datos de calibración se podrán guardar en memoria en distintos registros del VNA.

Tras la calibración del VNA, se realiza la medida de los parámetros S sobre la línea de transmisión. Por último, el VNA presentará los resultados, los cuales se podrán guardar en memoria o exportar, normalmente en formato *SnP*, donde n indica el número de puertos empleados en la medida y, por tanto, el número de parámetros S guardados. En la Figura 2.21, se puede observar el contenido de un archivo tipo `*.s1p`, donde `HZ` indica que la frecuencia se muestra en Hz, `S` señala que el archivo contiene parámetros S, `RI` indica que el archivo muestra la parte real e imaginaria de los parámetros S y `R 50` muestra que la impedancia característica de la medida es 50 Ω.

```
# HZ S RI R 50
10000 1.0032199753477187 -0.01557304777251817
15009900 0.9943592231497785 -0.10635142666315273
30009800 0.9771936572536339 -0.21200946470592713
45009700 0.9485899211488402 -0.31569270218004647
60009600 0.9118802077507263 -0.4119012747686779
75009500 0.8639333631929779 -0.5049221156028996
90009400 0.8010876758883214 -0.6009201407374561
105009300 0.7339780063617631 -0.6824093735020929
120009200 0.6575344064235786 -0.7534607833709694
135009100 0.575031245086454 -0.8162512588566079
150009000 0.4859509564674492 -0.87245918442889
165008900 0.39232860526534474 -0.917793638696625
180008800 0.29335799528005574 -0.9516249970806904
195008700 0.19472263375702487 -0.9728302516361341
210008600 0.09427183378925243 -0.9890746602210408
```

Figura 2.21: Contenido de un archivo `*.s1p`.

2.2.6. Medidas del VNA

Los parámetros que se obtienen de un VNA son de reflexión y de transmisión, y pueden estar representados de distintas formas. Habitualmente, suelen estar representados según la Carta de Smith, un diagrama de magnitud y fase o un diagrama de parte real e imaginaria. Las siguientes figuras muestran un ejemplo de cada tipo de representación:

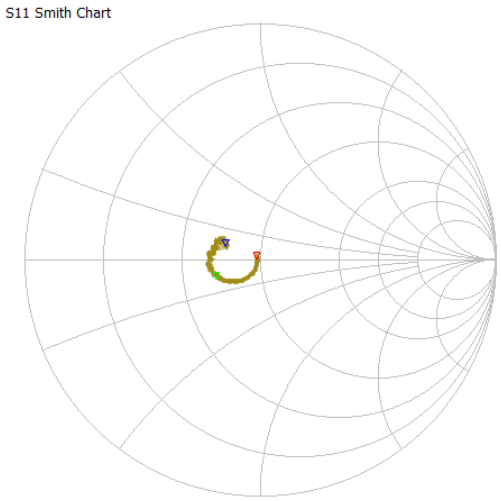


Figura 2.22: Representación en Carta de Smith.

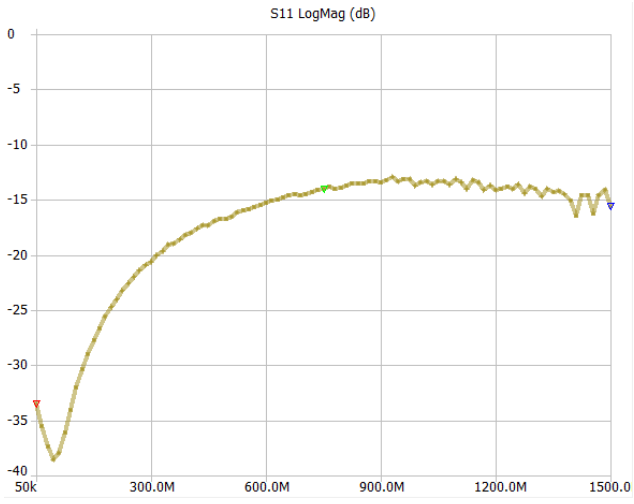


Figura 2.23: Representación en diagrama de magnitud (dB). El eje horizontal representa la frecuencia (Hz).

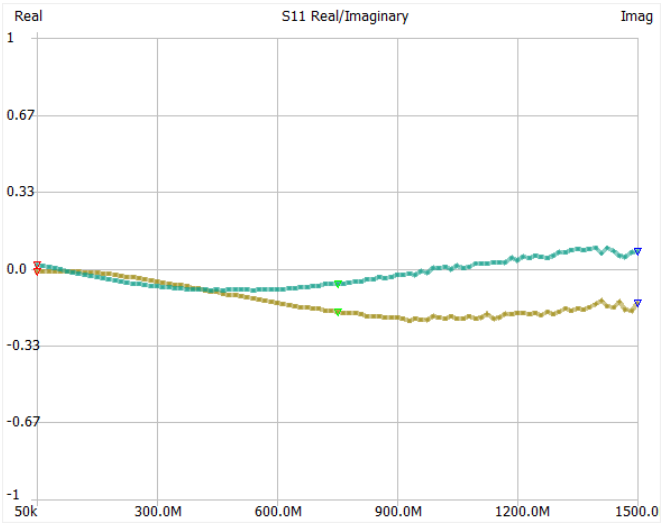


Figura 2.24: Representación en diagrama de parte real (marrón) e imaginaria (azul). El eje horizontal representa la frecuencia (Hz).

- Reflexión (S_{11}, S_{22}):
  - Carta de Smith: se obtienen los valores de la impedancia normalizada a distintas frecuencias. A partir de estos datos, se puede calcular el coeficiente de reflexión.
  - Magnitud y fase: representa la magnitud (en dB, normalmente) y la fase del coeficiente de reflexión en función de la frecuencia, que puede estar en escala logarítmica. De estos valores, se puede calcular la impedancia.
  - Parte real y parte imaginaria: representa la parte real e imaginaria del coeficiente de reflexión en función de la frecuencia.
- Transmisión (S_{21}, S_{12}):
  - Magnitud y fase: representa la magnitud y la fase del coeficiente de transmisión en función de la frecuencia.
  - Parte real y parte imaginaria: representa la parte real e imaginaria del coeficiente de transmisión en función de la frecuencia.

En un diagrama de magnitud y fase, la representación de la magnitud suele ser en decibelios (dB), lo cual se define como:

$$S_k(dB) = 20 \log |S_k| \tag{2.53}$$

Por tanto, se puede obtener la magnitud de la siguiente forma:

$$|S_k| = 10^{\frac{S_k(dB)}{20}} \tag{2.54}$$

Si, a esa frecuencia, su fase es ϕ_k, el valor del vector con magnitud y fase es:

$$S_k = |S_k| e^{j\phi_k} \tag{2.55}$$

De la misma manera, a partir del diagrama de parte real e imaginaria, se puede obtener el valor de la magnitud y la fase del vector de la siguiente forma:

$$|S_k| = \sqrt{\mathrm{Re}(S_k)^2 + \mathrm{Im}(S_k)^2} \tag{2.56}$$

$$\phi_k = \arctan\left(\frac{\mathrm{Im}(S_k)}{\mathrm{Re}(S_k)}\right) \tag{2.57}$$

2.3. Medios dieléctricos

Un medio material constituye el conjunto de puntos del espacio por donde se transmiten las ondas. Las propiedades electromagnéticas del medio determinarán su interacción con campos electromagnéticos y cómo se propagan las ondas electromagnéticas en el mismo. El comportamiento de un medio material ante la excitación con un campo electromagnético, caracterizada por las magnitudes vectoriales **E** y **H**, que representan el campo eléctrico y la excitación magnética, respectivamente, consiste en observar su respuesta ante dicha excitación, que puede caracterizarse mediante las magnitudes vectoriales **D**, **J** y **B**, que representan el desplazamiento eléctrico, la densidad de corriente y la inducción magnética, respectivamente. Los parámetros que caracterizan cada medio, y que relacionan la excitación electromagnética con su respuesta en el medio, son ε, σ y μ, que representan, respectivamente, la permitividad eléctrica, la conductividad eléctrica y le permeabilidad magnética. Las relaciones son las siguientes:

$$\mathbf{D} = \varepsilon\mathbf{E} \tag{2.58}$$

$$\mathbf{J} = \sigma\mathbf{E} \tag{2.59}$$

$$\mathbf{B} = \mu\mathbf{H} \tag{2.60}$$

Las magnitudes anteriores no siempre están en fase, por lo que las magnitudes ε, σ y μ tienen una representación como magnitudes complejas:

$$\varepsilon = \varepsilon' - j\varepsilon'' \tag{2.61}$$

$$\sigma = \sigma' + j\sigma'' \tag{2.62}$$

$$\mu = \mu' - j\mu'' \tag{2.63}$$

Los parámetros ε, σ y μ permiten realizar una clasificación de materiales. De esta forma, se encuentran los siguientes materiales:

- Dieléctricos: materiales con una conductividad muy baja y una respuesta magnética débil. Estos materiales se caracterizan mediante la permitividad eléctrica.

- Conductores: materiales con una conductividad muy alta, con campos muy pequeños en su interior, con una respuesta magnética débil y con una respuesta dieléctrica inapreciable.
- Magnéticos: materiales con una respuesta magnética importante.

Centrándose en los medios dieléctricos, si se excita el material con un campo eléctrico, se produce una reordenación de cargas con la aparición de dipolos eléctricos, fenómeno que se conoce como polarización del dieléctrico, la cual está asociada con la permitividad ε, donde ε' está relacionada con la energía almacenada al formarse los dipolos y ε'' está relacionada con las pérdidas de energía durante este proceso. La polarización puede ser por distorsión, aplicando un campo eléctrico externo que provoca que las cargas negativas se desplacen con respecto a las positivas, apareciendo un dipolo inducido, la cual se denomina polarización electrónica o atómica, o también puede ser por orientación, denominada polarización dipolar, en la que algunos materiales presentan moléculas con un momento dipolar permanente y, al aplicar un campo eléctrico, se provoca un cambio en la orientación de los dipolos.

La polarización del dieléctrico se caracteriza mediante el vector de polarización $\mathbf{P}$, que depende del campo eléctrico aplicado:

$$\mathbf{P} = \varepsilon_0 \chi_e \mathbf{E} \tag{2.64}$$

Donde χ_e es la susceptibilidad eléctrica y ε_0 es la permitividad del vacío. Podemos establecer la relación entre la susceptibilidad eléctrica y la permitividad a partir de la definición del vector desplazamiento eléctrico:

$$\mathbf{D} = \varepsilon \mathbf{E} = \varepsilon_0 \mathbf{E} + \mathbf{P} = \varepsilon_0 (1 + \chi_e) \mathbf{E} \tag{2.65}$$

Por tanto:

$$\varepsilon = \varepsilon_0 (1 + \chi_e) \tag{2.66}$$

El hecho de que la permitividad sea compleja indica que la polarización no es instantánea, sino que tarda un tiempo en establecerse, de manera que hay un desfase entre el campo eléctrico aplicado y la polarización inducida. Si el campo eléctrico aplicado al dieléctrico es variable en el tiempo, los dipolos tratarán de seguir las variaciones del campo eléctrico con un tiempo de respuesta variable y, por tanto, a frecuencias distintas. Esto da lugar a una variación de la permitividad con la frecuencia, lo que se conoce como dispersión dieléctrica. Más concretamente, lo que produce la variación de la permitividad con la frecuencia son los diferentes fenómenos de polarización, los cuales dan lugar a fenómenos de relajación en el espectro dieléctrico. Dependiendo de las fuerzas de cohesión entre las moléculas y las interacciones moleculares, cada medio presenta determinadas frecuencias de relajación a lo largo del espectro en las cuales la reorientación de las cargas o polarización siguiendo el ritmo alternante del campo eléctrico aplicado marcado por su frecuencia deja de poder seguir la velocidad de alternación de dicho campo eléctrico, ya que las moléculas tardan un tiempo en reorientarse. Este tiempo de retraso se denomina tiempo de relajación. Esto puede suceder por diferentes fenómenos que dan lugar a los diferentes tipos de polarización, como pueden ser la dipolar, la atómica o la electrónica.

En el caso de la polarización dipolar, al aplicar un campo eléctrico alternante, las cargas del dieléctrico se van desplazando de un lado a otro de manera alternante a la velocidad marcada por la frecuencia del campo aplicado. Cuanto mayor es la frecuencia, más aumenta el número de choques entre las cargas al desplazarse. Esto hace que se produzcan dos tipos de pérdidas: por calor y porque en esos choques se pierden cargas por el camino que no acaban contribuyendo a la polarización. Estas pérdidas son las que recoge la parte imaginaria de la permitividad, y el máximo de la misma coincide con el punto de inflexión de la parte real, porque lo que pierde una lo gana la otra, aunque no en términos absolutos.

Por tanto, en un barrido de frecuencias, la parte real de la permitividad (ε') presenta escalones descendentes que representan la desaparición de contribuciones dipolares al aumentar la frecuencia, mientras que la parte imaginaria de la permitividad (ε'') presenta máximos en las frecuencias de esos escalones, que corresponden a un aumento de la energía transferida al material a través de las interacciones de los dipolos con el entorno, y que se opone a las variaciones de su orientación.

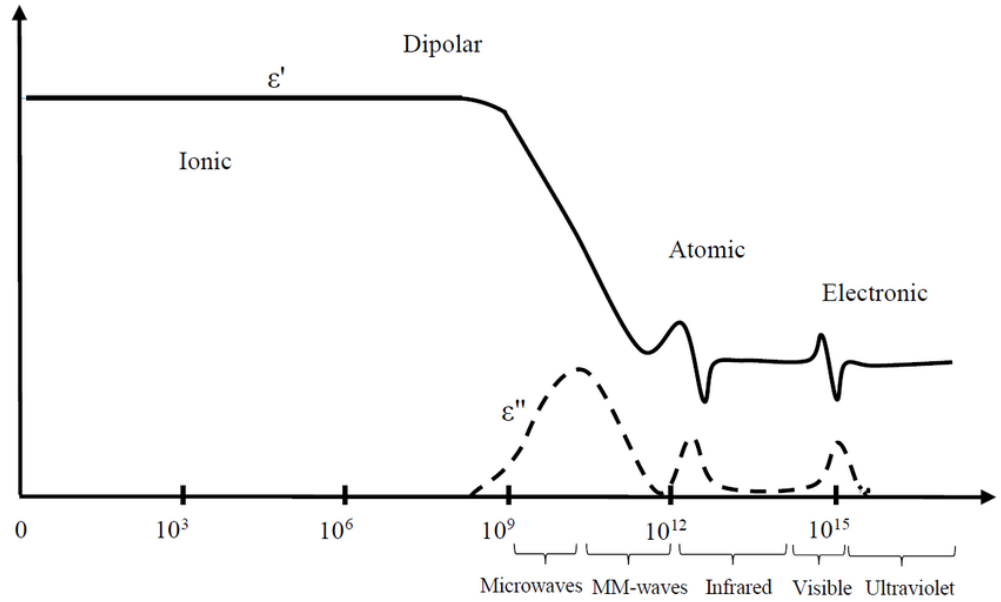


Figura 2.25: Dispersión dieléctrica. Permitividad en función de la frecuencia [13].

Por consiguiente, ε' está relacionada con el mecanismo de polarización y la energía almacenada al formarse los dipolos y ε'' está relacionada con las pérdidas de energía durante este proceso.

En conclusión, se puede afirmar que la permitividad eléctrica caracteriza completamente un material dieléctrico. Por otra parte, existen más parámetros que caracterizan medios dieléctricos, que son los siguientes parámetros derivados: la permitividad relativa (ε_r), la tangente de pérdidas ($\tan\delta$) y el índice de refracción (n). Estos parámetros se definen como:

$$\varepsilon_r = \frac{\varepsilon}{\varepsilon_0} \tag{2.67}$$

$$\tan\delta = \frac{\varepsilon''}{\varepsilon'} \tag{2.68}$$

$$n = \sqrt{\varepsilon_r} = \sqrt{\frac{\varepsilon}{\varepsilon_0}} \tag{2.69}$$

2.3.1. Caracterización de medios dieléctricos

La caracterización de medios dieléctricos consiste en utilizar métodos para determinar la permitividad eléctrica de los distintos materiales. Estos métodos se basan en la caracterización electromagnética del material en las bandas de frecuencias de microondas.

Fundamentalmente, existen dos grandes grupos en los que se dividen los métodos de caracterización de medios dieléctricos, según la forma de realizar la medida: métodos resonantes y métodos no resonantes. Para tener un conocimiento preciso de las propiedades dieléctricas de un material, se combina el método no resonante con el método resonante. Primero, se obtiene la información mediante el método no resonante para un rango de frecuencias y, posteriormente, se precisa esta información mediante el método resonante.

La resonancia describe el conjunto de fenómenos relacionados con los movimientos periódicos en los que se intensifica una oscilación al someter el sistema a oscilaciones de una frecuencia determinada, denominada frecuencia resonante.

2.3.1.1. Métodos no resonantes

Los métodos no resonantes se utilizan para tener una idea general de las propiedades electromagnéticas en un rango de frecuencias. Permiten realizar estudios de banda ancha. Las propiedades se obtendrán a partir de la impedancia y de la velocidad de onda de los materiales.

Si se consideran dos medios separados por una superficie, cuando una onda electromagnética se propaga por el primer medio e incide en esta superficie, se producirá una reflexión y una transmisión de la onda, a la vez que se refracta, lo cual se puede determinar mediante la Ley de Snell. La onda reflejada se propagará por el primer medio, mientras que la onda transmitida se propagará por el segundo medio. Este segundo medio tendrá unas características diferentes al primero, por lo que la velocidad de la onda y la impedancia del material variarán. La medida de la parte de la onda que se refleja o se transmite a través del material proporcionará información sobre la permitividad del mismo.

Dentro de los métodos no resonantes, se pueden diferenciar entre métodos de reflexión y métodos de transmisión/reflexión. Para poder dirigir la onda electromagnética hasta el material y poder medir la parte que se refleja y la que se transmite a través de él, se necesita una línea de transmisión, que podrá ser una sonda coaxial, una guía o el espacio libre.

- **Método de reflexión en una sonda coaxial**

En este método, una sonda coaxial acabada en abierto se pone en contacto directo con el material a medir, ya sea sólido, líquido o semisólido. La caracterización de materiales mediante métodos de reflexión consiste en observar el cambio en la impedancia de la línea de transmisión al introducir el material bajo estudio. Se calcularán las propiedades dieléctricas del material a partir de la medición del coeficiente de reflexión al final de la línea. El método de reflexión en una sonda coaxial también puede realizarse en cortocircuito.

Presenta las ventajas de que es un método no destructivo, se trata de una medida simple y cubre un gran rango de frecuencias, pero ofrece una precisión limitada y puede haber presencia de aire entre la sección final de la sonda coaxial y el plano del material bajo estudio, modificando el resultado de las mediciones. Se puede observar la realización del método en la Figuras 2.26 y 2.27.

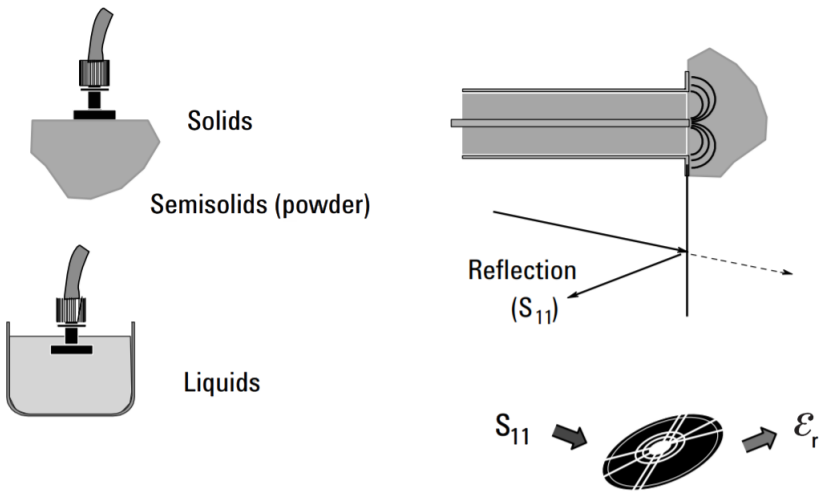


Figura 2.26: Método de reflexión en una sonda coaxial [16].

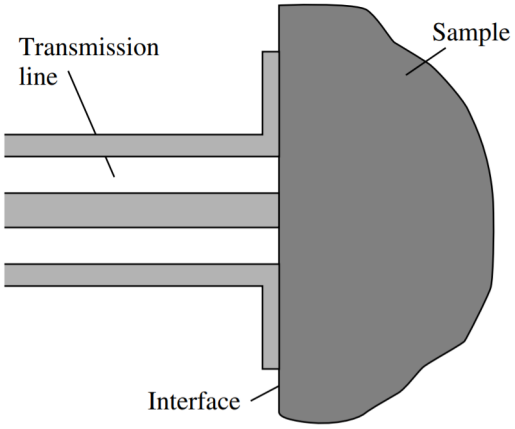


Figura 2.27: Método de reflexión en una sonda coaxial de final abierto. Detalle de la sonda en contacto con el material bajo estudio [13].

- **Método de la guía de ondas**

Este método emplea una guía de ondas y una muestra del material bajo estudio. Para caracterizar un material dieléctrico con este método, habrá que observar el cambio en la impedancia de la línea de transmisión al introducir una muestra del material, calculando sus propiedades dieléctricas a partir de los parámetros de reflexión y transmisión. El método de la guía de ondas puede ser de reflexión en circuito abierto o en cortocircuito (método de Robert Von Hippel) o de reflexión/transmisión.

Se trata de una técnica de banda ancha que permite medir materiales magnéticos, pero la muestra debe tener una forma y tamaño específicos. Se puede ver la aplicación del método en las Figuras 2.28 y 2.29.

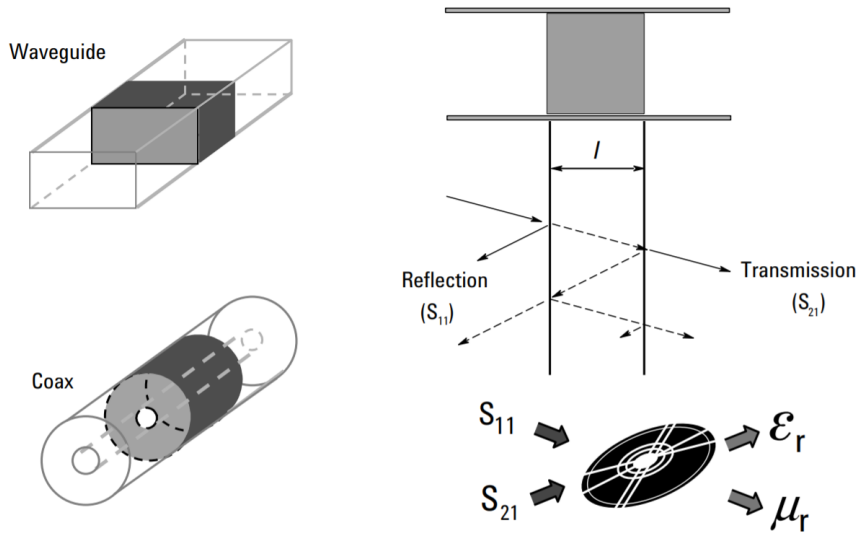


Figura 2.28: Método de reflexión/transmisión de la guía de ondas [16].

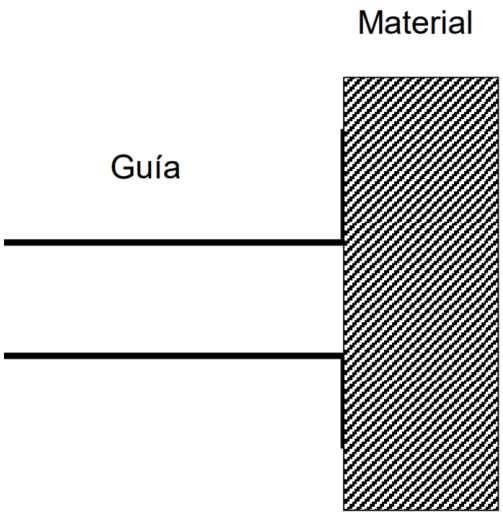


Figura 2.29: Método de reflexión de la guía de ondas [34].

- **Método del espacio libre**

El método del espacio libre utiliza antenas para aplicar las ondas de microondas a través del espacio y analizar la interacción con el material, que se encontrará en un punto intermedio del espacio libre, midiendo la atenuación y el desfase de las señales. A partir de estos valores, mediante ecuaciones, se obtienen las propiedades electromagnéticas del material. Para que las medidas sean correctas, se debe asegurar que se está trabajando en un campo lejano, de forma que las ondas se puedan considerar como ondas planas. Además, se debe escoger un tamaño adecuado de la muestra, tal que sea mayor que la longitud de la onda electromagnética para evitar la difracción y otros efectos de los bordes, y el lugar donde se realizan las medidas debe evitar las reflexiones no deseadas, por ejemplo, una cámara anecoica. El método del espacio libre puede ser de reflexión o de reflexión/transmisión.

Como ventajas, presenta que no es destructivo ni intrusivo, no necesita contacto y no se requiere una preparación especial de la muestra. Sin embargo, ofrece una precisión limitada por la difracción en los bordes de la muestra y las múltiples reflexiones que pueden ocurrir durante la medición. El esquema del método del espacio libre se muestra en las Figuras 2.30 y 2.31.

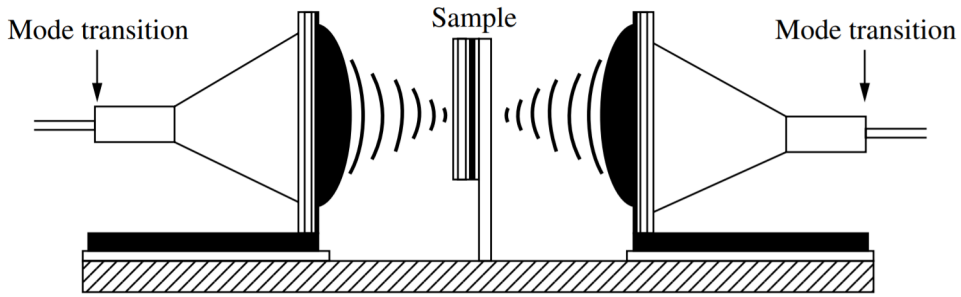


Figura 2.30: Método del espacio libre [13].

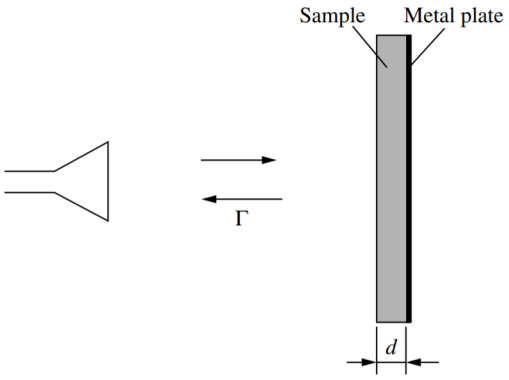


Figura 2.31: Método de reflexión del espacio libre [13].

2.3.1.2. Métodos resonantes

Los métodos resonantes se usan cuando se quiere información precisa de las propiedades dieléctricas de un material a una determinada frecuencia. Con este método, se pueden obtener las propiedades de un material para varias frecuencias discretas. Los métodos resonantes son mucho más precisos que los métodos no resonantes. Los métodos resonantes se dividen en: método resonador y método de perturbación resonante.

- **Método resonador**

El método resonador necesita conocer las dimensiones de un resonador dieléctrico. A partir de ello, se puede calcular la permitividad a partir de la frecuencia de resonancia y el factor de calidad de dicho resonador. Este método se utiliza para materiales que poseen una permitividad elevada. El método consiste en situar la muestra del material entre dos láminas conductoras para que funcione como resonador. En la Figura 2.32, se puede observar la geometría de la configuración de un resonador Courtney, que esencialmente consiste en una guía de ondas cilíndrica dieléctrica cortocircuitada por dos placas conductoras situadas en ambos extremos de la guía, transforman la línea de transmisión en un resonador.

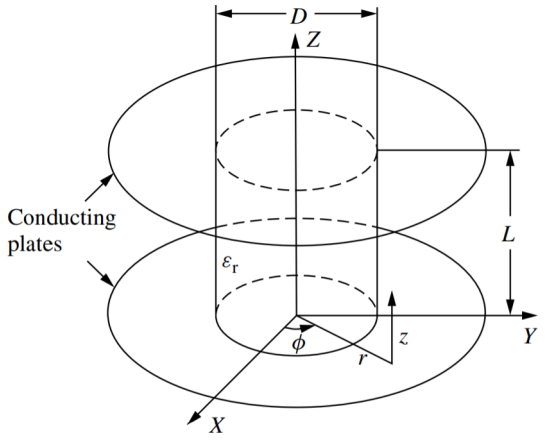


Figura 2.32: Configuración de un resonador Courtney [13].

- **Método de perturbación resonante**

Este método utiliza una estructura resonante en la que se introduce la muestra del material. Se podrá calcular la permitividad a partir de la frecuencia de resonancia y el factor de calidad, al igual que en el método resonador, pero ésta también dependerá de las condiciones de contorno, que varían según perturbaciones en la forma de la cavidad, en las pérdidas de las paredes del resonador o en el material. La estructura o cavidad resonante consiste en una guía de ondas cortocircuitada en ambos extremos, produciéndose ondas estacionarias en la dirección de propagación. En la cavidad, se introduce una pequeña muestra de material, lo cual provoca una variación de la frecuencia de resonancia y del factor de calidad. A partir de ello, se puede determinar la permitividad y permeabilidad del material. En la Figura 2.33, se muestra un esquema de la cavidad resonante con una muestra del material bajo estudio.

El método de perturbación resonante presenta las ventajas de rapidez y precisión en la medida, empleo de muestras de material de tamaño reducido y realización de medidas a temperaturas altas. Sin embargo, se trata de un método destructivo y es complicado fabricar la cavidad resonante con precisión.

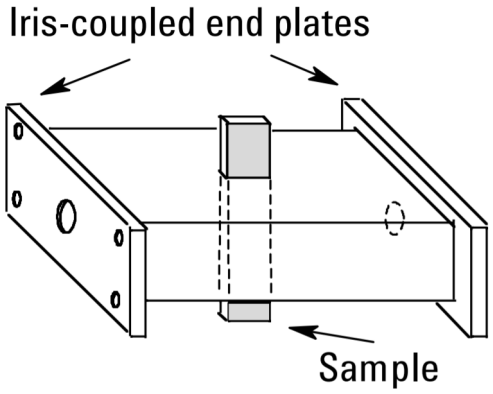


Figura 2.33: Método de perturbación resonante [16].

Capítulo 3

Diseño de la solución

En este capítulo, se ha desarrollado todo el proceso llevado a cabo para realizar el diseño del analizador de redes vectorial.

Una vez presentados los objetivos del presente proyecto y analizado el contexto en torno al mismo, se puede observar que la tecnología basada en la radiofrecuencia para la obtención de la permitividad de los materiales a partir de los parámetros S es una de las más utilizadas y de las más interesantes, debido a su precisión, alta velocidad para obtener los resultados y reducido coste, destacando sobre otras tecnologías. Además, se está desarrollando cada vez más una electrónica que hace posible el diseño de analizadores de redes vectoriales más precisos y rápidos a la hora de realizar las medidas y la calibración. Añadido a esto, se puede lograr el desarrollo de un VNA de tamaño reducido utilizando componentes pequeños, precisos y de bajo coste.

El diseño del analizador de redes vectorial del presente proyecto se basa en el diseño del NanoVNA llevado a cabo por Tomohiro Takahashi [42], que se trata de un proyecto *open source* (libre para usarlo y modificarlo). El objetivo consiste en adaptar el diseño a un VNA más simple de un solo puerto, logrando reducir su coste y su tamaño. Esta etapa consta de una fase de diseño del esquemático del circuito electrónico, una fase de diseño de la placa de circuito impreso (PCB, *Printed Circuit Board*), una fase de adquisición de los componentes necesarios de distintos fabricantes, una fase de montaje, donde se sueldan los distintos componentes electrónicos a la placa, y una fase de programación, en la que se programará el Firmware para el microcontrolador del VNA.

3.1. Diseño del esquemático

Para diseñar el analizador de redes vectorial, hay que tener en cuenta las distintas partes básicas de las que consta la arquitectura de este tipo de dispositivos. En primer lugar, dispondrá de la parte de alimentación, que se encargará de proporcionar energía a todo el sistema. También dispondrá de un generador de señales, que realizará el barrido en frecuencia y proporcionará la señal de prueba. Además, tendrá separadores de señal formados por divisores de tensión y un puente de Wheatstone para el puerto de reflexión. A todo lo anterior se le añade receptores de señales, formados por mezcladores de frecuencias y un códec, que se encargarán de recibir y procesar la señal y, por último, se tiene un procesador que procesará la señal procedente del receptor, así como puertos externos para poder acceder a ciertas partes del procesador desde el exterior.

3.1.1. Diagrama de bloques

A continuación, se muestra el diagrama de bloques correspondiente al diseño que se ha realizado para un VNA de 2 puertos y para un VNA de 1 puerto, mostrando finalmente el esquemático diseñado para cada uno a partir de su correspondiente diagrama de bloques.

En este proyecto, se centra la atención en el diseño e implementación del VNA de 1 puerto, para el que se realiza su diagrama de bloques, su esquemático, su PCB, se fabrica el prototipo físicamente y se realizan experimentos con el mismo. Pero se ha realizado el diseño de un VNA de 2 puertos (diagrama de bloques y esquemático) con más funcionalidades para futuras aplicaciones y trabajos futuros.

3.1.1.1. VNA de 2 puertos

El diagrama de bloques correspondiente al VNA de 2 puertos diseñado está formado por una parte de alimentación, un generador de señales, separadores de señal, receptores de señal, procesador y pantalla y puertos externos, tal y como se observa en la Figura 3.1.

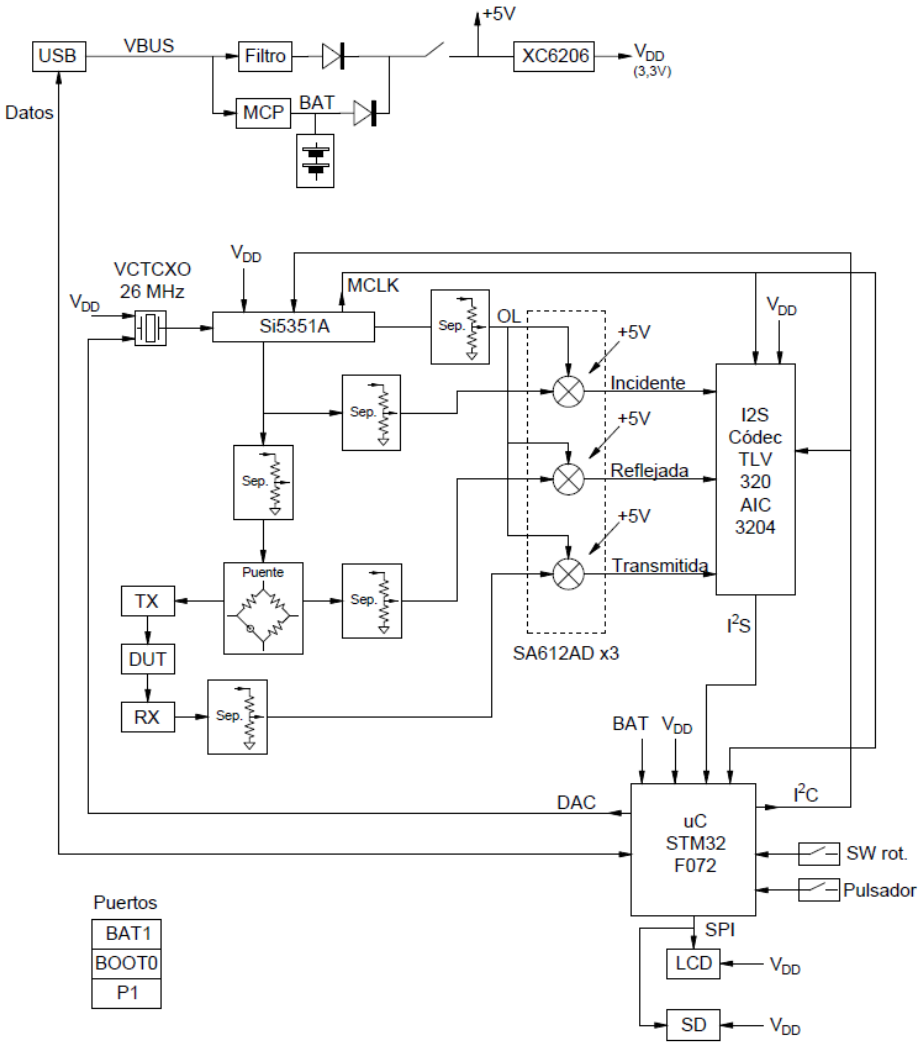


Figura 3.1: Diagrama de bloques del VNA de 2 puertos.

La alimentación parte del USB, cuya tensión pasa por un filtro y por un controlador de carga de batería MCP73831T que sirve para cargar una batería externa de Litio. Tras esto, pasa por el interruptor general y por un regulador de tensión lineal XC6206P331MR de forma que, antes del regulador, se obtienen +5 V y, después del regulador, se obtiene V_{DD} (3,3 V). La señal de V_{DD} alimenta al oscilador, al generador de señales Si5351A, al códec TLV320AIC3204, al microcontrolador STM32F072, a la pantalla LCD y a la tarjeta SD. La señal de +5 V alimenta a los mezcladores de frecuencias SA612AD. La batería también alimentará el microcontrolador.

El generador de señales Si5351A está alimentado por un oscilador VCTCXO de 26 MHz que, a su vez, está controlado por una señal DAC del microcontrolador. El Si5351A se comunica con el microcontrolador por I^2C, y ofrece tres señales: una señal para el oscilador local de los mezcladores de frecuencias, la señal de prueba que hace el barrido en frecuencia y una señal de reloj MCLK que sincroniza el microcontrolador y el códec.

Tras el generador de señales, se encuentran los separadores de señales. La señal de oscilación local de los mezcladores de frecuencias y la señal incidente presentan un separador de señal en forma de divisor de tensión resistivo. La señal reflejada posee como separadores de señal un divisor de tensión, un puente de Wheatstone para el puerto de reflexión y un divisor de tensión a la salida de este último. Por último, la señal transmitida tiene un atenuador en forma de pi equilibrado como separador de señal.

Tras los separadores de señales, la señal llega a los receptores de señales. Las señales que proceden de los separadores de señal entran en los mezcladores de frecuencias SA612AD, tanto las señales incidente, reflejada y transmitida como la señal de oscilación local. Tras los mezcladores, se dispone de un filtro paso bajo y, tras ello, las distintas señales entran a los canales de entrada del códec TLV320AIC3204. El códec se comunica por I^2C con el microcontrolador, y le manda por I^2S el resultado del procesamiento digital de las señales entrantes.

Las señales procesadas llegan al procesador por I^2S. El componente que se encarga del procesamiento es el microcontrolador STM32F072, que se comunica por I^2C con el códec y el generador de señales, controla el oscilador de 26 MHz con una señal DAC, recibe señales de un interruptor giratorio de tres posiciones y de un pulsador, se comunica por SPI con la pantalla LCD y la tarjeta SD e intercambia datos con el USB. El procesador mostrará el resultado de los datos en la pantalla LCD y podrá exportar los datos en una tarjeta SD.

Por último, se dispone de puertos externos con los que se podrá acceder a distintas partes del procesador, que son el puerto BAT1, que permite conectar la batería externa de Litio, el puerto BOOT0, que permite puentear el pin BOOT0 del microcontrolador para actualizar el Firmware, y el puerto P1, que permite realizar la depuración y programación del Firmware a través de SWD.

Tras el diseño y el análisis del diagrama de bloques para el VNA de 2 puertos, se ha procedido al desarrollo de su esquemático, que ofrece las funcionalidades descritas. Las distintas partes del esquemático se pueden ver en la Figuras 3.2, 3.3 y 3.4, aunque el esquemático completo se adjunta en el Anexo III.1 para su completa visualización.

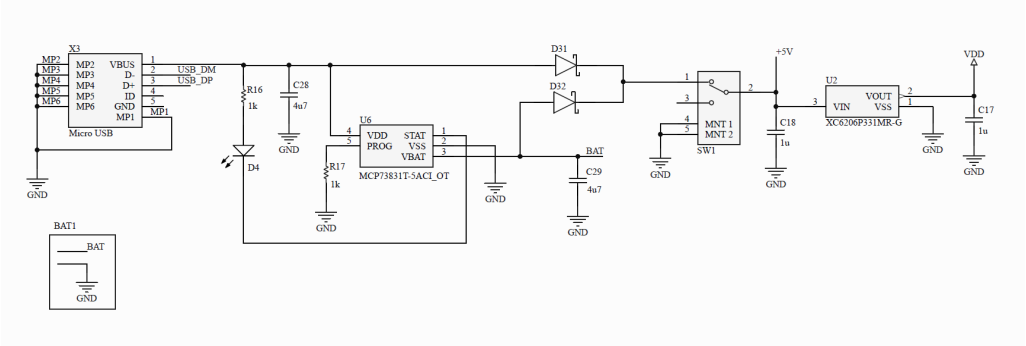


Figura 3.2: Esquemático del VNA de 2 puertos. USB, alimentación y puerto BAT1.

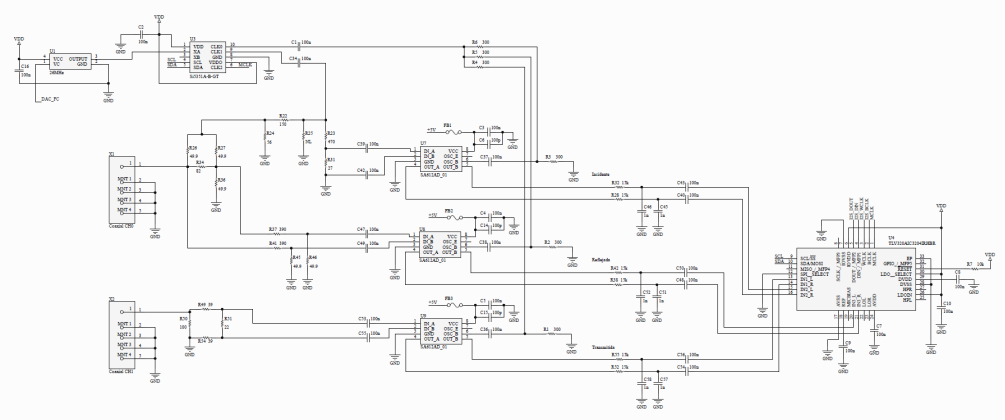


Figura 3.3: Esquemático del VNA de 2 puertos. Generador de señales, separadores de señales y receptores de señales.

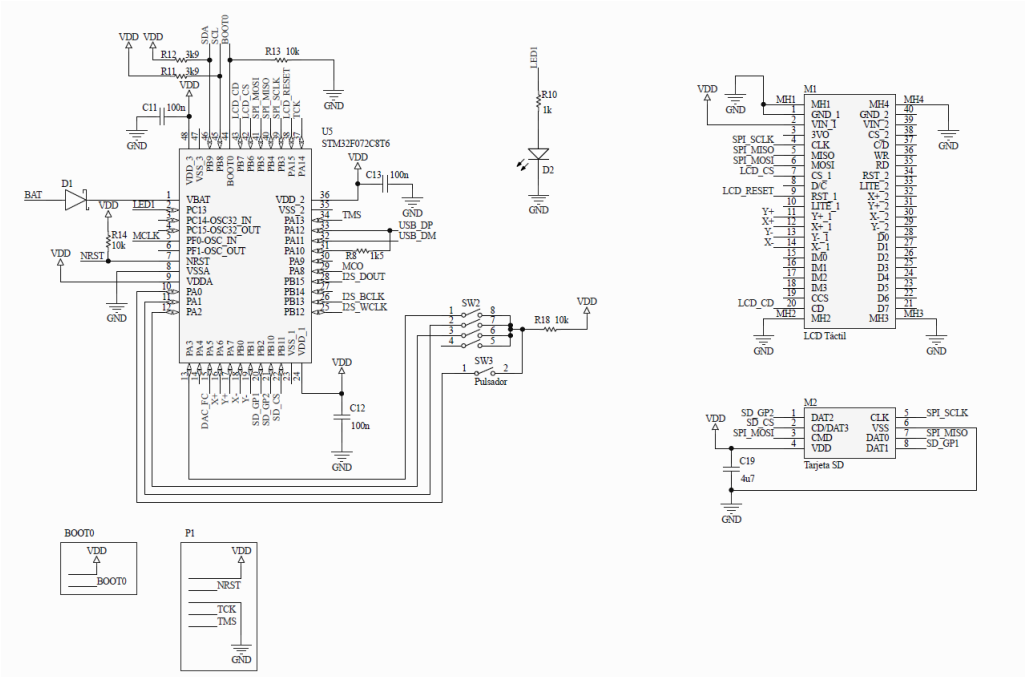


Figura 3.4: Esquemático del VNA de 2 puertos. Microcontrolador, pantalla LCD, tarjeta SD y puertos BOOT0 y P1.

3.1.1.2. VNA de 1 puerto

El diagrama de bloques correspondiente al diseño del VNA de 1 puerto está formado, de forma similar al VNA de 2 puertos, por una parte que se encarga de la alimentación del sistema, un generador de señales, separadores de señal, receptores de señal, procesador y puertos externos. Este es el VNA con el que se trabajará en este proyecto y, por tanto, los bloques por los que está formado se explicarán con más detalle en los siguientes apartados. El diagrama de bloques se puede ver en la Figura 3.5:

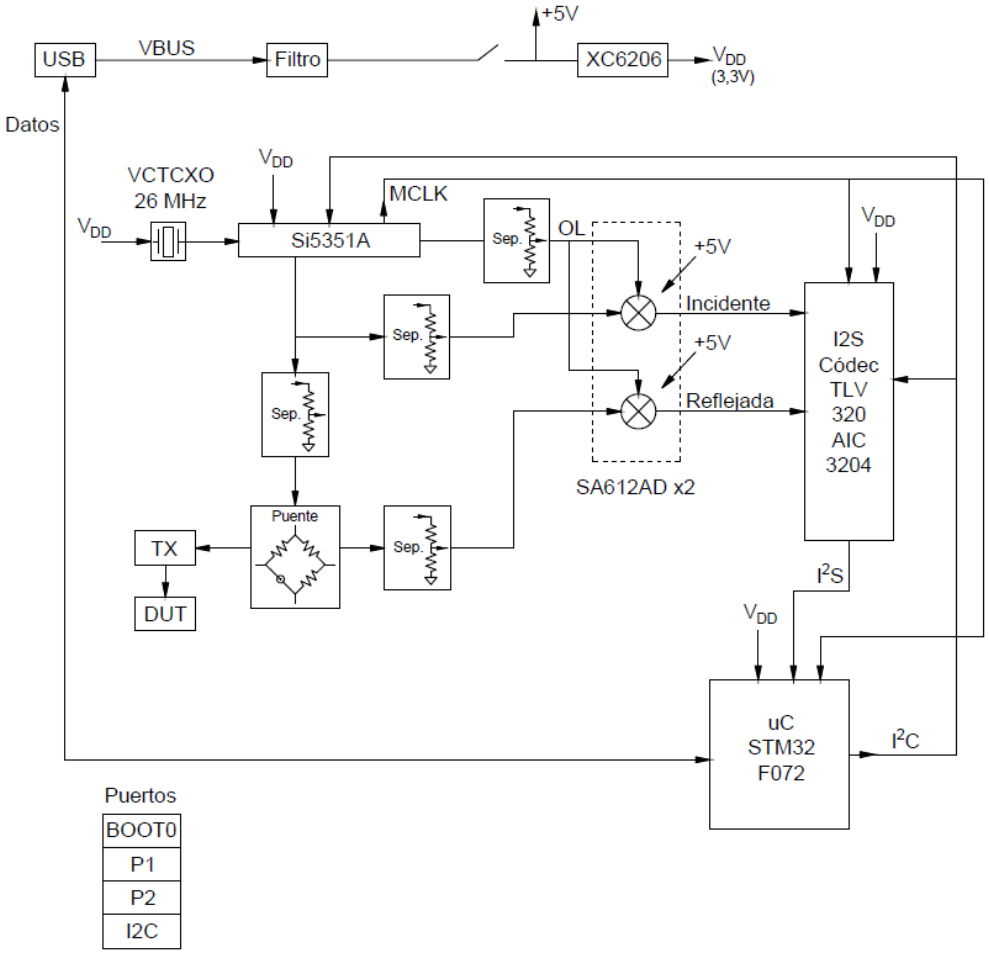


Figura 3.5: Diagrama de bloques del VNA de 1 puerto.

El funcionamiento y la relación existente entre todos los bloques que componen el VNA de 1 puerto se corresponden con la explicación dada para el VNA de 2 puertos en el apartado anterior, salvo algunas diferencias. La versión del VNA de 1 puerto no presenta un cargador de batería ni una batería externa, el oscilador de 26 MHz no está controlado por el microcontrolador, no incorpora el puerto de transmisión ni el atenuador ni el mezclador de frecuencias SA612AD correspondiente a la onda transmitida, no tiene el interruptor giratorio ni el pulsador, tampoco incorpora la pantalla LCD ni la tarjeta SD y no tiene el puerto BAT1, añadiendo el puerto externo P2 que permite realizar la comunicación con el microcontrolador a través de UART. Además, las alimentaciones para todos los componentes están convenientemente filtradas, para reducir el ruido en todo el sistema.

Una vez diseñado y analizado el diagrama de bloques para el VNA de 1 puerto, que es el que se implementará en el proyecto, se ha realizado el diseño de su esquemático. Las distintas partes del esquemático se pueden ver en la Figuras 3.6, 3.7 y 3.8, aunque el esquemático completo se adjunta en el Anexo III.2 para su completa visualización.

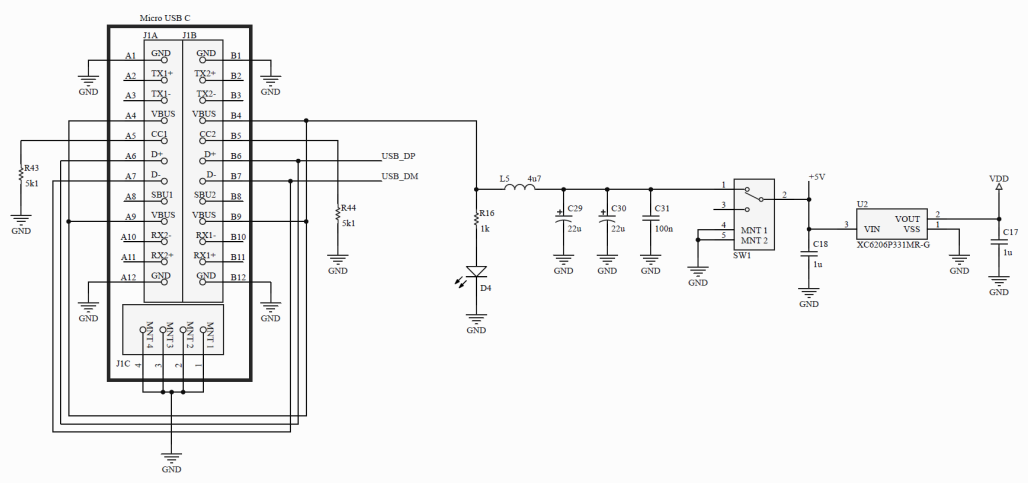


Figura 3.6: Esquemático del VNA de 1 puerto. USB y alimentación.

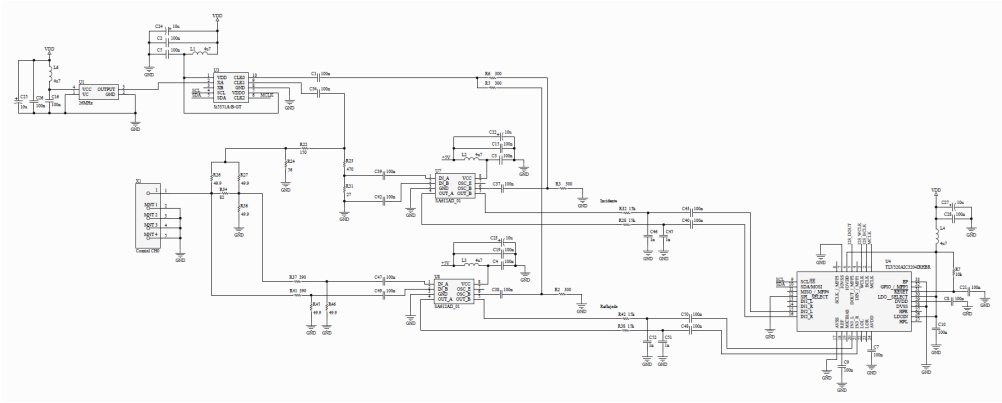


Figura 3.7: Esquemático del VNA de 1 puerto. Generador de señales, separadores de señales y receptores de señales.

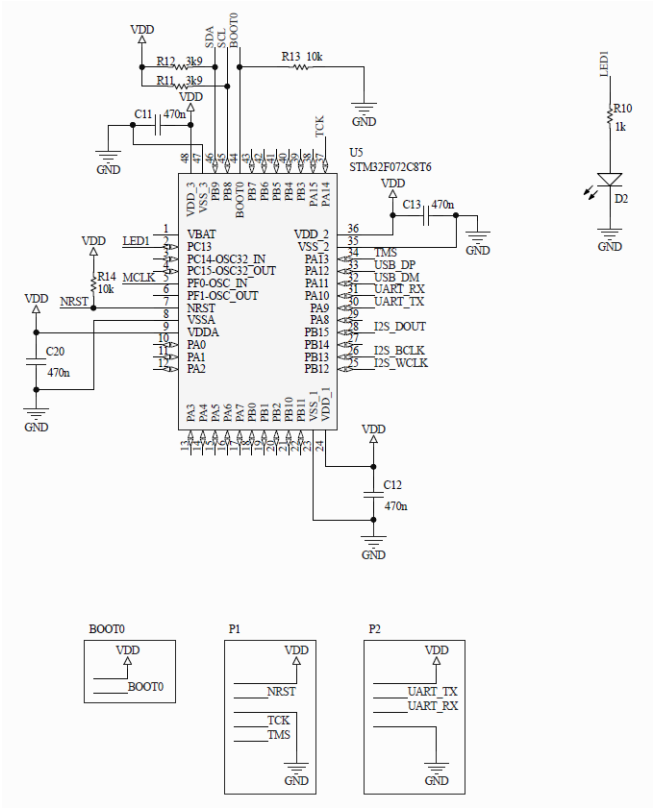


Figura 3.8: Esquemático del VNA de 1 puerto. Microcontrolador y puertos BOOT0, P1 y P2.

También se ha realizado otra versión del VNA de 1 puerto. Los cambios incluidos en esta versión han sido cambiar el conector tipo micro-USB C por sus cuatro pines esenciales: dos pines de alimentación y dos pines de datos, para facilitar la soldadura ya que, en varias pruebas que se hicieron con el otro modelo, resultaba difícil corregir la soldadura en caso de que se produjeran uniones entre pads, y también se ha incluido un nuevo puerto I2C que permite realizar la comunicación con el microcontrolador de forma externa a través de I^2C. El funcionamiento de esta versión sigue siendo el mismo. Cambiará, esencialmente, la PCB con esta versión, ya que su esquemático es prácticamente igual. Se pueden visualizar los cambios en el esquemático en las Figura 3.9, y el esquemático completo se adjunta en el Anexo III.3 para su completa visualización.

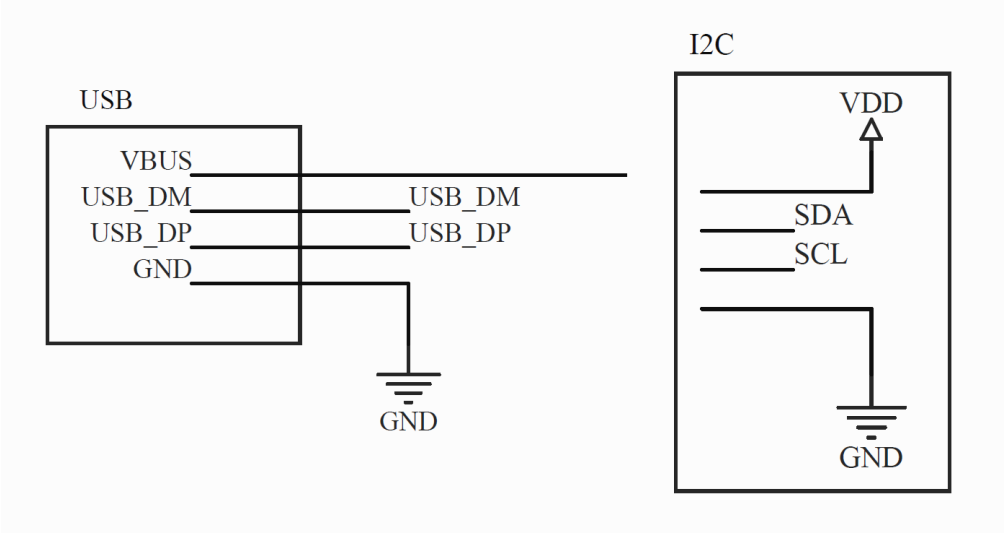


Figura 3.9: Cambios en la última versión del esquemático del VNA de 1 puerto. USB y puerto I2C.

3.1.2. Bloques detallados del esquemático

Como ya se ha mencionado, en este proyecto se realizará el diseño e implementación del VNA de 1 puerto. Por tanto, tras el análisis de su diagrama de bloques y la visualización de su esquemático, se realizará a continuación el análisis más detallado de cada uno de los bloques que componen su esquemático.

3.1.2.1. Alimentación

La alimentación comienza con el USB, que aporta la energía a todo el sistema a través de VBUS. La tensión proveniente de VBUS enciende un diodo LED para indicar que el USB está conectado, luego atraviesa una etapa de filtrado paso bajo para aumentar la estabilidad de la tensión, pasa por el interruptor general, que permitirá que la energía le llegue o no al resto del sistema y, tras el interruptor, la tensión llega al regulador de tensión lineal XC6206P331MR.

De esta forma, la alimentación obtenida para los componentes del VNA es de +5 V antes del regulador lineal y de V_{DD} (3,3 V) tras el regulador. La señal de V_{DD} alimenta al oscilador, al generador de señales Si5351A, al códec TLV320AIC3204 y al microcontrolador STM32F072. Por otra parte, la señal de +5 V alimenta a los mezcladores de frecuencias SA612AD. Estas alimentaciones están convenientemente filtradas para todos los componentes, para reducir el ruido en todo el sistema. Además, es importante señalar que se utilizarán los pines de datos USB_DP y USB_DM del USB para intercambiar datos con el microcontrolador. Se puede visualizar esta etapa en la Figura 3.10:

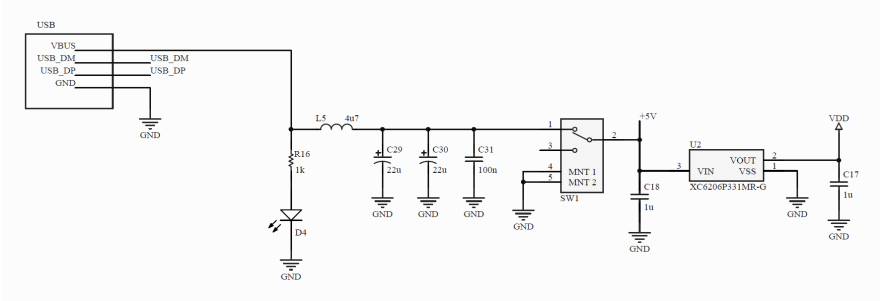


Figura 3.10: Bloque de alimentación.

3.1.2.2. Generador de señales

La generación de señales haciendo un barrido en frecuencia la realiza el componente Si5351A. A su vez, el generador de señales está alimentado por un oscilador VCTCXO (oscilador de cristal con compensación de temperatura y controlado por tensión) de 26 MHz, que ofrece una onda senoidal recortada. Como se puede observar, ambos componentes están alimentados con una tensión V_{DD} a la que se ha aplicado un filtro paso bajo para reducir el posible ruido.

El generador de señales Si5351A se comunica con el microcontrolador por I^2C, correspondiente a las señales SCL y SDA, de forma que el microcontrolador le indica el barrido de frecuencias que tiene que realizar en cada momento. Ofrece a su salida tres señales distintas de oscilación, que consisten en una señal para el oscilador local de los mezcladores de frecuencias SA612AD, la señal de prueba para el dispositivo bajo prueba que hace el barrido en frecuencia y una señal de reloj MCLK que permite la sincronización del microcontrolador y el códec TLV320AIC3204. Las señales de oscilación local y la señal de prueba pasan por condensadores de acoplo para eliminar posibles componentes de continua presentes en las señales.

El protocolo I^2C (Inter-Integrated Circuit) es un protocolo de comunicación serie síncrono. Utiliza dos líneas de señal: SDA (*Serial Data*), que es el flujo de datos, y SCL (*Serial Clock*), que es el reloj. Estas señales necesitan resistencias de pull-up, las cuales se pueden observar en el microcontrolador. En la Figura 3.11 se puede visualizar este bloque:

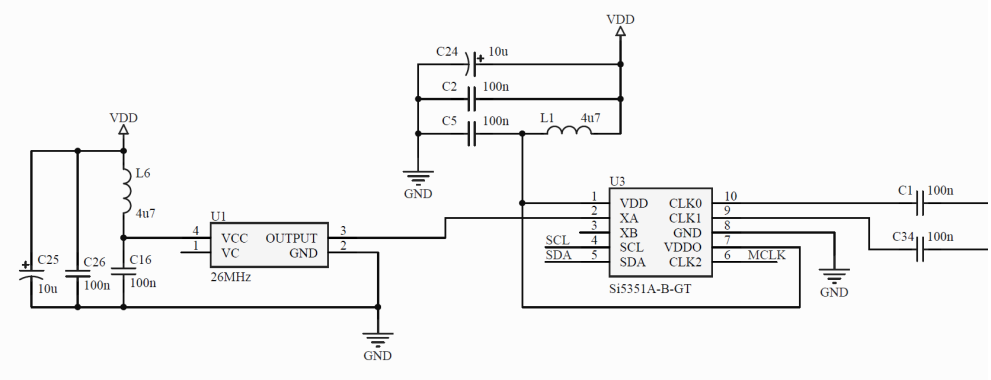


Figura 3.11: Bloque del generador de señales.

3.1.2.3. Separadores de señal

Después del generador de señales, se encuentran los separadores de señales, que se encargan de obtener la medida de la señal incidente como referencia y de separar la onda incidente de la onda reflejada a la entrada del dispositivo bajo prueba.

La señal de oscilación local de los mezcladores de frecuencias SA612AD y la señal incidente presentan un separador de señal en forma de divisor de tensión resistivo. Por su parte, la señal reflejada posee como separadores de señal un divisor de tensión, un puente de Wheatstone para el puerto de reflexión formado por resistencias de 49,9 Ω para compararlas con la resistencia de la línea de transmisión conectada al puerto y dos divisores de tensión resistivos a la salida del puente. Se puede visualizar esta parte remarcada en la Figura 3.12:

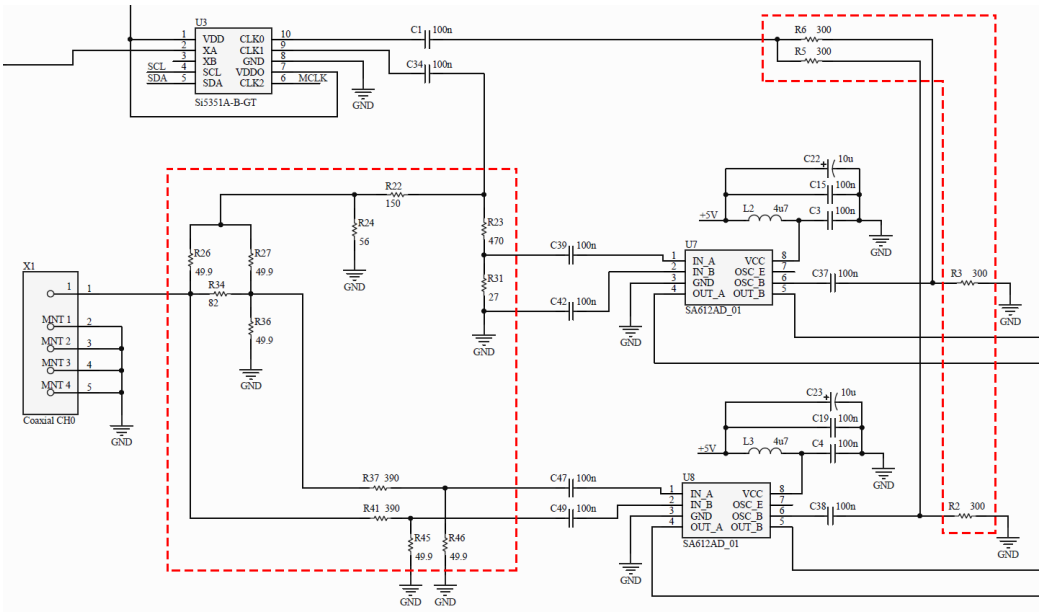


Figura 3.12: Bloque de los separadores de señal.

3.1.2.4. Receptores de la señal

Las señales, tras los separadores de señales, llegan a los receptores de señales, que se encargan de realizar las mediciones y el procesamiento de las mismas. Las señales procedentes de los separadores de señales entran en los mezcladores de frecuencias SA612AD, tanto las señales incidente y reflejada como la señal de oscilación local de cada uno, todas ellas pasando por condensadores de acoplo.

La alimentación de los mezcladores se realiza con +5 V, que son convenientemente filtrados con una etapa de filtro paso bajo en forma de pi para reducir el posible rizado de la señal. El primer mezclador toma la señal incidente y el segundo mezclador recibe la señal reflejada por el puerto de reflexión. Los mezcladores de frecuencias son circuitos que multiplican las dos señales de entrada, produciendo a su salida una señal que es la mezcla de las señales de entrada, con una ganancia. Se dispone de una señal de alta frecuencia en una de sus entradas y de un oscilador local en la otra entrada, de forma que a la salida se tendrá el producto de ambas, que será una señal a frecuencia intermedia, y que incluye la suma y la diferencia de las frecuencias de las señales de entrada. El mezclador de frecuencias SA612AD utiliza una célula de Gilbert para realizar la operación del producto de las señales entrantes.

Tras los mezcladores de frecuencias, se dispone de un filtro paso bajo para cada una de sus salidas y, tras el filtro, las distintas señales llegan a los canales de entrada del códec TLV320AIC3204 a través de condensadores de acoplo. El códec posee un ADC (conversor analógico/digital), se comunica por I^2C con el microcontrolador y le manda por I^2S el resultado del procesamiento digital de las señales entrantes. El códec también está alimentado convenientemente con la señal V_{DD} filtrada para aumentar su estabilidad.

El protocolo I^2S (Inter-IC Sound, Integrated Interchip Sound) es un protocolo de comunicación serie síncrono usado para la transmisión de audio digital. Utiliza tres líneas de señal: BCLK (*Bit Clock*), que es el reloj de bits, WCLK (*Word Clock*) que es el reloj de palabras, y líneas de datos DIN (*Data in*) y DOUT (*Data out*).

El códec de audio TLV320AIC3204 incluye un conjunto de algoritmos que permiten codificar y decodificar las señales entrantes. Para ello, tras un ADC, comprime (codifica) la señal de audio, reduciendo su tamaño, y

formando un flujo de bits codificado, llamado stream. Luego, descomprime (decodifica) dicho flujo de bits para poder reproducir la señal, en un formato más apropiado. Es decir, la parte del codificador comprime la señal en un flujo de bits codificado en binario y la parte del decodificador restaura la señal para que se pueda reproducir. El resultado final es reducir el tamaño de la señal transmitida manteniendo una alta calidad final. Finalmente, el códec TLV320AIC3204 realiza el procesamiento digital de la señal y da el resultado por I^2S, que mandará al microcontrolador. Con esto, se obtiene información tanto de la magnitud como de la fase de la señal. La etapa de los receptores de señal se puede visualizar en la Figura 3.13:

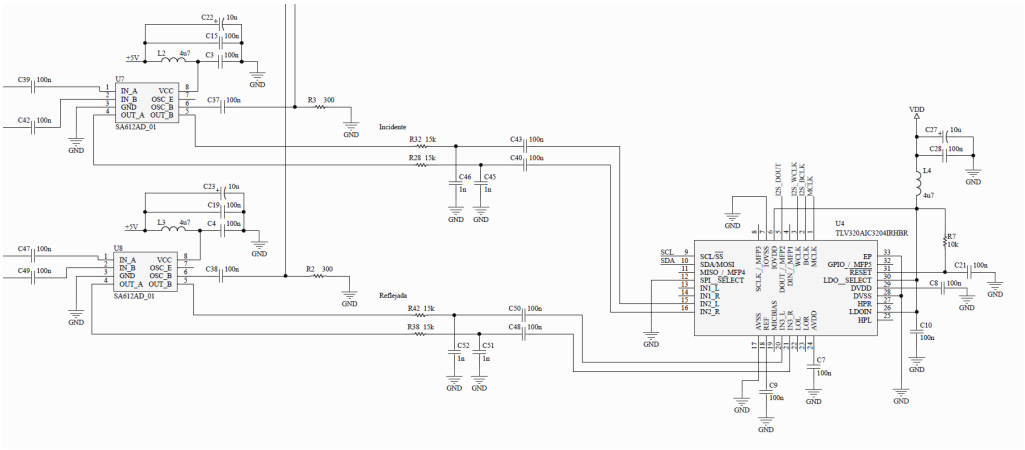


Figura 3.13: Bloque de los receptores de señal.

3.1.2.5. Procesador

Las señales procesadas por el códec TLV320AIC3204 llegan al procesador por I^2S. El componente que realiza el procesamiento de dichas señales consiste en el microcontrolador STM32F072, que posee un núcleo ARM de 32 bits Cortex-M0, una frecuencia de 48 MHz, una memoria Flash de 64 KB y una memoria RAM de 16 KB. El microcontrolador se comunica por I^2C con el códec y el generador de señales e intercambia datos con el USB a través de las señales USB_DP y USB_DM.

La alimentación del microcontrolador se realiza a través de V_{DD}, que se estabiliza con condensadores de 470 nF. Del procesador parten las señales SCL y SDA de I^2C con sus resistencias de pull-up, la señal de BOOT0 con una resistencia de pull-down, la señal de reset NRST con una resistencia de pull-up, las señales TMS y TCK para la comunicación por SWD y las señales de UART_RX y UART_TX para la comunicación por UART. Además, recibe las señales de I^2S del códec I2S_BCLK, I2S_WCLK e I2S_DOUT y la señal de reloj MCLK del generador de señales. Por último, ofrece una señal LED1 que activará el diodo LED conectado a ella. En la Figura 3.14 se puede ver el microcontrolador STM32F072 con todas sus conexiones:

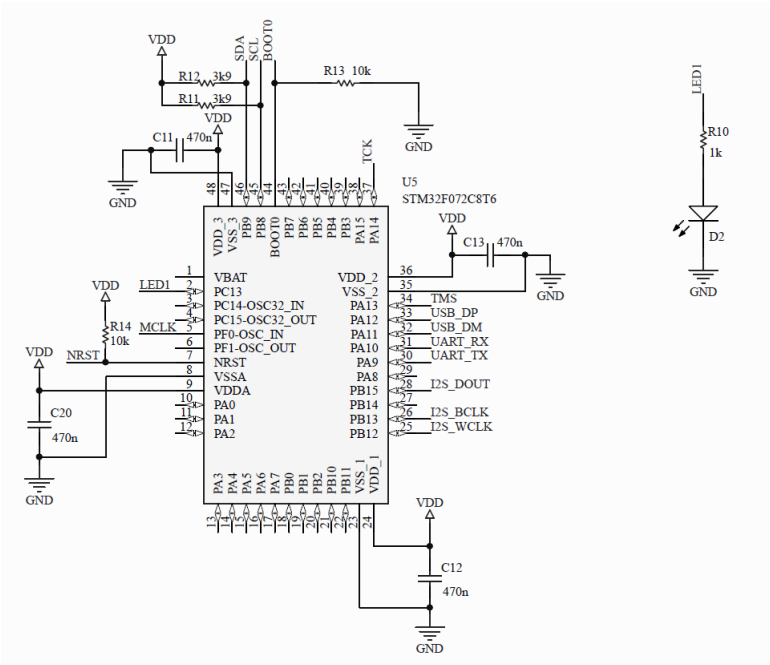


Figura 3.14: Bloque del microcontrolador.

3.1.2.6. Puertos externos

Finalmente, el sistema dispone de puertos externos que permiten acceder a distintas partes del microcontrolador STM32F072. Se trata del puerto BOOT0, formado por las señales BOOT0 y V_{DD}, que permite puentear el pin BOOT0 del microcontrolador con V_{DD} para actualizar el Firmware, el puerto P1, formado por las señales V_{DD}, NRST, GND, TCK y TMS, que permite realizar la depuración y programación del Firmware a través de SWD, el puerto P2, formado por las señales V_{DD}, UART_TX, UART_RX y GND, que permite realizar la comunicación con el microcontrolador a través de UART, y el puerto I2C, formado por las señales V_{DD}, SDA, SCL y GND, que posibilita la comunicación externa con el microcontrolador a través de I^2C. En la Figura 3.15 se pueden ver todos los puertos externos:

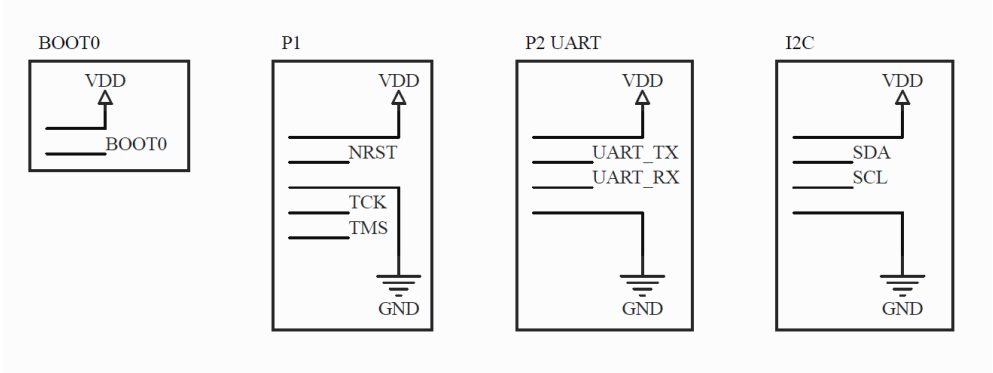


Figura 3.15: Bloque de los puertos externos.

3.2. Diseño de la PCB

Para el diseño del esquemático y de la PCB se ha utilizado el software de automatización de diseño electrónico y PCB para placas de circuito impreso Altium Designer. Se trata de un software de diseño de PCB que combina esquemáticos, diseño y todo lo necesario en un solo entorno para diseñar placas de circuito impreso.

Figura 3.16: Altium Designer. Software de diseño de PCB.

3.2.1. Elección de componentes

En primer lugar, a la hora de diseñar el esquemático, para luego convertirlo en la PCB, hay que hacer una elección de los componentes utilizados, buscándolos en las librerías de componentes de Altium o buscando los modelos en páginas web que los ofrecen, como *Component Search Engine*.

Los modelos de los componentes deben tener el símbolo del esquemático, la huella para la PCB y, opcionalmente, su modelo en 3D. Los componentes finalmente seleccionados y utilizados se han explicado en los apartados anteriores, y se realizará un listado de los mismos en la siguiente sección. En la Figura 3.17, se puede ver la búsqueda de componentes usando el buscador de Altium, el cual ofrece varias opciones para cada componente, información detallada del mismo y una vista previa del símbolo del esquemático, la huella para la PCB y el modelo en 3D.

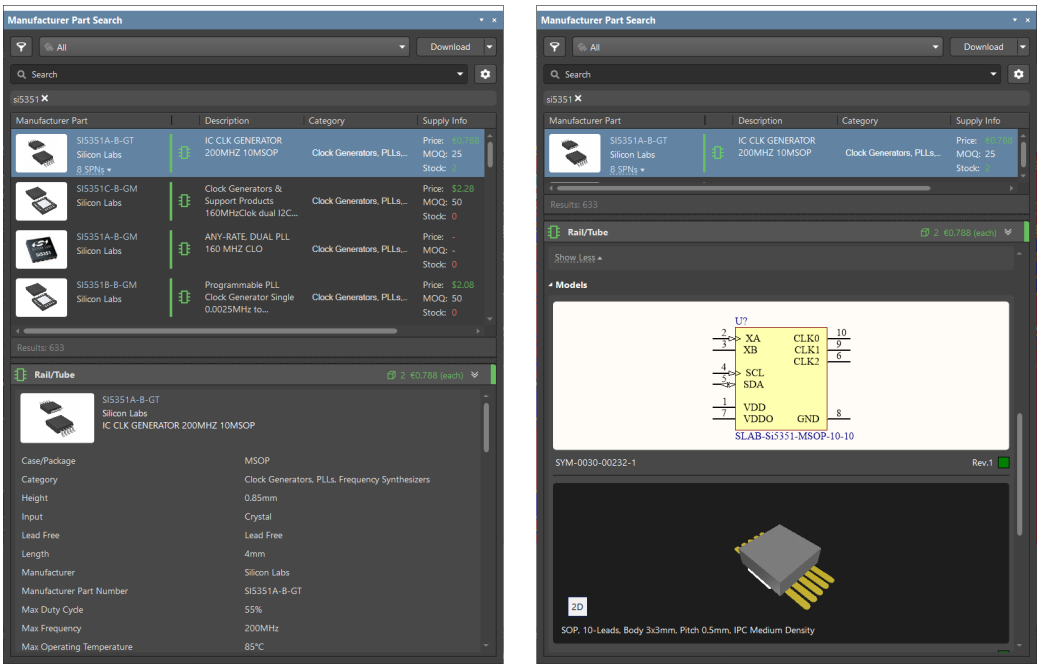


Figura 3.17: Búsqueda de componentes con Altium Designer.

3.2.2. Colocación de componentes y tamaño de la PCB

Una vez realizado el esquemático con todos los componentes conectados entre sí, se procede a la generación de la PCB. Para ello, se importan todos los componentes utilizados en el esquemático en la PCB, de forma que se visualizará la huella de cada componente en la misma. Altium permite alternar entre una vista 2D y una vista 3D de la PCB, en caso de que los componentes dispongan del modelo en 3D. Para la edición de la PCB, se utilizará la vista en 2D.

Tras incorporar todos los componentes en la PCB, se procede a la colocación de los mismos en la posición más óptima posible, de forma que se minimicen las longitudes de las pistas lo máximo posible, e intentando ocupar el menor espacio posible.

La posición del interruptor general, del USB y del puerto de reflexión con conector SMA deben estar en el borde de la placa, de forma que se pueda acceder desde el exterior a los mismos fácilmente. La parte de regulación de

tensión se sitúa cerca del interruptor general y del USB. El microcontrolador STM32F072 se coloca en una posición que le permite conectarse con el mayor número de componentes posible de una forma sencilla. Tras este último, se encuentran el generador de señales Si5351A junto con el oscilador VCTCXO de 26 MHz y el códec TLV320AIC3204. Por último, se encuentra la etapa de los mezcladores de frecuencias SA612AD y, tras ellos, se tiene el resto de componentes pasivos y el puerto de reflexión. Además, se debe dejar un espacio en las esquinas para incluir orificios que permitan atornillar la placa y se ha dejado un espacio en la parte izquierda para incluir los puertos externos.

Al finalizar la colocación final de los componentes, se procede a delimitar el tamaño que tendrá la placa físicamente, indicando los bordes de la misma. Se puede ver en la Figura 3.18 los componentes convenientemente colocados, según lo indicado, y la forma de la PCB ya definida.

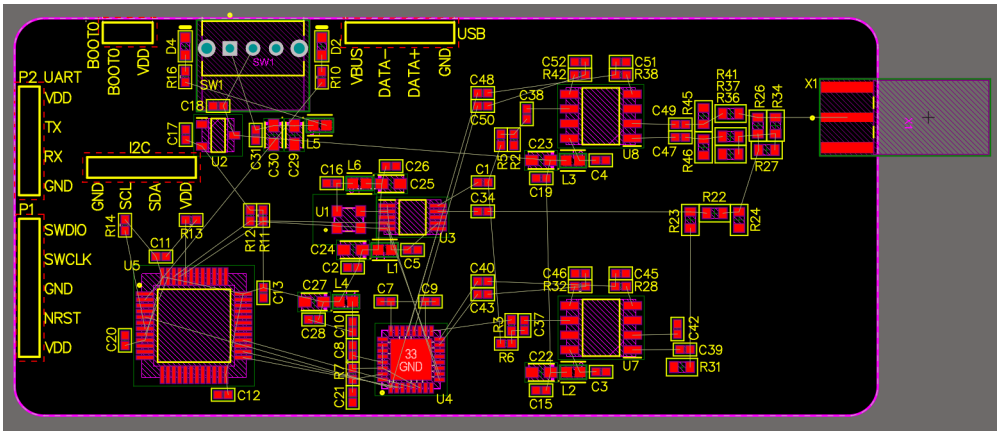


Figura 3.18: Colocación de los componentes y definición de la PCB.

3.2.3. Trazado de las pistas y finalización

Teniendo todos los componentes convenientemente colocados en la PCB, falta realizar el trazado de las pistas que conectará los componentes de la placa entre sí. Durante la edición de la PCB, se observan unas líneas indicadoras que unen los pads de los distintos componentes. Estas líneas indican la conexión que tienen los componentes entre sí, correspondientes a las conexiones realizadas en el esquemático entre dichos componentes. Por tanto, se deben realizar todas las conexiones indicadas por esas líneas mediante pistas en la PCB. Se pueden observar dichas líneas en la anterior Figura 3.18.

Antes de proceder al trazado de las pistas, se colocan los orificios donde se podrá atornillar la placa, los cuales irán conectados a la red de GND, y se colocan los pads correspondientes a cada una de las señales que forman los distintos puertos externos, los cuales están conectados a sus respectivas señales.

El tamaño de las pistas utilizado ha sido de 0,5 mm para la parte de potencia, 0,2 mm para las conexiones con las patillas del microcontrolador y 0,3 mm para el resto de pistas. Estos tamaños se recogen en la siguiente tabla, en la que también se especifican, además del tamaño preferido de la pista, el tamaño mínimo y máximo que puede tener para casos puntuales en los que se requiera:

Tabla 3.1: Tamaño de las pistas de la PCB.

Pista	Preferido	Mínimo	Máximo
Potencia	0,5 mm	0,3 mm	0,8 mm
Microcontrolador	0,2 mm	0,2 mm	0,2 mm
General	0,3 mm	0,2 mm	0,5 mm

Se puede visualizar la realización de esta configuración en el programa en la Figura 3.19:

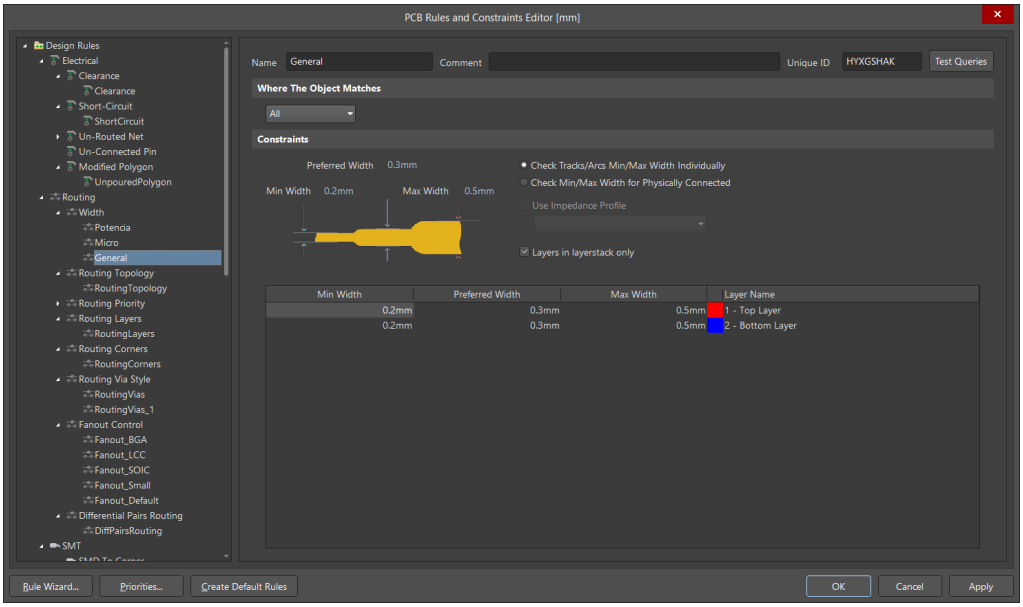


Figura 3.19: Configuración del tamaño de las pistas.

Una vez trazadas todas las pistas necesarias que conectan todos los componentes y los pads añadidos entre sí, utilizando tanto la capa superior como la capa inferior, se realiza un paso final, que consiste en hacer un *pour* de la red GND. Esto consiste en rellenar toda la placa de cobre de forma que las pistas conectadas a la red GND se funden con ese relleno y las pistas que son distintas quedan separadas del relleno. Este relleno se realiza tanto en la capa superior como en la capa inferior.

Para favorecer la disipación y aumentar la conectividad entre las dos capas de la PCB, se ha realizado una serie de vías en toda la PCB, de forma que el relleno de GND queda unido en muchos puntos en las dos capas. Además, se ha quitado una parte del relleno en la zona que se sitúa entre el generador de señales y los mezcladores de frecuencias. La colocación de los componentes, el tamaño de las pistas, el trazado de las pistas, el relleno de GND, las vías entre las dos capas de la PCB y la eliminación de la parte indicada del relleno de GND permiten reducir al máximo posible la presencia de ruido y aumentar la estabilidad y precisión del sistema.

Finalmente, solo queda colocar los nombres de cada uno de los componentes y de los pads de forma que se vean correctamente, en su capa correspondiente. Además, se ha añadido una serie de indicaciones en esta última capa, que son el logo de la UPCT, el nombre del autor, el nombre del dispositivo, el símbolo que representa la función del puerto de reflexión (S11) y el código al que pertenece el proyecto. En la Figura 3.20, se puede ver la PCB completamente terminada:

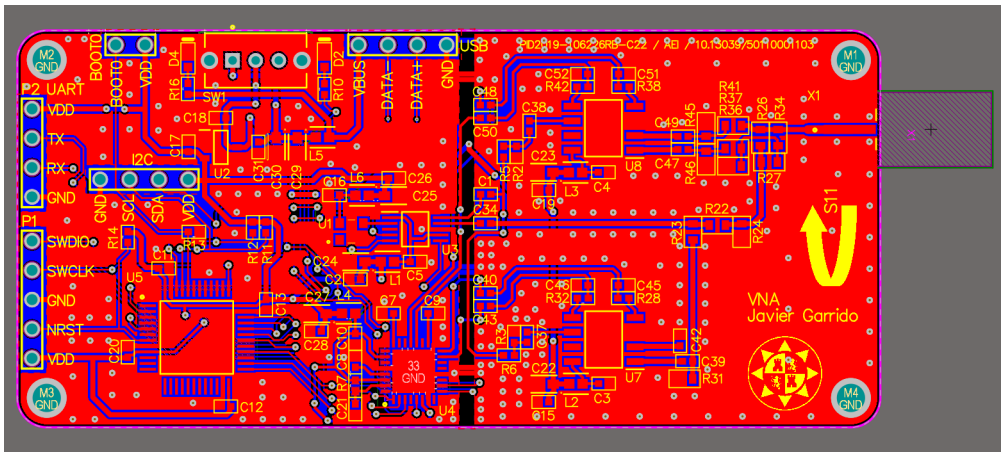


Figura 3.20: PCB del VNA terminada.

En las Figuras 3.21 y 3.22 se pueden visualizar las pistas de la parte superior e inferior de la PCB, las cuales se adjuntan en el Anexo IV.2.

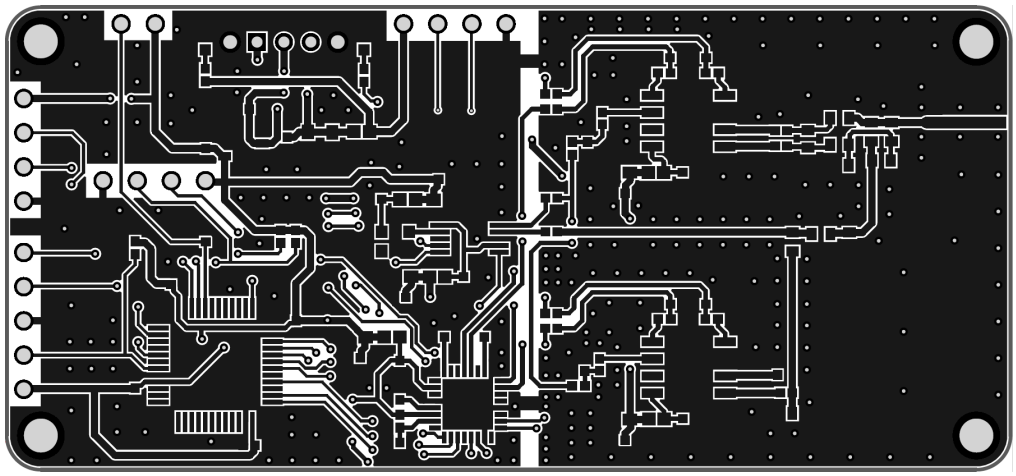


Figura 3.21: Pistas de la PCB. Parte superior.

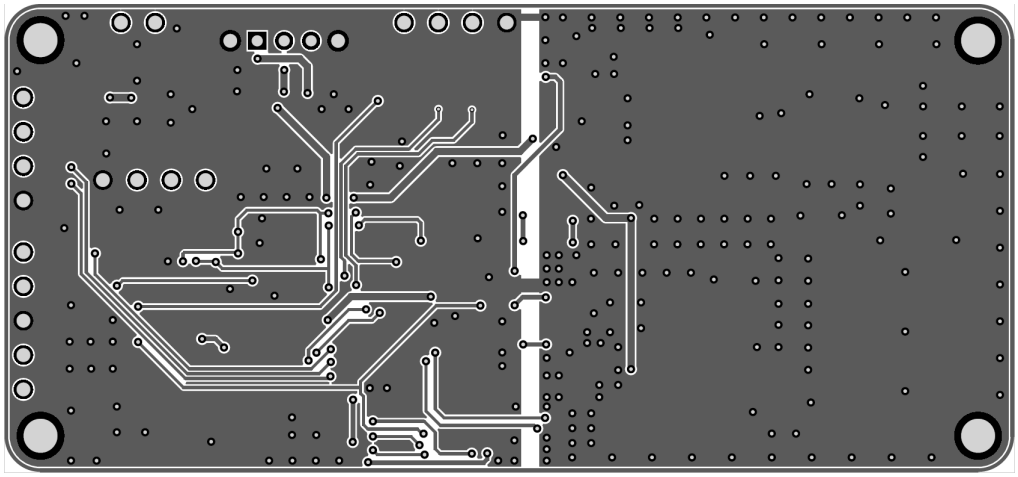


Figura 3.22: Pistas de la PCB. Parte inferior.

Además, en las Figuras 3.23 y 3.24 se puede visualizar una vista en 3D de la PCB terminada con todos los componentes, tanto de la parte superior como de la parte inferior de la misma.

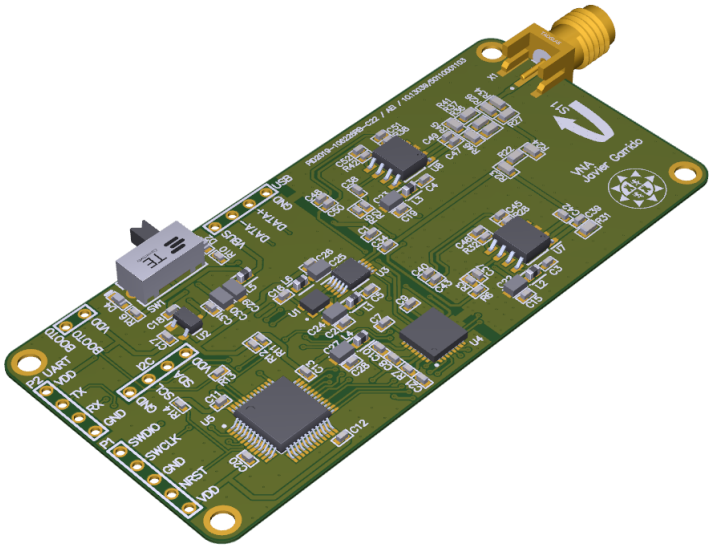


Figura 3.23: Vista 3D superior de la PCB.

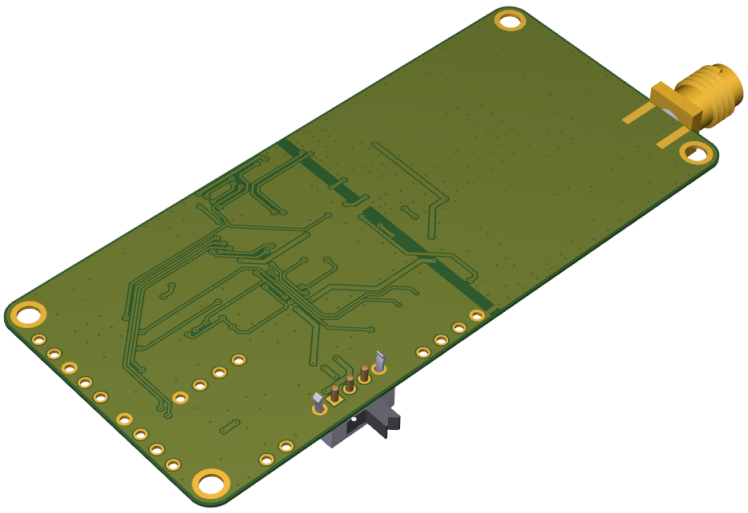


Figura 3.24: Vista 3D inferior de la PCB.

3.2.4. Archivos Gerber

Tras el diseño completo de la PCB, se procede a la generación de los archivos Gerber desde el programa de diseño electrónico Altium Designer, incluyendo la información de los taladros.

Gerber es un formato de archivo que contiene la información necesaria para la fabricación de la PCB, y se utilizará para realizar el pedido de la misma. El formato de estos archivos está diseñado para el control de las impresoras de transparencias (fotolito), que son máquinas de Control Numérico Computarizado (CNC) que se mueven en un plano y realizan la fabricación de la PCB. El estándar utilizado ha sido Gerber X2. En la Figura 3.25, se puede observar la generación de estos archivos desde el programa.

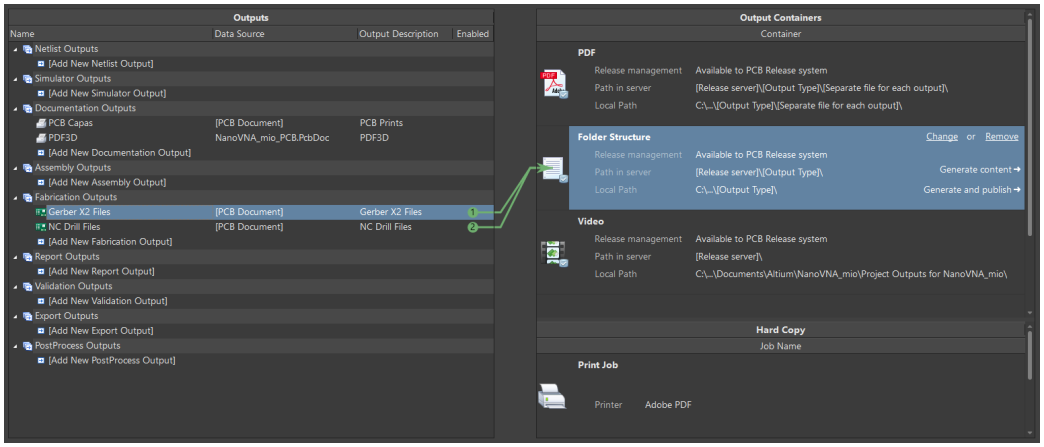


Figura 3.25: Generación de los archivos Gerber.

3.3. Adquisición de los componentes electrónicos

Tras el diseño del esquemático y la PCB, donde se concretan los componentes necesarios para el funcionamiento y montaje del VNA, se procede a la adquisición de dichos componentes electrónicos. Los proveedores elegidos para adquirir los componentes han sido TME, MOUSER y Aliexpress. En la siguiente tabla, se muestra un resumen de los componentes pedidos para construir el VNA, junto con el proveedor:

Tabla 3.2: Listado de componentes electrónicos.

Componente	**Valor**	**Cantidad**	**Fabricante**	**Proveedor**
Resistencia 0402	300 Ω	4	ROYAL OHM	TME
Resistencia 0402	1 kΩ	2	VISHAY	TME
Resistencia 0402	3,9 kΩ	2	ROYAL OHM	TME
Resistencia 0402	5,1 kΩ	2	YAGEO	TME
Resistencia 0402	10 kΩ	3	YAGEO	TME
Resistencia 0402	15 kΩ	4	ROYAL OHM	TME
Resistencia 0603	27 Ω	1	ROYAL OHM	TME
Resistencia 0603	49,9 Ω	5	ROYAL OHM	TME
Resistencia 0603	56 Ω	1	VISHAY	TME
Resistencia 0603	82 Ω	1	VISHAY	TME
Resistencia 0603	150 Ω	1	YAGEO	TME
Resistencia 0603	390 Ω	2	ROYAL OHM	TME
Resistencia 0603	470 Ω	1	YAGEO	TME
Condensador 0402	1 nF	4	AVX	TME
Condensador 0402	100 nF	27	KEMET	TME
Condensador 0402	470 nF	4	WALSIN	TME
Condensador 0402	1 μF	2	AVX	TME
Condensador Tántalo 0805	10 μF	5	AVX	MOUSER
Condensador Tántalo 0805	22 μF	2	AVX	TME
Bobina 0603	4,7 μH	6	KEMET	MOUSER

Diodo LED 0603	Blanco	2	LUCKYLIGHT	TME
Oscilador	26 MHz	1	NDK America	Aliexpress
Regulador de tensión	XC6206P 331MR-G	1	Torex Semiconductor	MOUSER
Generador de señales	Si5351A-B-GT	1	Silicon Labs	Aliexpress
Códec de audio	TLV320 AIC3204 IRHBR	1	Texas Instruments	Aliexpress
Mezclador de frecuencias	SA612AD	2	NXP Semiconductors	MOUSER
Microcontrolador	STM32F072 C8T6	1	STMicroelectronics	Aliexpress
Conector SMA	Hembra	1	Amphenol ICC	TME
Interruptor deslizante	-	1	C&K Switches	Aliexpress
Micro USB C	-	1	Molex	MOUSER

3.3.1. Precio del VNA

A continuación, se recoge de forma resumida el precio de todos los componentes que conforman la última versión del VNA diseñado en la Tabla 3.3, de forma que se puede observar un precio total para el VNA inferior a 17 €.

Tabla 3.3: Precio de los componentes electrónicos.

Componente	**Precio**
STM32F072C8T6	1,55 €
TLV320AIC3204IRHBR	2,20 €
Si5351A-B-GT	0,79 €
SA612AD	1,64 € × 2
XC6206P331MR-G	0,19 €
Oscilador 26 MHz	1,99 €
Conector SMA	2,36 €
Interruptor deslizante	0,20 €
Pasivos	4,36 €
Total	**16,92 €**

3.4. Montaje de la PCB

Utilizando los archivos Gerber generados previamente con Altium Designer, se ha realizado el pedido de la PCB a la empresa de prototipado de PCB JLCPCB. Con los archivos Gerber ya se indican las características de la placa, señalando que posee dos capas de cobre (superior e inferior) y unas dimensiones de 75 × 35 mm. Además, se realiza el pedido de una stencil, que consiste en una plantilla de acero inoxidable con agujeros que se coloca sobre la PCB para aplicar la pasta de soldadura de forma homogénea en los lugares requeridos. Teniendo en cuenta el tamaño de los pads de los componentes, se ha escogido un grosor de stencil de 0,1 mm, para lograr un mejor reparto de la pasta de soldadura. Una vez realizado el pedido, se obtiene la PCB y la stencil, que se pueden ver en la siguiente figura:

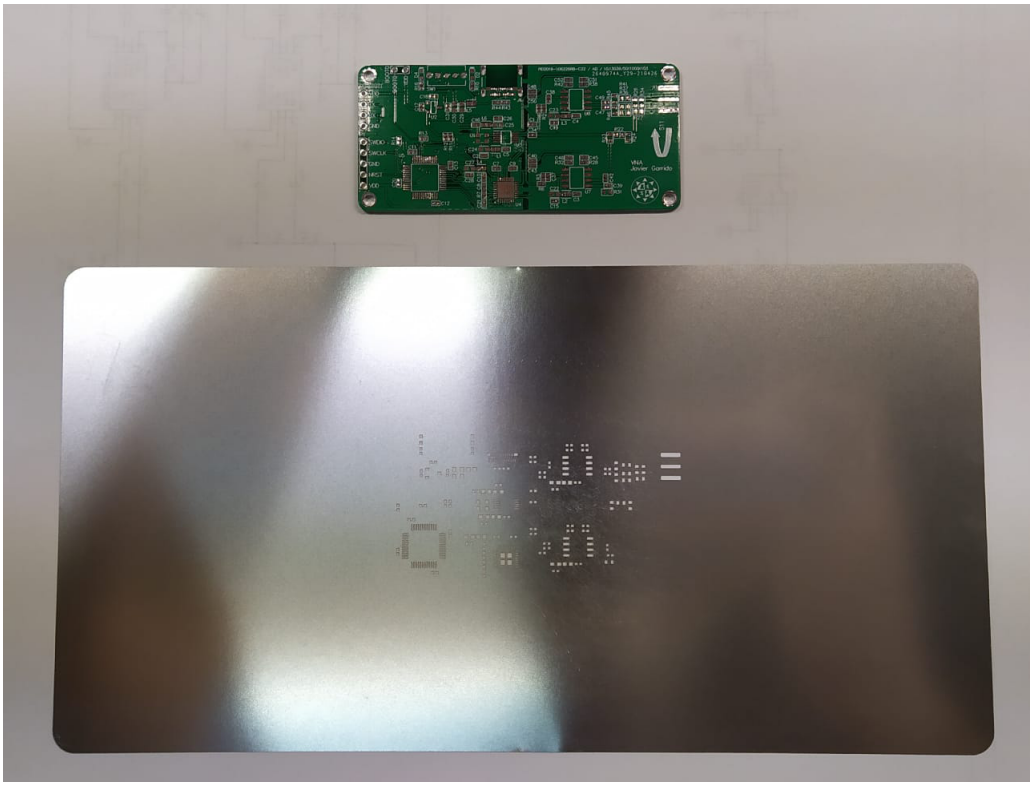


Figura 3.26: PCB y stencil.

En primer lugar, se coloca la stencil sobre la PCB haciendo coincidir los agujeros de la stencil con los pads de la PCB. Tras ello, se aplica la pasta de soldadura con una espátula cubriendo toda la zona de agujeros de la stencil, de forma que la pasta quedará aplicada de manera homogénea sobre los pads de la PCB. Después, se colocan los componentes sobre la PCB de forma minuciosa, siguiendo un cierto orden para facilitar el trabajo. Se comienza con los componentes más pequeños, como son las resistencias, condensadores, bobinas, diodos LED y el oscilador, y se termina colocando los de mayor tamaño, que son el regulador de tensión, el generador de señales, el códec de audio, el mezclador de frecuencias, el microcontrolador, el conector SMA, el interruptor deslizante y el USB.

Después de colocar todos los componentes electrónicos, se introduce la PCB en un horno especial para soldadura SMD (*Surface-Mount Device*, dispositivo de montaje superficial), que calienta la pasta de soldadura de forma progresiva según un perfil de temperatura predeterminado. Cuando termina el proceso, todos los componentes quedan soldados en su correspondiente lugar. Tras el proceso del horno, la placa alcanza una temperatura elevada, por lo que hay que esperar un tiempo hasta que la placa alcance la temperatura ambiente para poder manipularla. Finalmente, se colocan y se sueldan los pines de agujero pasante de los puertos externos de forma manual con una estación de soldadura. En las siguientes figuras se pueden ver las dos versiones de la PCB completamente soldadas.

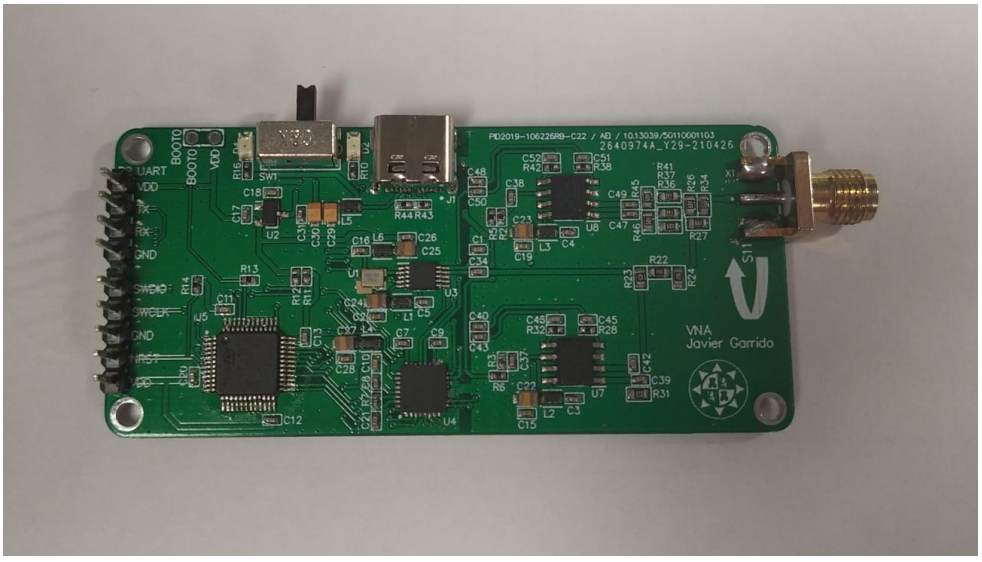


Figura 3.27: PCB soldada. Primera versión.

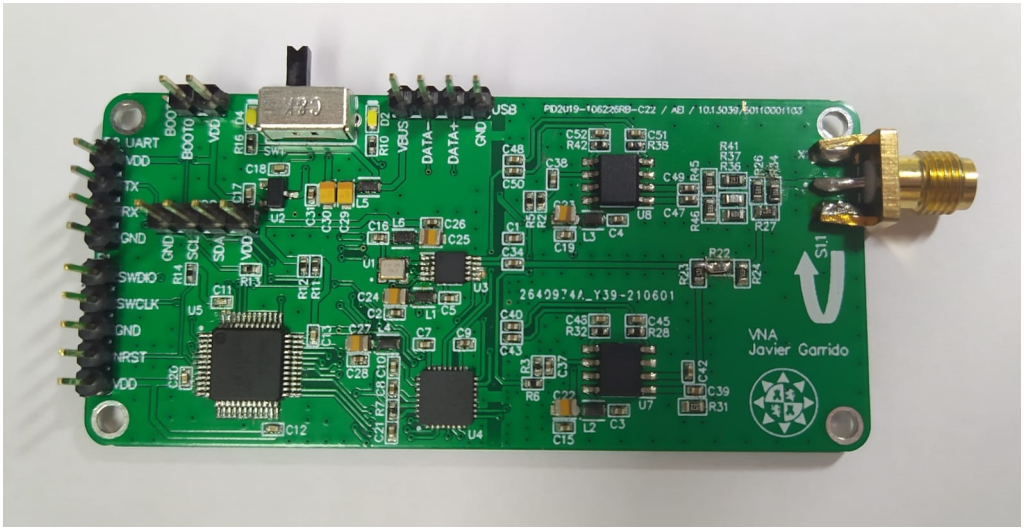


Figura 3.28: PCB soldada. Última versión.

Una vez realizado el montaje, se comprueba detalladamente, de forma visual, si hay uniones de pads tras la soldadura en el horno por un exceso de pasta de soldadura, por ejemplo, en el códec o en el microcontrolador. Esto se soluciona retirando parte del estaño manualmente con un soldador con una malla de cobre para desoldar, y volviendo a aplicar estaño si es necesario.

Luego, se comprueba la conductividad entre las pistas, para comprobar el correcto conexionado entre los distintos componentes y para evitar cortocircuitos que puedan dañar componentes, como podría ser el cortocircuito entre V_{DD} y GND. Tras determinar que no hay fallos aparentes, se procede a alimentar la placa y verificar que funciona correctamente. Las dimensiones del VNA diseñado, que se pueden ver en el Anexo IV.4 son muy reducidas, lo cual era uno de los objetivos del proyecto.

3.5. Programación del microcontrolador

Para realizar la programación del Firmware del microcontrolador STM32F072 se ha utilizado el sistema operativo en tiempo real ChibiOS, que ofrece una solución completa para dispositivos embebidos. Posee un kernel ChibiOS/RT que está diseñado para aplicaciones integradas en microcontroladores de 8, 16 y 32 bits, el cual es utilizado para programar microcontroladores STM32 con núcleo ARM de 32 bits Cortex-M. Además, incluye ChibiOS/HAL, una capa de abstracción de hardware (HAL, *Hardware Abstraction Layer*) compatible con ChibiOS/RT, que ofrece controladores de dispositivos hardware (drivers) como I^2C, I^2S, UART o USB y funciona como una interfaz entre el hardware y el software del sistema. La programación se ha realizado con el lenguaje de programación C en el entorno *Eclipse IDE for C/C++ Developers*, realizando toda la programación en el sistema operativo Ubuntu, una distribución de Linux.

Se puede ver el código implementado *main.c* en el Anexo V.1. Además, en la Figura 3.29, se muestra un flujograma que resume el funcionamiento del programa del microcontrolador, el cual gestiona todo el sistema del VNA. Se puede visualizar completamente este flujograma en el Anexo II.3. Para la implementación del código, se ha tomado como referencia el código diseñado para el NanoVNA de Tomohiro Takahashi [42].

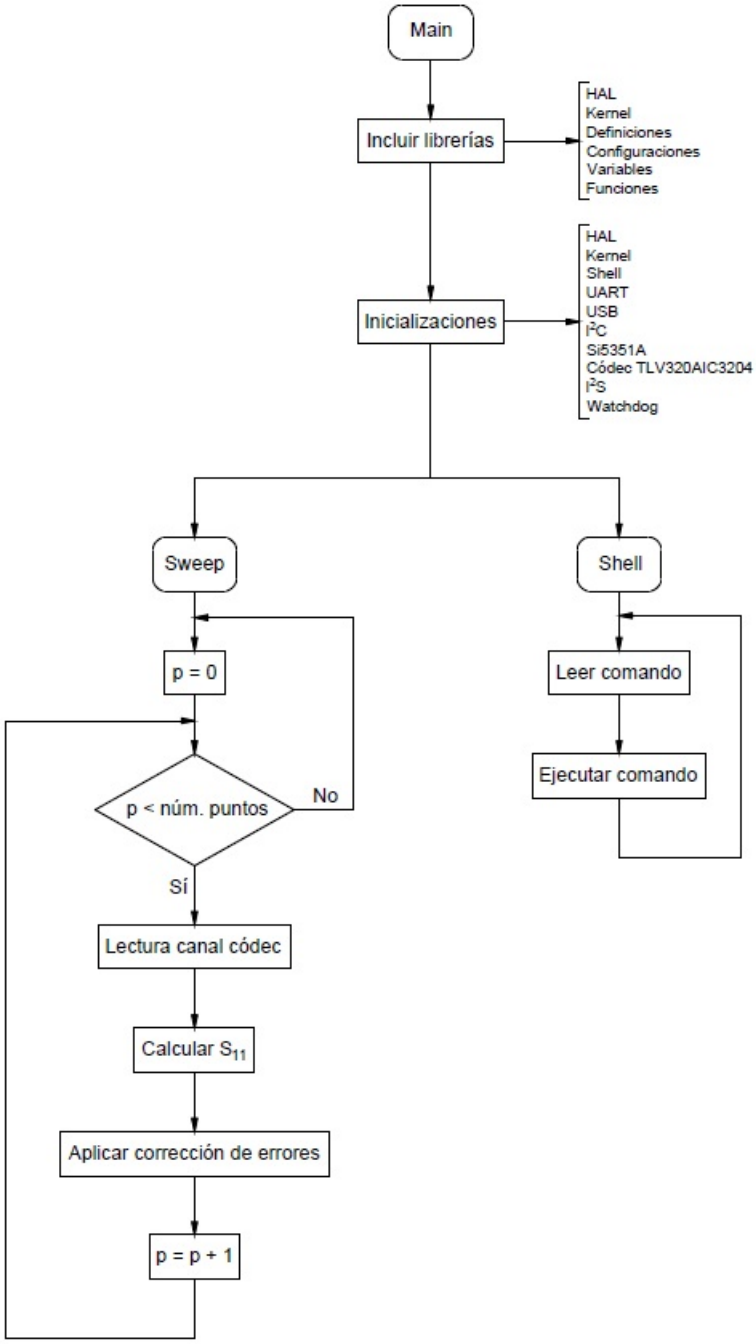


Figura 3.29: Flujograma del código del microcontrolador.

3.5.1. Código fuente

En primer lugar, se incluyen las librerías necesarias que poseen las definiciones, configuraciones, variables y funciones que se utilizarán en el desarrollo del programa, incluyendo las del sistema ChibiOS, así como librerías que permiten controlar el generador de señales Si5351A y el códec TLV320AIC3204.

El funcionamiento del programa consiste en, básicamente, el funcionamiento de dos hilos de forma concurrente. Uno de los hilos realizará constantemente el barrido en frecuencia que permitirá obtener el valor del coeficiente de reflexión medido y el otro hilo se ejecuta directamente en el *main*, realizando la lectura y ejecución de comandos a través de la Shell del sistema. Además de esto, el programa posee una serie de funciones y variables que se utilizarán para establecer y leer valores en el códec y el generador de señales, actualizar y calcular variables, definir configuraciones, realizar la medida de reflexión, realizar la corrección de error de las variables medidas por el VNA y una serie de funciones que hacen funcionar la Shell del sistema, la cual permite cambiar configuraciones o pedir datos para visualizarlos. También, se inicia la comunicación con la UART y el USB.

En la función principal *main*, se inicializan las librerías del HAL y el kernel, se carga la configuración por defecto, se inicializa la conexión con la Shell, se inicia la comunicación I^2C, se inicia la comunicación con el generador de señales y el códec, se inicia la comunicación I^2S y se inicia el Watchdog, con sus respectivas configuraciones. Tras ello, el hilo que se encarga de hacer el barrido de frecuencias comienza su ejecución y, concurrentemente, en el propio *main*, se realiza la lectura y ejecución de comandos por la Shell. El hilo que realiza el barrido de frecuencias calcula, para un total de 101 puntos, el coeficiente de reflexión para cada frecuencia a partir de los valores leídos del códec y, tras ello, aplica la corrección de error a cada punto medido. Al mismo tiempo, hace parpadear un LED para comprobar que la lectura se está realizando. En la Tabla 3.4, se recogen los distintos comandos que interpreta la Shell del VNA y su función.

Tabla 3.4: Listado de comandos de la Shell del VNA.

Comando	**Función**
`scan`	Hace un barrido entre dos frecuencias y envía los datos formateados
`scan_bin`	Hace un barrido entre dos frecuencias y envía los datos en forma de bytes
`data`	Devuelve los datos del último barrido
`frequencies`	Muestra las frecuencias actuales de los puntos del barrido
`freq`	Establece la frecuencia del Si5351A-B-GT
`sweep`	Configura las frecuencias de barrido del VNA
`power`	Establece la potencia del Si5351A-B-GT
`bandwidth`	Establece el ancho de banda
`saveconfig`	Guarda la configuración actual en la memoria flash
`clearconfig`	Borra la configuración almacenada en la memoria flash
`pause`	Detiene el barrido
`resume`	Reanuda el barrido
`cal`	Realiza la calibración del VNA indicando el estándar conectado
`save`	Guarda la calibración en la memoria flash
`recall`	Carga la configuración indicada en el comando
`edelay`	Establece un delay para corregir el desfase de la señal
`reset`	Reinicia el VNA
`usart_cfg`	Configura la comunicación UART
`usart`	Escribe el buffer almacenado en la UART
`usart_data`	Habilita el envío de los datos medidos a través de la UART
`transform`	Establece el dominio de los datos
`threshold`	Establece el umbral de frecuencia para el Si5351A-B-GT
`help`	Muestra todos los comandos de la Shell

3.5.2. Compilación y programación

Una vez realizada la programación, se utilizará el comando `make` desde una terminal de Ubuntu para generar el programa que se subirá al microcontrolador STM32F072. El comando `make` es una herramienta de gestión de las dependencias que existen entre los archivos que componen el código fuente de un programa que determina qué partes deben ser recompiladas para generar el programa. Para ello, lee las instrucciones para generar el programa del fichero *Makefile*. Por tanto, en primer lugar, se genera el archivo *Makefile* desde Eclipse, indicando las dependencias entre los archivos del programa y las especificaciones del microcontrolador. Se trata de un fichero de texto que utiliza `make` para llevar a cabo la gestión de la compilación de los programas, informando sobre las dependencias entre las diferentes partes del proyecto.

Para realizar la compilación y generación del programa, en primer lugar, se descargan las herramientas de compilación de ARM y se añaden esas herramientas al PATH del sistema, una variable de entorno del sistema operativo en la que se especifican las rutas donde el intérprete de comandos busca los programas a ejecutar. Después, se instala la herramienta DFU (*Device Firmware Update*, actualización del Firmware del dispositivo), que permitirá actualizar el Firmware del microcontrolador a través del USB. Todo esto se realiza escribiendo los siguientes comandos en la terminal de Ubuntu:

```
$ wget https://developer.arm.com/-/media/Files/downloads/
gnu-rm/8-2018q4/gcc-arm-none-eabi-8-2018-q4-major-
linux.tar.bz2

$ sudo tar xfj -C /usr/local gcc-arm-none-eabi-8-2018-q4-
major-linux.tar.bz2

$ PATH=/usr/local/gcc-arm-none-eabi-8-2018-q4-major/
bin:$PATH

$ sudo apt install -y dfu-util
```

Luego, desde la terminal de Ubuntu, estando en la carpeta que contiene el *Makefile* y el resto de ficheros del programa, se realiza la compilación del Firmware ejecutando el comando:

```
$ make
```

Finalmente, se hace que el microcontrolador entre en modo DFU conectando el puente que une BOOT0 con V_{DD}, se conecta el microcontrolador al ordenador con Ubuntu a través del USB y se ejecuta el siguiente comando en la terminal para actualizar el Firmware a través del USB:

```
$ dfu-util -d 0483:df11 -a 0 -s 0x08000000 -D build/ch.bin
```

Igualmente, se puede acceder a la orden `flash` que existe en el *Makefile* y que llama al mismo comando, escribiendo:

```
$ make flash
```

Con estos pasos, el microcontrolador del VNA contiene el Firmware programado y ya está listo para realizar las mediciones necesarias.

3.6. Datalogger con ESP32 en MicroPython

Para realizar la comunicación y recogida de datos con el VNA diseñado, se ha implementado un Datalogger en MicroPython con una ESP32 usando el IDE para MicroPython *Thonny*, cuyo código se puede ver en el Anexo V.2. La ESP32 se comunica con el VNA por UART enviándole comandos que la Shell del sistema del VNA interpretará y ejecutará, realizando las acciones pertinentes.

Los comandos enviados por UART por la ESP32 sirven para calibrar el VNA con el estándar OSL (*Open-Short-Load*) y para recoger los datos del coeficiente de reflexión de un barrido de frecuencias. Además, el código almacena los datos en ficheros de texto para luego procesarlos más fácilmente. En la Figura 3.30 se puede ver la ESP32 utilizada para realizar las pruebas de esta tarea.

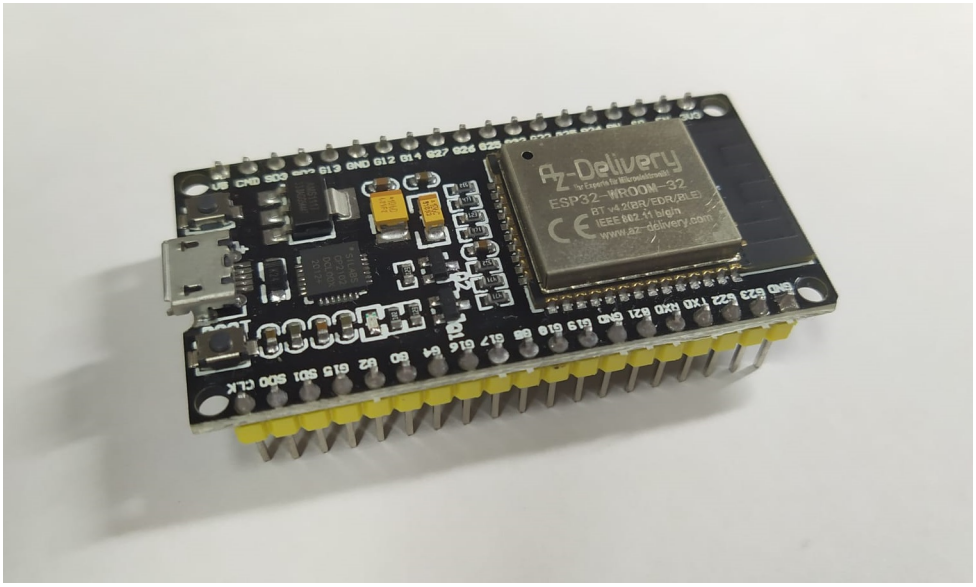


Figura 3.30: ESP32 utilizada para realizar las pruebas del Datalogger.

3.6.1. Protocolo SDI-12

Además de las funcionalidades indicadas anteriormente, se ha implementado en la ESP32 un protocolo de comunicación SDI-12 en MicroPython para el envío de los datos medidos a un datalogger que utilice el mismo protocolo. Es decir, la ESP32 funciona como sensor SDI-12, y envía la información de las mediciones al maestro SDI-12.

El protocolo SDI-12 (*Serial Digital Interface at 1200 baud*, interfaz digital serie a 1200 baudios) es un protocolo de comunicación serie asíncrono entre un maestro, datalogger o *data recorder* (registrador de datos) y los esclavos o sensores basados en microprocesadores. El protocolo sigue una configuración maestro-esclavo donde un datalogger solicita datos de los sensores, cada uno identificado con una dirección única. Permite la conexión de hasta 62 sensores.

La comunicación se realiza en una sola línea y es half-duplex. Solo el sensor con la dirección escogida responderá al maestro, mientras que el resto de sensores permanecerán en modo de bajo consumo hasta que se llamen. Este protocolo se emplea especialmente cuando se requiere un bajo consumo, un coste reducido y el uso de un simple datalogger con múltiples sensores empleando un solo cable. El protocolo SDI-12 emplea tres líneas de señal: línea de datos de 0 V - 5 V, línea de alimentación de 12 V y línea de tierra. Es decir, solo se necesita una línea para transmitir datos por SDI-12. El esquema de conexión del datalogger con los distintos sensores sigue la estructura de la Figura 3.31.

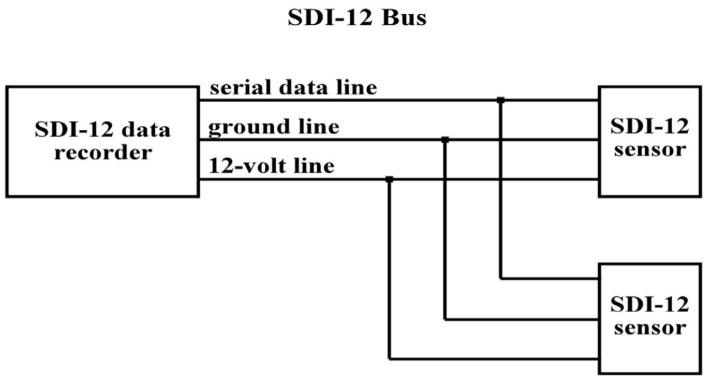


Figura 3.31: Esquema de conexión SDI-12 [36].

Para el funcionamiento del protocolo, el sensor debe tener una dirección, que se suele designar por "a". Después, existen una serie de comandos que el datalogger le envía al sensor, cuya estructura consiste en una cadena de caracteres que comienza con la dirección del sensor y termina con el carácter '!'. Tras el envío del comando, el sensor responde con otra cadena que comienza también con su dirección y que concluye con los caracteres <CR><LF> seguidos, que se corresponden con un retorno de carro y un salto de línea, respectivamente. Algunos de los comandos más relevantes y sus respuestas se muestran en la Tabla 3.5.

Tabla 3.5: Comandos relevantes en el protocolo SDI-12.

Definición	**Comando**	**Respuesta**
Consulta de dirección	?!	a<CR><LF>
Confirmación activo	a!	a<CR><LF>
Enviar identificación	aI!	allccccccccmmmmmmmvvvx...x <CR><LF>
Cambiar dirección	aAb!	b<CR><LF>
Comenzar medida	aM!	atttn<CR><LF>
Enviar datos	aD0!	a<valores><CR><LF>

Normalmente, el procedimiento seguido para realizar la comunicación SDI-12 entre los sensores y el datalogger para obtener una medida consiste en que el datalogger despierta a todos los sensores del bus con un *break*, envía un comando a un sensor específico para que tome una medida (aM!), el sensor responde devolviendo el tiempo máximo en el que las medidas estarán listas y el número de valores a devolver, el datalogger le pide las medidas al sensor (aD0!) y el sensor devuelve las medidas realizadas, tras lo cual se pone en modo de bajo consumo.

Para conseguir implementar este protocolo en la ESP32, se ha programado una UART a 1200 baudios que permite la comunicación en ambos sentidos. Para conseguir una sola línea de datos de 0 V - 5 V a partir de la UART, se ha diseñado el circuito de la Figura 3.32, empleando como referencia el artículo descrito en [37].

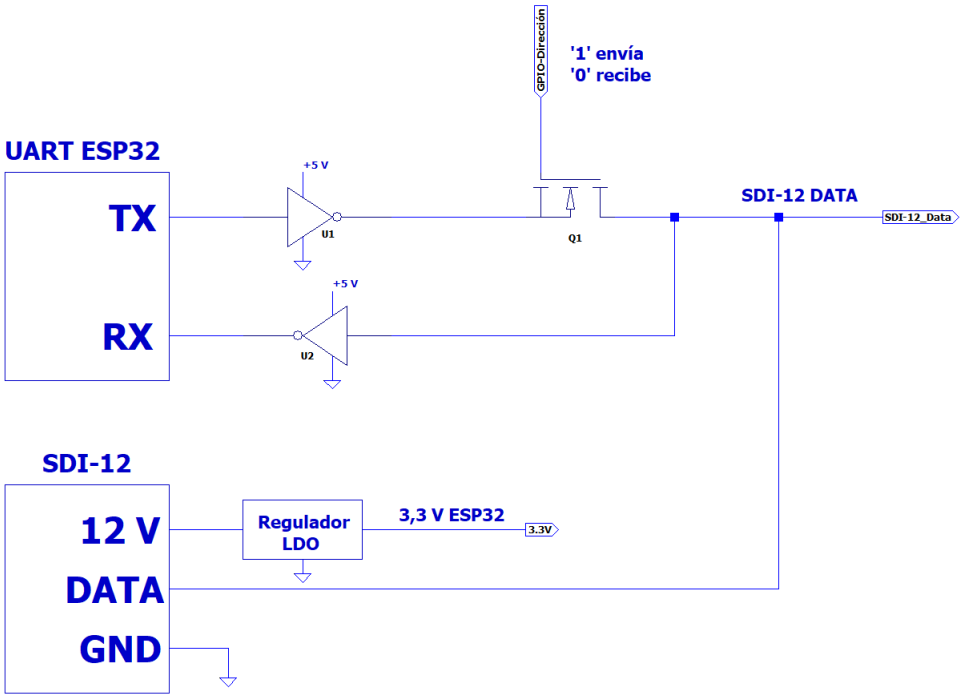


Figura 3.32: Circuito de adaptación UART/SDI-12.

En este circuito, el MOSFET está controlado por una GPIO (*General Purpose Input/Output*, entrada/salida de propósito general) de la ESP32, indicando la dirección de los datos. Cuando la UART está en modo de transmisión, envía la señal desde el pin TX a través de un inversor alimentado a 5 V. Esto invierte la señal y la convierte a niveles de 0 V - 5 V. A continuación, el GPIO activa la puerta del MOSFET para permitir que la transmisión invertida pase a través de la línea de datos SDI-12. Una vez que la transmisión se ha completado, el GPIO desactiva la puerta del MOSFET, permitiendo que lleguen señales de la línea de datos SDI-12 a un segundo inversor donde se invierten y se envían al pin RX. Además, para completar el estándar de los pines SDI-12, la ESP32 se alimenta a través de los 12 V del maestro después de pasar por un regulador de tensión, que los transforma a 3,3 V. El componente utilizado para los inversores es el inversor Schmitt doble de Texas Instruments SN74LVC2G14, para el MOSFET es el FDV303N de Fairchild y para el regulador LDO (*Low-dropout*, baja caída) se ha empleado el REG102 a 3,3 V de Texas Instruments.

El código implementado en MicroPython para realizar esta función forma parte del código previamente introducido, que se puede ver en el Anexo V.2. La ESP32 es capaz de responder a los comandos básicos de ?!, a!, aI!, aAb!, aM!, aD0! y aD1!. Por defecto, la ESP32 toma una dirección de '1', y tras los comandos aD0! y aD1! envía los datos correspondientes a las medidas del coeficiente de reflexión y al cálculo de la permitividad compleja, respectivamente. La parte del cálculo de la permitividad se detalla en el siguiente capítulo.

3.6.2. Diseño de la PCB del Datalogger

Para integrar todas las partes del Datalogger, se ha diseñado una PCB que incluye todos los elementos necesarios para su funcionamiento. Para ello, primero, se ha diseñado el esquemático y, en segundo lugar, la PCB.

3.6.2.1. Esquemático del Datalogger

Mediante el uso de Altium, en primer lugar, se ha diseñado el esquemático, el cual se puede ver en la Figura 3.33, y además se adjunta en el Anexo III.4 para su completa visualización:

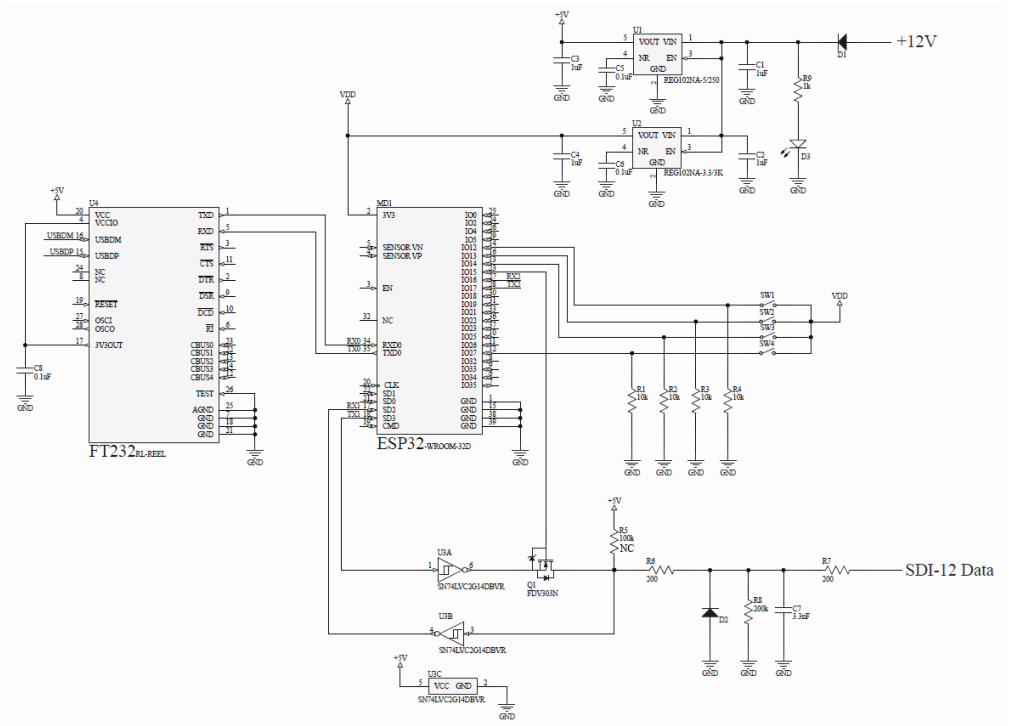


Figura 3.33: Esquemático del Datalogger basado en una ESP32.

En primer lugar, se ha incluido una parte de alimentación que parte de los +12 V del maestro SDI-12 y, tras pasar por dos reguladores de tensión U1 y U2, se obtienen tensiones de +5 V y +3,3 V (VDD), respectivamente. La tensión de +5 V servirá para alimentar los inversores del circuito, y la tensión

VDD permitirá alimentar la ESP32. Además hay un diodo LED que indica que el Datalogger está conectado al maestro. Para realizar la comunicación y programación de la ESP32, se ha añadido una pasarela de USB a UART, el FT232RL, el cual se alimenta también a partir de los +5 V. Los datos parten del USB a través de los pines USBDP y USBDM y, a la salida del FT232RL, las señales de RXD y TXD se conectan con la UART0 de la ESP32, formada por los pines RX0 y TX0.

Tras estas etapas, se encuentra la ESP32, que gestiona todo el sistema del Datalogger. Primero, dispone de cuatro interruptores SW1, SW2, SW3 y SW4 que permiten al Datalogger realizar las acciones de efectuar las mediciones desde el VNA y guardar un barrido de frecuencia, hacer la calibración *Load*, hacer la calibración *Open* y hacer la calibración *Short*, respectivamente. Finalmente, se encuentra la etapa de adaptación UART/SDI-12 que se ha descrito en el apartado anterior, alimentada a +5 V y formada por dos inversores con histéresis y un MOSFET controlado por la ESP32 que gestiona la dirección del flujo de datos. Tras esto, se ha añadido una red de protección de la línea de datos SDI-12.

3.6.2.2. PCB del Datalogger

Una vez que se ha diseñado el esquemático del Datalogger, se procede al diseño de su PCB siguiendo la estrategia indicada en el apartado del diseño de la PCB del VNA 3.2. En primer lugar, se seleccionan y se colocan los componentes convenientemente en la PCB, de forma que se minimicen las longitudes de las pistas en la medida de lo posible, e intentando ocupar el menor espacio para el conjunto de la PCB. La posición de la ESP32 se encuentra en la zona central superior, para que sea accesible por todos los componentes, el FT232RL y la conexión USB se encuentran en el borde derecho, la etapa de alimentación y las conexiones con SDI-12 y la UART del VNA se posicionan en el borde izquierdo y, por último, los interruptores y la parte que gestiona la adaptación UART/SDI-12 se encuentran en la parte inferior.

Una vez colocados los componentes, se delimita el tamaño y forma de la PCB. Después, se realiza el trazado de las pistas, que conectará los distintos componentes de la placa entre sí. Antes de realizar el trazado de las pistas, se colocan en las cuatro esquinas los orificios donde se podrá atornillar la placa, y estarán conectados a la red de GND. Además, se colocan los pads

de las señales de los puertos externos indicados anteriormente. El tamaño de las pistas, que se recoge resumidamente en la Tabla 3.6, ha sido de 0,4 mm para la parte de potencia y de 0,25 mm para el resto de pistas.

Tabla 3.6: Tamaño de las pistas de la PCB del Datalogger.

Pista	Preferido	Mínimo	Máximo
Potencia	0,25 mm	0,25 mm	0,25 mm
General	0,4 mm	0,3 mm	0,5 mm

Justo a continuación del trazado de las pistas, se ha realizado un *pour* o relleno de la red GND, tanto en la capa superior como en la capa inferior. Además, para aumentar la disipación se ha realizado una serie de vías a lo largo de la PCB que conectan ambas capas en la red de GND. Por último, se ha colocado el nombre de los componentes y puertos, y también se ha añadido el logo de la UPCT, el nombre del autor y el nombre del dispositivo. Se puede ver en la Figura 3.34 la PCB terminada en Altium.

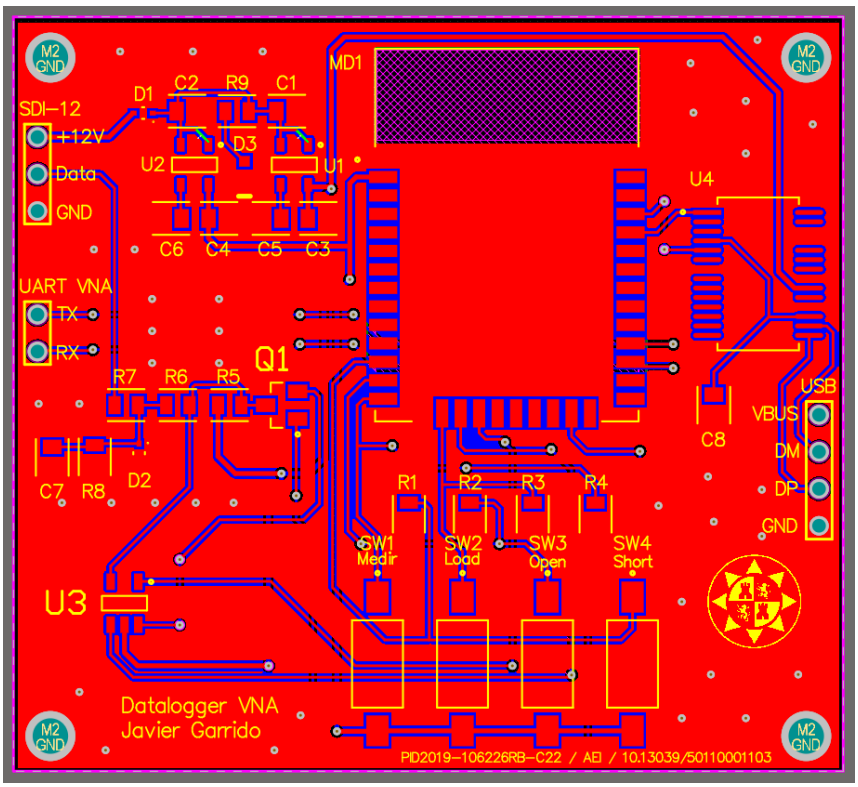


Figura 3.34: PCB del Datalogger terminada.

En las Figuras 3.35 y 3.36, se pueden visualizar las pistas de las capas superior e inferior de la PCB, y también se adjuntan en el Anexo IV.3.

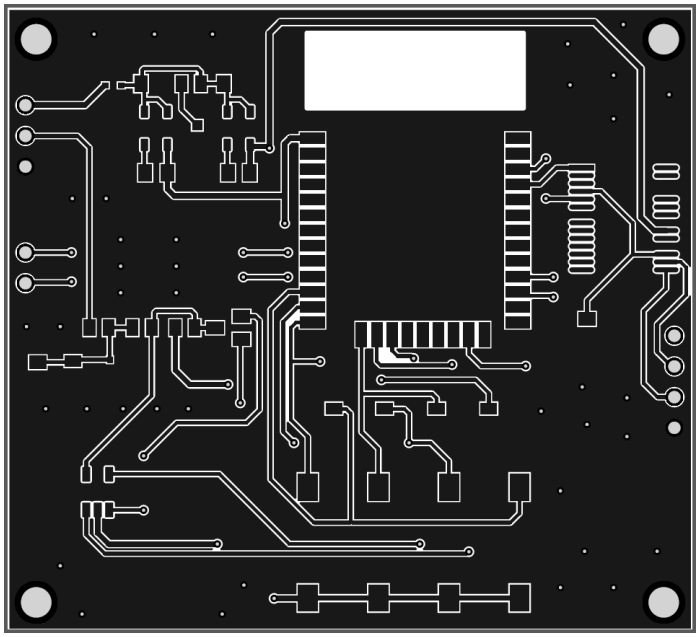


Figura 3.35: Pistas de la PCB del Datalogger. Parte superior.

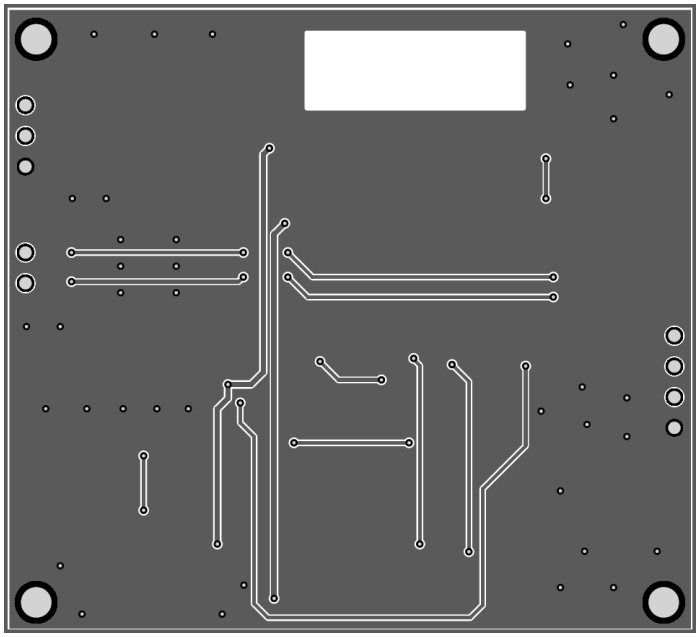


Figura 3.36: Pistas de la PCB del Datalogger. Parte inferior.

Añadido a esto, en las Figuras 3.37 y 3.38 se puede contemplar una vista en 3D de la parte superior e inferior de la PCB terminada con todos los componentes.

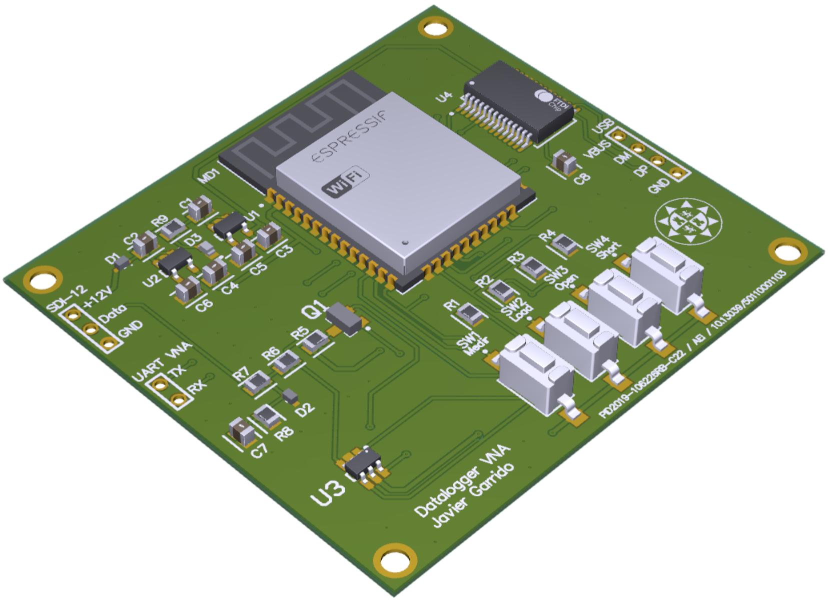


Figura 3.37: Vista 3D superior de la PCB del Datalogger.

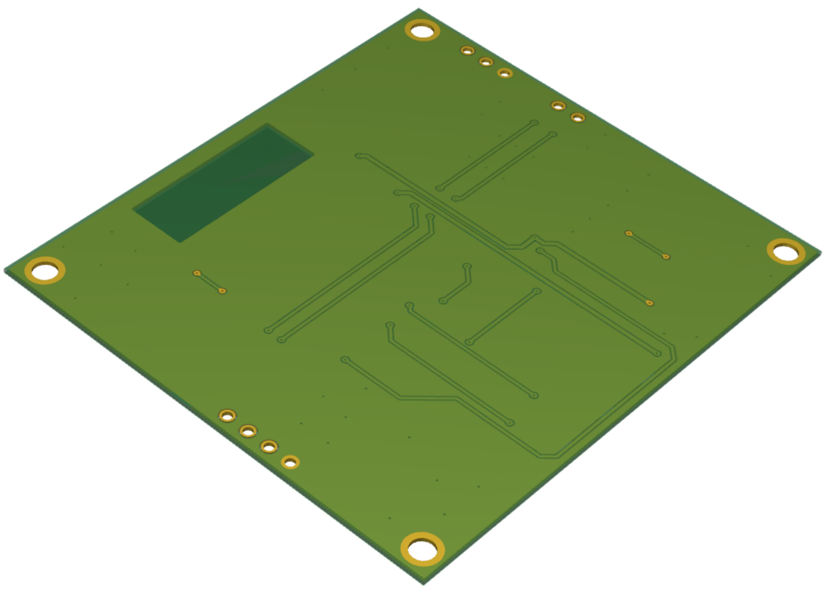


Figura 3.38: Vista 3D inferior de la PCB del Datalogger.

Finalmente, por falta de tiempo a la hora de hacer el pedido de la PCB y los componentes no se ha podido realizar la fabricación de la PCB de forma física. No obstante, sí se ha logrado probar el funcionamiento del diseño realizando el mismo montaje sobre una placa de prototipado rápido, obteniéndose los resultados esperados, lo cual permitirá la obtención de los datos del VNA diseñado en la fase de experimentación. En la Figura 3.39 se puede observar el correcto funcionamiento del protocolo implementado en el Datalogger, donde se ha hecho una prueba con un maestro SDI-12 a través de un terminal, y el Datalogger ha respondido correctamente.

```
>> ?!
1
>> 1!
1
>> 1I!
114jgarrido000001001VNA
>> 1M!
10011
>> 1D0!
1+1.0-0.0+0.9-0.1+0.9-0.2+0.9-0.3+0.8-0.4
>> 1D1!
1-26.2+22.7+20.5-0.4+20.5-0.8+20.3-1.3+20.2-1.8
>> 1A5!
5
>> 5!
5
```

Figura 3.39: Funcionamiento del SDI-12 implementado en el Datalogger.

Capítulo 4

Estudio experimental

En la fase experimental del proyecto, se ha realizado la medida de la permitividad de distintos líquidos y el aire utilizando el VNA diseñado. Las mediciones se han realizado utilizando el método de reflexión en una sonda coaxial y, para ello, ha sido necesario acoplar una sonda coaxial acabada en abierto al VNA. También será preciso tomar unos medios de referencia para saber lo que se está midiendo, mediante la calibración del VNA con la sonda coaxial tras las mediciones.

4.1. Pruebas preliminares del VNA diseñado

Antes de comenzar con la fase de experimentación, se procede a validar que la PCB del VNA diseñado está correctamente soldada, y también que realiza la calibración y las mediciones del estándar OSL de forma correcta, ofreciendo resultados razonables, y comparándolos con los del VNA de referencia [42]. En primer lugar, aunque ya se realizó tras la finalización del montaje de la PCB, se vuelve a comprobar exhaustivamente la conductividad entre las distintas pistas y los distintos pines de los circuitos integrados, para comprobar que están correctamente conectados y para evitar cortocircuitos no deseados. Tras realizar estas comprobaciones sobre la PCB del VNA diseñado, no se ha encontrado ningún fallo aparente al respecto. Además, de forma visual, se comprueba también que todos los pines están soldados adecuadamente, de forma aparente, aunque no se puede ver la totalidad de los

mismos, por ejemplo, en el códec, ya que dispone sus pines debajo del integrado donde, además, hay una conexión a tierra que puede dar problemas.

Seguidamente, se ha realizado la medición en el osciloscopio de distintos puntos del circuito: la señal generada por el oscilador de 26 MHz, la señal que proporciona el generador de señales controlado por el microcontrolador, donde se pudo comprobar que realizaba el barrido de frecuencias correctamente, las señales entrantes y salientes de los mezcladores de frecuencias y la onda estacionaria que se forma en el puerto de reflexión. Todos estos resultados, visualmente en el osciloscopio, incluyendo sus frecuencias, eran razonables y aparentemente correctos.

Una vez realizadas todas estas comprobaciones sobre la PCB, se procede a realizar las mediciones indicadas con el estándar OSL, y los resultados registrados no se corresponden con medidas razonables para el coeficiente de reflexión, incluso tras la calibración. Por tanto, tras la realización de muchas pruebas, se concluyó que, a pesar de que el diseño del VNA es correcto y de que se realizara el montaje completo de la PCB, no se disponía de los medios y la tecnología suficientes para realizar correctamente las soldaduras y manipulación de los distintos componentes que forman la PCB, ya que tenían un tamaño muy reducido y se puede haber ocasionado la destrucción de algún componente por electricidad estática en el cuerpo o por sobrecalentamiento al intentar corregir las soldaduras, como puede ser el caso del códec, un componente difícil de comprobar y muy sensible. Esta situación era esperable tras el montaje de la PCB con los medios disponibles.

Para completar la parte final del proyecto, que consiste en experimentar con el VNA diseñado para obtener modelos que proporcionan el cálculo de la permitividad a partir del coeficiente de reflexión, se ha procedido a utilizar el VNA de referencia [42] desoldando las partes asociadas con el puerto de transmisión, la pantalla y la batería, de forma que la PCB contiene prácticamente el mismo diseño que se ha realizado en el presente proyecto, y esto permitirá validar el funcionamiento del mismo. De aquí en adelante, se seguirá haciendo referencia al VNA de referencia modificado como VNA diseñado, ya que mantiene el mismo diseño. Además, el código del microcontrolador y la parte del Datalogger con la ESP32 son perfectamente aplicables a esto.

4.2. Experimentación

En primer lugar, se ha realizado una serie de mediciones sobre distintos líquidos con el VNA diseñado para comprobar su correcto funcionamiento, comparando sus resultados con el NanoVNA diseñado por Tomohiro Takahashi [42], tomado como modelo de referencia por su semejanza física y en cuanto al principio de funcionamiento.

El VNA diseñado ofrece como salida el coeficiente de reflexión S_{11} del medio dieléctrico medido en función de la frecuencia. Se ha realizado la medida de cinco líquidos de referencia distintos y del aire con los dos analizadores, que se recogen en las Tablas 4.1 y 4.2, junto con la temperatura a la que se realizó cada medición.

Tabla 4.1: Listado de materiales medidos con el NanoVNA y su temperatura.

Material	**Temperatura**
Aire	$26,4\ ^{\circ}C$
Agua destilada	$26,0\ ^{\circ}C$
Acetona	$25,9\ ^{\circ}C$
Isopropanol	$25,8\ ^{\circ}C$
Metanol	$25,8\ ^{\circ}C$
Etilenglicol	$26,5\ ^{\circ}C$

Tabla 4.2: Listado de materiales medidos con el VNA diseñado y su temperatura.

Material	**Temperatura**
Aire	$26,2\ ^{\circ}C$
Agua destilada	$25,9\ ^{\circ}C$
Acetona	$26,2\ ^{\circ}C$
Isopropanol	$26,2\ ^{\circ}C$
Metanol	$25,9\ ^{\circ}C$
Etilenglicol	$26,5\ ^{\circ}C$

Para cada uno, la medida se ha realizado tres veces para obtener el valor medio reduciendo, así, el posible error en la medida, y para descartar una de las tres medidas en caso de que sea muy distinta de las otras dos, dando por hecho que no es un resultado correcto. Esto último no ha ocurrido en los ensayos. El diagrama de bloques correspondiente al montaje realizado para hacer los ensayos se puede ver en la Figura 4.1 y en el Anexo II.4, donde el VNA, que se programa a través del USB desde el PC, realiza las mediciones sobre el material bajo estudio (MUT, *Material Under Test*) a través de la sonda acabada en abierto, y el Datalogger, que se comunica también a través del USB con el PC, le pide los datos de las mediciones realizadas al VNA por UART, compartiendo dicha información a través del protocolo SDI-12 con un maestro SDI-12 si estuviera conectado.

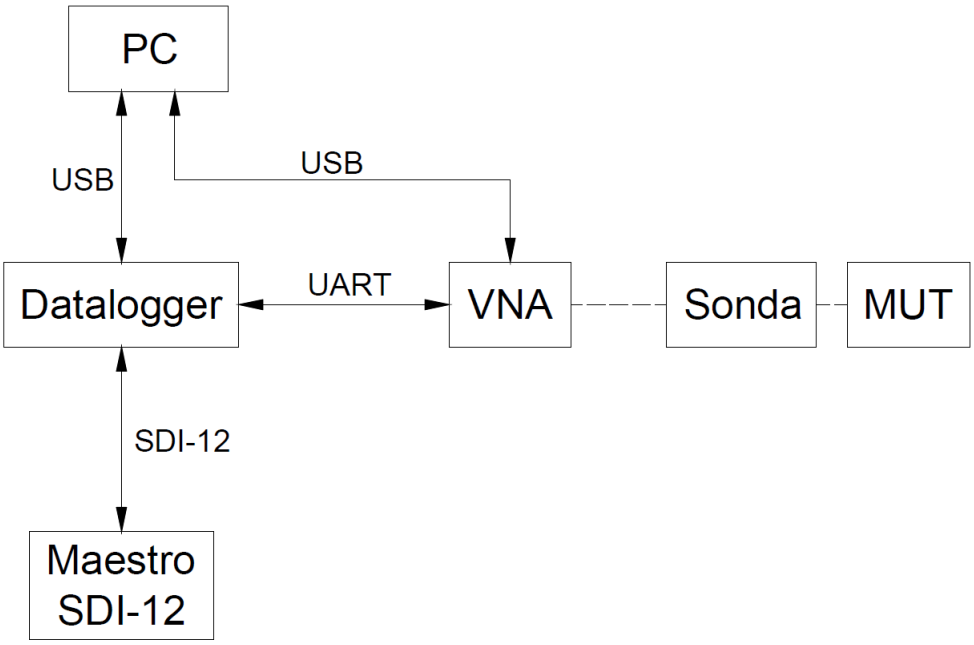


Figura 4.1: Diagrama de bloques para la realización de ensayos.

4.2.1. Materiales

Los materiales utilizados para realizar las mediciones sobre los distintos líquidos han sido los siguientes:

- **Vaso de precipitado**

Se ha utilizado un vaso de precipitado Pyrex de 600 ml de capacidad para contener la muestra de todos los líquidos sobre los que se ha realizado la medida del coeficiente de reflexión. El vaso de precipitado debe quedar limpio entre medición y medición para evitar la contaminación de las muestras de líquidos. Se puede visualizar el vaso de precipitado en la Figura 4.2.

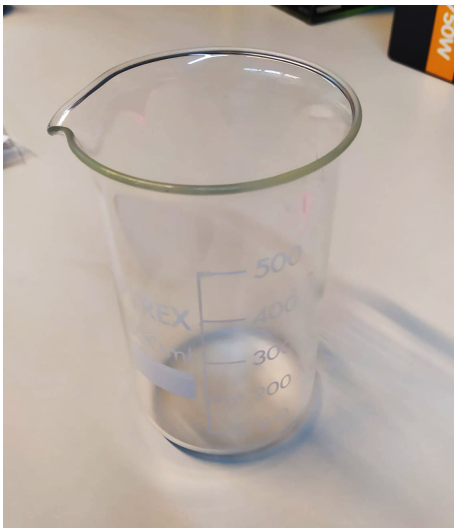


Figura 4.2: Vaso de precipitado de 600 ml.

- **Cable coaxial SMA**

La conexión directa de la sonda coaxial de final abierto con la que se ha realizado la medición con el conector SMA situado en la PCB del VNA dificultaba la realización de las medidas sobre los líquidos contenidos en el vaso de precipitado. Para solventarlo, se conectó un cable coaxial SMA con el objetivo de alargar este conector, para conectar en su extremo la sonda que realizó las mediciones y facilitar su contacto con el líquido medido.

Figura 4.3: Cable coaxial SMA conectado a la sonda coaxial de final abierto.

- **Sonda coaxial de latón y teflón de final abierto**

Para realizar las mediciones del coeficiente de reflexión de todos los materiales, se ha utilizado una sonda coaxial de latón y teflón de final abierto basada en el diseño realizado en el artículo [3]. Las medidas de sus diámetros son, de dentro hacia fuera: 12,5 mm, 40 mm y 45 mm, que quedan recogidas en la Tabla 4.3. Se puede observar la sonda coaxial en la Figura 4.4.

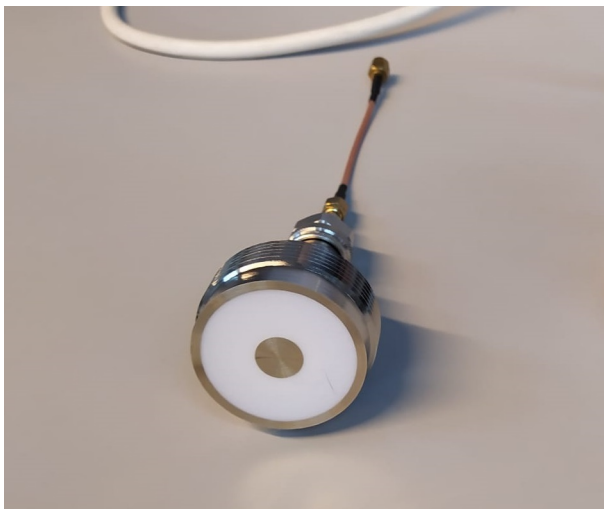


Figura 4.4: Sonda coaxial de latón y teflón de final abierto.

Tabla 4.3: Dimensiones de la sonda coaxial de final abierto.

Diámetro	Medida
Interior	12,5 mm
Intermedio	40 mm
Exterior	45 mm

- **Termopar**

Para saber la temperatura a la que se estaban realizando las mediciones sobre los distintos materiales, se ha utilizado el termopar de Comark C28 tipo K, que tiene un fondo de escala de 800 $°C$ con un rango de -200 $°C$ a 600 $°C$, una precisión del $0{,}1\,\% \pm 0{,}2$ $°C$ y una resolución de 0,1 $°C$ si la temperatura es superior a -100 $°C$, mientras que la resolución baja a 1 $°C$ si la temperatura desciende por debajo de -100 $°C$. Se puede visualizar en la Figura 4.5.

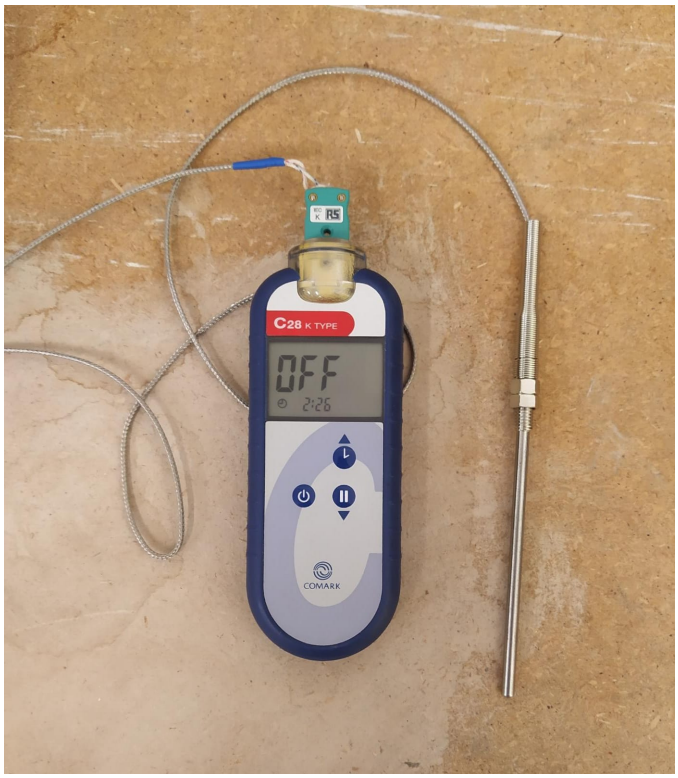


Figura 4.5: Termopar Comark C28 tipo K.

4.2.2. Procedimiento

En primer lugar, se fija el VNA en un sitio firme a una determinada altura para que no se mueva y pueda realizar las mediciones correctamente sobre el vaso de precipitado con la sonda coaxial de final abierto conectada en el extremo del cable coaxial SMA.

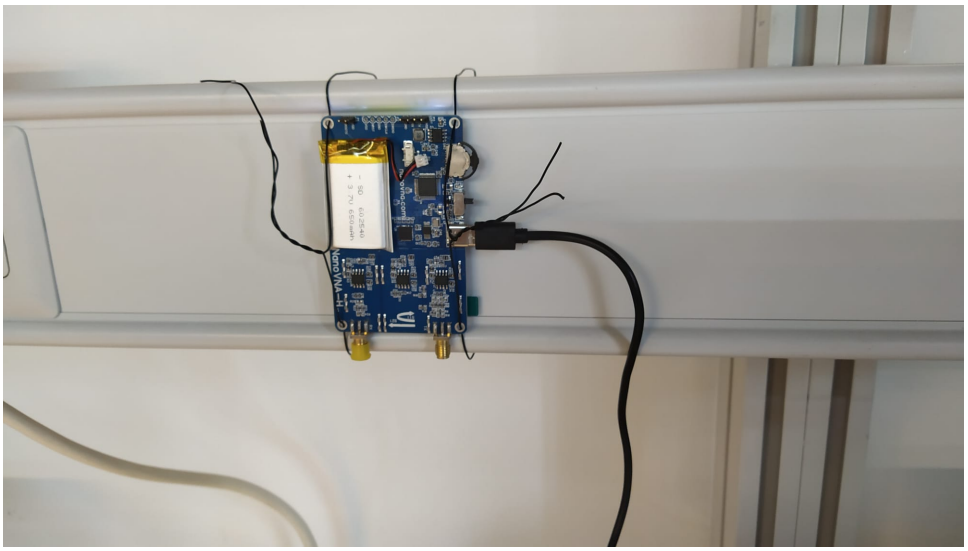


Figura 4.6: VNA firmemente fijado.

Seguidamente, se conecta el cable coaxial SMA para alargar la distancia de medida. Tras esto, se calibra el VNA en la punta del cable, para que realice correctamente las mediciones. Con la calibración hecha, se conecta en el extremo la sonda coaxial con la que se realizarán las mediciones sobre los distintos líquidos, de forma que queda colgando en un plano paralelo a la superficie de la mesa.

Con todo conectado, se realiza una primera medición sin vaso de precipitado, para obtener la medida del aire. Con esta medida registrada, se procede a realizar la medición sobre los cinco líquidos restantes, que se hace de la misma forma en todos los casos. Primero, se llena el vaso de precipitado con un volumen del líquido ligeramente superior a 100 ml, y se mide su temperatura con el termopar. Después, se introduce la sonda coaxial de final abierto en el interior del vaso de precipitado, de forma que el plano inferior de la sonda queda sumergido en el líquido. Antes de proceder a registrar la medida del coeficiente de reflexión del líquido, se debe asegurar

que no se ha formado ninguna microburbuja entre el plano de la sonda y el líquido, ya que esto variará la capacidad que estamos midiendo y, por tanto, alterará los resultados del coeficiente de reflexión obtenidos. Con esto comprobado, se realiza la medida del líquido en cuestión, y se repite este procedimiento descrito para los cinco líquidos mencionados. Entre medida y medida, se ha realizado la limpieza completa del vaso de precipitado, del termopar y de la sonda coaxial de final abierto para evitar la contaminación de las muestras. Por último, se ha realizado también la calibración de la sonda coaxial, cuyo procedimiento se explica en el siguiente apartado. Las medidas del coeficiente de reflexón S_{11} para todos los materiales se ha realizado entre una frecuencia de 10 kHz y una frecuencia de 1,5 GHz. La forma en la que se han realizado las medidas se puede ver en la Figura 4.7.

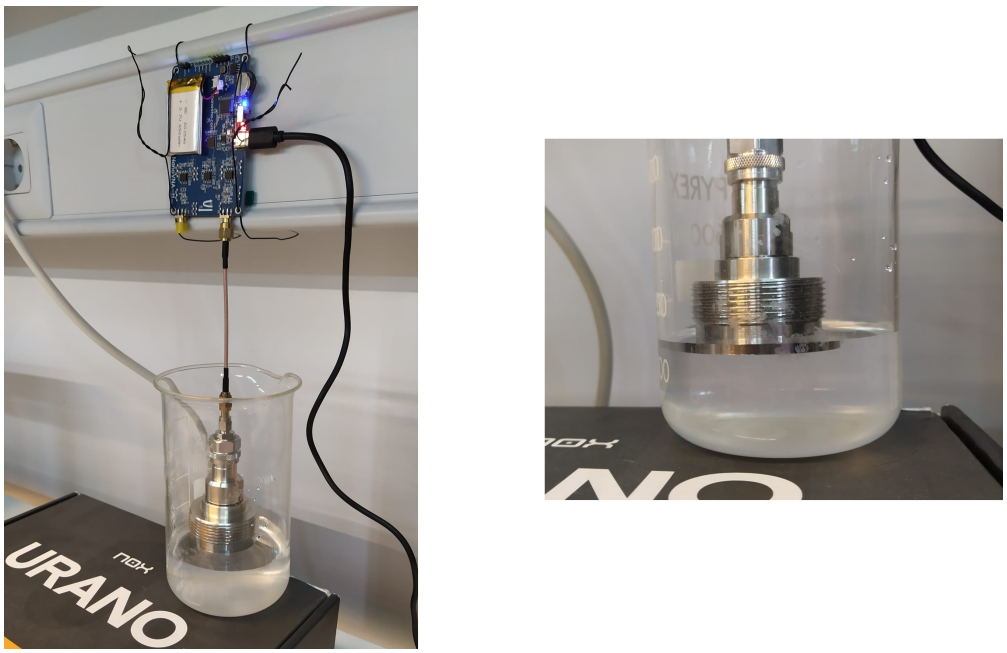


Figura 4.7: Cable coaxial y sonda coaxial de final abierto conectados al VNA. Sonda coaxial sumergida en el líquido medido.

Para registrar las medidas del coeficiente de reflexión de los distintos materiales con el NanoVNA, se ha utilizado el programa de código abierto *NanoVNA Saver*, que está programado en Python y permite realizar tanto la calibración desde el propio programa, guardando dicha calibración, como las mediciones entre dos frecuencias programadas y exportar las mismas en formato `.s1p` (medidas de 1 puerto) o `.s2p` (medidas de 2 puertos). En este caso, las medidas son de 1 puerto. Además, el propio programa permite realizar la representación de las mediciones en distintos formatos, como la Carta de Smith, parte real e imaginaria o representación de magnitud logarítmica, entre otros. Se puede ver la interfaz del programa en la siguiente Figura 4.8.

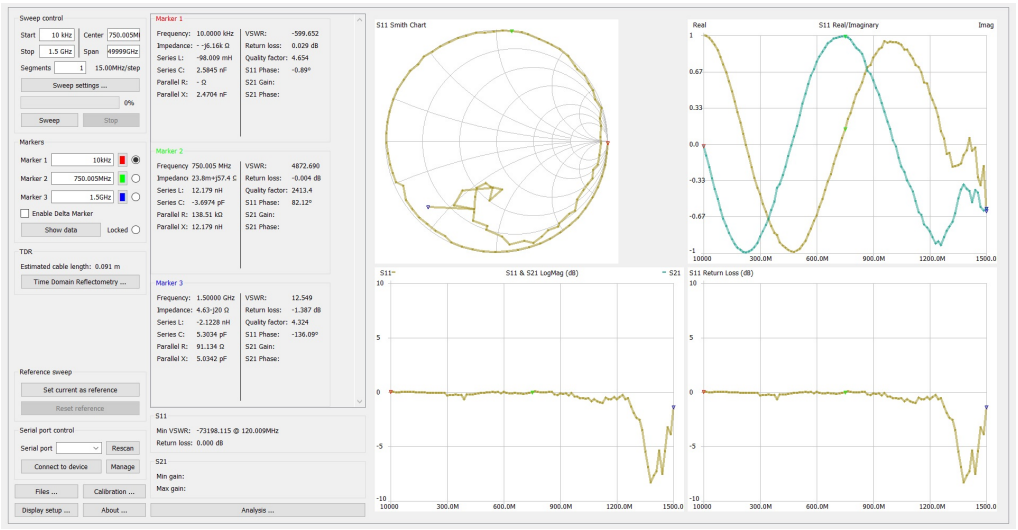


Figura 4.8: NanoVNA Saver.

Para el registro de los datos del coeficiente de reflexión medido con el VNA diseñado, se ha utilizado el Datalogger formado por la ESP32, que recoge los datos mediante la comunicación por UART con el VNA, tal y como se ha explicado en la sección 3.6 y se ha introducido al comienzo de esta sección, a la hora de explicar el diagrama de bloques utilizado durante la realización de los ensayos.

4.3. Calibración del VNA

La calibración del VNA se ha realizado en dos partes. En primer lugar, se ha efectuado la calibración en la punta del cable coaxial SMA mediante la conexión del estándar de calibración *Open-Short-Load* (50 Ω) en dicho extremo. Este estándar se puede ver en la Figura 4.9.

Figura 4.9: Estándar *Open-Short-Load*.

4.3.1. Calibración OWL

En segundo lugar, a partir de las medidas realizadas con el VNA diseñado y el NanoVNA de referencia en el apartado anterior, se procede a la calibración del VNA con la sonda coaxial de final abierto, conectada en el extremo del cable coaxial SMA. Para ello, se ha realizado la calibración de Circuito abierto-Agua-Líquido (OWL, *Open-Water-Liquid*) descrita en el artículo científico de Norman Wagner et al. [2], para calcular la permitividad compleja $\varepsilon_r^\star$ a partir del coeficiente de reflexión complejo S_{11} medido con el VNA.

Para valores del coeficiente de reflexión y permitividad del aire (*Open*, Circuito abierto) de S_{11}^O y $\varepsilon_{r,O}$, respectivamente, del coeficiente de reflexión y permitividad del agua destilada (*Water*, Agua) de S_{11}^W y $\varepsilon_{r,W}$, respectivamente y de los coeficientes de reflexión y permitividades de los líquidos medidos (*Liquid*, Líquido) de $S_{11}^{L,i}$ y $\varepsilon_{r,L,i}$, respectivamente, la calibración OWL consiste en determinar el valor de las constantes c_1, c_2 y c_3 para cada frecuencia resolviendo el siguiente sistema de ecuaciones:

$$\begin{aligned} S_{11}^{O} c_1 - c_2 - \varepsilon_{r,O} c_3 &= -\varepsilon_{r,O} S_{11}^{O} \\ S_{11}^{W} c_1 - c_2 - \varepsilon_{r,W} c_3 &= -\varepsilon_{r,W} S_{11}^{W} \\ S_{11}^{L,1} c_1 - c_2 - \varepsilon_{r,L,1} c_3 &= -\varepsilon_{r,L,1} S_{11}^{L,1} \\ &\vdots \\ S_{11}^{L,n} c_1 - c_2 - \varepsilon_{r,L,n} c_3 &= -\varepsilon_{r,L,n} S_{11}^{L,n} \end{aligned} \tag{4.1}$$

O, en forma matricial:

$$\begin{pmatrix} S_{11}^{O} & -1 & -\varepsilon_{r,O} \\ S_{11}^{W} & -1 & -\varepsilon_{r,W} \\ S_{11}^{L,1} & -1 & -\varepsilon_{r,L,1} \\ & \vdots & \\ S_{11}^{L,n} & -1 & -\varepsilon_{r,L,n} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} -\varepsilon_{r,O} S_{11}^{O} \\ -\varepsilon_{r,W} S_{11}^{W} \\ -\varepsilon_{r,L,1} S_{11}^{L,1} \\ \vdots \\ -\varepsilon_{r,L,n} S_{11}^{L,n} \end{pmatrix} \tag{4.2}$$

En forma compacta, esta relación se puede expresar como:

$$\mathbf{M} \cdot \mathbf{c} = \mathbf{e} \tag{4.3}$$

En el caso de utilizar tres medidas (aire, agua y un líquido), el sistema se puede resolver fácilmente mediante:

$$\mathbf{c} = \mathbf{M}^{-1} \cdot \mathbf{e} \tag{4.4}$$

Con el valor de las constantes c_1, c_2 y c_3 para cada frecuencia, ya se puede calcular el valor de la permitividad compleja $\varepsilon_r^{\star}$ del líquido desconocido a partir del coeficiente de reflexión S_{11} medido para ese líquido, mediante la expresión (4.5):

$$\varepsilon_r^{\star} = \frac{c_1 S_{11} - c_2}{c_3 - S_{11}} \tag{4.5}$$

Esta calibración se ha realizado utilizando aire, agua y un líquido, el cual ha ido cambiando entre metanol, isopropanol, acetona y etilenglicol. Realizando esto, se ha obtenido el valor de c_1, c_2 y c_3 para todas las frecuencias cuatro veces distintas, rotando el líquido utilizado, de manera que los líquidos no utilizados para calibrar en cada caso se han empleado en validar la calibración. De esta forma, los valores de c_1, c_2 y c_3 seleccionados se corresponden con la calibración que menor error ha generado.

Para poder aplicar la calibración OWL, es necesario conocer el valor de la permitividad de los distintos materiales en el espectro de frecuencias $(\varepsilon_{r,O}, \varepsilon_{r,W}, \varepsilon_{r,L,i})$. Para tal fin, se han utilizado valores de referencia encontrados en la bibliografía, basados en modelos de dispersión dieléctrica, descritos en la siguiente sección. Estos valores de referencia se han utilizado para aplicarlos en el método de calibración OWL, así como para calcular el error cometido en la estimación de la permitividad de los líquidos no usados en la misma. El indicador de error utilizado es el error cuadrático medio $RMSE$ (*Root Mean Square Error*), que se define como:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}{n}} \tag{4.6}$$

Aplicado a la calibración realizada, se calcula el $RMSE$ sobre los tres líquidos desconocidos no utilizados en la calibración y de los que se ha estimado el valor de su permitividad:

$$RMSE = \sqrt{\frac{\sum_{k=1}^{3} \sum_{i=1}^{n} \left(\varepsilon^{\star}_{r,L,k,i} - \varepsilon_{r,L,k,i}\right)^2}{3n}} \tag{4.7}$$

Donde n es el número de muestras medidas con el VNA. Para la realización del proceso completo de calibración OWL con todos los líquidos, se ha implementado un código en MATLAB, que realiza todo lo descrito, y que se puede ver en el Anexo V.3.

El código comienza con una limpieza de todas las variables que pudieran existir antes de la ejecución del mismo. Seguidamente, carga el *Workspace*, es decir, el espacio de trabajo, donde se encuentran todos los datos del coeficiente de reflexión de los distintos materiales. Para que esto se pueda realizar, hay que preparar los datos en una tabla recogida dentro de un archivo `*.mat`. Para ello, se introducen en una hoja de cálculo `MEDIDAS_VNA.xlsx` todos los datos correspondientes a los distintos coeficientes de reflexión (parte real e imaginaria) y, tras ello, se genera el archivo `*.mat` ejecutando lo siguiente desde MATLAB:

```
[num, txt, raw] = xlsread('MEDIDAS_VNA.xlsx');

VNArawS11_2021 = cell2table(raw(2:end, :));

VNArawS11_2021.Properties.VariableNames = raw(1, :);

save VNArawS11_2021.mat VNArawS11_2021
```

Con esto ya listo, se cargan los datos de reflexión preparados y se definen las temperaturas de trabajo para cada material. Seguidamente, se realiza una interpolación para normalizar las frecuencias de los datos del S_{11} del VNA. Luego, se definen los modelos ideales para la permitividad de los distintos materiales y, finalmente, se obtiene la permitividad a partir del coeficiente de reflexión medido por el VNA, mediante la calibración OWL. Paralelamente, se calcula el error obtenido tras la realización de la calibración con cada líquido.

4.3.2. Modelos dieléctricos de dispersión

Para determinar las permitividades complejas teóricas de los distintos materiales medidos en función de la frecuencia y la temperatura a la que se han realizado las mediciones, se ha empleado una serie de modelos que ofrecen ecuaciones de relajación de la permitividad, y son los modelos de Debye, Cole-Cole y Davidson-Cole, cuyas ecuaciones y tablas de parámetros empleadas para estos materiales están incluidas en la bibliografía [4], [5], [6]. Estos modelos se explican a continuación, y obtienen una curva en el espectro de frecuencias para la permitividad compleja que presenta el resultado de la Figura 4.10 para la parte real e imaginaria de la permitividad.

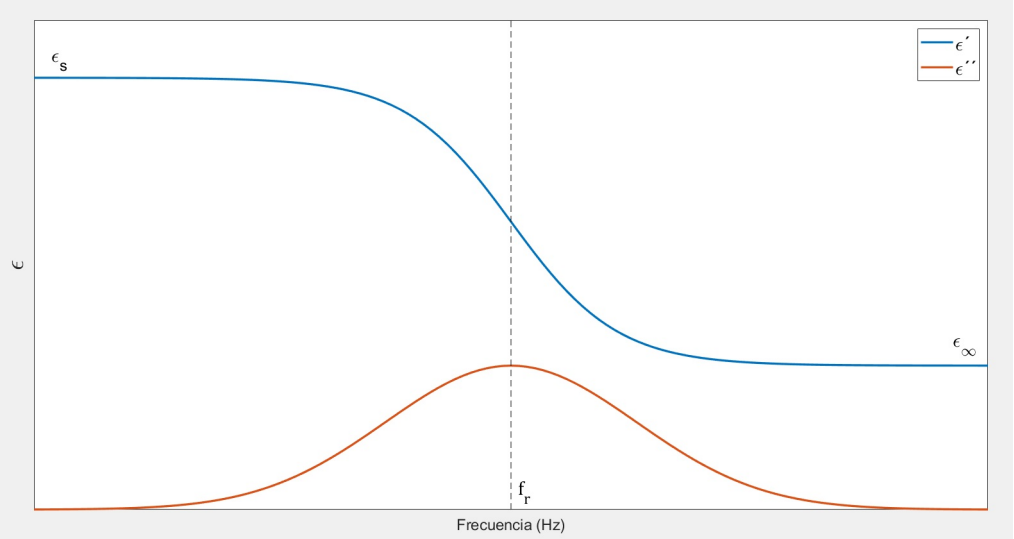


Figura 4.10: Modelo para la permitividad compleja.

Los materiales medidos han empleado los siguientes modelos para determinar su permitividad compleja: para el agua destilada y el metanol se ha empleado el modelo de Debye, para el isopropanol se ha utilizado el modelo de Double Debye, para la acetona se ha usado el modelo de Cole-Cole y para el etilenglicol se ha empleado el modelo de Davidson-Cole. Por último, para el aire, se ha tomado su permitividad relativa como constante en todo el rango de frecuencias: $\varepsilon_{r,O} = 1$. Esto queda recogido en la siguiente tabla resumen para todos los materiales.

Tabla 4.4: Modelos utilizados para la permitividad de los distintos materiales.

Material	Modelo
Agua destilada	Debye
Metanol	Debye
Isopropanol	Double Debye
Acetona	Cole-Cole
Etilenglicol	Davidson-Cole
Aire	$\varepsilon_{r,O} = 1$

4.3.2.1. Modelo de Debye

El modelo de la ecuación de relajación de Debye para la permitividad compleja $\varepsilon_r^\star$ en función de la frecuencia f depende del valor de la permitividad estática ε_s, que es la permitividad a una frecuencia de 0 Hz, del valor de la permitividad en el infinito o a alta frecuencia ε_∞, que es la permitividad a una frecuencia mucho mayor que la frecuencia de relajación, y del tiempo (τ) o frecuencia (f_r) de relajación:

$$\varepsilon_r^\star = \varepsilon_\infty + \frac{\varepsilon_s - \varepsilon_\infty}{1 + j\omega\tau} = \varepsilon_\infty + \frac{\varepsilon_s - \varepsilon_\infty}{1 + j\frac{f}{f_r}} \tag{4.8}$$

4.3.2.2. Modelo de Double Debye

El modelo de Double Debye para la permitividad compleja $\varepsilon_r^\star$ en función de la frecuencia f emplea dos relajaciones dieléctricas, y depende del valor de la permitividad estática ε_s, del valor de la permitividad de alta frecuencia de la relajación de la frecuencia más baja ε_h, del valor de la permitividad en el infinito ε_∞ y de los tiempos (τ_1, τ_2) o frecuencias (f_{r_1}, f_{r_2}) de relajación:

$$\varepsilon_r^\star = \varepsilon_\infty + \frac{\varepsilon_s - \varepsilon_h}{1 + j\omega\tau_1} + \frac{\varepsilon_h - \varepsilon_\infty}{1 + j\omega\tau_2} = \varepsilon_\infty + \frac{\varepsilon_s - \varepsilon_h}{1 + j\frac{f}{f_{r_1}}} + \frac{\varepsilon_h - \varepsilon_\infty}{1 + j\frac{f}{f_{r_2}}} \tag{4.9}$$

4.3.2.3. Modelo de Cole-Cole

El modelo de la ecuación de Cole-Cole para la permitividad compleja $\varepsilon_r^\star$ en función de la frecuencia f depende del valor de la permitividad estática ε_s, del valor de la permitividad en el infinito ε_∞, del tiempo (τ) o frecuencia (f_r) de relajación y de un parámetro β que caracteriza la distribución de relajación:

$$\varepsilon_r^\star = \varepsilon_\infty + \frac{\varepsilon_s - \varepsilon_\infty}{1 + (j\omega\tau)^\beta} = \varepsilon_\infty + \frac{\varepsilon_s - \varepsilon_\infty}{1 + \left(j\frac{f}{f_r}\right)^\beta} \tag{4.10}$$

4.3.2.4. Modelo de Davidson-Cole

El modelo de Davidson-Cole para la permitividad compleja $\varepsilon_r^\star$ en función de la frecuencia f, al igual que el modelo anterior, depende del valor de la permitividad estática ε_s, del valor de la permitividad en el infinito ε_∞, del tiempo (τ) o frecuencia (f_r) de relajación y de un parámetro β que proporciona información acerca de la distribución y simetría de los tiempos de relajación:

$$\varepsilon_r^\star = \varepsilon_\infty + \frac{\varepsilon_s - \varepsilon_\infty}{(1 + j\omega\tau)^\beta} = \varepsilon_\infty + \frac{\varepsilon_s - \varepsilon_\infty}{\left(1 + j\frac{f}{f_r}\right)^\beta} \tag{4.11}$$

4.3.2.5. Representación gráfica de los modelos para la permitividad

Estos modelos para la permitividad compleja proporcionan una curva ideal para el espectro de la misma en función del tipo de material. Si se realiza la representación gráfica de la permitividad para los materiales medidos según el modelo utilizado, el cual se indica en la Tabla 4.4 para cada material, se obtienen las siguientes curvas para sus permitividades:

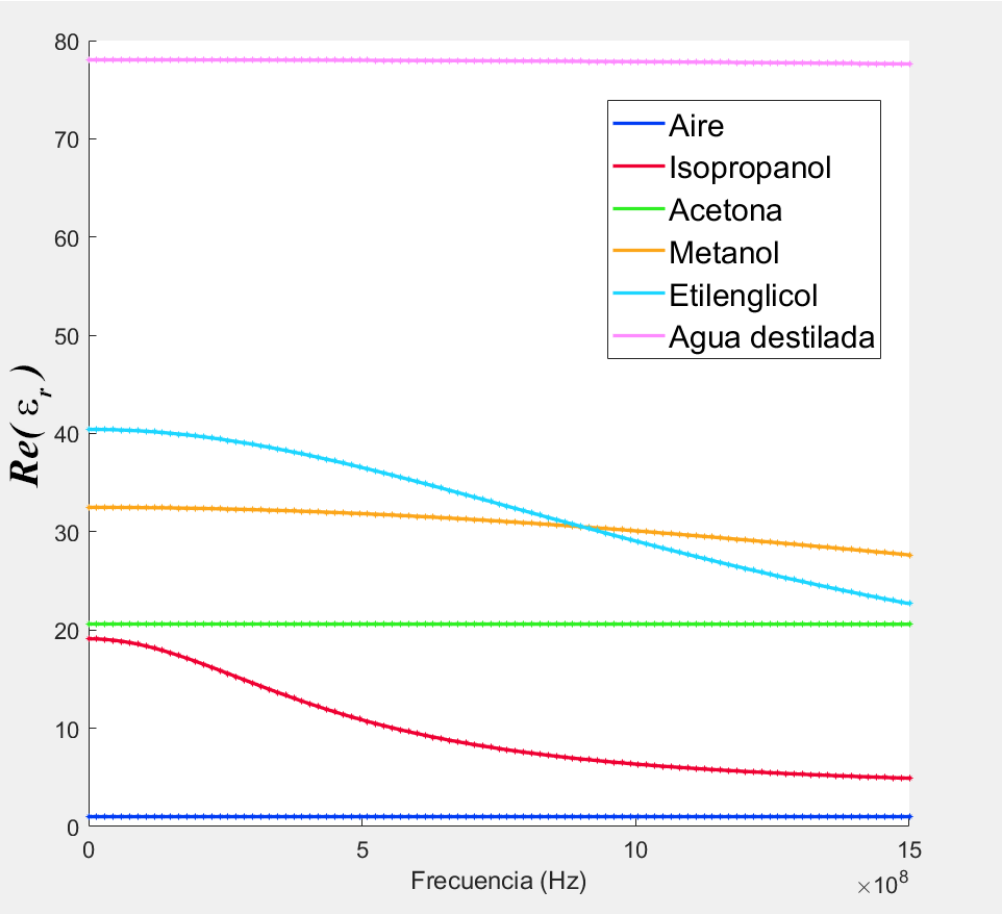


Figura 4.11: Representación gráfica de los modelos para la permitividad de los distintos materiales. Parte real.

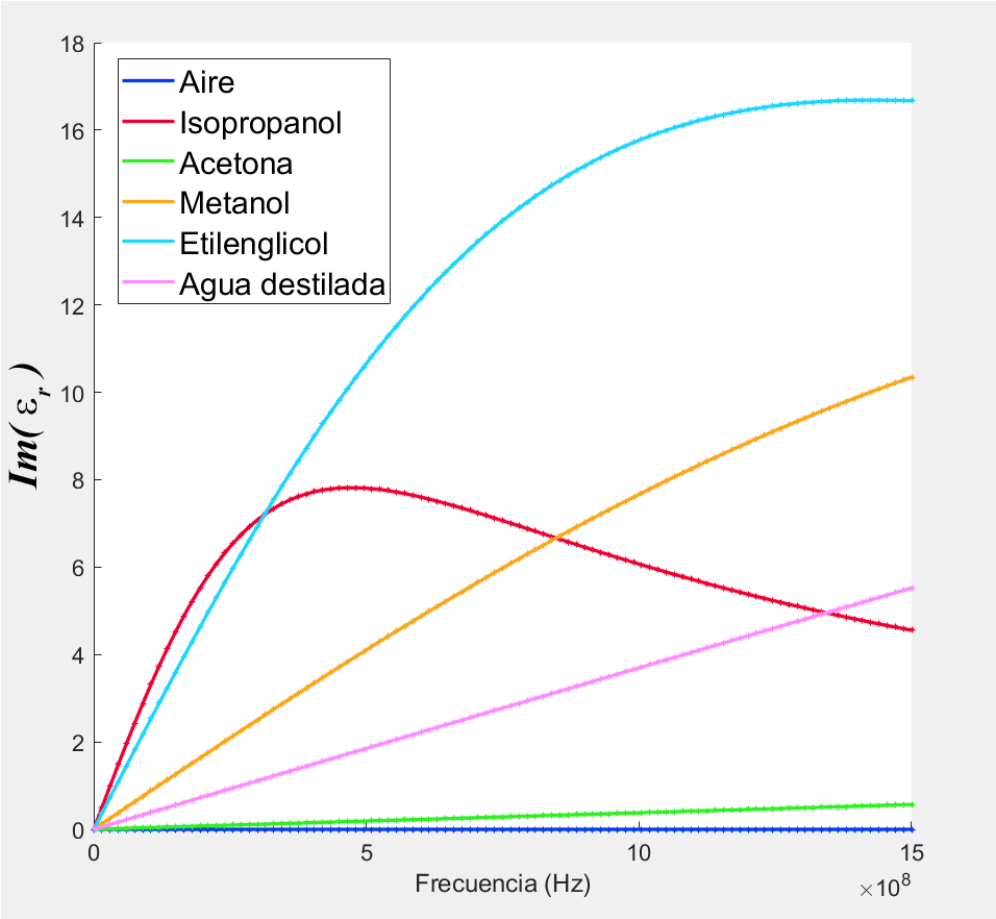


Figura 4.12: Representación gráfica de los modelos para la permitividad de los distintos materiales. Parte imaginaria.

Capítulo 5

Resultados

En este capítulo, se exponen los resultados obtenidos tras realizar las mediciones con el VNA diseñado, comparándolos con el de referencia. Se presentan, en primer lugar, los resultados obtenidos de la medición del parámetro S_{11} a distintas frecuencias para los diferentes materiales. Luego, se muestran y analizan los resultados obtenidos tras la calibración del VNA para obtener la permitividad de dichos materiales, a partir del código desarrollado en MATLAB. Por último, tras haber determinado la calibración más exacta, se han introducido las constantes en el Datalogger y se ha ampliado el código del mismo para aplicar la ecuación que permite obtener el valor de la permitividad compleja a partir del coeficiente de reflexión, que se corresponde con la expresión (4.5). Los resultados del coeficiente de reflexión y de la permitividad, tal y como se ha mencionado, se guardan en ficheros de texto para poder exportarlos y procesarlos fácilmente.

5.1. Coeficiente de reflexión

En esta sección, se observan los resultados obtenidos del coeficiente de reflexión tras la realización de las mediciones por parte de ambos analizadores de redes, comparándolos para validar el funcionamiento del diseño del VNA del presente proyecto.

5.1.1. Coeficiente de reflexión con el VNA de referencia

Los resultados relativos al coeficiente de reflexión del VNA de referencia se pueden visualizar en la Figura 5.1, en formato de parte real e imaginaria. Se puede observar que la parte real del coeficiente de reflexión, a baja frecuencia, toma el valor de 1, mientras que la parte imaginaria comienza con un valor de 0. También se aprecia que, en alta frecuencia, el coeficiente de reflexión medido empieza a describir un cierto rizado, tanto en la parte real como en la imaginaria en todos los materiales. Se ha empleado línea continua para la parte real y línea discontinua para la parte imaginaria.

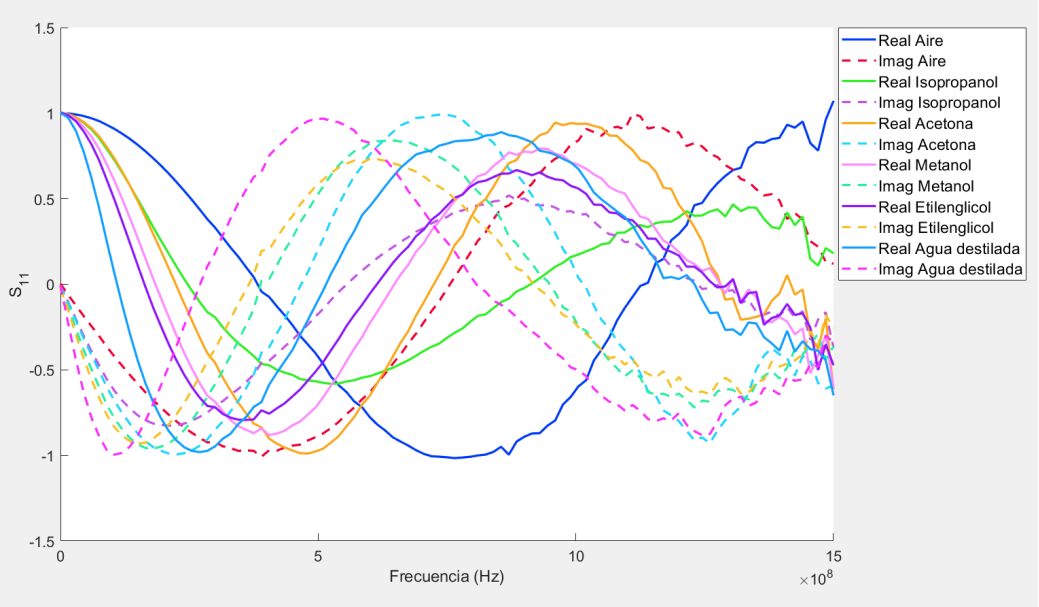


Figura 5.1: Coeficiente S_{11} medido sobre los distintos materiales con el VNA de referencia.

5.1.2. Coeficiente de reflexión con el VNA diseñado

De la misma forma, en la Figura 5.2 se pueden ver las curvas obtenidas para coeficiente de reflexión con el VNA diseñado, en el mismo formato. La forma de estas curvas es muy similar al caso anterior. Además, se puede observar que, a alta frecuencia, las curvas reducen su rizado y se enderezan ligeramente en comparación con los resultados anteriores. Esto último se puede comprobar en la Figura 5.3, donde se ha escogido el coeficiente de reflexión de la acetona de ambos analizadores. La mayor diferencia ocurre a alta frecuencia, donde el VNA diseñado (colores rojo y verde oscuro) presenta una curva para el S_{11} más suave, mientras que el VNA de referencia (colores naranja y verde claro) presenta esos picos al final. También se verifica que los resultados con ambos analizadores son prácticamente iguales.

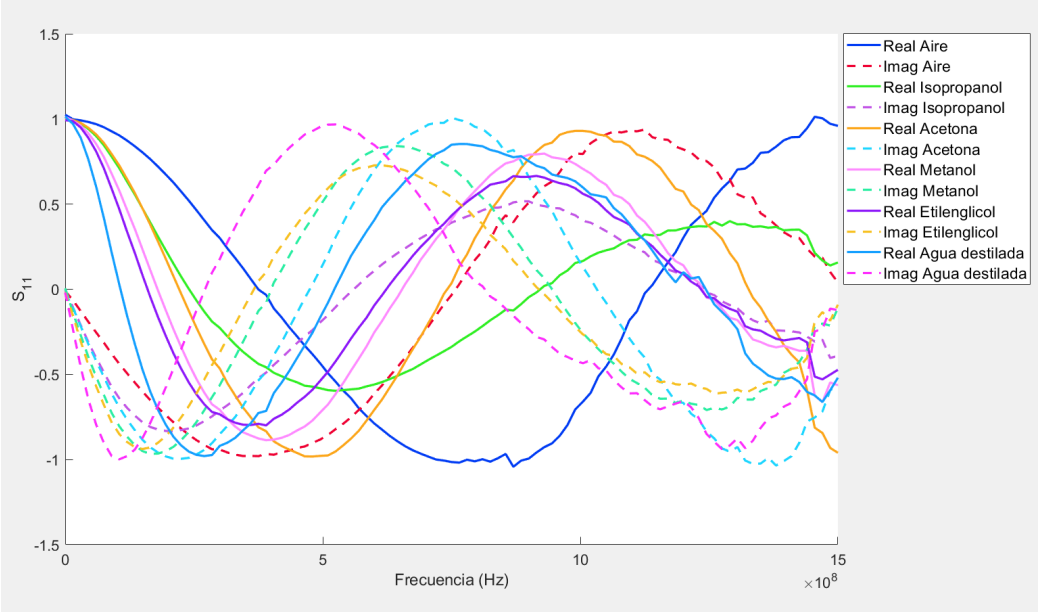


Figura 5.2: Coeficiente S_{11} medido sobre los distintos materiales con el VNA diseñado.

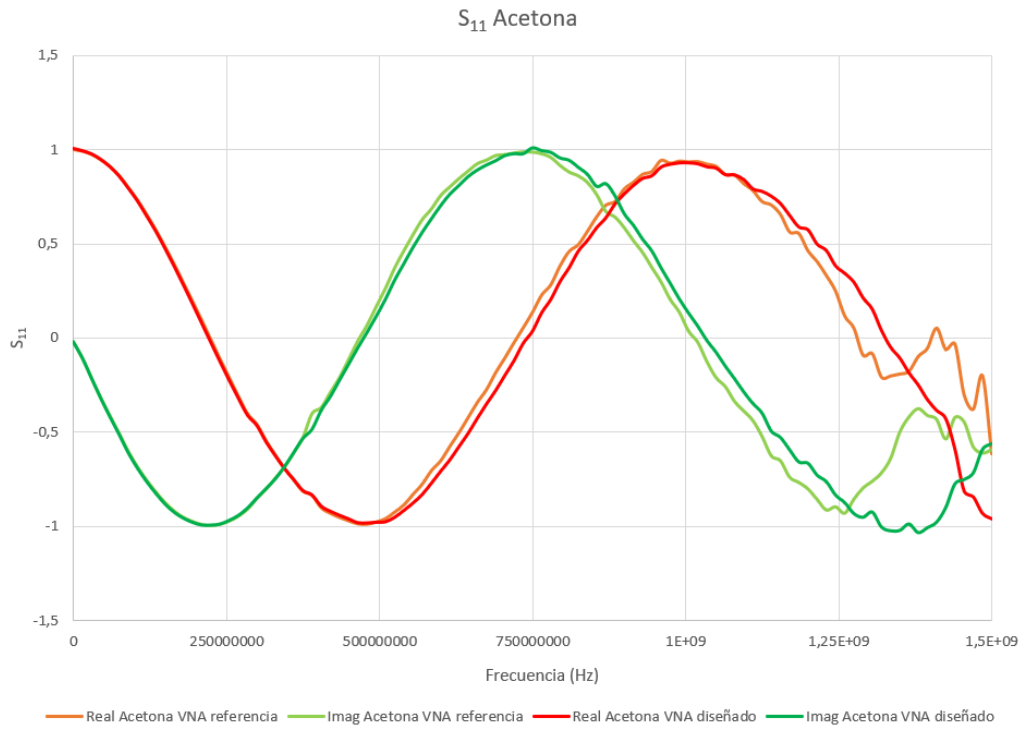


Figura 5.3: Comparativa del coeficiente de reflexión de la acetona de ambos analizadores de redes.

5.2. Permitividad compleja

Con los datos medidos del coeficiente de reflexión para ambos analizadores de redes, se procede a realizar la calibración OWL mediante el código implementado en MATLAB. Tal y como se ha explicado en la fase experimental, se ha realizado la calibración con aire, agua y un líquido que ha ido rotando entre metanol, isopropanol, acetona y etilenglicol, de manera que los líquidos no empleados para la calibración en cada caso se han utilizado para validarla. De esta forma, se ha obtenido el valor para las constantes c_1, c_2 y c_3 cuatro veces distintas, seleccionando aquellas que se corresponden con la calibración que menor error $RMSE$ ha generado. Se guardarán las constantes correspondientes a los resultados con menor error del VNA diseñado, para introducirlas en el Datalogger con el objetivo de que sea capaz de calcular la permitividad a partir de los datos del coeficiente de reflexión.

5.2.1. Calibración OWL del VNA de referencia

Los datos obtenidos tras la realización de la calibración OWL con los distintos líquidos con el VNA de referencia se pueden visualizar en las Figuras 5.4, 5.5, 5.6 y 5.7, correspondientes a la calibración realizada con metanol, isopropanol, acetona y etilenglicol, respectivamente. Se representa con una línea continua el modelo dieléctrico de dispersión ideal para cada permitividad, según lo indicado en la Tabla 4.4 y, con asteriscos, los valores correspondientes al cálculo de la permitividad tras aplicar la calibración. El líquido empleado para la calibración en cada caso se indica en el título de la gráfica correspondiente. Se puede observar un aumento de la dispersión cuando se alcanza la alta frecuencia, alrededor de 1 GHz, donde los cálculos comienzan a diferir de los modelos para las permitividades.

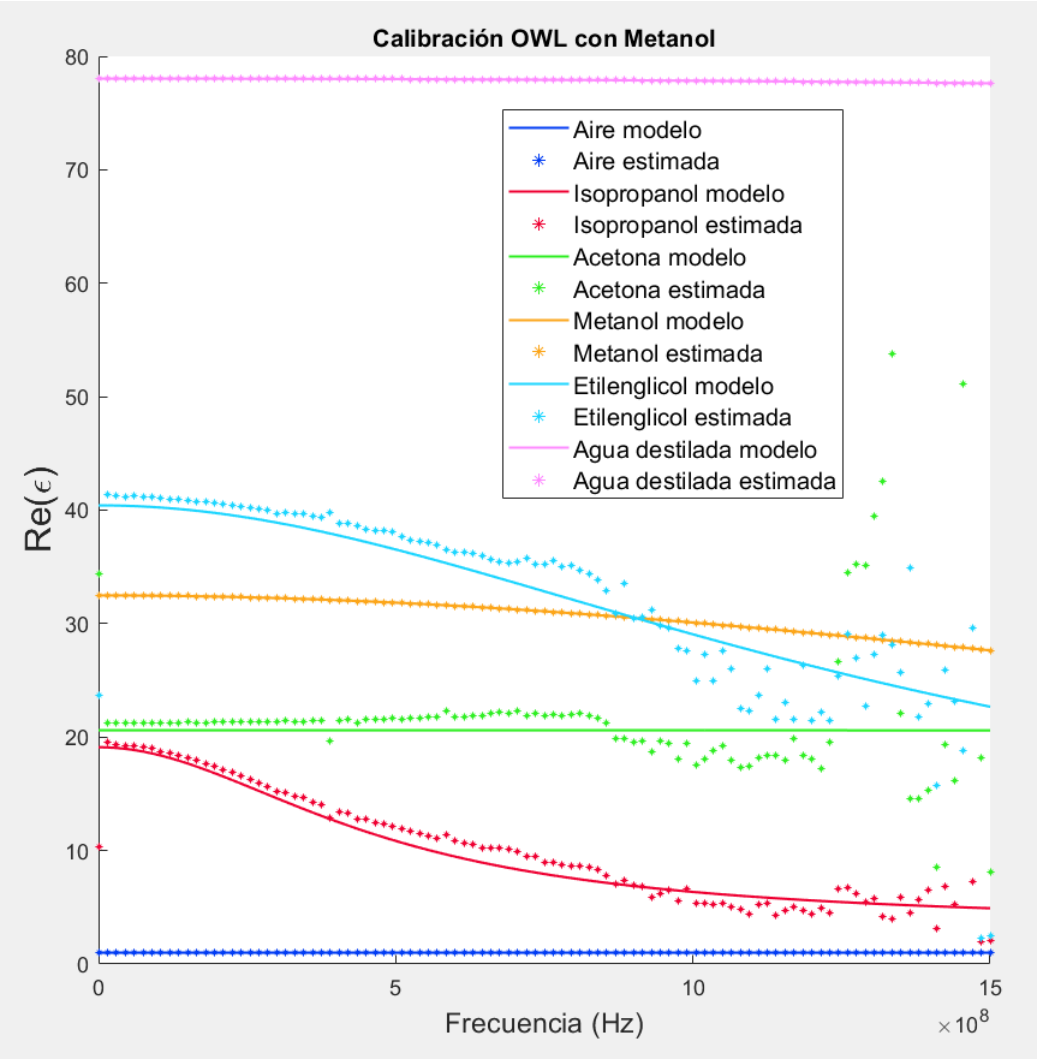


Figura 5.4: Calibración OWL con metanol. VNA de referencia.

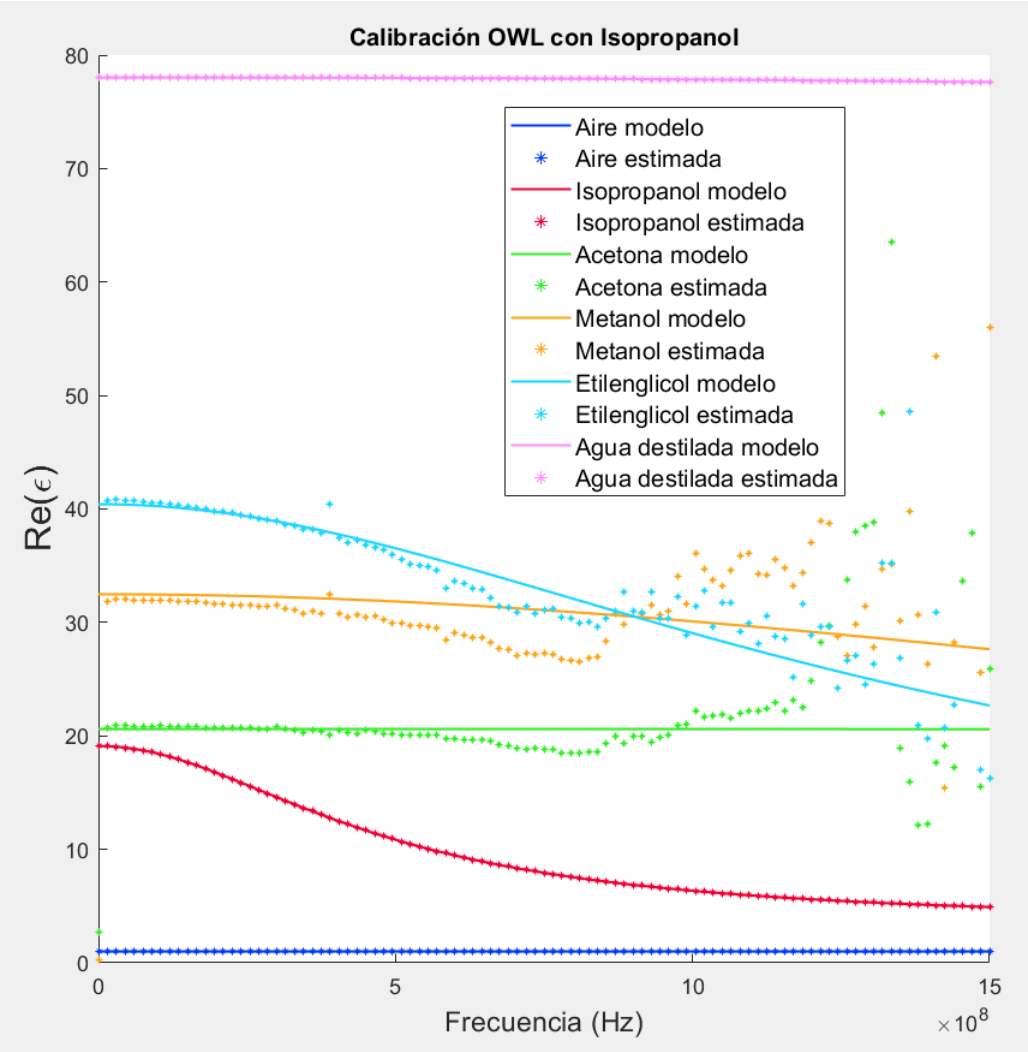


Figura 5.5: Calibración OWL con isopropanol. VNA de referencia.

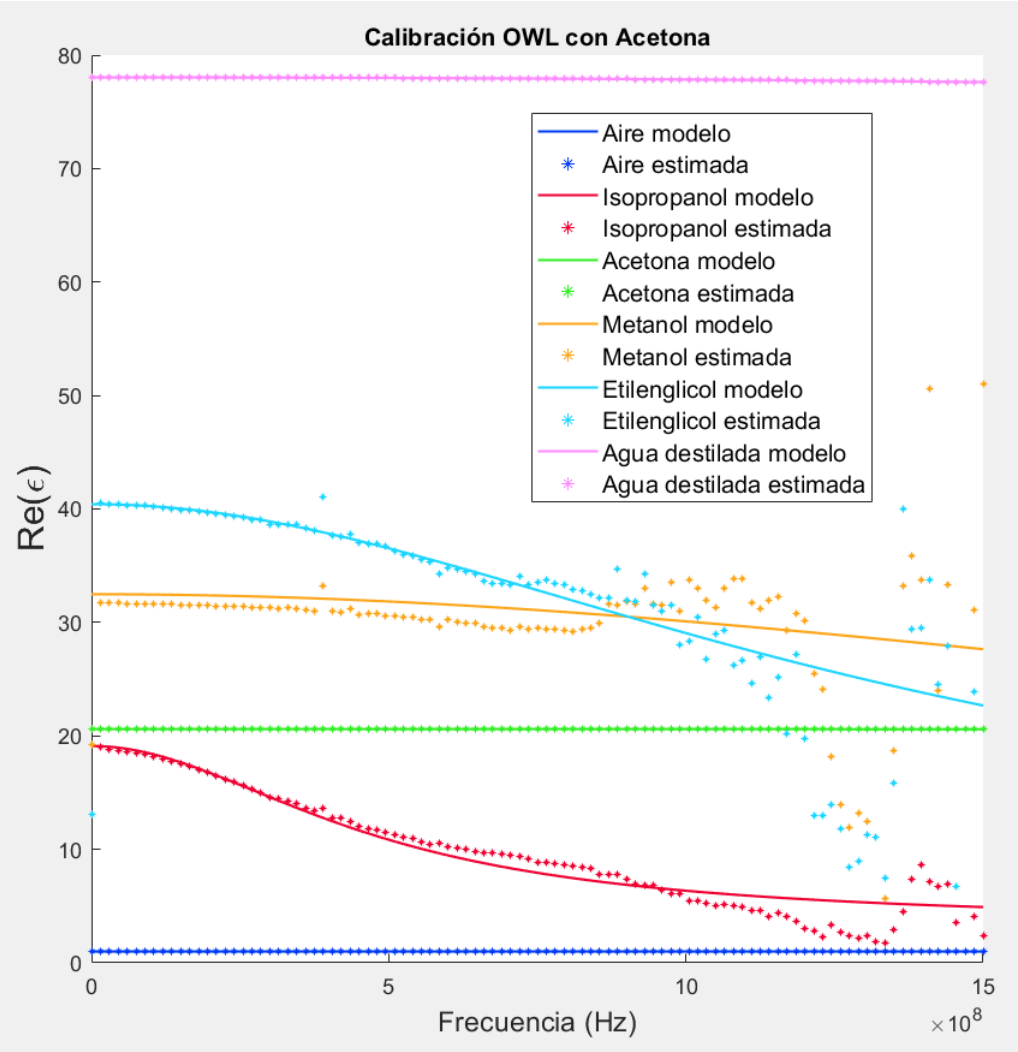


Figura 5.6: Calibración OWL con acetona. VNA de referencia.

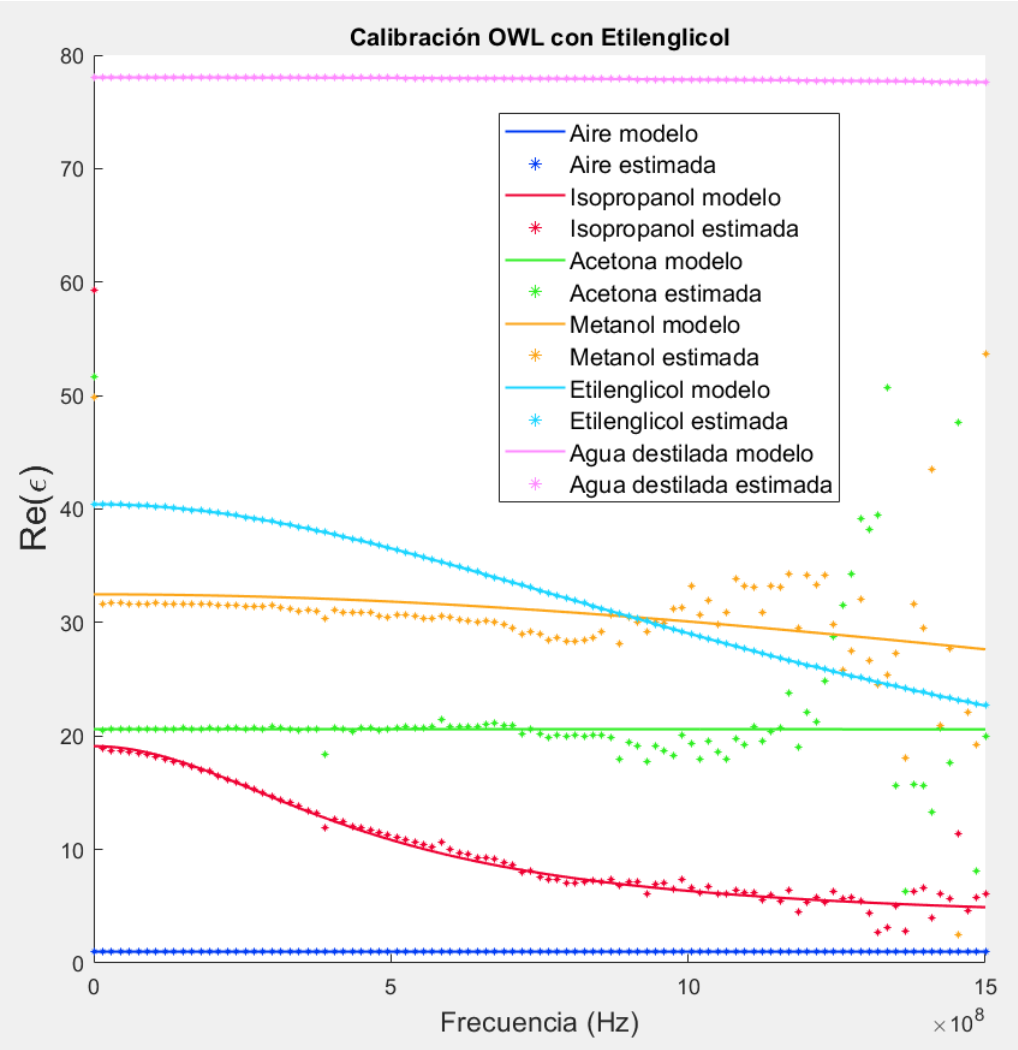


Figura 5.7: Calibración OWL con etilenglicol. VNA de referencia.

Como ya se ha mencionado, la precisión obtenida empleando la calibración OWL es alta hasta una frecuencia de 1 GHz, aproximadamente. Esto puede ser debido a una limitación de frecuencia del algoritmo o a la capacidad del VNA de proporcionar una mejor precisión a partir de esas frecuencias. Si, por ejemplo, se seleccionan los datos obtenidos tras la calibración con acetona hasta 1 GHz, los cálculos se aproximan bastante a los modelos de la permitividad, tal y como se observa en la Figura 5.8.

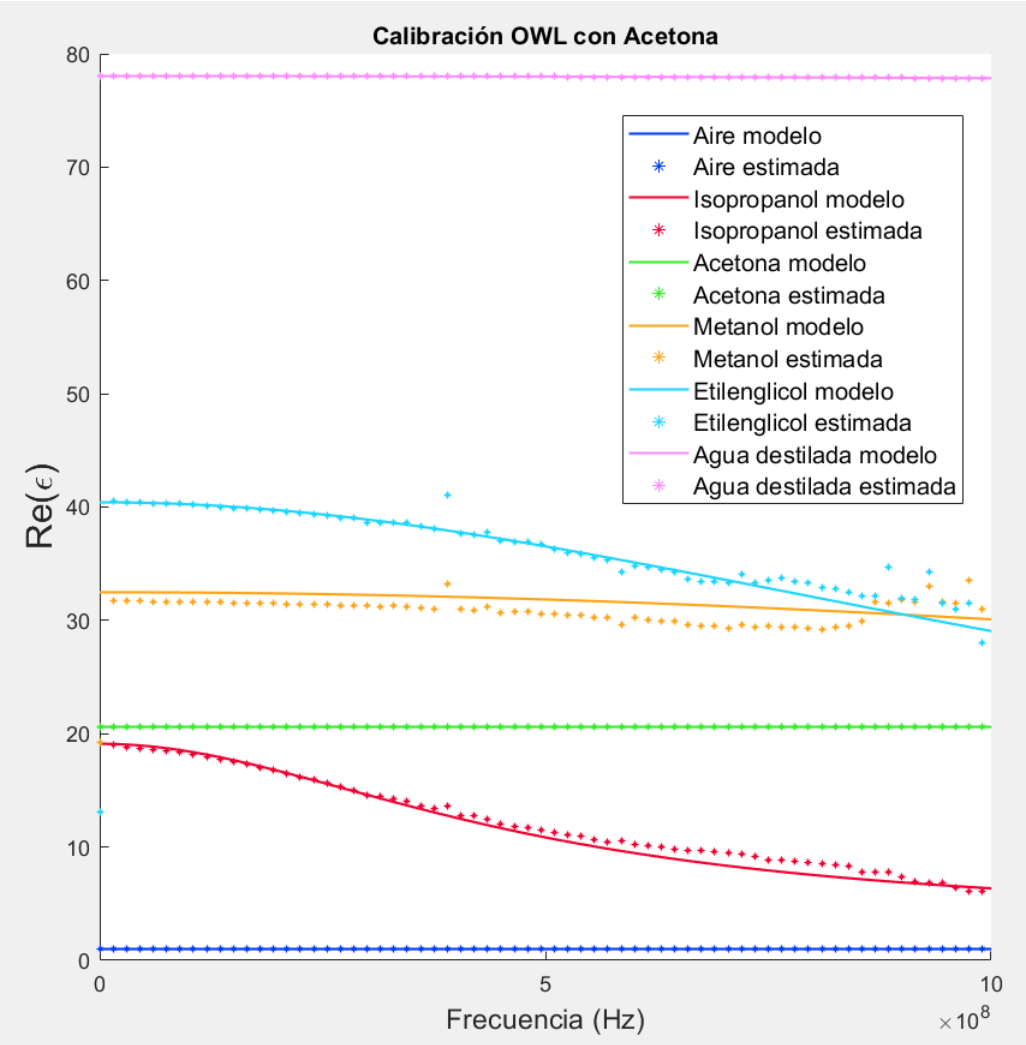


Figura 5.8: Calibración OWL con acetona hasta 1 GHz. VNA de referencia.

5.2.1.1. Error en la calibración OWL del VNA de referencia

Una vez realizada la calibración OWL con metanol, isopropanol, acetona y etilenglicol, se procede a evaluar cuál ha sido la calibración que menor error ha generado, observando el valor calculado del error $RMSE$ para cada una, que se recoge en la Tabla 5.1. Se puede observar en dicha tabla que la calibración que se corresponde con el menor error generado ha sido la calibración con acetona, por lo que se usarían las constantes c_1, c_2 y c_3 obtenidas durante su calibración. El error se ha calculado teniendo en cuenta todo el ancho de banda, desde los 10 kHz hasta los 1,5 GHz. Por otra parte, si se selecciona el ancho de banda hasta 1 GHz, el líquido óptimo puede cambiar.

Tabla 5.1: Errores $RMSE$ durante la calibración OWL con los distintos líquidos para el VNA de referencia.

Líquido empleado para la calibración OWL	**RMSE**
Metanol	12,397
Isopropanol	9,263
Acetona	6,378
Etilenglicol	27,804

5.2.2. Calibración OWL del VNA diseñado

De igual forma, los datos obtenidos tras la realización de la calibración OWL con los distintos liquidos con el VNA diseñado se pueden ver en las Figuras 5.9, 5.10, 5.11 y 5.12, correspondientes a la calibración realizada con metanol, isopropanol, acetona y etilenglicol, respectivamente. Como ya se ha adelantado en la anterior calibración, se puede observar un aumento de la dispersión a altas frecuencias, sobre 1 GHz, y los cálculos empiezan a diferir de los modelos ideales para las permitividades.

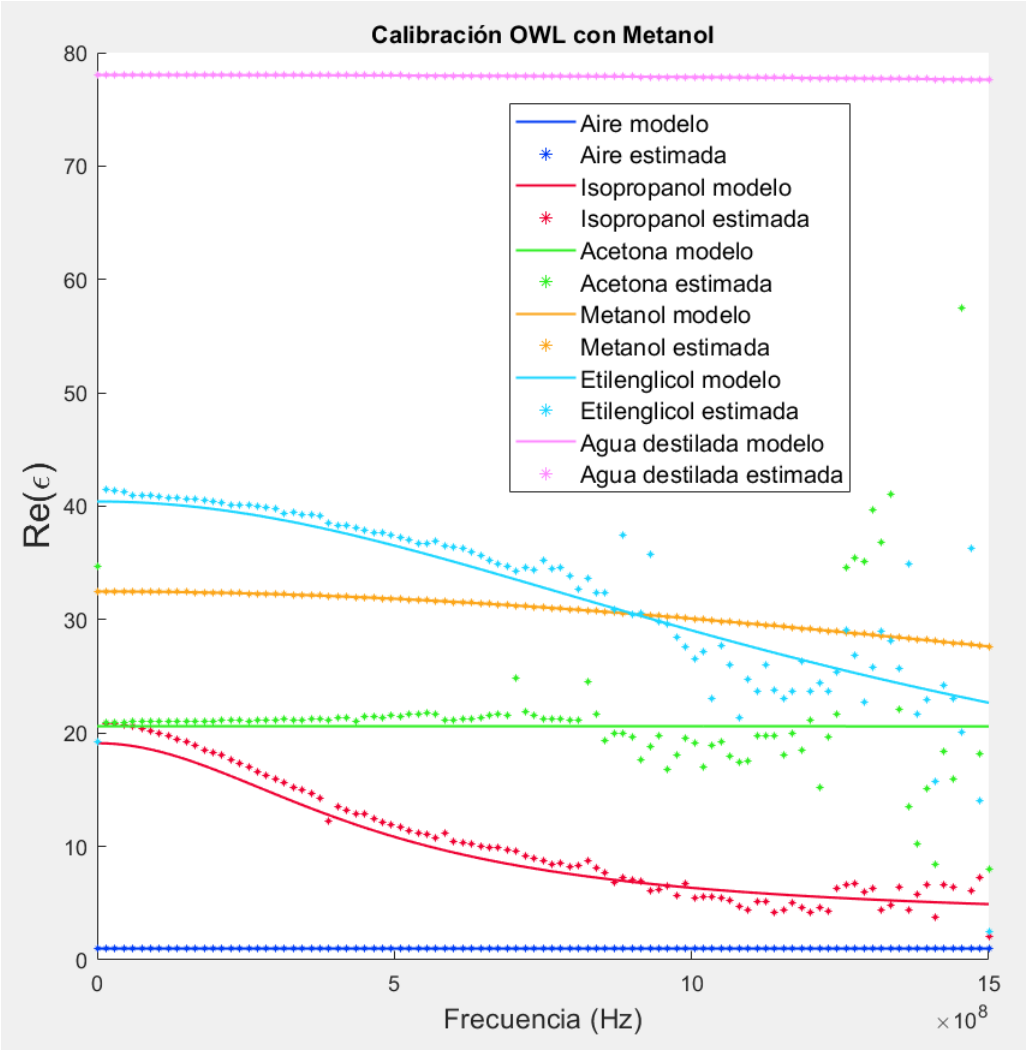


Figura 5.9: Calibración OWL con metanol. VNA diseñado.

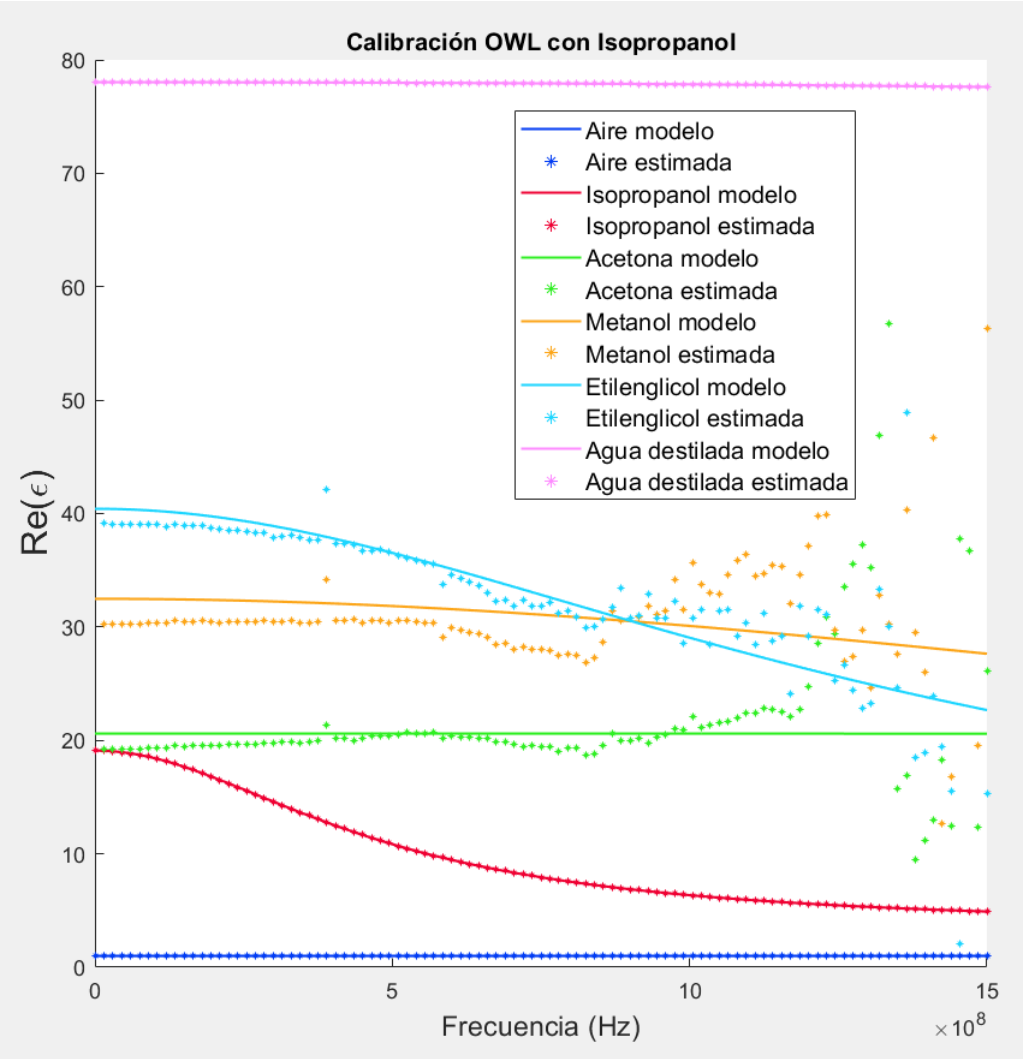


Figura 5.10: Calibración OWL con isopropanol. VNA diseñado.

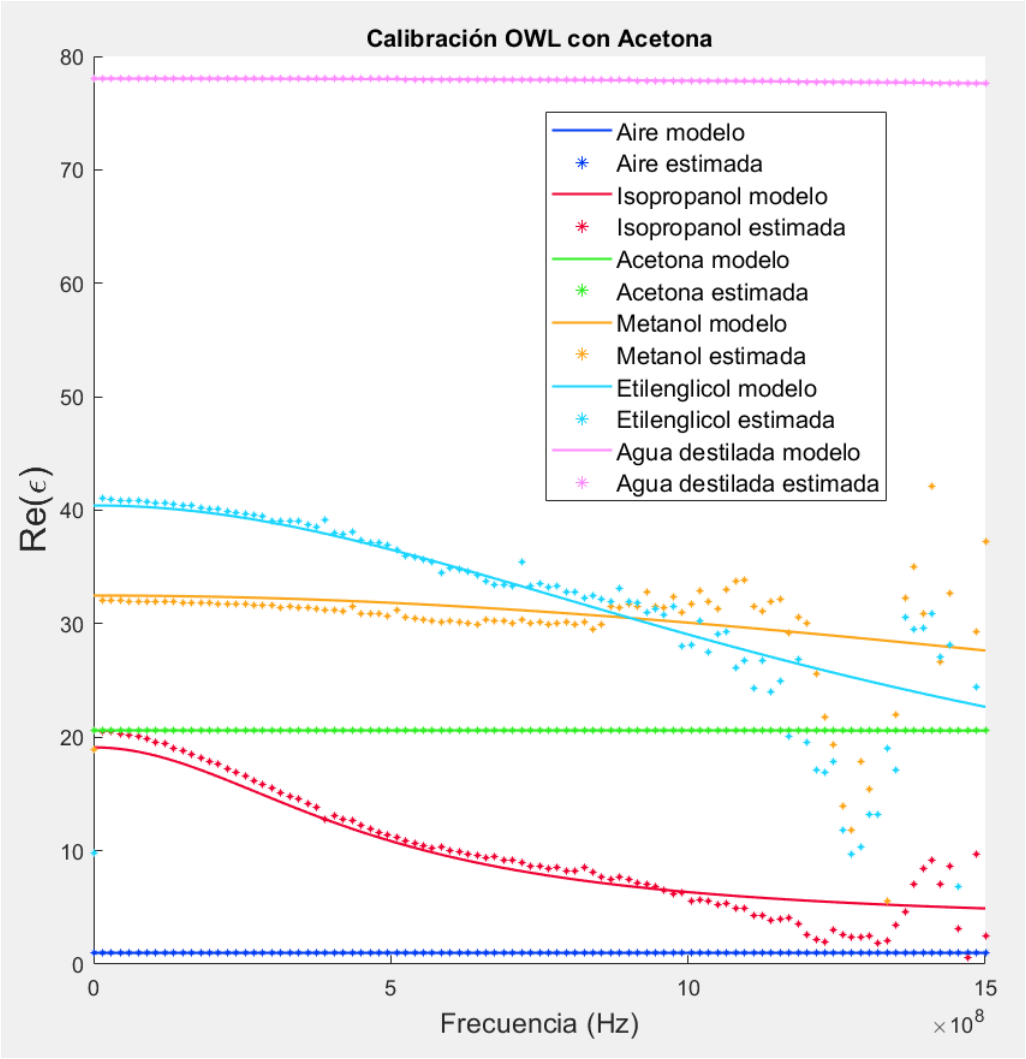


Figura 5.11: Calibración OWL con acetona. VNA diseñado.

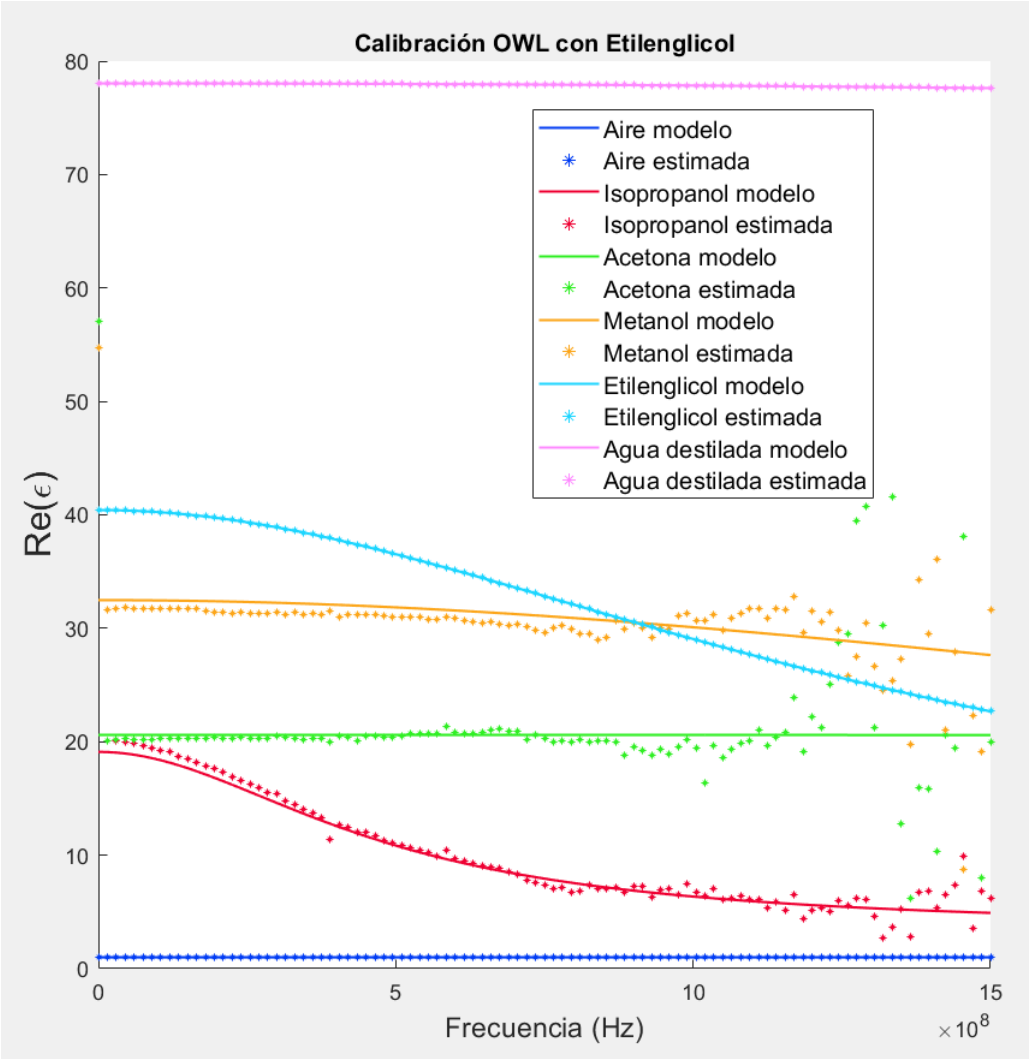


Figura 5.12: Calibración OWL con etilenglicol. VNA diseñado.

La precisión obtenida empleando la calibración OWL, al igual que el caso anterior, es alta hasta una frecuencia de 1 GHz, aproximadamente y, de la misma manera, esto puede ser debido a una limitación de frecuencia del algoritmo o a la capacidad del VNA diseñado de proporcionar una mejor precisión a partir de esas frecuencias. También puede ser debido a la longitud eléctrica equivalente de la sonda, que podría dar lugar a resonancias en ese ancho de banda. Si se seleccionan los datos obtenidos tras la calibración con acetona hasta 1 GHz, los cálculos se aproximan bastante a los modelos de la permitividad. Esto se puede observar gráficamente en la Figura 5.13.

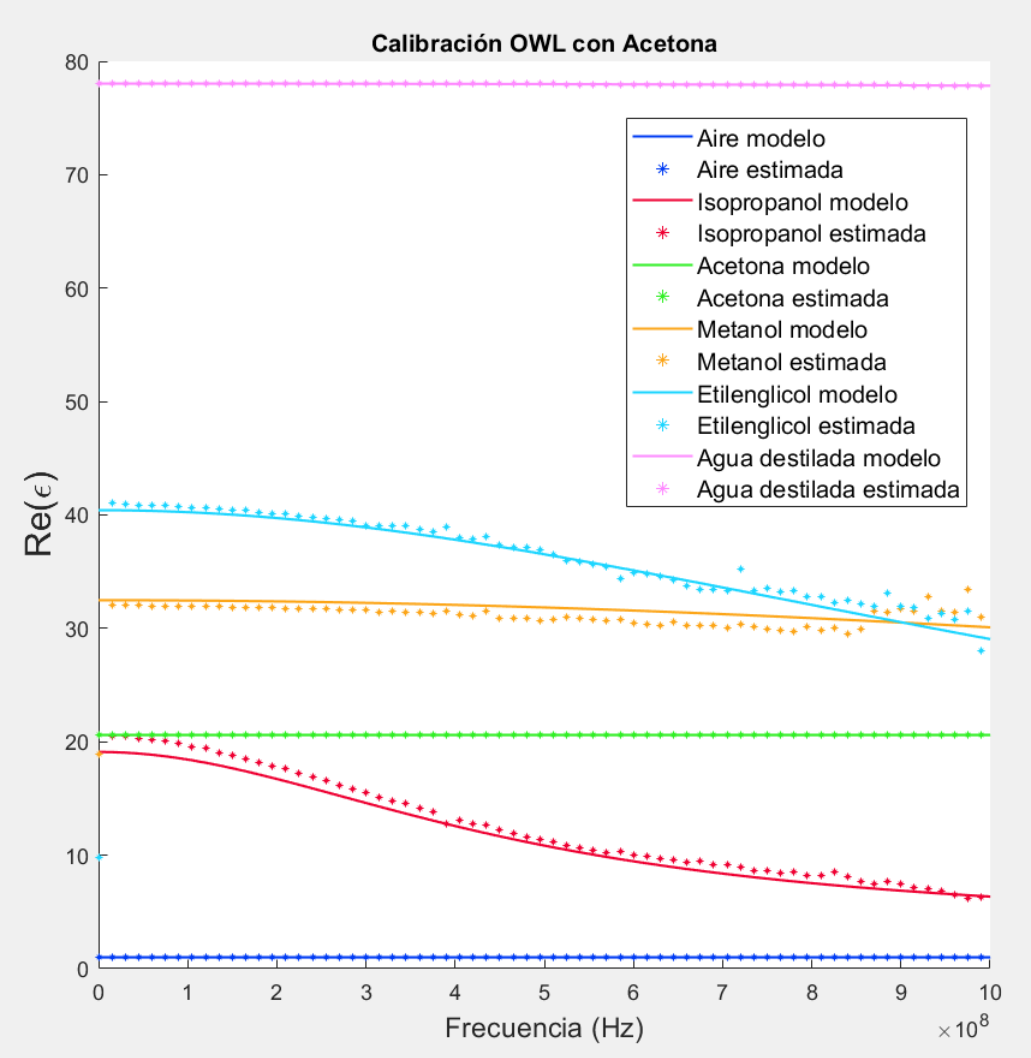


Figura 5.13: Calibración OWL con acetona hasta 1 GHz. VNA diseñado.

5.2.2.1. Error en la calibración OWL del VNA diseñado

Después de haber realizado la calibración OWL con metanol, isopropanol, acetona y etilenglicol, se comprueba cuál ha sido la calibración que menor error ha generado, considerando el valor calculado del error $RMSE$, el cual se recoge en la Tabla 5.2 para cada calibración realizada. En la tabla se puede observar que la calibración que se corresponde con el menor error generado ha sido la calibración con acetona, al igual que el caso anterior, por lo que se usarán las constantes c_1, c_2 y c_3 obtenidas durante su calibración para ampliar el código del Datalogger, de forma que pueda calcular la permitividad a partir del coeficiente de reflexión. Igualmente, el error se ha calculado teniendo en cuenta todo el ancho de banda, desde los 10 kHz hasta los 1,5 GHz.

Tabla 5.2: Errores $RMSE$ durante la calibración OWL con los distintos líquidos para el VNA diseñado.

Líquido empleado para la calibración OWL	**RMSE**
Metanol	11,883
Isopropanol	10,977
Acetona	6,888
Etilenglicol	23,340

5.3. Ampliación del código del Datalogger

Partiendo de los resultados del apartado anterior, para el VNA diseñado, se ha obtenido el menor error $RMSE$ mediante la calibración OWL con acetona. Por tanto, se ha realizado la ampliación del código en MicroPython del Datalogger incluyendo el valor de las constantes c_1, c_2 y c_3 obtenidas durante esta calibración para todas las frecuencias, desde los 10 kHz hasta los 1,5 GHz. De esta manera, el Datalogger podrá calcular la permitividad a partir del coeficiente de reflexión medido por el VNA diseñado mediante el uso de la ecuación (4.5). El valor de las constantes complejas c_1, c_2 y c_3 se recoge en la Tabla 5.3, y el código del Datalogger completo está adjunto en el Anexo V.2, como ya se indicó.

Tabla 5.3: Constantes c_1, c_2 y c_3 introducidas en el código del Datalogger para el cálculo de la permitividad.

$\mathbf{c_1}$	$\mathbf{c_2}$	$\mathbf{c_3}$
$-1{,}69278558689803$ $-7{,}32078856888990j$	$-1{,}79085140212646$ $-7{,}48797144948293j$	$1{,}00218193003791$ $-0{,}007325145329137j$
$-83{,}9288077320716$ $-867{,}505945311766j$	$-137{,}771022786579$ $-860{,}574782924139j$	$-0{,}847010053224312$ $+0{,}227391504204256j$
$13{,}1600190538397$ $-460{,}703512871530j$	$-44{,}0544740727128$ $-458{,}847313400280j$	$-1{,}00080374975980$ $+0{,}0712928648190214j$
$17{,}0348682903276$ $-313{,}725220028775j$	$-41{,}4303752298404$ $-311{,}760363036411j$	$-1{,}06318919128172$ $+0{,}108987050480106j$
$2{,}08772092627441$ $-234{,}079430155553j$	$-55{,}3592985558708$ $-228{,}007336455912j$	$-0{,}988830310605097$ $+0{,}250557380212901j$
$-1{,}69952238096380$ $-183{,}185725497395j$	$-57{,}2007744251372$ $-174{,}370160022090j$	$-0{,}921080530784791$ $+0{,}315997159945976j$
$0{,}921849703809841$ $-153{,}684611145884j$	$-55{,}5016538184971$ $-143{,}639039986244j$	$-0{,}952853228451252$ $+0{,}365242607474260j$
$1{,}61288519498791$ $-131{,}259635536709j$	$-54{,}2301539199865$ $-120{,}174161136474j$	$-0{,}925708243335361$ $+0{,}417460310500782j$
$0{,}846658627160821$ $-114{,}665360563608j$	$-54{,}0930321376740$ $-101{,}543157312076j$	$-0{,}885675142180305$ $+0{,}480964079300289j$
$0{,}0648585200422218$ $-102{,}543383588138j$	$-54{,}4031270590194$ $-87{,}2521798488465j$	$-0{,}848243994239404$ $+0{,}550268530116380j$
$-0{,}129280593973334$ $-92{,}0842708670848j$	$-53{,}8049284547317$ $-75{,}1347046006361j$	$-0{,}808778008325127$ $+0{,}603957730794162j$

$-0{,}508084848503870$ $-83{,}4768911975476j$	$-53{,}2103241540267$ $-64{,}7925995000244j$	$-0{,}757814587762305$ $+0{,}656218751846441j$
$-0{,}608686258688195$ $-75{,}8322365039702j$	$-51{,}7371803970695$ $-55{,}6130444021835j$	$-0{,}711955408739971$ $+0{,}692319892839584j$
$-0{,}489719025877385$ $-70{,}6280593023714j$	$-51{,}1418387345553$ $-48{,}7718426002589j$	$-0{,}668523306604103$ $+0{,}746520074703764j$
$-0{,}729642178889669$ $-65{,}6771672687686j$	$-50{,}4993522334090$ $-42{,}1850264321947j$	$-0{,}617619183044899$ $+0{,}789817797992800j$
$-1{,}13901728509071$ $-60{,}9778496622192j$	$-49{,}4080549290103$ $-35{,}6946493342460j$	$-0{,}560846594500899$ $+0{,}818425746393837j$
$-0{,}842473720520516$ $-57{,}3217004352436j$	$-48{,}4173917306460$ $-30{,}8081481456223j$	$-0{,}511777086774004$ $+0{,}858747688101709j$
$-1{,}03419161918396$ $-54{,}1134951362127j$	$-47{,}5459740168053$ $-25{,}9376570483382j$	$-0{,}450152112471841$ $+0{,}891203548138251j$
$-1{,}19290573074744$ $-50{,}8666063640035j$	$-46{,}2114510514759$ $-21{,}3419193039304j$	$-0{,}389823202417408$ $+0{,}912934190928087j$
$-0{,}999704785920748$ $-48{,}2459530520609j$	$-45{,}0064049950353$ $-17{,}7029251534522j$	$-0{,}330828820764208$ $+0{,}940992654938144j$
$0{,}135235119198668$ $-46{,}7173140167968j$	$-43{,}3413366175389$ $-16{,}1000832931407j$	$-0{,}294148748025757$ $+0{,}966586335526108j$
$-0{,}930940181155162$ $-43{,}5596343921505j$	$-41{,}5361027919905$ $-11{,}4356689358676j$	$-0{,}209107691738486$ $+0{,}944278761804964j$
$-0{,}850572186287369$ $-41{,}5944940245335j$	$-40{,}2246299895751$ $-8{,}71430710867175j$	$-0{,}152909970475220$ $+0{,}967843928381929j$
$-1{,}94004904549613$ $-39{,}4588741924704j$	$-38{,}6884845663263$ $-4{,}92083368625007j$	$-0{,}0690855541823011$ $+0{,}950016974332349j$
$-1{,}59353649746416$ $-38{,}0230158486680j$	$-37{,}2343744485437$ $-2{,}71197283657359j$	$-0{,}0211239657041125$ $+0{,}965791923087891j$
$-1{,}66718751648925$ $-36{,}2231078244920j$	$-35{,}5411912601769$ $-0{,}260760125395848j$	$0{,}0388465691492020$ $+0{,}963146181204751j$
$2{,}04308632648365$ $-32{,}0083156925058j$	$-31{,}0217593731586$ $-2{,}88916400055098j$	$-0{,}0449750409705570$ $+1{,}00525075710192j$
$-2{,}19845335248838$ $-33{,}3353407578179j$	$-32{,}4190743573485$ $+3{,}98764531188858j$	$0{,}163808283583220$ $+0{,}945237612906646j$
$-2{,}61787299763066$ $-32{,}2043515252994j$	$-30{,}7799626733388$ $+5{,}94059060799104j$	$0{,}223969651603002$ $+0{,}926211186633641j$
$-2{,}58660850009014$ $-31{,}3473795472861j$	$-29{,}6410740648150$ $+7{,}60975406782565j$	$0{,}280369838998358$ $+0{,}909425706816795j$

$-2{,}29470244506033$ $-30{,}1973134420441j$	$-28{,}1243939246566$ $+8{,}63721565141450j$	$0{,}331212162000316$ $+0{,}917417602384866j$
$-2{,}57002592907881$ $-28{,}8188486264782j$	$-26{,}4813240505565$ $+10{,}1150955526356j$	$0{,}396019539360332$ $+0{,}890697417229885j$
$-3{,}05036823952861$ $-28{,}2068961641553j$	$-25{,}1444429745730$ $+11{,}7606304994880j$	$0{,}467807758252665$ $+0{,}856192810849891j$
$-3{,}00018114777014$ $-27{,}0119114953472j$	$-23{,}4955645498995$ $+12{,}6972207148635j$	$0{,}522205952303857$ $+0{,}823663515338345j$
$-3{,}12857697680783$ $-25{,}9750581840515j$	$-21{,}7491432924869$ $+13{,}8545660944021j$	$0{,}572599285630087$ $+0{,}777551997155809j$
$-3{,}39573712879821$ $-24{,}5206575298782j$	$-19{,}4558813558729$ $+14{,}5683862724584j$	$0{,}614291763525577$ $+0{,}721066354861456j$
$-3{,}42058416467723$ $-23{,}6595966954278j$	$-17{,}8893425439124$ $+15{,}3254986409336j$	$0{,}661568500782271$ $+0{,}678237908842138j$
$-3{,}48936156503186$ $-22{,}8288268150156j$	$-16{,}2704368625024$ $+15{,}8751750928213j$	$0{,}710511409385768$ $+0{,}623335574351898j$
$-3{,}43267605325420$ $-22{,}1946917913760j$	$-14{,}7776813416331$ $+16{,}6666728405710j$	$0{,}751424874631388$ $+0{,}569091494928307j$
$-3{,}70083331479136$ $-21{,}7685053610642j$	$-13{,}3563339635757$ $+17{,}5505255313216j$	$0{,}799758961560533$ $+0{,}472859910185284j$
$-3{,}62139055766100$ $-21{,}2693105439848j$	$-11{,}8094393570063$ $+17{,}9313653401593j$	$0{,}821781188905111$ $+0{,}444756829564047j$
$-3{,}48914316509254$ $-20{,}6632108036201j$	$-10{,}4557652581060$ $+18{,}0519652602817j$	$0{,}841222048528141$ $+0{,}386057328286968j$
$-3{,}50117003939603$ $-20{,}2896333074339j$	$-9{,}05418773291735$ $+18{,}4256668242790j$	$0{,}864976874270084$ $+0{,}321317206683485j$
$-3{,}53441698654052$ $-19{,}7614967374633j$	$-7{,}64816476291221$ $+18{,}5426422645816j$	$0{,}882227364826000$ $+0{,}257096711234196j$
$-3{,}82053253082305$ $-19{,}2449022121610j$	$-6{,}01074040618900$ $+18{,}6948287605092j$	$0{,}889559016737744$ $+0{,}182643895165642j$
$-4{,}15371797704654$ $-19{,}0998365417229j$	$-4{,}43402108658460$ $+19{,}0473949030269j$	$0{,}885856120704003$ $+0{,}0994822058361545j$
$-4{,}17688292268593$ $-18{,}5829880274276j$	$-2{,}95668123910334$ $+18{,}7750595758736j$	$0{,}885919031837848$ $+0{,}0417915535295876j$
$-4{,}14455325653577$ $-18{,}0266385829330j$	$-1{,}70440523178456$ $+18{,}4176205876028j$	$0{,}865760933377515$ $-0{,}009665839628095j$
$-4{,}35216415002429$ $-17{,}3915511886543j$	$-0{,}157348320236487$ $+17{,}9697596074150j$	$0{,}834836249543764$ $-0{,}0643887683129762j$

$-4{,}55154612181866$ $-17{,}2104034010992j$	$1{,}21183565468842$ $+17{,}7582825407304j$	$0{,}827854462683665$ $-0{,}125831375076595j$
$-4{,}57359873691478$ $-16{,}6186684272123j$	$2{,}25189783972764$ $+17{,}1492416009997j$	$0{,}811799087135023$ $-0{,}163168385440999j$
$-4{,}57830427880674$ $-16{,}0452783351733j$	$3{,}32006291244287$ $+16{,}4381661149146j$	$0{,}797614386691524$ $-0{,}205210181476714j$
$-4{,}86210082312007$ $-16{,}3660126992123j$	$4{,}79312604881478$ $+16{,}2558057486157j$	$0{,}765466026494276$ $-0{,}272990539596616j$
$-4{,}59701376772005$ $-15{,}6513601701636j$	$5{,}56850596106113$ $+15{,}5275764148532j$	$0{,}744389674842561$ $-0{,}326129798082031j$
$-4{,}56983106302158$ $-15{,}3352257916183j$	$6{,}26436414147366$ $+14{,}8848679517384j$	$0{,}731561792833551$ $-0{,}374341925648769j$
$-3{,}95921229051848$ $-15{,}5256273777522j$	$6{,}75707528997580$ $+15{,}2784867035983j$	$0{,}744227013368981$ $-0{,}415675967078150j$
$-3{,}69340661851199$ $-15{,}4072189121726j$	$7{,}81284462024118$ $+14{,}7234142317781j$	$0{,}726822369383678$ $-0{,}458640297567145j$
$-2{,}91297635764923$ $-14{,}5204164196263j$	$7{,}40583910672447$ $+13{,}6767858828403j$	$0{,}752450738337823$ $-0{,}461857981757774j$
$-2{,}65628643130253$ $-12{,}3594752798833j$	$5{,}83709213386472$ $+12{,}6868024938611j$	$0{,}755087509807581$ $-0{,}452220738057148j$
$-2{,}50544843370877$ $-12{,}7110109655656j$	$6{,}61900558981561$ $+12{,}5936300680111j$	$0{,}732872242274808$ $-0{,}493531057379066j$
$-2{,}74985178116037$ $-12{,}0001085445240j$	$7{,}15293437890742$ $+11{,}5568669364562j$	$0{,}701482220802574$ $-0{,}515730036140174j$
$-2{,}56466804674436$ $-11{,}5435070129712j$	$7{,}32579351673043$ $+10{,}7144454485984j$	$0{,}678509683638261$ $-0{,}533185555919474j$
$-3{,}09575838531658$ $-10{,}9376500288324j$	$7{,}94905180920610$ $+9{,}41130828988079j$	$0{,}613571831483808$ $-0{,}536355895024802j$
$-2{,}73403302456613$ $-10{,}6035611457968j$	$7{,}66881986886830$ $+9{,}07520647391573j$	$0{,}606765893026886$ $-0{,}574912503232810j$
$-2{,}95696802706124$ $-10{,}0107052048201j$	$7{,}98668405014531$ $+7{,}85850574712893j$	$0{,}583127551255861$ $-0{,}590300597654352j$
$-2{,}21633975796519$ $-9{,}60179176520526j$	$7{,}67568375049459$ $+7{,}35069957478452j$	$0{,}568323052238377$ $-0{,}612422689721290j$
$-2{,}07525240197926$ $-9{,}79878603448475j$	$8{,}03525605710823$ $+6{,}71780137321238j$	$0{,}536724011147663$ $-0{,}656632379062454j$
$-2{,}21380628510467$ $-8{,}81827645042891j$	$7{,}29456570625680$ $+5{,}49767578904770j$	$0{,}510982024876819$ $-0{,}641849701861813j$

$-2{,}60140294613383$ $-8{,}83499007985952j$	$8{,}12332212024471$ $+4{,}66121375754579j$	$0{,}428946372653173$ $-0{,}656380146188602j$
$-2{,}75484527512829$ $-9{,}05799531582912j$	$8{,}58918125613724$ $+3{,}78982118573992j$	$0{,}340953079786751$ $-0{,}709670538155653j$
$-2{,}76908614204341$ $-8{,}79028921987965j$	$8{,}49460323381956$ $+3{,}12382213748046j$	$0{,}299053530116777$ $-0{,}734001100660474j$
$-2{,}17963118949383$ $-8{,}95593862825690j$	$8{,}48697047822142$ $+3{,}10919001107698j$	$0{,}264132908265298$ $-0{,}770102589934800j$
$-2{,}52776958985582$ $-8{,}63029512993491j$	$8{,}15845008878802$ $+1{,}86249543487744j$	$0{,}232396705932837$ $-0{,}759622573272546j$
$-1{,}80051359881780$ $-8{,}79288928864775j$	$8{,}13342423159763$ $+1{,}66218420541451j$	$0{,}198767003372969$ $-0{,}795901706311812j$
$-2{,}31975270367181$ $-8{,}12244268927813j$	$7{,}54094322540710$ $+0{,}562119784805327j$	$0{,}164170772865714$ $-0{,}736820578129763j$
$-2{,}20889919308858$ $-8{,}04152387895250j$	$7{,}48390841817735$ $-0{,}187153980538412j$	$0{,}0867139243766885$ $-0{,}752705631906620j$
$-2{,}90258373966401$ $-8{,}10055027812730j$	$7{,}16098309872426$ $-1{,}12868641738120j$	$0{,}0185777129216143$ $-0{,}774145387980639j$
$-2{,}67246570396692$ $-8{,}43729021459523j$	$7{,}32299173191656$ $-1{,}48555932482908j$	$-0{,}0467194351780270$ $-0{,}778261948097659j$
$-3{,}26421853263738$ $-7{,}92178706366832j$	$6{,}73165206394902$ $-2{,}50507274248062j$	$-0{,}135272038459249$ $-0{,}731298736645041j$
$-3{,}12849231347870$ $-7{,}85064393454400j$	$6{,}23436990534521$ $-2{,}74988028106882j$	$-0{,}122723369373488$ $-0{,}742487735720120j$
$-3{,}11286969327294$ $-6{,}58130084355369j$	$4{,}86327565417353$ $-3{,}03358965332426j$	$-0{,}0999846854755972$ $-0{,}692838065325815j$
$-2{,}10995157942624$ $-5{,}46721147074420j$	$4{,}03095291194681$ $-2{,}17784394175509j$	$-0{,}0596487525019280$ $-0{,}781140515546858j$
$-1{,}97987275893466$ $-5{,}12480229837866j$	$3{,}60095136231878$ $-2{,}08156701744526j$	$-0{,}102879244145643$ $-0{,}815845275574224j$
$-0{,}972071648164580$ $-5{,}44682747032566j$	$4{,}31992248703115$ $-1{,}63461338399662j$	$-0{,}208392296760385$ $-0{,}850196055773570j$
$-1{,}02458494017768$ $-4{,}54368563507561j$	$3{,}50306710924899$ $-1{,}62444240500234j$	$-0{,}264534471678079$ $-0{,}838481463954136j$
$-0{,}811108502609516$ $-3{,}71020806548881j$	$2{,}74331533757694$ $-1{,}30069091742669j$	$-0{,}238437517951559$ $-0{,}773561645689788j$
$-1{,}16746653171871$ $-3{,}21731005524726j$	$2{,}29462951118920$ $-1{,}48526738029115j$	$-0{,}324035622738556$ $-0{,}668797092583347j$

$-0{,}917917387629233$ $-3{,}49397930733674j$	$2{,}32894291103622$ $-1{,}78159315860355j$	$-0{,}323265022050557$ $-0{,}630458390208569j$
$-0{,}951579202382344$ $-1{,}80960147707951j$	$1{,}33370677980865$ $-0{,}620260571891096j$	$-0{,}326806817422356$ $-0{,}635964979406297j$
$0{,}270979695232162$ $+0{,}512216772069746j$	$0{,}896765955217786$ $+1{,}80472882986470j$	$-0{,}228890561605064$ $-0{,}739025267251832j$
$2{,}17117139466353$ $+2{,}33030321702346j$	$1{,}73243484656967$ $+4{,}05119206927465j$	$-0{,}231920869400852$ $-0{,}768995003860186j$
$3{,}85553571064121$ $+1{,}31810602717587j$	$3{,}71771640963222$ $+3{,}77547553677974j$	$-0{,}355793975367884$ $-0{,}694401164770779j$
$7{,}38641548746293$ $+0{,}858374378784431j$	$7{,}15747774850470$ $+4{,}60341178261907j$	$-0{,}454874416142366$ $-0{,}743768120804538j$
$9{,}17673882914708$ $-0{,}804142711257435j$	$9{,}72866650966487$ $+3{,}47252620664509j$	$-0{,}556902898240804$ $-0{,}771306683033608j$
$7{,}12259099819431$ $-3{,}84859357100227j$	$8{,}87222839679363$ $-0{,}346111145338551j$	$-0{,}435195681309770$ $-0{,}585282637072333j$
$4{,}75610496236057$ $-4{,}82176979065465j$	$7{,}14160627798468$ $-2{,}02392877214963j$	$-0{,}545842336636727$ $-0{,}557922432682316j$
$7{,}00112486563618$ $-1{,}61098788969930j$	$8{,}42512137762821$ $+1{,}00714799754689j$	$-0{,}478216242036665$ $-0{,}621475949755697j$
$0{,}460337111042593$ $-0{,}262072502754800j$	$1{,}93651308294160$ $+0{,}492026137174233j$	$-0{,}407277782390163$ $-0{,}476794901446108j$
$-1{,}65093631707250$ $-1{,}43272540349731j$	$0{,}00174465230759679$ $-1{,}10708328106067j$	$-0{,}391041621120480$ $-0{,}451118657101884j$
$-0{,}907112515505215$ $-7{,}67023640399875j$	$1{,}36662694849404$ $-7{,}28219172009737j$	$-0{,}497216865479225$ $-0{,}162197271982651j$
$-2{,}18787357371833$ $-3{,}73221677781250j$	$-0{,}344652550226041$ $-3{,}37381324421837j$	$-0{,}648533024664051$ $-0{,}254126510220367j$

5.4. Ensayo con el sistema completo

Con todo el sistema completo, se ha procedido a realizar la medición del coeficiente de reflexión del isopropanol con el VNA diseñado, a la vez que se registraban los datos del S_{11} en el Datalogger. Tras el procesamiento del mismo, se han obtenido los valores del coeficiente de reflexión y de la permitividad del isopropanol en ficheros de texto, tal y como se observa en la Figura 5.14. Finalmente, se ha procedido a graficar los datos, y se pueden ver los resultados en las Figuras 5.15 y 5.16, los cuales son curvas que se asemejan mucho a los valores correspondientes para dicho líquido. En el caso de la permitividad, como ya se ha mencionado, aparece una desviación a muy alta frecuencia, pero el sistema es bastante preciso hasta más de 1 GHz.

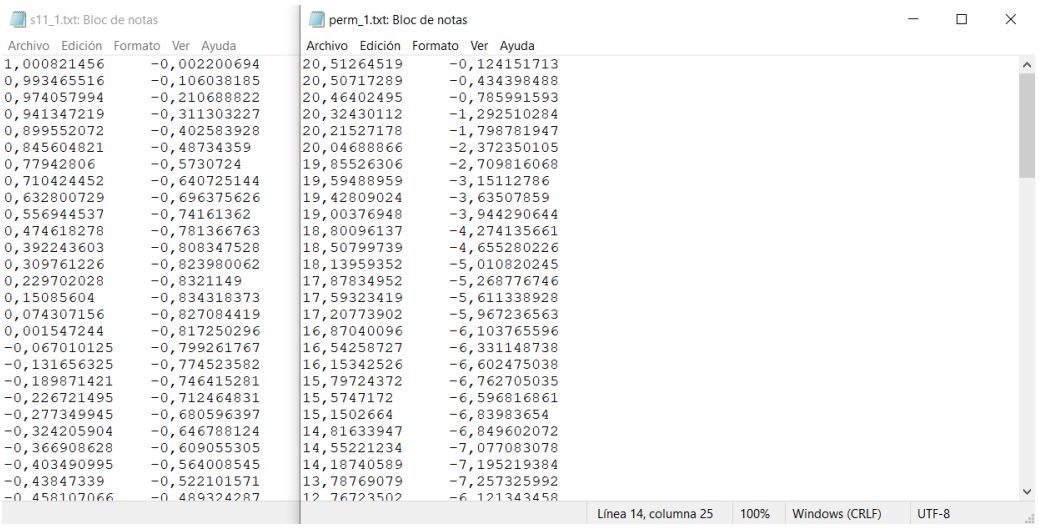


Figura 5.14: Ficheros de texto con el valor de la parte real e imaginaria del coeficiente de reflexión y de la permitividad, generados por el Datalogger.

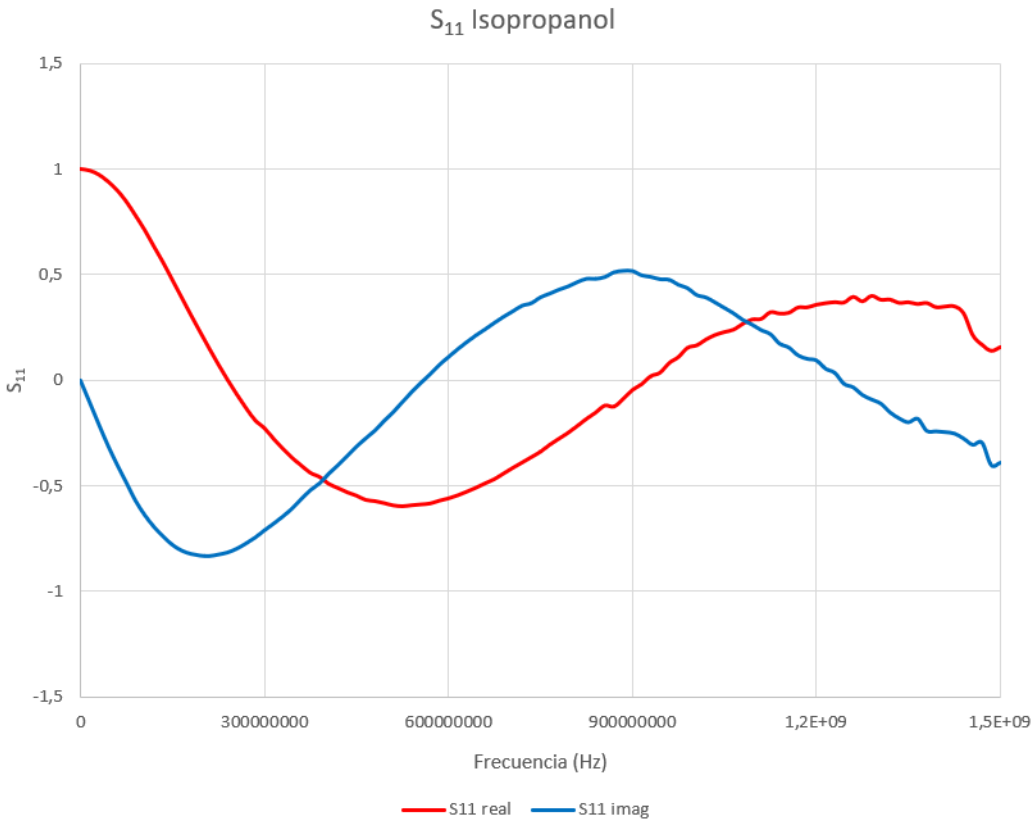


Figura 5.15: Coeficiente de reflexión del isopropanol con los datos del fichero de texto del Datalogger.

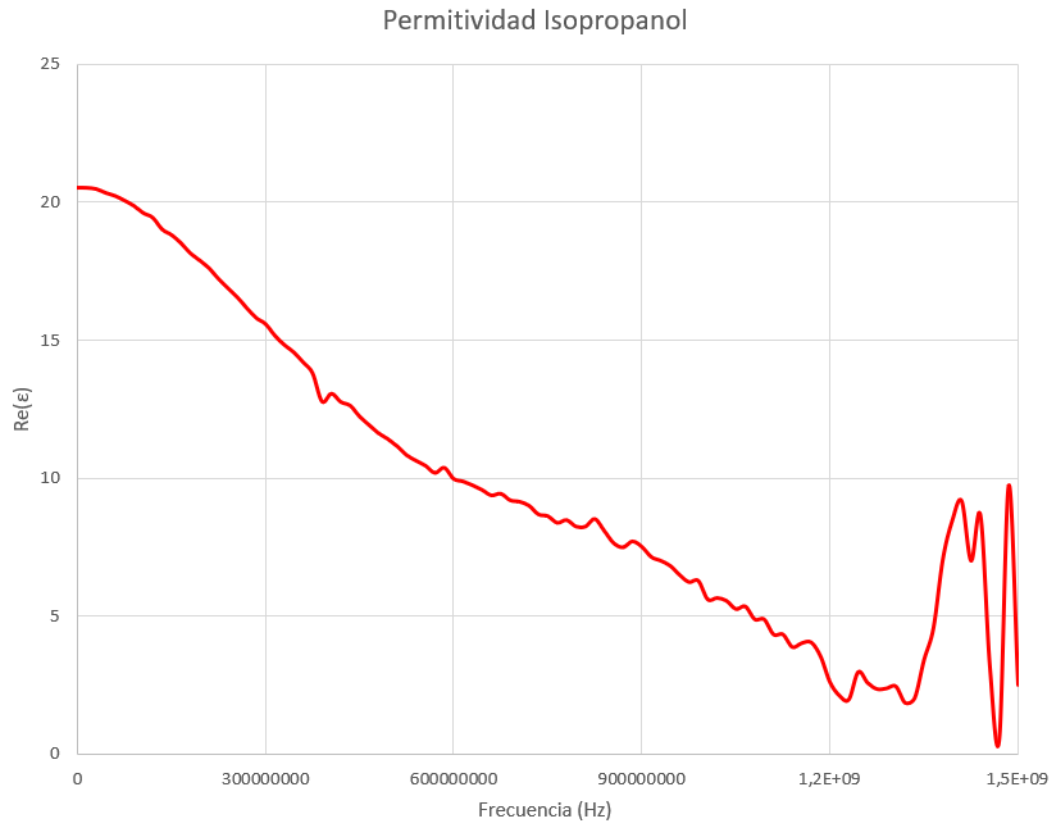


Figura 5.16: Permitividad del isopropanol con los datos del fichero de texto del Datalogger.

Capítulo 6

Conclusiones y trabajos futuros

Los objetivos principales del presente proyecto eran realizar el diseño y la fabricación de un analizador de redes vectorial con un tamaño y coste reducidos que ofreciera una buena calidad y precisión en la medida, así como la capacidad de realizar su calibración. Además, este VNA se utilizaría para la caracterización de medios dieléctricos, obteniendo la permitividad de los mismos a partir de su coeficiente de reflexión. Como objetivos particulares, se establecían realizar un estudio de la arquitectura y funcionamiento de un analizador de redes vectorial, desarrollar el diseño electrónico del VNA utilizando herramientas de diseño de PCB, así como la programación de su microcontrolador, realizar el diseño y fabricación de un Datalogger basado en una ESP32 que registre los datos medidos por el VNA a través de UART, implementar un sistema de comunicación SDI-12 para el intercambio de datos con este protocolo, adquirir todos lo componentes electrónicos para la fabricación del prototipo y realizar ensayos con el VNA diseñado para validar su funcionamiento en la medida de la permitividad, extendiendo su calibración a distintos líquidos.

En este sentido, el diseño de un analizador de redes vectorial de tamaño reducido que ofrece alta precisión de medida se ha realizado. Su coste, teniendo en cuenta todos los componentes que lo forman, es inferior a 17 €, lo cual se trata de un precio muy reducido considerando el precio de VNA comerciales, como el modelo Agilent 4395A, que realiza las mediciones hasta 500 MHz y tiene un coste de alrededor de 5.000 €. Por otra parte, a pesar no haber dispuesto de la tecnología suficiente para realizar el montaje de la PCB correctamente, sí se ha podido comprobar el diseño completo durante

los ensayos llevados a cabo, partiendo de un NanoVNA [42] que se ha reducido hasta dejarlo a la mínima expresión, obteniendo el diseño aquí realizado, y todos los bloques que lo componen funcionan correctamente en conjunto, haciendo que el sistema que forma el VNA diseñado efectúe adecuadamente las mediciones a la hora de realizar la calibración y a la hora de realizar las medidas sobre los distintos materiales de los ensayos. De esta forma, se cumple con los objetivos principales del trabajo.

Del mismo modo, antes de comenzar con el diseño del VNA, se ha realizado un estudio previo acerca de la arquitectura y funcionamiento de un VNA, con el objetivo de lograr hacer un diseño que redujera al máximo el número de componentes necesario para lograr su correcto funcionamiento. Además, a la hora de realizar el diseño, se ha utilizado la herramienta de diseño de PCB Altium Designer, que ha permitido diseñar el esquemático y, a partir de ahí, construir la PCB, incluyendo todos los componentes fundamentales para su funcionamiento. Asimismo, se ha realizado la programación del microcontrolador STM32F072 en lenguaje C, empleando el sistema operativo en tiempo real ChibiOS, de forma que gestiona todo el sistema para que el VNA funcione de forma correcta.

Adicionalmente, se ha realizado el diseño del esquemático y la PCB de un Datalogger basado en una ESP32 mediante el uso de Altium Designer, incluyendo la posibilidad de poder registrar los datos medidos por el VNA en un barrido de frecuencias y de poder realizar la calibración OSL, todo esto mediante una comunicación UART con el VNA. También se ha incluido un adaptador USB/UART para realizar la comunicación fácilmente desde un ordenador y un circuito adaptador UART/SDI-12 para exportar los datos desde la ESP32 hacia un maestro SDI-12 que emplee este mismo estándar. Para que todo esto funcione, se ha programado la ESP32 en MicroPython, incluyendo la parte de comunicación con el VNA y de comunicación a través de SDI-12. No obstante, como ya se ha indicado en su correspondiente apartado, por falta de tiempo a la hora de pedir los componentes y la PCB física, no se ha podido construir profesionalmente, pero sí se ha probado el diseño funcional mediante el montaje del circuito en una placa de prototipado rápido, corroborando que su funcionamiento es correcto.

Por último, a la hora de realizar los ensayos, se ha puesto en práctica el procedimiento para obtener la permitividad del método de la sonda coaxial de final abierto, obteniendo los datos del coeficiente de reflexión de los distintos materiales a través de las medidas del VNA efectuadas en la punta de dicha sonda. Una vez registrados todos los datos, se ha empleado un código

en MATLAB para realizar la calibración OWL, obteniendo unas constantes que permiten calcular la permitividad del material a partir de su coeficiente de reflexión. Con estos datos ya calculados, se ha incluido una última parte en el código en MicroPython de la ESP32 que realiza estos cálculos, según lo indicado en su correspondiente apartado. De esta forma, el Datalogger registra los datos relativos al coeficiente de reflexión y la permitividad compleja del material bajo estudio, y los exporta en formato de texto para su posterior procesamiento. De este modo, se logra cumplir todos los objetivos propuestos en el presente proyecto.

6.1. Trabajos futuros

Partiendo de las conclusiones que se han obtenido, en esta sección se plantean recomendaciones y líneas de actuación para el desarrollo de trabajos futuros a partir del VNA diseñado en el presente proyecto.

En primer lugar, tras comprobar que los resultados obtenidos después de realizar las mediciones son precisos, queda pendiente el envío del diseño a fabricar con los medios oportunos para poder validar con total certeza el funcionamiento del equipo diseñado. Por otra parte, una línea interesante a desarrollar es la ampliación del rango de frecuencias que es capaz de medir el VNA. Debido a su reducido tamaño y al objetivo de intentar conseguir un coste reducido para lograr una mayor accesibilidad del mismo, los componentes utilizados hacen que la banda de frecuencias y la potencia utilizadas se vean limitadas. Sería interesante lograr mediciones en un rango de frecuencias superior para obtener más características del medio bajo estudio a altas frecuencias y para poder caracterizar otro tipo de materiales, así como los distintos tipos de polarización existentes. También, queda abierta la posibilidad de utilizar la comunicación I^2C a través del puerto I2C incluido en el diseño del VNA para realizar la comunicación con cualquier otro dispositivo para el intercambio de datos, así como el uso del puerto P1 para la programación y depuración del microcontrolador a través de SWD. A su vez, queda pendiente el diseño y definición de un encapsulado y sistema de aislamiento con impresión 3D para proteger la circuitería del VNA y mejorar su portabilidad. Además, tras haber obtenido una buena precisión en las mediciones y los resultados y haber incluido en el código del Datalogger la forma de obtener la permitividad directamente a partir de la medida realizada del coeficiente de reflexión, se pretende llevar a cabo su validación en

campo, generando un gran número de medidas y tomando como referencia analizadores de redes vectoriales comerciales. De la misma forma, queda pendiente el uso más exhaustivo de la comunicación SDI-12 implementada con distintos maestros SDI-12 para validar su funcionamiento.

Asimismo, queda abierta la posibilidad de aplicar el VNA diseñado en laboratorios de biomedicina, con el objetivo de realizar el estudio de las propiedades dieléctricas de los tejidos para la detección de enfermedades como el cáncer, dada la gran comodidad de su uso en laboratorio por su reducido tamaño, además de su coste. Adicionalmente, se plantea la posibilidad de ampliar las funcionalidades del Datalogger, permitiendo que realice una conexión Wi-Fi con la finalidad de que pueda compartir los datos registrados con cualquier parte del mundo a través de Internet. Además, la ESP32 que contiene el Datalogger diseñado ya está preparada para realizar dicha conexión Wi-Fi.

Por último, con el avance reciente de la inteligencia artificial y del *Machine Learning* debido a la alta capacidad de computación actual, se podría realizar el entrenamiento de una red neuronal artificial, con el objetivo de lograr la clasificación del medio bajo estudio a partir de los datos del espectro de su permitividad compleja. Para ello, sería necesario utilizar una gran cantidad de datos de entrada o *dataset*, para lo cual será necesario realizar muchas mediciones de muchos materiales distintos, con el objetivo de que la red aprenda entrenándose con ese conjunto de datos. El entrenamiento se llevaría a cabo en un ordenador que tuviera una alta potencia computacional y que precisara de una GPU (*Graphics Processing Unit*, unidad de procesamiento gráfico) de gran capacidad, utilizando las librerías de TensorFlow y realizando la programación en Python. Tras ello, se exportaría el modelo entrenado para después importarlo en un microcontrolador, siendo necesario el uso de TensorFlow Lite para microcontroladores, y teniendo en cuenta los microcontroladores compatibles con dicha librería.

Anexos

Anexo I

Hojas de características

I.1. VNA

I.1.1. STM32F072C8T6

life.augmented

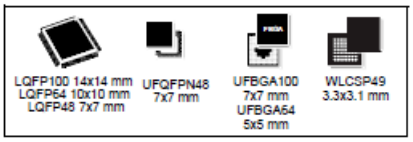
STM32F072x8 STM32F072xB

ARM®-based 32-bit MCU, up to 128 KB Flash, crystal-less USB FS 2.0, CAN, 12 timers, ADC, DAC & comm. interfaces, 2.0 - 3.6 V

Datasheet - production data

Features

- Core: ARM® 32-bit Cortex®-M0 CPU, frequency up to 48 MHz
- Memories
  - 64 to 128 Kbytes of Flash memory
  - 16 Kbytes of SRAM with HW parity
- CRC calculation unit
- Reset and power management
  - Digital and I/O supply: V_{DD} = 2.0 V to 3.6 V
  - Analog supply: V_{DDA} = V_{DD} to 3.6 V
  - Selected I/Os: V_{DDIO2} = 1.65 V to 3.6 V
  - Power-on/Power down reset (POR/PDR)
  - Programmable voltage detector (PVD)
  - Low power modes: Sleep, Stop, Standby
  - V_{BAT} supply for RTC and backup registers
- Clock management
  - 4 to 32 MHz crystal oscillator
  - 32 kHz oscillator for RTC with calibration
  - Internal 8 MHz RC with x6 PLL option
  - Internal 40 kHz RC oscillator
  - Internal 48 MHz oscillator with automatic trimming based on ext. synchronization
- Up to 87 fast I/Os
  - All mappable on external interrupt vectors
  - Up to 68 I/Os with 5V tolerant capability and 19 with independent supply V_{DDIO2}
- Seven-channel DMA controller
- One 12-bit, 1.0 µs ADC (up to 16 channels)
  - Conversion range: 0 to 3.6 V
  - Separate analog supply: 2.4 V to 3.6 V
- One 12-bit D/A converter (with 2 channels)
- Two fast low-power analog comparators with programmable input and output
- Up to 24 capacitive sensing channels for touchkey, linear and rotary touch sensors



- Calendar RTC with alarm and periodic wakeup from Stop/Standby
- 12 timers
  - One 16-bit advanced-control timer for six-channel PWM output
  - One 32-bit and seven 16-bit timers, with up to four IC/OC, OCN, usable for IR control decoding or DAC control
  - Independent and system watchdog timers
  - SysTick timer
- Communication interfaces
  - Two I^2C interfaces supporting Fast Mode Plus (1 Mbit/s) with 20 mA current sink, one supporting SMBus/PMBus and wakeup
  - Four USARTs supporting master synchronous SPI and modem control, two with ISO7816 interface, LIN, IrDA, auto baud rate detection and wakeup feature
  - Two SPIs (18 Mbit/s) with 4 to 16 programmable bit frames, and with I^2S interface multiplexed
  - CAN interface
  - USB 2.0 full-speed interface, able to run from internal 48 MHz oscillator and with BCD and LPM support
- HDMI CEC wakeup on header reception
- Serial wire debug (SWD)
- 96-bit unique ID
- All packages ECOPACK®2

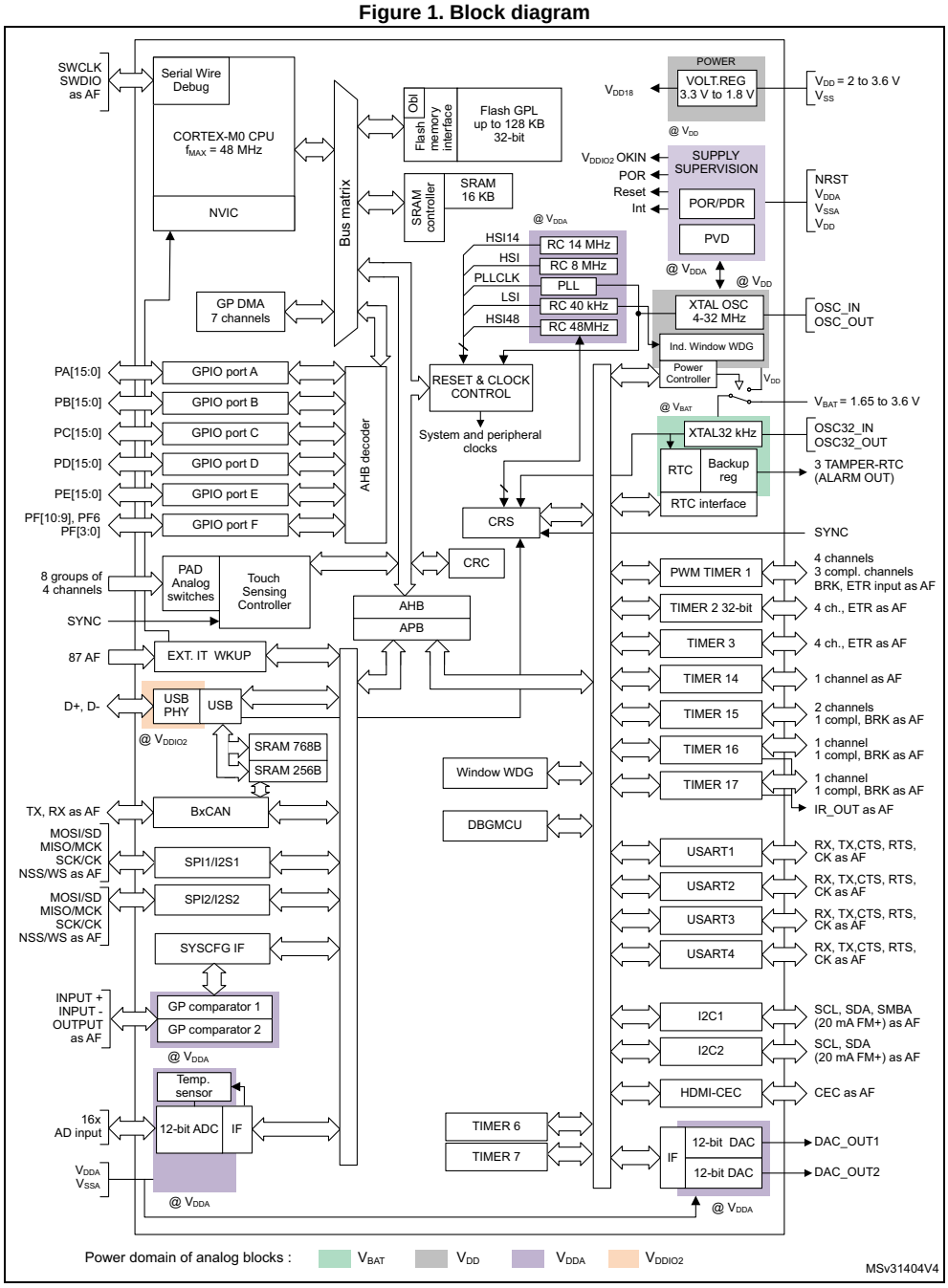
Table 1. Device summary

Reference	Part number
STM32F072x8 STM32F072xB	STM32F072C8, STM32F072R8, STM32F072V8, STM32F072CB, STM32F072RB, STM32F072VB

January 2017 DocID025004 Rev 5 1/128

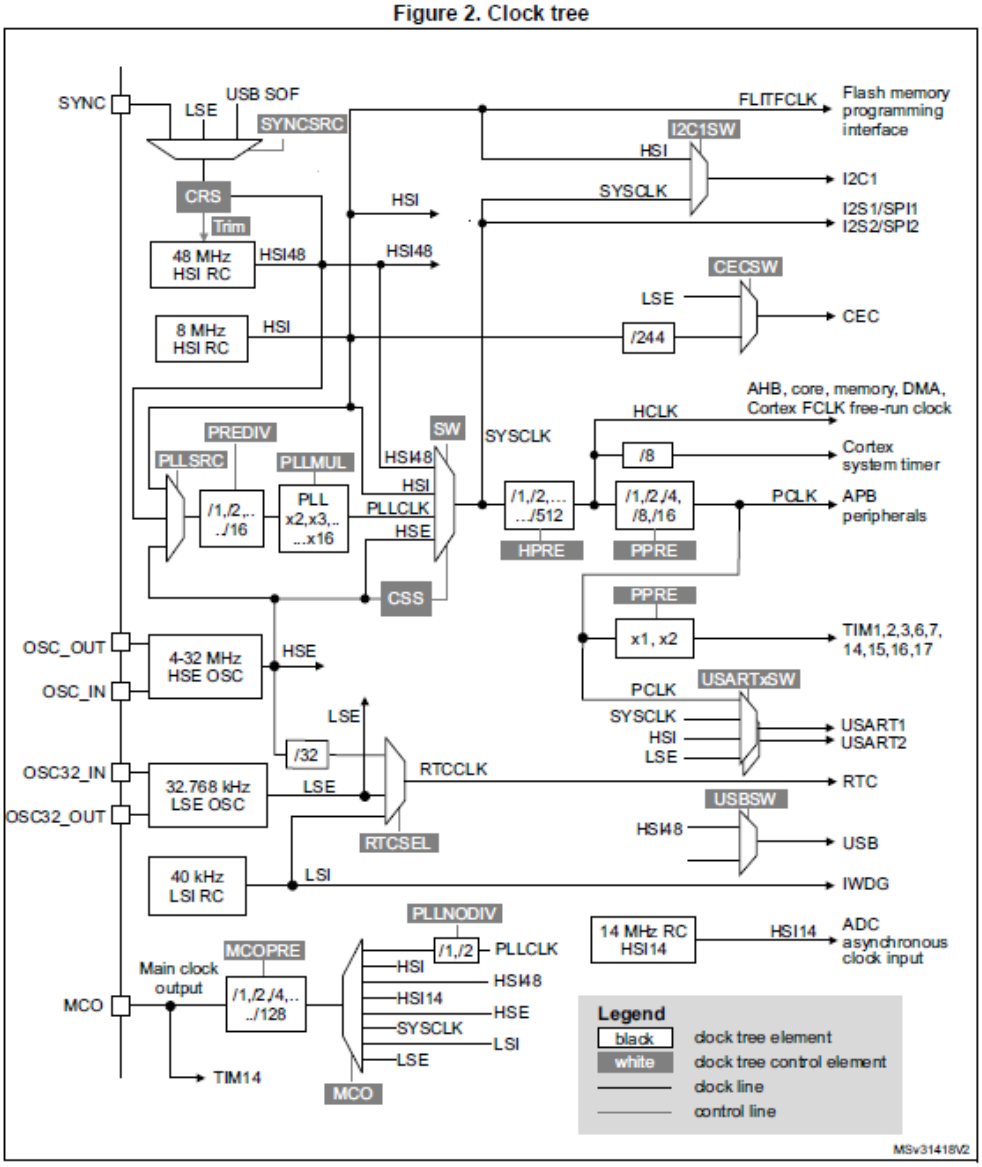
This is information on a product in full production. www.st.com

Figure 1. Block diagram



back to the internal RC oscillator. A software interrupt is generated if enabled. Similarly, full interrupt management of the PLL clock entry is available when necessary (for example on failure of an indirectly used external crystal, resonator or oscillator).

Figure 2. Clock tree



Several prescalers allow the application to configure the frequency of the AHB and the APB domains. The maximum frequency of the AHB and the APB domains is 48 MHz.

Figure 6. LQFP64 package pinout

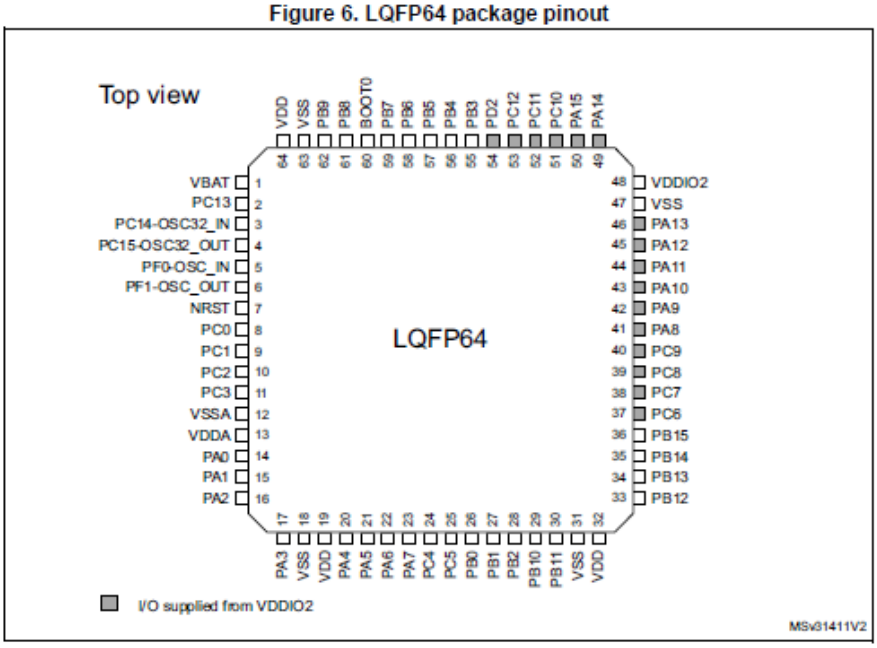
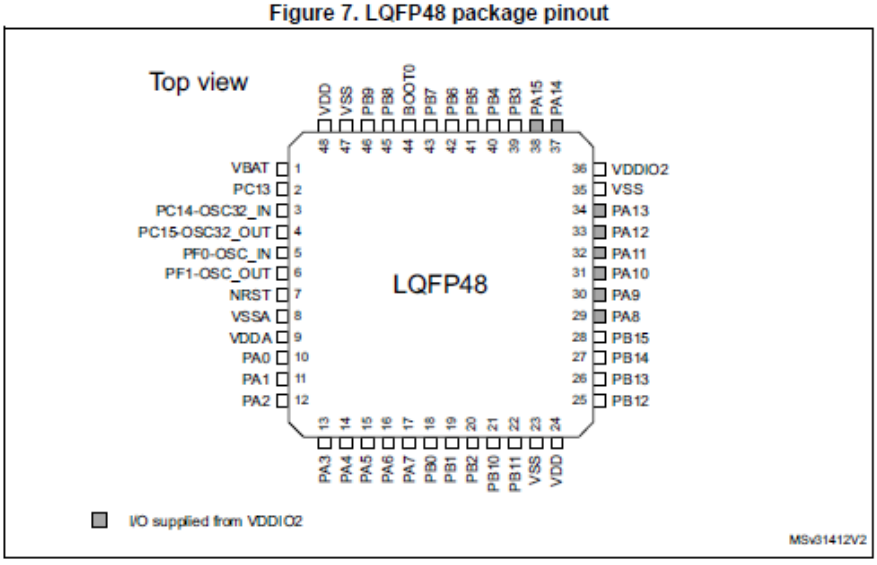


Figure 7. LQFP48 package pinout



I.1.2. Códec TLV320AIC3204

TEXAS INSTRUMENTS

TLV320AIC3204

SLOS602E – SEPTEMBER 2008 – REVISED SEPTEMBER 2019

TLV320AIC3204 Ultra Low Power Stereo Audio Codec

1 Features

- Stereo Audio DAC with 100dB SNR
- 4.1mW Stereo 48ksps DAC Playback
- Stereo Audio ADC with 93dB SNR
- 6.1mW Stereo 48ksps ADC Record
- PowerTune™
- Extensive Signal Processing Options
- Six Single-Ended or 3 Fully-Differential Analog Inputs
- Stereo Analog and Digital Microphone Inputs
- Stereo Headphone Outputs
- Stereo Line Outputs
- Very Low-Noise PGA
- Low Power Analog Bypass Mode
- Programmable Microphone Bias
- Programmable PLL
- Integrated LDO
- 5-mm × 5-mm, 32-Pin VQFN Package

2 Applications

- Portable Navigation Devices (PND)
- Portable Media Player (PMP)
- Mobile Handsets
- Communication
- Portable Computing

3 Description

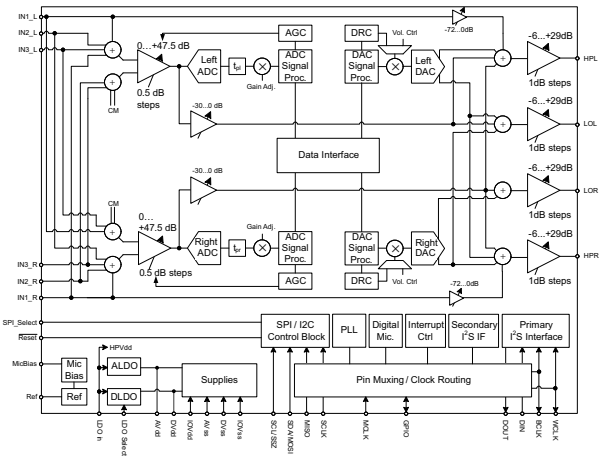
The TLV320AIC3204 (also called the AIC3204) is a flexible, low-power, low-voltage stereo audio codec with programmable inputs and outputs, PowerTune capabilities, fixed predefined and parameterizable signal-processing blocks, integrated PLL, integrated LDOs and flexible digital interfaces.

Device Information(1)

PART NUMBER	PACKAGE	BODY SIZE (NOM)
TLV320AIC3204	VQFN (32)	5.00 mm x 5.00 mm

(1) For all available packages, see the orderable addendum at the end of the datasheet.

Simplified Block Diagram



I.1.3. Si5351A-B-GT

SILICON LABS

Si5351A/B/C-B

I²C-PROGRAMMABLE ANY-FREQUENCY CMOS CLOCK GENERATOR + VCXO

Features

- www.silabs.com/custom-timing
- Generates up to 8 non-integer-related frequencies from 2.5 kHz to 200 MHz
- I²C user definable configuration
- Exact frequency synthesis at each output (0 ppm error)
- Highly linear VCXO
- Optional clock input (CLKIN)
- Low output period jitter: < 70 ps pp, typ
- Configurable spread spectrum selectable at each output
- Operates from a low-cost, fixed frequency crystal: 25 or 27 MHz
- Supports static phase offset
- Programmable rise/fall time control
- Glitchless frequency changes
- Separate voltage supply pins provide level translation:
  - Core VDD: 2.5 or 3.3 V
  - Output VDDO: 1.8, 2.5, or 3.3 V
- Excellent PSRR eliminates external power supply filtering
- Very low power consumption
- Adjustable output delay
- Available in 2 packages types:
- 10-MSOP: 3 outputs
- 20-QFN (4x4 mm): 8 outputs
- PCIE Gen 1 compatible
- Supports HCSL compatible swing

10-MSOP

20-QFN

Ordering Information:
See page 29

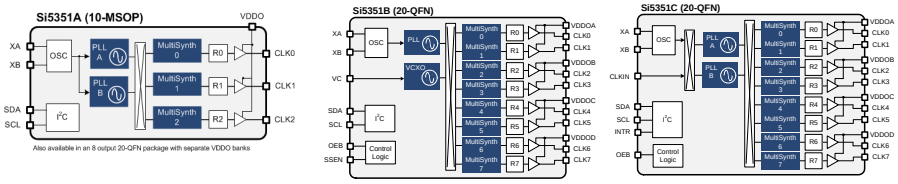
Applications

- HDTV, DVD/Blu-ray, set-top box
- Audio/video equipment, gaming
- Printers, scanners, projectors
- Handheld Instrumentation
- Residential gateways
- Networking/communication
- Servers, storage
- XO replacement

Description

The Si5351 is an I²C configurable clock generator that is ideally suited for replacing crystals, crystal oscillators, VCXOs, phase-locked loops (PLLs), and fanout buffers in cost-sensitive applications. Based on a PLL/VCXO + high resolution MultiSynth fractional divider architecture, the Si5351 can generate any frequency up to 200 MHz on each of its outputs with 0 ppm error. Three versions of the Si5351 are available to meet a wide variety of applications. The Si5351A generates up to 8 free-running clocks using an internal oscillator for replacing crystals and crystal oscillators. The Si5351B adds an internal VCXO and provides the flexibility to replace both free-running clocks and synchronous clocks. It eliminates the need for higher cost, custom pullable crystals while providing reliable operation over a wide tuning range. The Si5351C offers the same flexibility but synchronizes to an external reference clock (CLKIN).

Functional Block Diagram



I.1.4. SA612AD

SA612A

Double-balanced mixer and oscillator

Rev. 3 — 4 June 2014 Product data sheet

1. General description

The SA612A is a low-power VHF monolithic double-balanced mixer with on-board oscillator and voltage regulator. It is intended for low cost, low-power communication systems with signal frequencies to 500 MHz and local oscillator frequencies as high as 200 MHz. The mixer is a 'Gilbert cell' multiplier configuration that provides gain of 14 dB or more at 45 MHz.

The oscillator can be configured for a crystal, a tuned tank operation, or as a buffer for an external LO. Noise figure at 45 MHz is typically below 6 dB and makes the device well-suited for high-performance cordless phone/cellular radio. The low power consumption makes the SA612A excellent for battery-operated equipment. Networking and other communications products can benefit from very low radiated energy levels within systems. The SA612A is available in an 8-lead SO (surface-mounted miniature package).

2. Features and benefits

- Low current consumption
- Low cost
- Operation to 500 MHz
- Low radiated energy
- Low external parts count; suitable for crystal/ceramic filter
- Excellent sensitivity, gain, and noise figure

3. Applications

- Cordless telephone
- Portable radio
- VHF transceivers
- RF data links
- Sonobuoys
- Communications receivers
- Broadband LANs
- HF and VHF frequency conversion
- Cellular radio mixer/oscillator

4. Ordering information

Table 1. Ordering information

Type number	Topside marking	Package		
		Name	Description	Version
SA612AD/01	SA612A	SO8	plastic small outline package; 8 leads; body width 3.9 mm	SOT96-1

4.1 Ordering options

Table 2. Ordering options

Type number	Orderable part number	Package	Packing method	Minimum order quantity	Temperature
SA612AD/01	SA612AD/01,112	SO8	Standard marking *IC's tube - DSC bulk pack	2000	T_{amb} = −40 °C to +85 °C
	SA612AD/01,118	SO8	Reel 13" Q1/T1 *Standard mark SMD	2500	T_{amb} = −40 °C to +85 °C

5. Block diagram

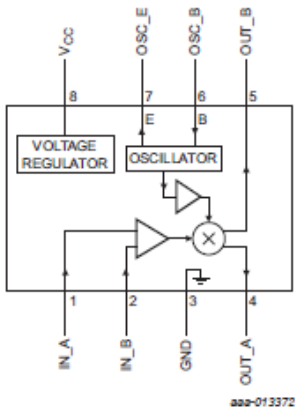


Fig 1. Block diagram

6. Pinning information

6.1 Pinning

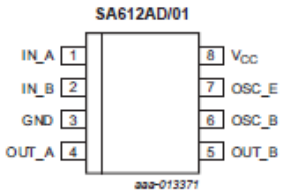


Fig 2. Pin configuration for SO8

6.2 Pin description

Table 3. Pin description

Symbol	Pin	Description
IN_A	1	RF input A
IN_B	2	RF input B
GND	3	ground
OUT_A	4	mixer output A
OUT_B	5	mixer output B
OSC_B	6	oscillator input (base)
OSC_E	7	oscillator output (emitter)
V_{CC}	8	supply voltage

7. Functional description

The SA612A is a Gilbert cell, an oscillator/buffer, and a temperature-compensated bias network as shown in Figure 3. The Gilbert cell is a differential amplifier (IN_A and IN_B pins) that drives a balanced switching cell. The differential input stage provides gain and determines the noise figure and signal handling performance of the system.

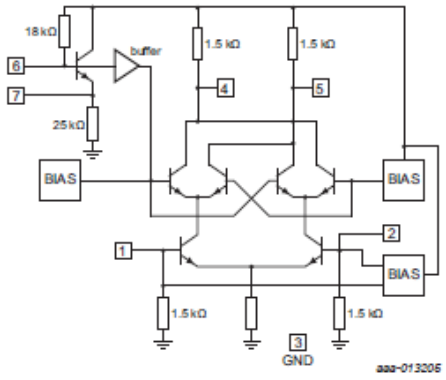


Fig 3. Equivalent circuit

The SA612A is designed for optimum low-power performance. When used with the SA614A as a 45 MHz cordless phone/cellular radio second IF and demodulator, the SA612A is capable of receiving −119 dBm signals with a 12 dB S/N ratio. Third-order intercept is typically −15 dBm (that is approximately +5 dBm output intercept because of the RF gain). The system designer must be cognizant of this large signal limitation. When designing LANs or other closed systems where transmission levels are high, and small-signal or signal-to-noise issues are not critical, the input to the SA612A should be appropriately scaled.

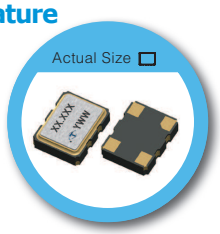
Besides excellent low-power performance well into VHF, the SA612A is flexible. The input, output and oscillator ports support various configurations provided the designer understands certain constraints, which are explained here.

The RF inputs (IN_A and IN_B pins) are biased internally. They are symmetrical. The equivalent AC input impedance is approximately 1.5 kΩ || 3 pF through 50 MHz. IN_A and IN_B pins can be used interchangeably, but they should not be DC biased externally. Figure 4 shows three typical input configurations.

I.1.5. Oscilador VCTCXO de 26 MHz

TAITIEN
Our Performance, Your Reputation

TX Type
3.2 x 2.5 mm SMD Voltage Controlled Temperature Compensated Crystal Oscillator

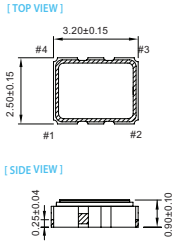


RoHS Compliant

FEATURE
- Typical 3.2 x 2.5 x 0.9 mm SMD.
- For automatic assembly.
- Compactness and lightweight.
- Low power consumption.
- VCTCXO available.
- Low thickness

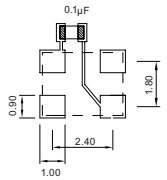
TYPICAL APPLICATION
- GPS
- WiMAX, WLAN
- Mobile Phone

DIMENSION (mm)



Pin#	Function
1	Vcon:VC-TCXO GND / NC:TCXO
2	GND
3	Output
4	VDD

SOLDER PAD LAYOUT (mm)



To ensure optimal oscillator performance, place a by-pass capacitor of 0.1µF as close to the part as possible between Vdd and GND pads.

ELECTRICAL SPECIFICATION

Parameter	3.3 / 3.0 / 2.8 V		2.5 V		1.8 V		Unit
	Min.	Max.	Min.	Max.	Min.	Max.	
Supply Voltage Variation (VDD)	2.66	3.465	2.375	2.625	1.71	1.89	V
Frequency Range	10	52	10	52	10	52	MHz
Standard Frequency	10, 12.8, 13, 16.367667, 16.368, 16.369, 19.2, 19.44, 20, 25, 26, 27, 30 30.72, 32, 38.4						
Frequency Tolerance*	–	±2.0	–	±2.0	–	±2.0	ppm
Frequency stability							
Vs Supply Voltage (±5%) change	–	±0.2	–	±0.2	–	±0.2	ppm
Vs Load (±10%) change	–	±0.2	–	±0.2	–	±0.2	
Vs Aging (@1st year)	–	±1.0	–	±1.0	–	±1.0	ppm
Supply Current 10 MHz ≦ Fo ≦ 26 MHz	–	1.5	–	1.5	–	1.5	mA
26 MHz < Fo ≦ 52 MHz	–	2.0	–	2.0	–	2.0	
Output Level (Clipped sine wave)	0.8	–	0.8	–	0.8	–	Vp-p
Load	10KΩ//10pF		10KΩ//10pF		10KΩ//10pF		
Control Voltage Range (VCTCXO)	0.5	2.5	0.4	2.4	0.3	1.5	V
Pulling Range (VCTCXO)	±5.0	–	±5.0	–	±5.0	–	ppm
Vc Input Impedance (VCTCXO)	500	–	500	–	500	–	kΩ
Phase Noise @ 19.2 MHz 100 Hz	-115		-115		-115		dBc/Hz
1 kHz	-135		-135		-135		
10 kHz	-148		-148		-148		
Start time	–	2	–	2	–	2	mSec
Storage Temp. Range	-40	85	-40	85	-40	85	°C

Standard frequencies are frequencies which the crystal has been designed and does not imply a stock position.
*Frequency at 25°C, 1 hour after reflow.

FREQ. STABILITY vs. TEMP. RANGE

Temp. (°C) \ ppm	±0.5	±1.0	±1.5	±2.0	±2.5
-20 ~ +70	○	○	○	○	○
-30 ~ +85	○	○	○	○	○
-40 ~ +85	○	○	○	○	○

* ○: Available △:Conditional X: Not available

Note: not all combination of options are available. Other specifications may be available upon request.

I.1.6. Regulador de tensión XC6206P331

TOREX

XC6206 Series

ETR0305_008

Low ESR Cap.Compatible Positive Voltage Regulators

■GENERAL DESCRIPTION

The XC6206 series are highly precise, low power consumption, 3 terminal, positive voltage regulators manufactured using CMOS and laser trimming technologies. The series provides large currents with a significantly small dropout voltage.
The XC6206 consists of a current limiter circuit, a driver transistor, a precision reference voltage and an error correction circuit. The series is compatible with low ESR ceramic capacitors. The current limiter's foldback circuit operates as a short circuit protection as well as the output current limiter for the output pin.
Output voltages are internally by laser trimming technologies. It is selectable in 0.1V increments within a range of 1.2V to 5.0V. SOT-23, SOT-89 and USP-6B packages are available.

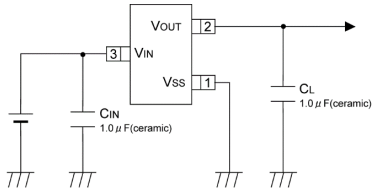
■APPLICATIONS

- ●Smart phones / Mobile phones
- ●Portable game consoles
- ●Digital still cameras / Camcorders
- ●Digital audio equipments
- ●Reference voltage sources
- ●Multi-function power supplies

■FEATURES

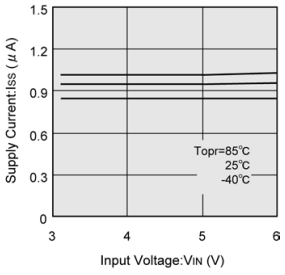
Maximum Output Current	: 200mA (3.0V type)
Dropout Voltage	: 250mV @ 100mA (3.0V type)
Maximum Operating Voltage	: 6.0V
Output Voltage Range	: 1.2V ~ 5.0V (0.1V increments)
Highly Accurate	: ±2%@V_{OUT}≧1.5V ±30mV@V_{OUT}<1.5V (±1% @V_{OUT}≧2.0V)
Low Power Consumption	: 1.0μA (TYP.)
Low ESR Capacitor	: Ceramic capacitor compatible
Protection	: Current Limit Circuit Built-in
Operating Ambient Temperature	: -40℃~ +85℃
Packages	: SOT-23 SOT-89 USP-6B
Environmentally Friendly	: EU RoHS Compliant, Pb Free

■TYPICAL APPLICATION CIRCUIT

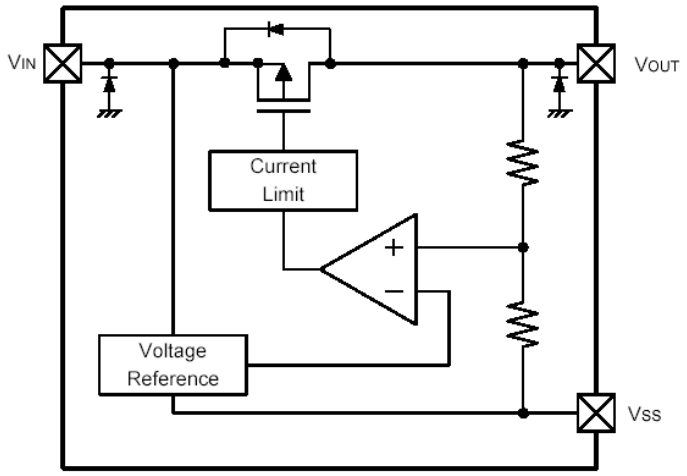


■TYPICAL PERFORMANCE CHARACTERISTICS

XC6206P302



■BLOCK DIAGRAM



*Diodes inside the circuit are an ESD protection diode and a parasitic diode.

■PRODUCT CLASSIFICATION

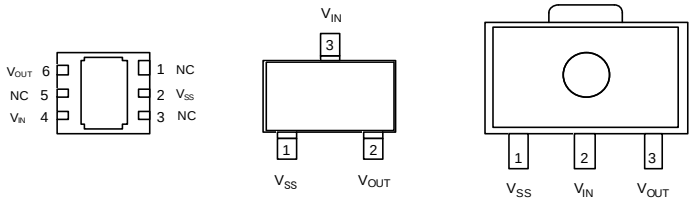
●Ordering Information

XC6206P①②③④⑤-⑥(*1)

DESIGNATOR	ITEM	SYMBOL	DESCRIPTION
①②	Output Voltage	12~50	e.g. VOUT: 3.0V→①=3, ②=0
③	Accuracy	2	±2% (VOUT≧1.5V), ±30mV (VOUT<1.5V)
		1	±1% (VOUT≧2.0V)
④⑤-⑥	Packages (Order Unit)	MR	SOT-23 (3,000pcs/Reel)
		MR-G	SOT-23 (3,000pcs/Reel)
		PR	SOT-89 (1,000pcs/Reel)
		PR-G	SOT-89 (1,000pcs/Reel)
		DR	USP-6B (3,000pcs/Reel)
		DR-G	USP-6B (3,000pcs/Reel)

(*1) The "-G" suffix denotes Halogen and Antimony free as well as being fully EU RoHS compliant.

■PIN CONFIGURATION



USP-6B
(BOTTOM VIEW)

SOT-23
(TOP VIEW)

SOT-89
(TOP VIEW)

*The dissipation pad for the USP-6B package should be solder-plated in recommended mount pattern and metal masking so as to enhance mounting strength and heat release.
If the pad needs to be connected to other pins, it should be connected to the pin number 4 (V_{IN}).

■PIN ASSIGNMENT

PIN NUMBER			PIN NAME	FUNCTIONS
SOT-23	SOT-89	USP-6B		
1	1	2	VSS	Ground
3	2	4	VIN	Power Input
2	3	6	VOUT	Output
-	-	1, 3, 5	NC	No Connection

■ABSOLUTE MAXIMUM RATINGS

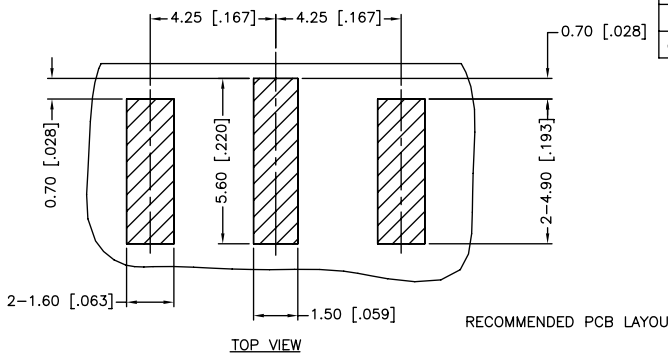
Ta=25°C

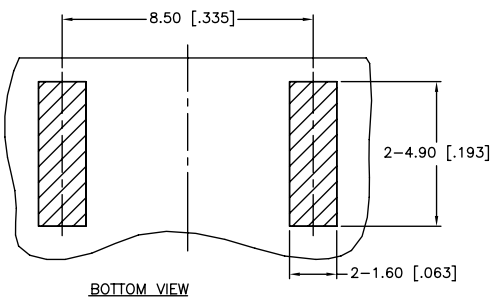
PARAMETER		SYMBOL	RATINGS	UNITS
Input Voltage		VIN	-0.3～+7.0	V
Output Current		IOUT	500 (*1)	mA
Output Voltage		VOUT	-0.3 ~ VIN + 0.3	V
Power Dissipation	SOT-23	Pd	250	mW
			500(40mm x 40mm Standard board) (*2)	
	SOT-89		500	
			1000(40mm x 40mm Standard board) (*2)	
	USP-6B		120	
			1000(40mm x 40mm Standard board) (*2)	
Operating Ambient Temperature		Topr	- 40 ~ + 85	°C
Storage Temperature		Tstg	- 55 ~ + 125	°C

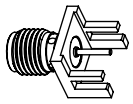
(*1) $I_{OUT} \leqq Pd / (V_{IN}-V_{OUT})$
(*2) The power dissipation figure shown is PCB mounted and is for reference only.
The mounting condition is please refer to PACKAGING INFORMATION.

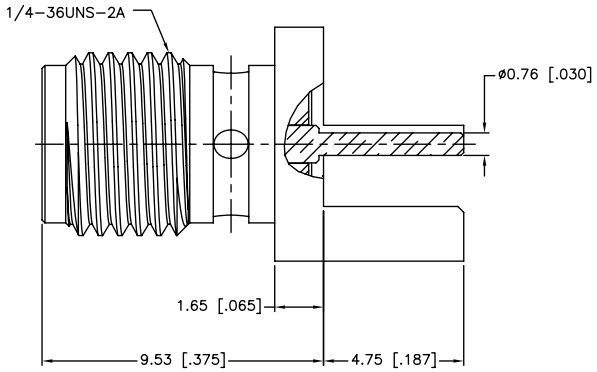
I.1.7. Conector SMA hembra

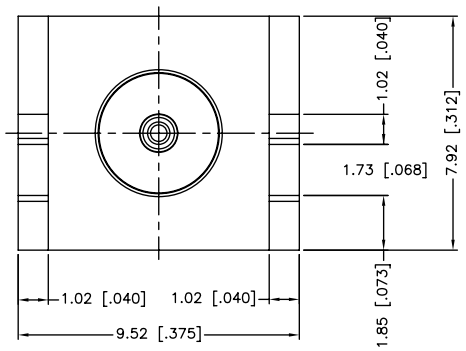
REV.	DATE	ECR NO.	DESCRIPTION
NC	10/29/98	XXXXXX	INITIAL RELEASE
A	08/13/06	XXXXXX	UPDATE DRAWING FORMAT
B	07/29/11	2208	ADD PCB LAYOUT AND NOTES
C	11/30/11	2254	UPDATE THE CONTACT AND INSULATOR











1. IMPEDANCE: 50 OHM
2. FREQUENCY RANGE: DC 0-18 GHZ
3. VSWR: 1.3 MAX.
4. DIELECTRIC WITHSTANDING VOLTAGE: 1000 VRMS, MIN.
5. BOARD THICKNESS=1.57 MM [.062"]
6. CONTACT PIN MUST WITHSTAND 6 LBS MIN. AXIAL IN BOTH DIRECTIONS AND 4 IN-OZ MIN. RADIAL FORCE.

	DESCRIPTION	MATERIAL	FINISH	QTY
3	CONTACT PIN	BER. COPPER	GOLD	1
2	INSULATOR	PTFE	NATURAL	1
1	BODY	BRASS	GOLD	1

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN MILLIMETERS. DIMENSIONS IN [ ] ARE IN INCHES AND FOR CUSTOMER REFERANCE ONLY.

UNLESS OTHERWISE SPECIFIED TOLERANCES FOR MILLIMETERS ARE:
0.5 - 8mm ± 0.20mm
8 - 30mm ± 0.40mm
30 - 120mm ± 0.50mm

UNLESS OTHERWISE SPECIFIED TOLERANCES FOR INCHES ARE:
.020 - .315 = ± 0.007"
.315 - 1.180 = ± 0.015"
1.180 - 4.724 = ± 0.020"

DO NOT SCALE DRAWING

APPROVALS	DATE
DRAWN M.ZHANG	11/30/11
CHECKED	
ISSUED	
SHEET 1 OF 1	

CAD FILE C:/CONNEX/132255.DWG

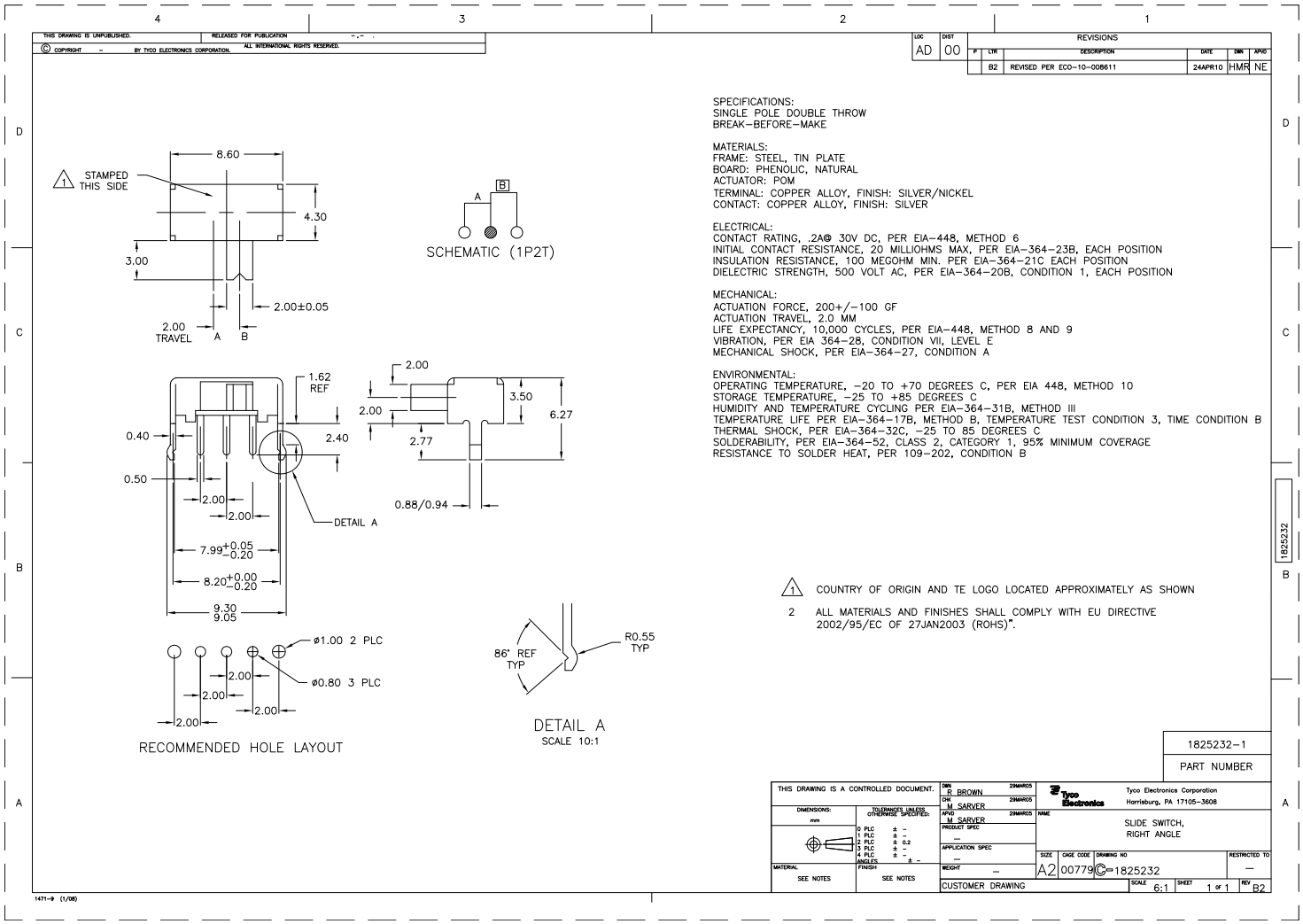
Amphenol Connex

PART DESCRIPTION: SMA END LAUNCH RECEPT. JACK, RIBBED FLANGE (.062" BOARD THICKNESS)

PART NO. 132255

DWG. NO. 132255.DWG | REV. C | SIZE A | SCALE NA

I.1.8. Interruptor deslizante



I.2. Datalogger
I.2.1. ESP32

1 Overview

ESP32 is a single 2.4 GHz Wi-Fi-and-Bluetooth combo chip designed with the TSMC ultra-low-power 40 nm technology. It is designed to achieve the best power and RF performance, showing robustness, versatility and reliability in a wide variety of applications and power scenarios.

The ESP32 series of chips includes ESP32-D0WD-V3, ESP32-D0WDQ6-V3, ESP32-D0WD, ESP32-D0WDQ6, ESP32-D2WD, ESP32-S0WD, and ESP32-U4WDH, among which, ESP32-D0WD-V3, ESP32-D0WDQ6-V3, and ESP32-U4WDH are based on ECO V3 wafer.

For details on part numbers and ordering information, please refer to Section 7.

For details on ECO V3 instructions, please refer to *ESP32 ECO V3 User Guide*.

1.1 Featured Solutions

1.1.1 Ultra-Low-Power Solution

ESP32 is designed for mobile, wearable electronics, and Internet-of-Things (IoT) applications. It features all the state-of-the-art characteristics of low-power chips, including fine-grained clock gating, multiple power modes, and dynamic power scaling. For instance, in a low-power IoT sensor hub application scenario, ESP32 is woken up periodically and only when a specified condition is detected. Low-duty cycle is used to minimize the amount of energy that the chip expends. The output of the power amplifier is also adjustable, thus contributing to an optimal trade-off between communication range, data rate and power consumption.

Note:
For more information, refer to Section 3.7 *RTC and Low-Power Management*.

1.1.2 Complete Integration Solution

ESP32 is a highly-integrated solution for Wi-Fi-and-Bluetooth IoT applications, with around 20 external components. ESP32 integrates an antenna switch, RF balun, power amplifier, low-noise receive amplifier, filters, and power management modules. As such, the entire solution occupies minimal Printed Circuit Board (PCB) area.

ESP32 uses CMOS for single-chip fully-integrated radio and baseband, while also integrating advanced calibration circuitries that allow the solution to remove external circuit imperfections or adjust to changes in external conditions. As such, the mass production of ESP32 solutions does not require expensive and specialized Wi-Fi testing equipment.

1.4 MCU and Advanced Features

1.4.1 CPU and Memory

- Xtensa® single-/dual-core 32-bit LX6 microprocessor(s), up to 600 MIPS (200 MIPS for ESP32-S0WD/ ESP32-U4WDH, 400 MIPS for ESP32-D2WD)
- 448 KB ROM
- 520 KB SRAM
- 16 KB SRAM in RTC
- QSPI supports multiple flash/SRAM chips

1.4.2 Clocks and Timers

- Internal 8 MHz oscillator with calibration
- Internal RC oscillator with calibration
- External 2 MHz ~ 60 MHz crystal oscillator (40 MHz only for Wi-Fi/BT functionality)
- External 32 kHz crystal oscillator for RTC with calibration
- Two timer groups, including 2 × 64-bit timers and 1 × main watchdog in each group
- One RTC timer
- RTC watchdog

1.4.3 Advanced Peripheral Interfaces

- 34 × programmable GPIOs
- 12-bit SAR ADC up to 18 channels
- 2 × 8-bit DAC
- 10 × touch sensors
- 4 × SPI
- 2 × I²S
- 2 × I²C
- 3 × UART
- 1 host (SD/eMMC/SDIO)
- 1 slave (SDIO/SPI)
- Ethernet MAC interface with dedicated DMA and IEEE 1588 support
- Two-Wire Automotive Interface (TWAI®, compatible with ISO11898-1)
- IR (TX/RX)
- Motor PWM
- LED PWM up to 16 channels
- Hall sensor

1.4.4 Security

- Secure boot
- Flash encryption
- 1024-bit OTP, up to 768-bit for customers
- Cryptographic hardware acceleration:
  - AES
  - Hash (SHA-2)
  - RSA
  - ECC
  - Random Number Generator (RNG)

1.6 Block Diagram

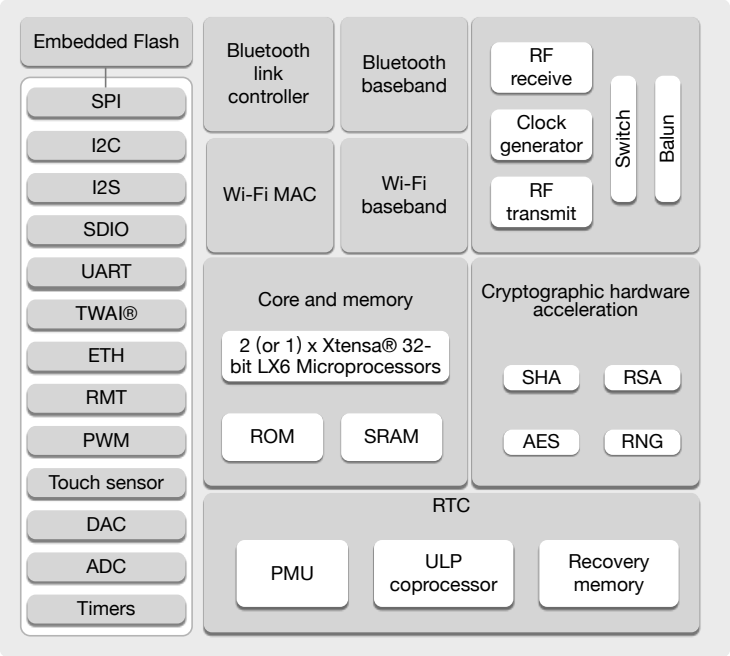


Figure 1: Functional Block Diagram

Note:
Products in the ESP32 series differ from each other in terms of their support for embedded flash and the number of CPUs they have. For details, please refer to Section 7 *Part Number and Ordering Information*.

2 Pin Definitions

2.1 Pin Layout

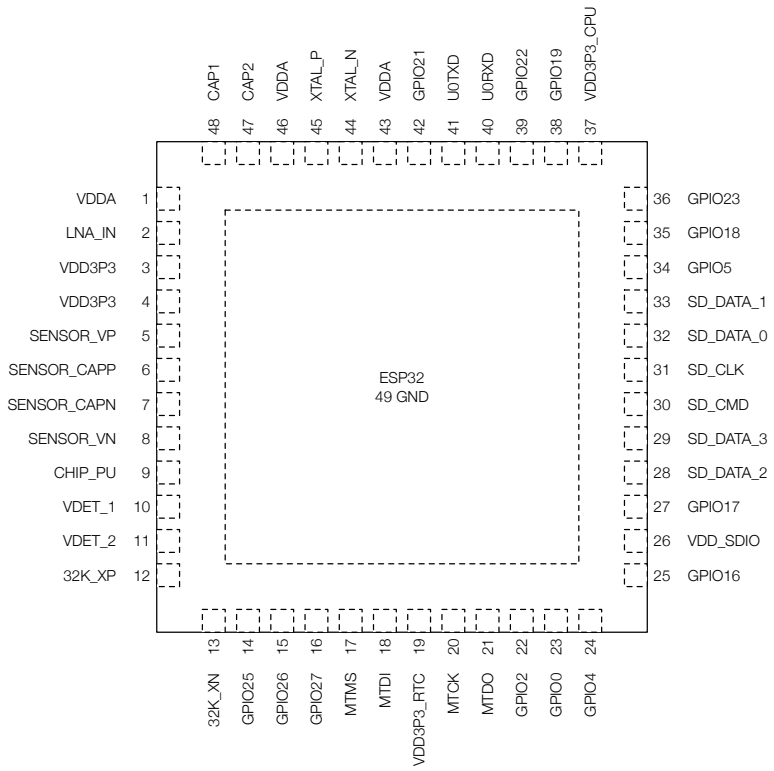


Figure 2: ESP32 Pin Layout (QFN 6*6, Top View)

I.2.2. FT232RL

FTDI Chip

FT232R USB UART IC Datasheet
Version 2.16

Document No.: FT_000053 Clearance No.: FTDI# 38

Future Technology Devices International Ltd.

FT232R USB UART IC Datasheet

The FT232R is a USB to serial UART interface with the following advanced features:

- Single chip USB to asynchronous serial data transfer interface.
- Entire USB protocol handled on the chip. No USB specific firmware programming required.
- Fully integrated 1024 bit EEPROM storing device descriptors and CBUS I/O configuration.
- Fully integrated USB termination resistors.
- Fully integrated clock generation with no external crystal required plus optional clock output selection enabling a glue-less interface to external MCU or FPGA.
- Data transfer rates from 300 baud to 3 Mbaud (RS422, RS485, RS232) at TTL levels.
- 128 byte receive buffer and 256 byte transmit buffer utilising buffer smoothing technology to allow for high data throughput.
- FTDI's royalty-free Virtual Com Port (VCP) and Direct (D2XX) drivers eliminate the requirement for USB driver development in most cases.
- Unique USB FTDIChip-ID™ feature.
- Configurable CBUS I/O pins.
- Transmit and receive LED drive signals.
- UART interface support for 7 or 8 data bits, 1 or 2 stop bits and odd / even / mark / space / no parity
- FIFO receives and transmits buffers for high data throughput.
- Synchronous and asynchronous bit bang interface options with RD# and WR# strobes.
- Device supplied pre-programmed with unique USB serial number.
- Supports bus powered, self-powered and high-power bus powered USB configurations.
- Integrated +3.3V level converter for USB I/O.
- Integrated level converter on UART and CBUS for interfacing to between +1.8V and +5V logic.
- True 5V/3.3V/2.8V/1.8V CMOS drive output and TTL input.
- Configurable I/O pin output drive strength.
- Integrated power-on-reset circuit.
- Fully integrated AVCC supply filtering - no external filtering required.
- UART signal inversion option.
- +3.3V (using external oscillator) to +5.25V (internal oscillator) Single Supply Operation.
- Low operating and USB suspend current.
- Low USB bandwidth consumption.
- UHCI/OHCI/EHCI host controller compatible.
- USB 2.0 Full Speed compatible.
- -40°C to 85°C extended operating temperature range.
- Available in compact Pb-free 28 Pin SSOP and QFN-32 packages (both RoHS compliant).

2 FT232R Block Diagram

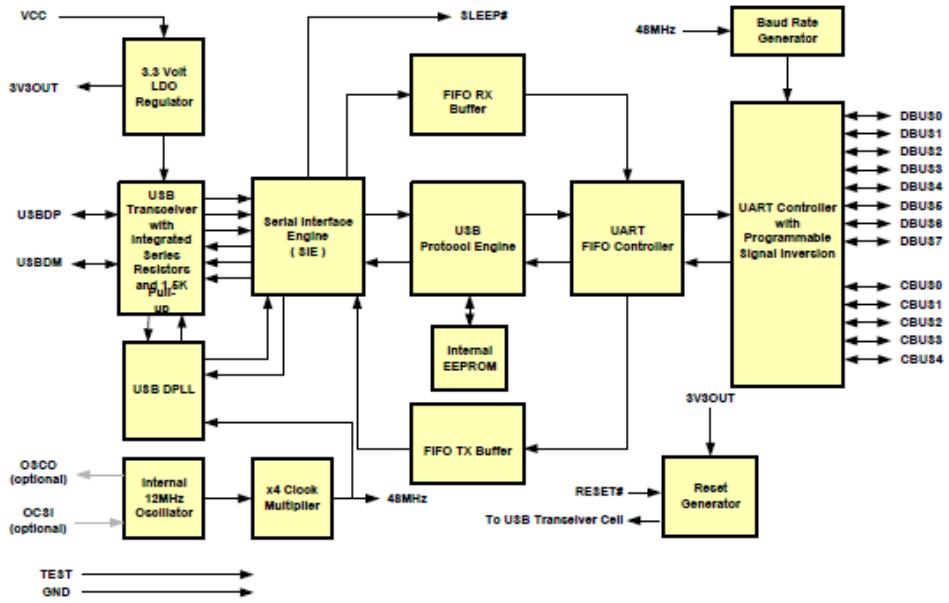


Figure 2.1 FT232R Block Diagram

For a description of each function please refer to Section **4.**

3 Device Pin Out and Signal Description

3.1 28-LD SSOP Package

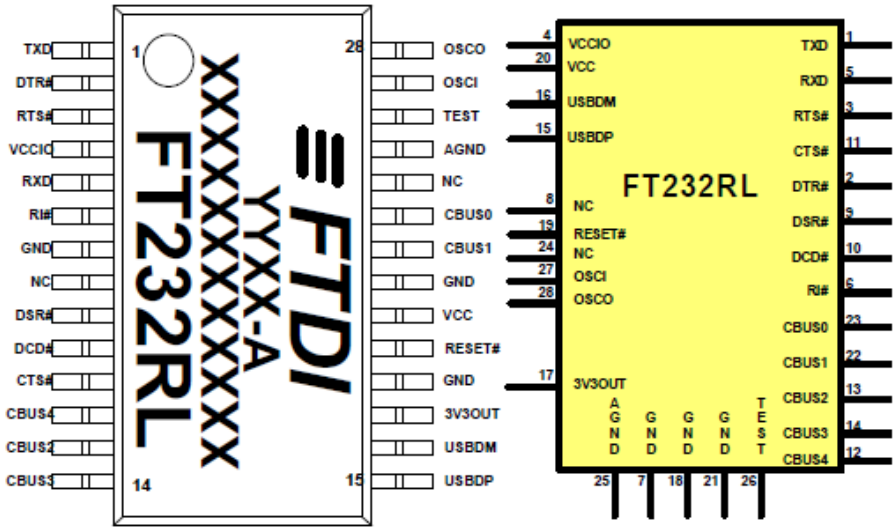


Figure 3.1 SSOP Package Pin Out and Schematic Symbol

3.2 SSOP Package Pin Out Description

Note: The convention used throughout this document for active low signals is the signal name followed by#

Pin No.	Name	Type	Description
15	USBDP	I/O	USB Data Signal Plus, incorporating internal series resistor and 1.5kΩ pull up resistor to 3.3V.
16	USBDM	I/O	USB Data Signal Minus, incorporating internal series resistor.

Table 3.1 USB Interface Group

Pin No.	Name	Type	Description
4	VCCIO	PWR	+1.8V to +5.25V supply to the UART Interface and CBUS group pins (1...3, 5, 6, 9...14, 22, 23). In USB bus powered designs connect this pin to 3V3OUT pin to drive out at +3.3V levels, or connect to VCC to drive out at 5V CMOS level. This pin can also be supplied with an external +1.8V to +2.8V supply in order to drive outputs at lower levels. It should be noted that in this case this supply should originate from the same source as the supply to VCC. This means that in bus powered designs a regulator which is supplied by the +5V on the USB bus should be used.
7, 18, 21	GND	PWR	Device ground supply pins
17	3V3OUT	Output	+3.3V output from integrated LDO regulator. This pin should be decoupled to ground using a 100nF capacitor. The main use of this pin is to provide the internal +3.3V supply to the USB transceiver cell and the internal 1.5kΩ pull up resistor on USBDP. Up to 50mA can be drawn from

Pin No.	Name	Type	Description
			this pin to power external logic if required. This pin can also be used to supply the VCCIO pin.
20	VCC	PWR	+3.3V to +5.25V supply to the device core. (see Note 1)
25	AGND	PWR	Device analogue ground supply for internal clock multiplier

Table 3.2 Power and Ground Group

Pin No.	Name	Type	Description
8, 24	NC	NC	No internal connection
19	RESET#	Input	Active low reset pin. This can be used by an external device to reset the FT232R. If not required can be left unconnected, or pulled up to VCC.
26	TEST	Input	Puts the device into IC test mode. Must be tied to GND for normal operation, otherwise the device will appear to fail.
27	OSCI	Input	Input 12MHz Oscillator Cell. Optional – Can be left unconnected for normal operation. (see Note 2)
28	OSCO	Output	Output from 12MHZ Oscillator Cell. Optional – Can be left unconnected for normal operation if internal Oscillator is used. (see Note 2)

Table 3.3 Miscellaneous Signal Group

Pin No.	Name	Type	Description
1	TXD	Output	Transmit Asynchronous Data Output.
2	DTR#	Output	Data Terminal Ready Control Output / Handshake Signal.
3	RTS#	Output	Request to Send Control Output / Handshake Signal.
5	RXD	Input	Receiving Asynchronous Data Input.
6	RI#	Input	Ring Indicator Control Input. When remote wake up is enabled in the internal EEPROM taking RI# low (20ms active low pulse) can be used to resume the PC USB host controller from suspend.
9	DSR#	Input	Data Set Ready Control Input / Handshake Signal.
10	DCD#	Input	Data Carrier Detect Control Input.
11	CTS#	Input	Clear To Send Control Input / Handshake Signal.
12	CBUS4	I/O	Configurable CBUS output only Pin. Function of this pin is configured in the device internal EEPROM. Factory default configuration is SLEEP#. See CBUS Signal Options, Table 3.9.
13	CBUS2	I/O	Configurable CBUS I/O Pin. Function of this pin is configured in the device internal EEPROM. Factory default configuration is TXDEN. See CBUS Signal Options, Table 3.9.
14	CBUS3	I/O	Configurable CBUS I/O Pin. Function of this pin is configured in the device internal EEPROM. Factory default configuration is PWREN#. See CBUS Signal Options, Table 3.9. PWREN# should be used with a 10kΩ resistor pull up.
22	CBUS1	I/O	Configurable CBUS I/O Pin. Function of this pin is configured in the device internal EEPROM. Factory default configuration is RXLED#. See CBUS Signal Options, Table 3.9.
23	CBUS0	I/O	Configurable CBUS I/O Pin. Function of this pin is configured in the device internal EEPROM. Factory default configuration is TXLED#. See CBUS Signal Options, Table 3.9.

Table 3.4 UART Interface and CUSB Group (see note 3)

Notes:

1. The minimum operating voltage VCC must be +4.0V (could use VBUS=+5V) when using the internal clock generator. Operation at +3.3V is possible using an external crystal oscillator.
2. For details on how to use an external crystal, ceramic resonator, or oscillator with the FT232R, please refer Section 7.6
3. When used in Input Mode, the input pins are pulled to VCCIO via internal 200kΩ resistors. These pins can be programmed to gently pull low during USB suspend (PWREN# = "1") by setting an option in the internal EEPROM.

I.2.3. MOSFET FDV303N

FAIRCHILD

July 2014

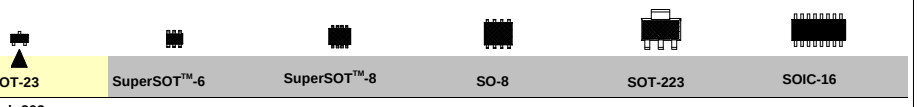
FDV303N
Digital FET, N-Channel

General Description

These N-Channel enhancement mode field effect transistors are produced using Fairchild's proprietary, high cell density, DMOS technology. This very high density process is tailored to minimize on-state resistance at low gate drive conditions. This device is designed especially for application in battery circuits using either one lithium or three cadmium or NMH cells. It can be used as an inverter or for high-efficiency miniature discrete DC/DC conversion in compact portable electronic devices like cellular phones and pagers. This device has excellent on-state resistance even at gate drive voltages as low as 2.5 volts.

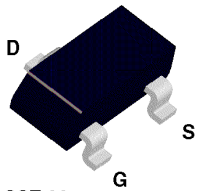
Features

- 25 V, 0.68 A continuous, 2 A Peak.
  $R_{DS(ON)}$ = 0.45 Ω @ V_{GS} = 4.5 V
  $R_{DS(ON)}$ = 0.6 Ω @ V_{GS}= 2.7 V.
- Very low level gate drive requirements allowing direct operation in 3V circuits. $V_{GS(th)}$ < 1V.
- Gate-Source Zener for ESD ruggedness.>6kV Human Body Model
- Compact industry standard SOT-23 surface mount package.
- Alternative to TN0200T and TN0201T.

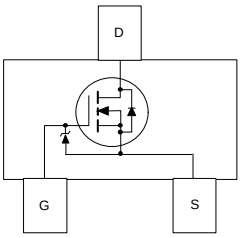


SOT-23	SuperSOT™-6	SuperSOT™-8	SO-8	SOT-223	SOIC-16

Mark:303



SOT-23



Absolute Maximum Ratings T_A = 25°C unless other wise noted

Symbol	Parameter	FDV303N	Units
V_{DSS}	Drain-Source Voltage, Power Supply Voltage	25	V
V_{GSS}	Gate-Source Voltage, V_{IN}	8	V
I_D	Drain/Output Current - Continuous	0.68	A
	- Pulsed	2	
P_D	Maximum Power Dissipation	0.35	W
T_J,T_{STG}	Operating and Storage Temperature Range	-55 to 150	°C
ESD	Electrostatic Discharge Rating MIL-STD-883D Human Body Model (100pf / 1500 Ohm)	6.0	kV
THERMAL CHARACTERISTICS			
$R_{\theta JA}$	Thermal Resistance, Junction-to-Ambient	357	°C/W

Electrical Characteristics (T_A = 25 °C unless otherwise noted)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
OFF CHARACTERISTICS						
BV_{DSS}	Drain-Source Breakdown Voltage	V_{GS} = 0 V, I_D = 250 µA	25			V
$\Delta BV_{DSS}/\Delta T_J$	Breakdown Voltage Temp. Coefficient	I_D = 250 µA, Referenced to 25 °C		26		mV / °C
I_{DSS}	Zero Gate Voltage Drain Current	V_{DS} = 20 V, V_{GS} = 0 V			1	µA
		T_J = 55°C			10	µA
I_{GSS}	Gate - Body Leakage Current	V_{GS} = 8 V, V_{DS} = 0 V			100	nA
ON CHARACTERISTICS (Note)						
$\Delta V_{GS(th)}/\Delta T_J$	Gate Threshold Voltage Temp. Coefficient	I_D = 250 µA, Referenced to 25 °C		-2.6		mV / °C
$V_{GS(th)}$	Gate Threshold Voltage	V_{DS} = V_{GS}, I_D = 250 µA	0.65	0.8	1	V
$R_{DS(ON)}$	Static Drain-Source On-Resistance	V_{GS} = 4.5 V, I_D = 0.5 A		0.33	0.45	Ω
		T_J =125°C		0.52	0.8	
		V_{GS} = 2.7 V, I_D = 0.2 A		0.44	0.6	
$I_{D(ON)}$	On-State Drain Current	V_{GS} = 2.7 V, V_{DS} = 5 V	0.5			A
g_{FS}	Forward Transconductance	V_{DS} = 5 V, I_D= 0.5 A		1.45		S
DYNAMIC CHARACTERISTICS						
C_{iss}	Input Capacitance	V_{DS} = 10 V, V_{GS} = 0 V, f = 1.0 MHz		50		pF
C_{oss}	Output Capacitance			28		pF
C_{rss}	Reverse Transfer Capacitance			9		pF
SWITCHING CHARACTERISTICS (Note)						
$t_{D(on)}$	Turn - On Delay Time	V_{DD} = 6 V, I_D = 0.5 A, V_{GS} = 4.5 V, R_{GEN} = 50 Ω		3	6	ns
t_r	Turn - On Rise Time			8.5	18	ns
$t_{D(off)}$	Turn - Off Delay Time			17	30	ns
t_f	Turn - Off Fall Time			13	25	ns
Q_g	Total Gate Charge	V_{DS} = 5 V, I_D = 0.5 A, V_{GS} = 4.5 V		1.64	2.3	nC
Q_{gs}	Gate-Source Charge			0.38		nC
Q_{gd}	Gate-Drain Charge			0.45		nC
DRAIN-SOURCE DIODE CHARACTERISTICS AND MAXIMUM RATINGS						
I_S	Maximum Continuous Drain-Source Diode Forward Current				0.3	A
V_{SD}	Drain-Source Diode Forward Voltage	V_{GS} = 0 V, I_S = 0.5 A (Note)		0.83	1.2	V

Note:

Pulse Test: Pulse Width ≤ 300µs, Duty Cycle ≤ 2.0%.

FDV303N Rev.D2

I.2.4. Inversor Schmitt doble SN74LVC2G14

TEXAS INSTRUMENTS

SN74LVC2G14

www.ti.com

SCES200M – APRIL 1999 – REVISED NOVEMBER 2013

Dual Schmitt-Trigger Inverter

Check for Samples: SN74LVC2G14

FEATURES

- Available in the Texas Instruments NanoFree™ Package
- Supports 5-V V_{CC} Operation
- Inputs Accept Voltages to 5.5 V
- Max t_{pd} of 5.4 ns at 3.3 V
- Low Power Consumption, 10-µA Max I_{CC}
- ±24-mA Output Drive at 3.3 V
- Typical V_{OLP} (Output Ground Bounce) <0.8 V at V_{CC} = 3.3 V, T_A = 25°C
- Typical V_{OHV} (Output V_{OH} Undershoot) >2 V at V_{CC} = 3.3 V, T_A = 25°C
- I_{off} Supports Live Insertion, Partial Power Down Mode, and Back Drive Protection
- Support Translation Down (5V to 3.3V; 3.3V to 1.8V)
- Latch-Up Performance Exceeds 100 mA Per JESD 78, Class II
- ESD Protection Exceeds JESD 22
  - 2000-V Human-Body Model (A114-A)
  - 1000-V Charged-Device Model (C101)

DESCRIPTION

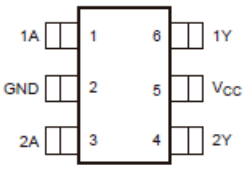
This dual Schmitt-trigger inverter is designed for 1.65-V to 5.5-V V_{CC} operation.

NanoFree™ package technology is a major breakthrough in IC packaging concepts, using the die as the package.

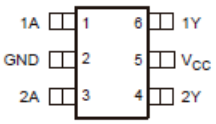
The SN74LVC2G14 contains two inverters and performs the Boolean function $Y = \overline{A}$. The device functions as two independent inverters, but because of Schmitt action, it may have different input threshold levels for positive-going (V_{T+}) and negative-going (V_{T-}) signals.

This device is fully specified for partial-power-down applications using I_{off}. The I_{off} circuitry disables the outputs, preventing damaging current backflow through the device when it is powered down.

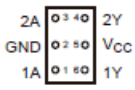
DBV PACKAGE
(TOP VIEW)



DCK PACKAGE
(TOP VIEW)



YZP PACKAGE
(BOTTOM VIEW)



See mechanical drawings for dimensions.

Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

NanoFree is a trademark of Texas Instruments.

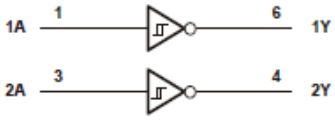
PRODUCTION DATA information is current as of publication date. Products conform to specifications per the terms of the Texas Instruments standard warranty. Production processing does not necessarily include testing of all parameters.

These devices have limited built-in ESD protection. The leads should be shorted together or the device placed in conductive foam during storage or handling to prevent electrostatic damage to the MOS gates.

Function Table
(Each Inverter)

INPUT A	OUTPUT Y
H	L
L	H

Logic Diagram (Positive Logic)



Absolute Maximum Ratings (1)

over operating free-air temperature range (unless otherwise noted)

			MIN	MAX	UNIT
V_{CC}	Supply voltage range		–0.5	6.5	V
V_I	Input voltage range (2)		–0.5	6.5	V
V_O	Voltage range applied to any output in the high-impedance or power-off state (2)		–0.5	6.5	V
V_O	Voltage range applied to any output in the high or low state (2) (3)		–0.5	V_{CC} + 0.5	V
I_{IK}	Input clamp current	V_I < 0		–50	mA
I_{OK}	Output clamp current	V_O < 0		–50	mA
I_O	Continuous output current			±50	mA
	Continuous current through V_{CC} or GND			±100	mA
θ_{JA}	Package thermal impedance (4)	DBV package		165	°C/W
		DCK package		259	
		YZP package		123	
T_{stg}	Storage temperature range		–65	150	°C

(1) Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.
(2) The input negative-voltage and output voltage ratings may be exceeded if the input and output clamp-current ratings are observed.
(3) The value of V_{CC} is provided in the recommended operating conditions table.
(4) The package thermal impedance is calculated in accordance with JESD 51-7.

I.2.5. Reguladores LDO REG102

Burr-Brown Products from Texas Instruments

REG102

SBVS024F – NOVEMBER 2000 – REVISED SEPTEMBER 2005

DMOS 250mA Low-Dropout Regulator

FEATURES

- NEW DMOS TOPOLOGY:
  Ultra Low Dropout Voltage:
  150mV typ at 250mA
  Output Capacitor *not* Required for Stability
- FAST TRANSIENT RESPONSE
- VERY LOW NOISE: 28µVrms
- HIGH ACCURACY: ±1.5% max
- HIGH EFFICIENCY:
  I_{GND} = 600µA at I_{OUT} = 250mA
  Not Enabled: I_{GND} = 0.01µA
- 2.5V, 2.85V, 3.0V, 3.3V, AND 5.0V ADJUSTABLE OUTPUT VERSIONS
- OTHER OUTPUT VOLTAGES AVAILABLE UPON REQUEST
- FOLDBACK CURRENT LIMIT
- THERMAL PROTECTION
- SMALL SURFACE-MOUNT PACKAGES:
  SOT23-5, SOT223-5, and SO-8

APPLICATIONS

- PORTABLE COMMUNICATION DEVICES
- BATTERY-POWERED EQUIPMENT
- PERSONAL DIGITAL ASSISTANTS
- MODEMS
- BAR-CODE SCANNERS
- BACKUP POWER SUPPLIES

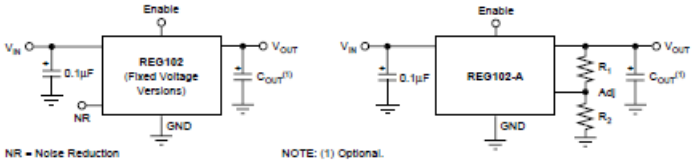
DESCRIPTION

The REG102 is a family of low-noise, low-dropout linear regulators with low ground pin current. The new DMOS topology provides significant improvement over previous designs, including low-dropout voltage (only 150mV typ at full load), and better transient performance. In addition, no output capacitor is required for stability, unlike conventional low-dropout regulators that are difficult to compensate and require expensive low ESR capacitors greater than 1µF.

Typical ground pin current is only 600µA (at I_{OUT} = 250mA) and drops to 10nA when not enabled. Unlike regulators with PNP pass devices, quiescent current remains relatively constant over load variations and under dropout conditions.

The REG102 has very low output noise (typically 28µVrms for V_{OUT} = 3.3V with C_{NR} = 0.01µF), making it ideal for use in portable communications equipment. On-chip trimming results in high output voltage accuracy. Accuracy is maintained over temperature, line, and load variations. Key parameters are tested over the specified temperature range (–40°C to +85°C).

The REG102 is well protected—internal circuitry provides a current limit that protects the load from damage; furthermore, thermal protection circuitry keeps the chip from being damaged by excessive temperature. The REG102 is available in SOT23-5, SOT223-5, and SO-8 packages.



Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

All trademarks are the property of their respective owners.

PRODUCTION DATA information is current as of publication date. Products conform to specifications per the terms of Texas Instruments standard warranty. Production processing does not necessarily include testing of all parameters.

TEXAS INSTRUMENTS
www.ti.com

ABSOLUTE MAXIMUM RATINGS(1)

Supply Input Voltage, V_{IN}	–0.3V to 12V
Enable Input Voltage, V_{EN}	–0.3V to V_{IN}
Feedback Voltage, V_{FB}	–0.3V to 6.0V
NR Pin Voltage, V_{NR}	–0.3V to 6.0V
Output Short-Circuit Duration	Indefinite
Operating Temperature Range (T_J)	–55°C to +125°C
Storage Temperature Range (T_A)	–65°C to +150°C
Lead Temperature (soldering, 3s)	+240°C

NOTE: (1) Stresses above these ratings may cause permanent damage. Exposure to absolute maximum conditions for extended periods may degrade device reliability.

ELECTROSTATIC DISCHARGE SENSITIVITY

This integrated circuit can be damaged by ESD. Texas Instruments recommends that all integrated circuits be handled with appropriate precautions. Failure to observe proper handling and installation procedures can cause damage.

ESD damage can range from subtle performance degradation to complete device failure. Precision integrated circuits may be more susceptible to damage because very small parametric changes could cause the device not to meet its published specifications.

PACKAGE/ORDERING INFORMATION(1)

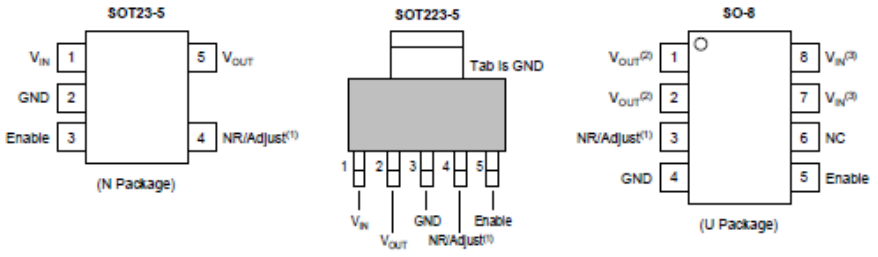
PRODUCT	V_{OUT}(2)
REG102xx-yyyy/zzz	XX is package designator. YYYY is typical output voltage (5 = 5.0V, 2.85 = 2.85V, A = Adjustable). ZZZ is package quantity.

(1) For the most current package and ordering information, see the Package Option Addendum at the end of this document, or see the TI website at www.ti.com.

(2) Output voltages from 2.5V to 5.1V in 50mV increments are available; minimum order quantities apply. Contact factory for details and availability.

PIN CONFIGURATIONS

Top View



NOTES: (1) For REG102A-A: voltage setting resistor pin.
All other models: noise reduction capacitor pin.
(2) Both pin 1 and pin 2 must be connected.
(3) Both pin 7 and pin 8 must be connected.

ELECTRICAL CHARACTERISTICS

Boldface limits apply over the specified temperature range, T_J = –40°C to +85°C.

At T_J = +25°C, V_{IN} = V_{OUT} + 1V (V_{OUT} = 2.5V for REG102-A), V_{ENABLE} = 1.8V, I_{OUT} = 5mA, C_{NR} = 0.01µF, and C_{OUT} = 0.1µF[1], unless otherwise noted.

PARAMETER		CONDITION	REG102NA REG102GA REG102UA MIN	TYP	MAX	UNITS
OUTPUT VOLTAGE						
Output Voltage Range	V_{OUT}					
REG102-2.5				2.5		V
REG102-2.8				2.8		V
REG102-2.85				2.85		V
REG102-3.0				3.0		V
REG102-3.3				3.3		V
REG102-5				5		V
REG102-A			2.5		5.5	V
Reference Voltage	V_{REF}			1.26		V
Adjust Pin Current	I_{ADJ}			0.2	1	µA
Accuracy				±0.5	±1.5	%
Over Temperature					**±2.3**	%
vs Temperature	**dV_{OUT}/dT**			**50**		**ppm/°C**
vs Line and Load		I_{OUT} = 5mA to 250mA, V_{IN} = (V_{OUT} + 0.4V) to 10V		±0.8	±2.0	%
Over Temperature		**V_{IN} = (V_{OUT} + 0.6V) to 10V**			**±2.8**	**%**
DC DROPOUT VOLTAGE[2]	V_{DROP}	I_{OUT} = 5mA		4	10	mV
For all models		I_{OUT} = 250mA		150	220	mV
Over Temperature		**I_{OUT} = 250mA**			**270**	mV
VOLTAGE NOISE						
f = 10Hz to 100kHz	V_n					
Without C_{NR} (all models)		C_{NR} = 0, C_{OUT} = 0		23µVrms/V • V_{OUT}		µVrms
With C_{NR} (all fixed voltage models)		C_{NR} = 0.01µF, C_{OUT} = 10µF		7µVrms/V • V_{OUT}		µVrms
OUTPUT CURRENT						
Current Limit[3]	I_{CL}		340	400	470	mA
Over Temperature			**300**		**490**	mA
Short-Circuit Current Limit	I_{SC}			150		mA
RIPPLE REJECTION						
f = 120Hz				65		dB
ENABLE CONTROL						
V_{ENABLE} High (output enabled)	V_{ENABLE}		1.8		V_{IN}	V
V_{ENABLE} Low (output disabled)			–0.2		0.5	V
I_{ENABLE} High (output enabled)	I_{ENABLE}	V_{ENABLE} = 1.8V to V_{IN}, V_{IN} = 1.8V to 6.5[4]		1	100	nA
I_{ENABLE} Low (output disabled)		V_{ENABLE} = 0V to 0.5V		2	100	nA
Output Disable Time		C_{OUT} = 1.0µF, R_{LOAD} = 13Ω		50		µs
Output Enable Softstart Time		C_{OUT} = 1.0µF, R_{LOAD} = 13Ω		1.5		ms
THERMAL SHUTDOWN						
Junction Temperature						
Shutdown				160		°C
Reset from Shutdown				140		°C
GROUND PIN CURRENT						
Ground Pin Current	I_{GND}	I_{OUT} = 5mA		400	500	µA
		I_{OUT} = 250mA		600	800	µA
Enable Pin Low		V_{ENABLE} ≤ 0.5V		0.01	0.2	µA
INPUT VOLTAGE	V_{IN}					
Operating Input Voltage Range[5]			1.8		10	V
Specified Input Voltage Range		V_{IN} > 1.8V	V_{OUT} + 0.4		10	V
Over Temperature		**V_{IN} > 1.8V**	**V_{OUT} + 0.6**		**10**	V
TEMPERATURE RANGE						
Specified Range	T_J		–40		+85	°C
Operating Range	T_J		–55		+125	°C
Storage Range	T_A		–65		+150	°C
Thermal Resistance						
SOT23-5 Surface-Mount	θ_{JA}	Junction-to-Ambient		200		°C/W
SO-8 Surface-Mount	θ_{JA}	Junction-to-Ambient		150		°C/W
SOT223-5 Surface-Mount	θ_{JC}	Junction-to-Case		15		°C/W
	θ_{JA}	Junction-to-Ambient		See Figure 8		°C/W

NOTES: (1) The REG102 does not require a minimum output capacitor for stability, however, transient response can be improved with proper capacitor selection.
(2) Dropout voltage is defined as the input voltage minus the output voltage that produces a 2% change in the output voltage from the value at V_{IN} = V_{OUT} + 1V at fixed load.
(3) Current limit is the output current that produces a 10% change in output voltage from V_{IN} = V_{OUT} + 1V and I_{OUT} = 5mA.
(4) For V_{ENABLE} > 6.5V, see typical characteristic *I_{ENABLE} vs V_{ENABLE}*.
(5) The REG102 no longer regulates when V_{IN} < V_{OUT} + $V_{DROP\ (MAX)}$. In dropout, the impedance from V_{IN} to V_{OUT} is typically less than 1Ω at T_J = +25°C.

I.2.6. Diodo BAS16XV2T1G

BAS16XV2T1G, BAS16XV2T5G, SBAS16XV2T1G

Switching Diode

Features

- High−Speed Switching Applications
- Lead Finish: 100% Matte Sn (Tin)
- Qualified Reflow Temperature: 260°C
- Extremely Small SOD−523 Package
- S Prefix for Automotive and Other Applications Requiring Unique Site and Control Change Requirements; AEC−Q101 Qualified and PPAP Capable
- These Devices are Pb−Free, Halogen Free/BFR Free and are RoHS Compliant

MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Continuous Reverse Voltage	V_R	100	V
Continuous Forward Current	I_F	200	mA
Peak Forward Surge Current	$I_{FM(surge)}$	500	mA
Repetitive Peak Forward Current	I_{FRM}	500	mA
Non−Repetitive Peak Forward Current (Square Wave, T_J = 25°C prior to surge) t = 1 μs t = 1 ms t = 1 s	I_{FSM}	 4.0 1.0 0.5	A

Stresses exceeding Maximum Ratings may damage the device. Maximum Ratings are stress ratings only. Functional operation above the Recommended Operating Conditions is not implied. Extended exposure to stresses above the Recommended Operating Conditions may affect device reliability.

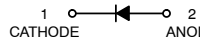
THERMAL CHARACTERISTICS

Characteristic	Symbol	Max	Unit
Total Device Dissipation, (Note 1) T_A = 25°C Derate above 25°C	P_D	200 1.57	mW mW/°C
Thermal Resistance, Junction−to−Ambient	$R_{\theta JA}$	635	°C/W
Junction and Storage Temperature	T_J, T_{stg}	−55 to 150	°C

1. FR-5 Minimum Pad.

ON Semiconductor®

http://onsemi.com



MARKING DIAGRAM

SOD−523
CASE 502
PLASTIC

A6 = Specific Device Code
M = Date Code
▪ = Pb−Free Package
(Note: Microdot may be in either location)

ORDERING INFORMATION

Device	Package	Shipping†
BAS16XV2T1G	SOD−523 (Pb−Free)	3000 / Tape & Reel
BAS16XV2T5G	SOD−523 (Pb−Free)	8000 / Tape & Reel
SBAS16XV2T1G	SOD−523 (Pb−Free)	3000 /T ape & Reel

†For information on tape and reel specifications, including part orientation and tape sizes, please refer to our Tape and Reel Packaging Specifications Brochure, BRD8011/D.

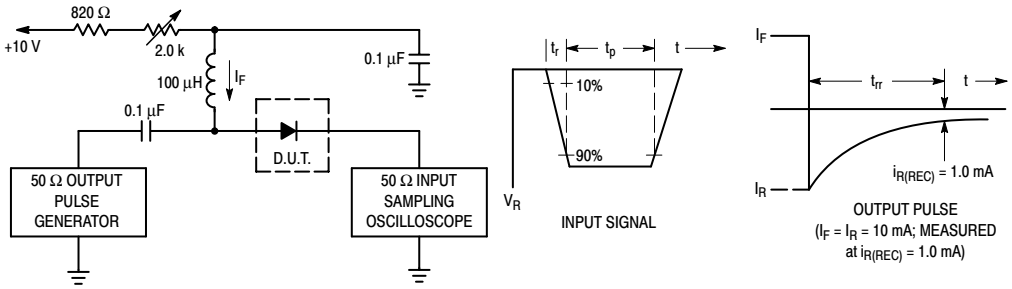
April, 2013 − Rev. 7

Publication Order Number:
BAS16XV2T1/D

BAS16XV2T1G, BAS16XV2T5G, SBAS16XV2T1G

ELECTRICAL CHARACTERISTICS (T_A = 25°C unless otherwise noted)

Characteristic	Symbol	Min	Max	Unit
OFF CHARACTERISTICS				
Reverse Voltage Leakage Current	I_R			μA
(V_R = 100 V)		–	1.0	
(V_R = 75 V, T_J = 150°C)		–	50	
(V_R = 25 V, T_J = 150°C)		–	30	
Reverse Breakdown Voltage (I_{BR} = 100 μA)	$V_{(BR)}$	100	–	V
Forward Voltage	V_F			mV
(I_F = 1.0 mA)		–	715	
(I_F = 10 mA)		–	855	
(I_F = 50 mA)		–	1000	
(I_F = 150 mA)		–	1250	
Diode Capacitance (V_R = 0, f = 1.0 MHz)	C_D	–	2.0	pF
Forward Recovery Voltage (I_F = 10 mA, t_r = 20 ns)	V_{FR}	–	1.75	V
Reverse Recovery Time (I_F = I_R = 10 mA, R_L = 50 Ω)	t_{rr}	–	6.0	ns
Stored Charge (I_F = 10mA to V_R = 5.0V, R_L = 500 Ω)	Q_S	–	45	pC



Notes: 1. A 2.0 kΩ variable resistor adjusted for a Forward Current (I_F) of 10 mA.
2. Input pulse is adjusted so $I_{R(peak)}$ is equal to 10 mA.
3. t_p » t_{rr}

Figure 1. Recovery Time Equivalent Test Circuit

I.2.7. Pulsador FSMSM

tyco Electronics — FSMSM/JM Series — Alcoswitch

Tactile Switches, .138 [3.5] x .236 [6], Surface Mount

MATERIAL SPECIFICATIONS:
Contacts .....Stainless steel, silver clad plate
Case Material .....FSMSM: PPS
Actuator Material .....FSMSM: Nylon
FSMJM: PPS
Terminals .....Copper alloy, silver plate or tin plate
Cover.....Stainless steel

ENVIRONMENTAL SPECIFICATIONS:
Operating Temperature.....-40°F to +185°F (-40°C to +85°C)
Storage Temperature.....-40°F to +185°F (-40°C to +85°C)
Solder Heat Resistance.....FSMSM IR Soldering EIA-364-56,
Procedure 5 Level 1, 215° C
FSMJM IR Soldering per EIA 364-56
Procedure 5, level 2, 464°F (240°C)

TYPICAL PERFORMANCE CHARACTERISTICS:
Contact Rating .....50 mA @ 12 VDC
Initial Contact Resistance.....100 Milliohms max.
Insulation Resistance .....100 Megohms min. @ 500 VDC
Dielectric Strength .....500 VAC for 1 minute
Actuator Travel.....010"(.254)
Life Expectancy .....20,000-40,000 Cycles

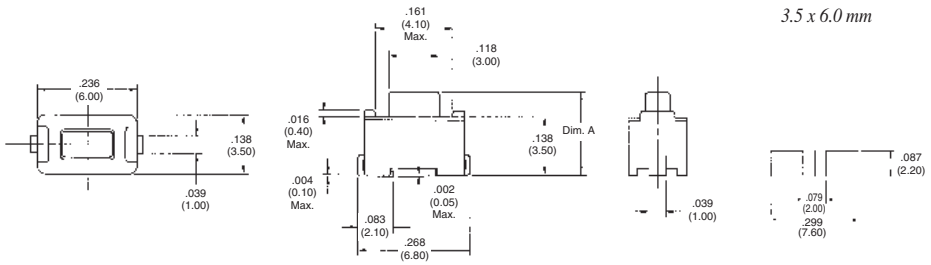
3.5 x 6.0 mm


FSMSM

Tyco Electronics	Alcoswitch	Type			Packaging	Actuation Force	Life Cycle
1437566-3	FSMSM	SPST	Off	(On)	Bulk	180±50	20,000
1437566-4	FSMSMTR				Reel	180±50	20,000

(On) indicates momentary position.

3.5 x 6.0 mm



Tyco Electronics	Alcoswitch	Type	Dim. A	Actuation Force	Packaging	Life Cycle
147873-1	FSMJM5.0ATR	SPST	.197 (5.00)	180±50	Reel	40,000
147873-2	FSMJM4.3ATR	SPST	.169 (4.30)	180±50	Reel	40,000
147873-3	FSMJM3.5ATR	SPST	.138 (3.50)	180±50	Reel	40,000
147942-1	FSMJM4.3	SPST	.169 (4.30)	130±50	Bulk	40,000
147942-2	FSMJM4.3TR	SPST	.169 (4.30)	130±50	Reel	40,000

E9

Catalog 1308390
Issued 9-04

www.tycoelectronics.com

Dimensions are in inches and millimeters unless otherwise specified. Values in parentheses or brackets are metric equivalents.

Dimensions are shown for reference purposes only. Specifications subject to change.

USA: 1-800-522-6752
Canada: 1-905-470-4425
Mexico: 01-800-733-8926
C. America: 52-55-5-729-0425

South America: 55-11-3611-1514
Hong Kong: 852-2735-1628
Japan: 81-44-844-8013
UK: 44-141-810-8967

E

FSMSM/JM Series

I.3. Termopar

Comark C20 Thermometers

COMARK

Performance with Style

Stylish new case, waterproof to IP67 standards

Countdown timer

Exceptional battery life of up to 14000 hours

Permanent clock display

Choice of models for use with thermistor/type T thermocouple probes or type K thermocouple probes

°C or °F scales

Data hold function

Easy to use keypad and large LCD display

Auto switch-off

Built-in protective boot

Interactive cell-phone style menu for selection of

- Scales
- Clock adjustment
- Countdown timer
  - Can be stopped during countdown
  - Uses include monitoring how long probe is inserted to obtain accurate reading or monitoring cooking and cooling times
  - Bleeps to advise end of timed period

Selectable auto switch-off, with choice of time period before the instrument turns off

- 3 minutes
- 10 minutes
- 30 minutes

Comark is the leading manufacturer and supplier of a wide range of electronic measurement instruments for temperature, humidity, pressure and pH.

Click on www.comarkltd.com for more information

Comark C20 Thermometers

COMARK

C20 Model Range

The two C20 models are: -

C22

- Use with wide range of Comark thermistor or type T thermocouple probes
- Automatic recognition of probe sensor type

C28 K Type

- For use with wide range of Comark type K thermocouple probes
- Measurement range covers food or industrial applications

C20 Model Planner

Model	C22	C28 K Type
Countdown Timer	Yes	Yes
Data Hold	Yes	Yes
Real Time Clock	Yes	Yes
Scales	°C/°F	°F/°C
Connector	Lumberg	Sub-miniature
Thermistor	Yes	No
Type T Thermocouple	Yes	No
Type K Thermocouple	No	Yes
Auto Off	Yes	Yes
IP67	Yes	Yes

C20 Thermometers with BioCote® Protection

BioCote® is a silver based anti-microbial agent that is impregnated into the instrument case during moulding. BioCote® effectively inhibits the function, growth and reproduction of a wide range of micro-organisms and its protection is increasingly accepted as part of HACCP, due diligence and health and safety procedures to reduce cross contamination. BioCote® protection lasts for the life of the instrument, because the agent is present throughout the case plastic and cannot rub off or be washed or leached out.

BioCote® active ingredients are registered with the US Environmental Protection Agency (EPA).

Although BioCote® inhibits the growth of micro-organisms on the instrument case, it does not protect individuals against such harmful organisms or remove the need to maintain the highest standards of personal and product hygiene and cleanliness.

Distributed by:

C20 SERIES TECHNICAL SPECIFICATIONS

Sensor Type	
C22	Thermistor/Type T thermocouple
C28 K Type	Thermocouple Type K
Measurement Range	
C22	
Thermistor	–50°C to +150°C, –58°F to +302°F
Type T	–200°C to +400°C, –328°F to +752°F
C28 K Type	–200°C to +600°C, –328°F to +1112°F
Scales	°C and °F
Displayed Resolution	
> –100°	0.1°
≤ –100°	1°
System Accuracy at +23°C/+73°F	between 0°C to +70°C, +32°F to +158°F
Thermistor	<±0.3°C/0.6°F
Type T*	±0.5°C/0.9°F
Typical accuracy with a Comark probe	
Instrument Accuracy at +23°C/+73°F, full range	
Type K	0.1% ± 0.2°C/0.4°F
Operating Temperature Range	–20°C to +50°C/–4°F to +122°F
Display	4 digit, 12.5mm LCD
Countdown Timer Interval	1 second to 24 hours
Environmental Protection	IP67 BS EN 60529 IEC 529
Battery	2 x Type I.E.C. LR6 Size AA
Battery Life (continuous)	
Using Thermocouple Probes	Up to 7000 hours
Using Thermistor Probes	Up to 14000 hours
Dimensions	
Length	152mm/6 inches
Width	58mm/2.3 inches
Depth	22/27mm/0.9/1.1 inches
Weight	178g/6.3 oz

WARRANTY

All Comark Instruments have a minimum one-year warranty unless otherwise stated. The warranty period for temperature probes is six months and all other probes and electrodes are unwarranted because the conditions of use are beyond our control.

The Comark warranty covers manufacturing defects and component failure and applies worldwide. The warranty does not affect your statutory rights.

In line with our policy of continuous development we reserve the right to alter any product specifications without notice.

All products are covered by our Quality Management System which is compliant with BS EN ISO 9001:2000 for the design, manufacture, supply, service, repair and recalibration of electronic measuring instruments and equipment.

Comark has an accredited UKAS calibration laboratory for temperature and humidity measurement and can offer full service and recalibration facilities.

Comark Limited Gunnels Wood Road, Stevenage, Hertfordshire SG1 2TA England
Tel: 01438 (+44 1438) 367367
Fax: 01438 (+44 1438) 367400
Email enquiries UK and Ireland: salesuk@comarkltd.com
Email enquiries International: salesint@comarkltd.com

Comark Instruments Inc.
9710 SW Sunshine Court, Beaverton, OR 97005, USA.
Tel: (503) 643 5204 Fax: (503) 644 5859
Email: sales@comarkUSA.com

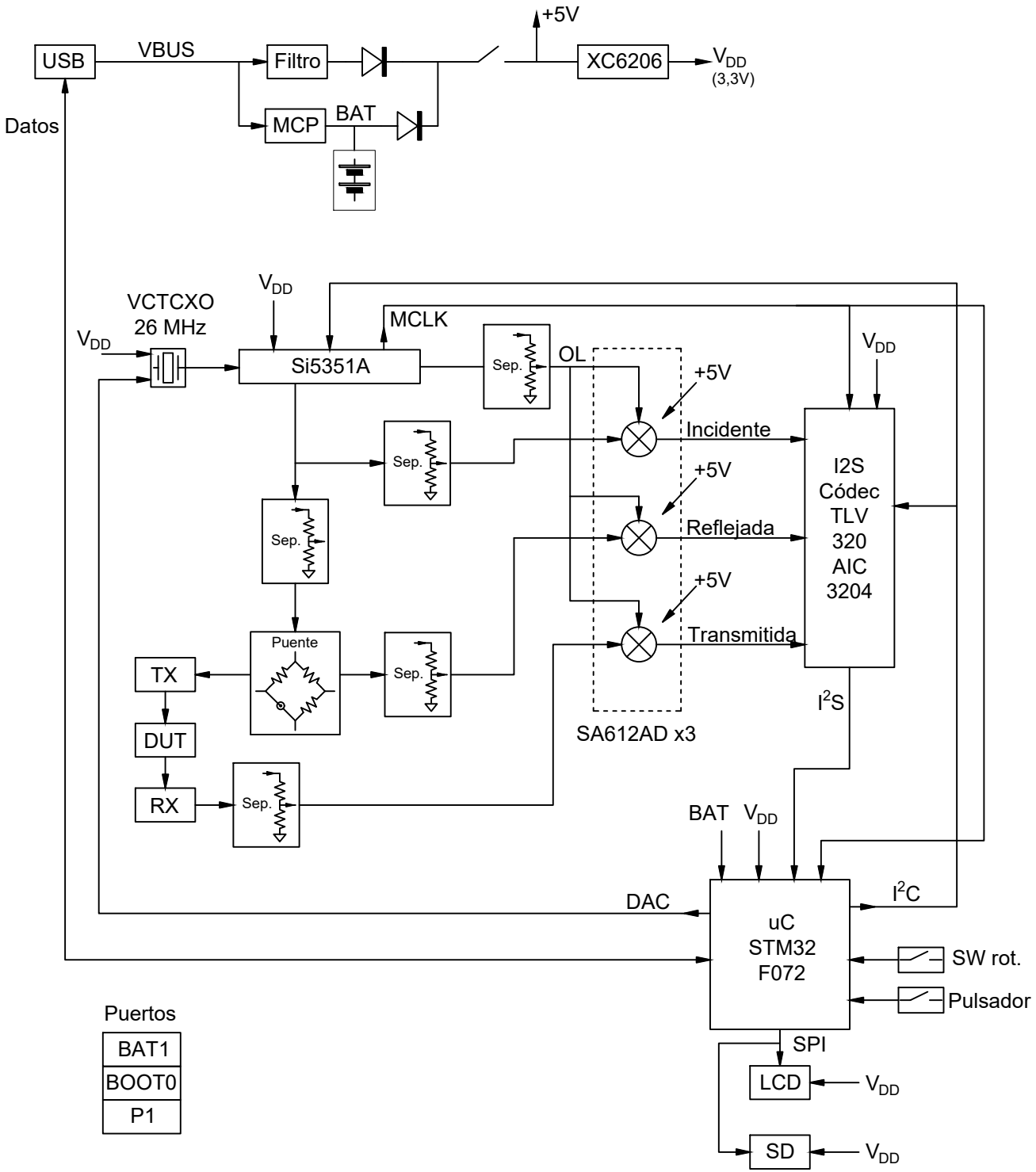
COMARK
a VIDA GROUP company

Click on www.comarkltd.com for more information

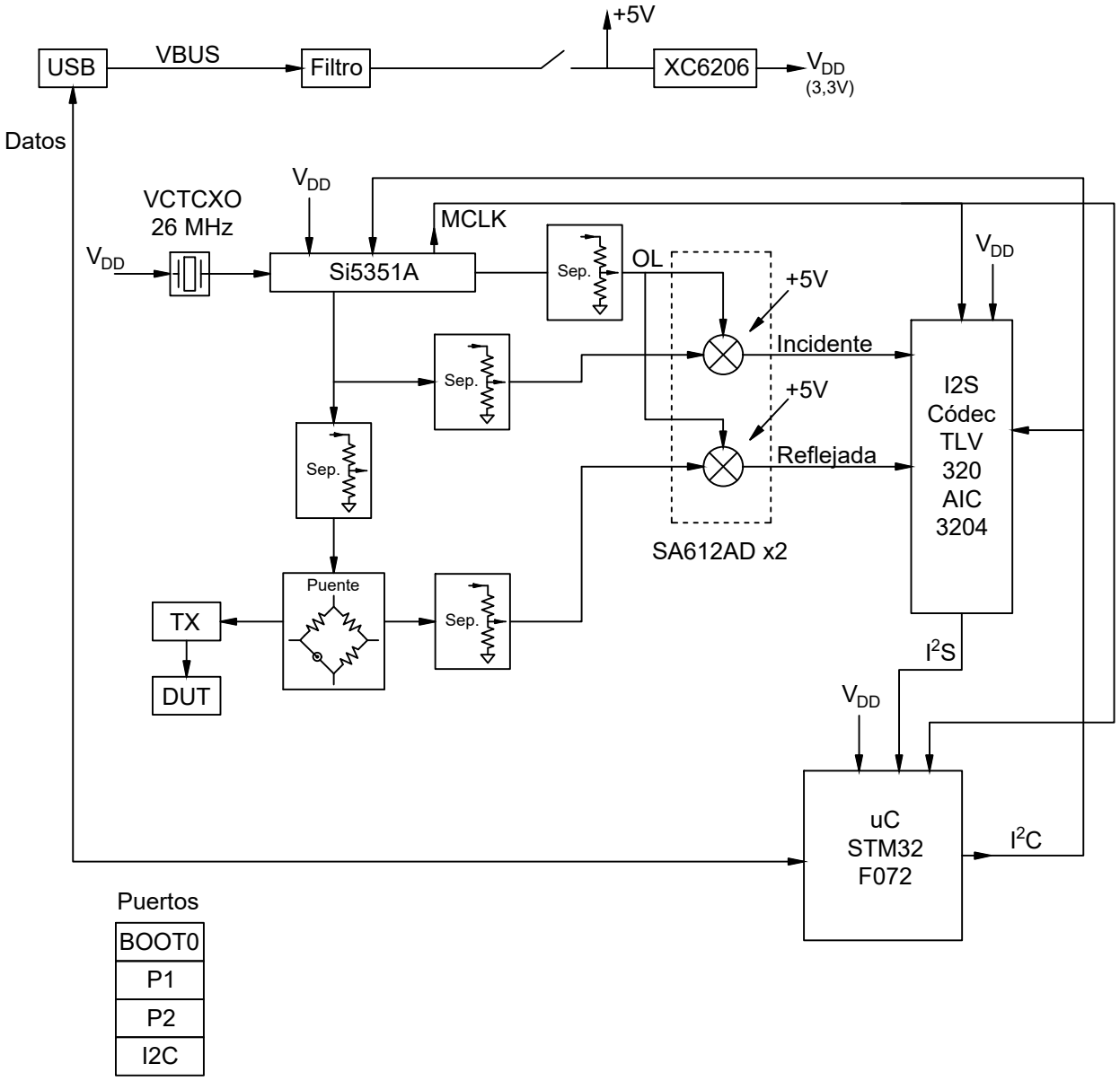
Anexo II

Diagramas de bloques

II.1. Diagrama de bloques del VNA de 2 puertos


+5V
USB
VBUS
Filtro
XC6206
V_{DD}
(3,3V)
Datos
MCP
BAT
VCTCXO
26 MHz
V_{DD}
MCLK
Si5351A
Sep.
OL
+5V
Incidente
Reflejada
Transmitida
I2S
Códec
TLV
320
AIC
3204
Puente
TX
DUT
RX
SA612AD x3
I^2S
BAT
DAC
I^2C
uC
STM32
F072
SW rot.
Pulsador
SPI
LCD
SD
Puertos
BAT1
BOOT0
P1

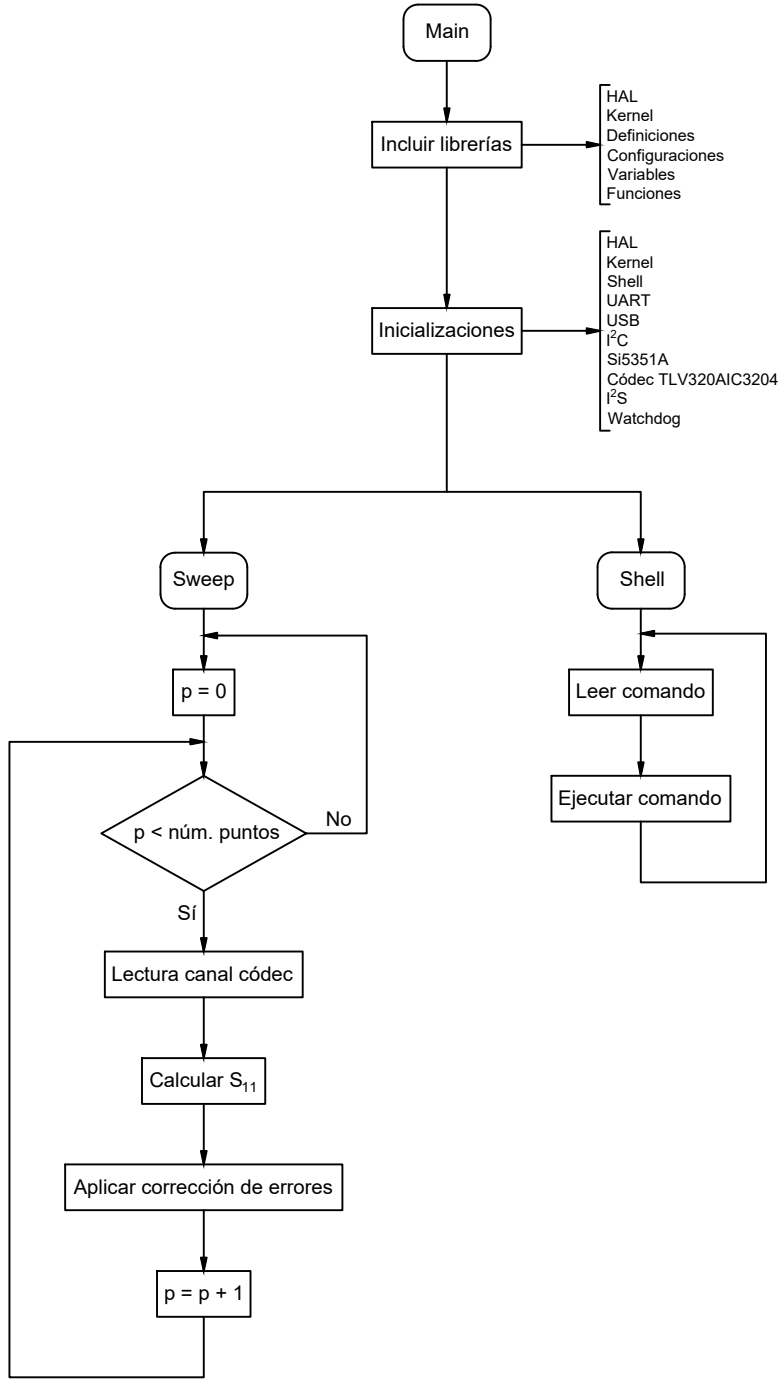
II.2. Diagrama de bloques del VNA de 1 puerto



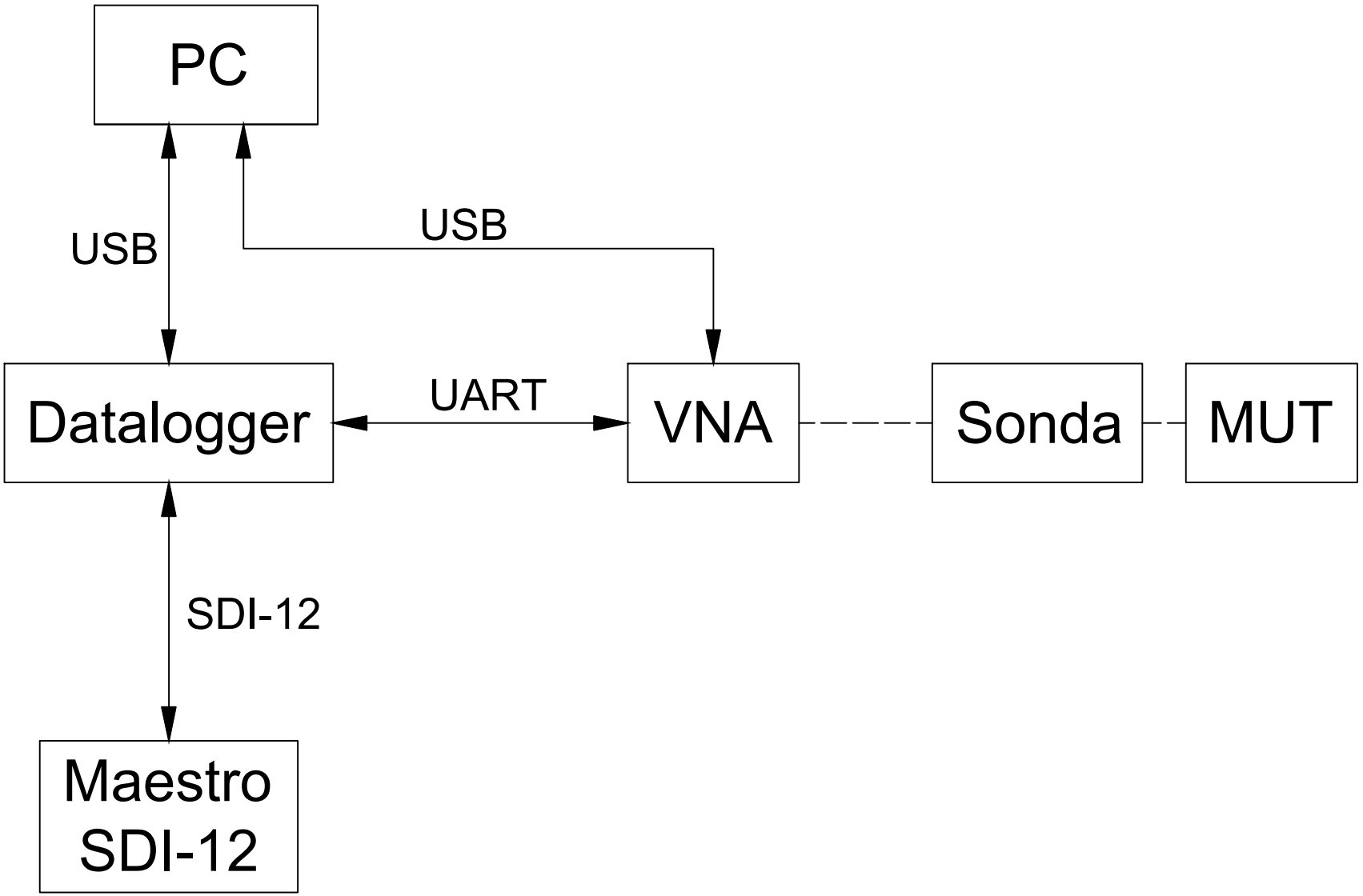
Puertos

BOOT0
P1
P2
I2C

II.3. Flujograma del código del microcontrolador del VNA


Main
Incluir librerías
HAL
Kernel
Definiciones
Configuraciones
Variables
Funciones
Inicializaciones
HAL
Kernel
Shell
UART
USB
I2C
Si5351A
Códec TLV320AIC3204
I2S
Watchdog
Sweep
p = 0
p < núm. puntos
No
Sí
Lectura canal códec
Calcular S11
Aplicar corrección de errores
p = p + 1
Shell
Leer comando
Ejecutar comando

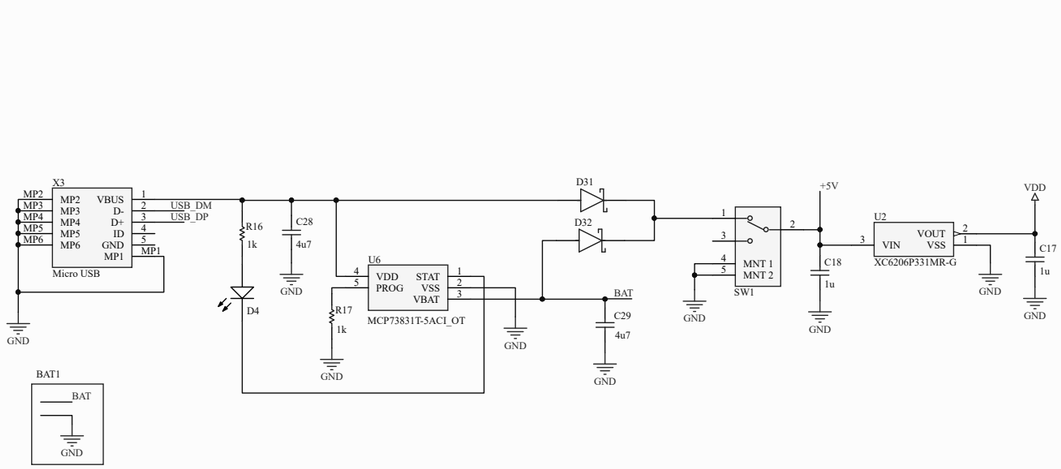
II.4. Diagrama de bloques para la realización de ensayos con el VNA

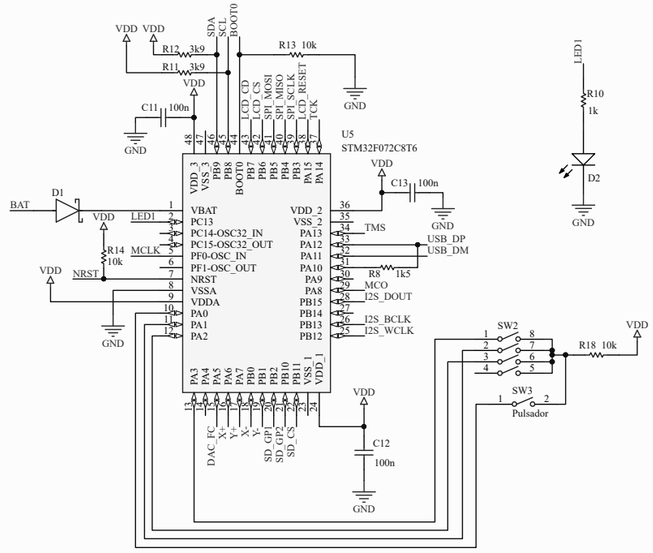

PC
USB
USB
Datalogger
UART
VNA
Sonda
MUT
SDI-12
Maestro
SDI-12

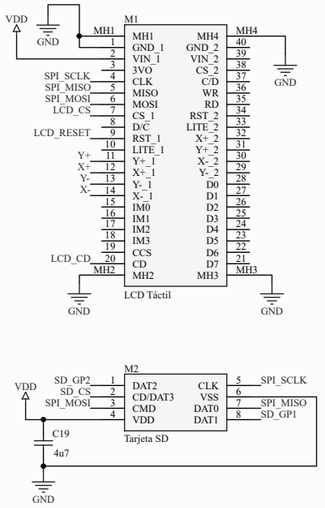
Anexo III

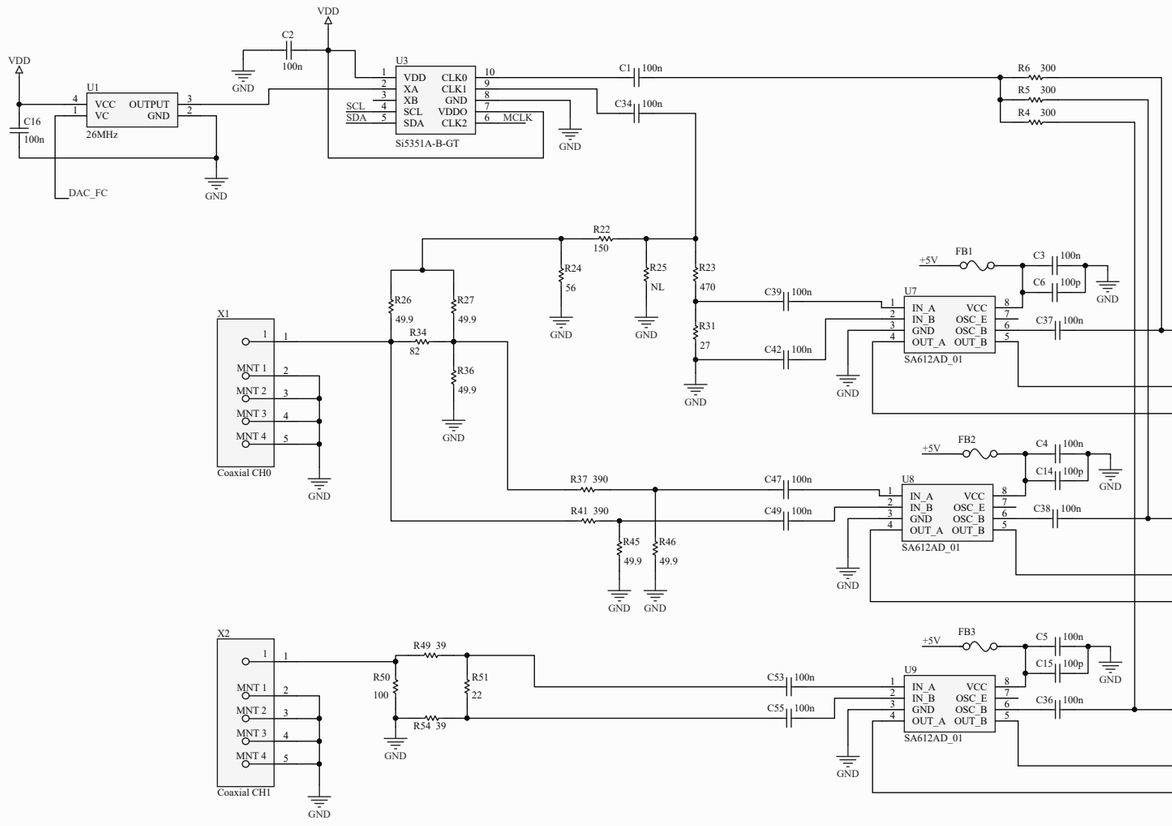
Esquemáticos

III.1. Esquemático del VNA de 2 puertos



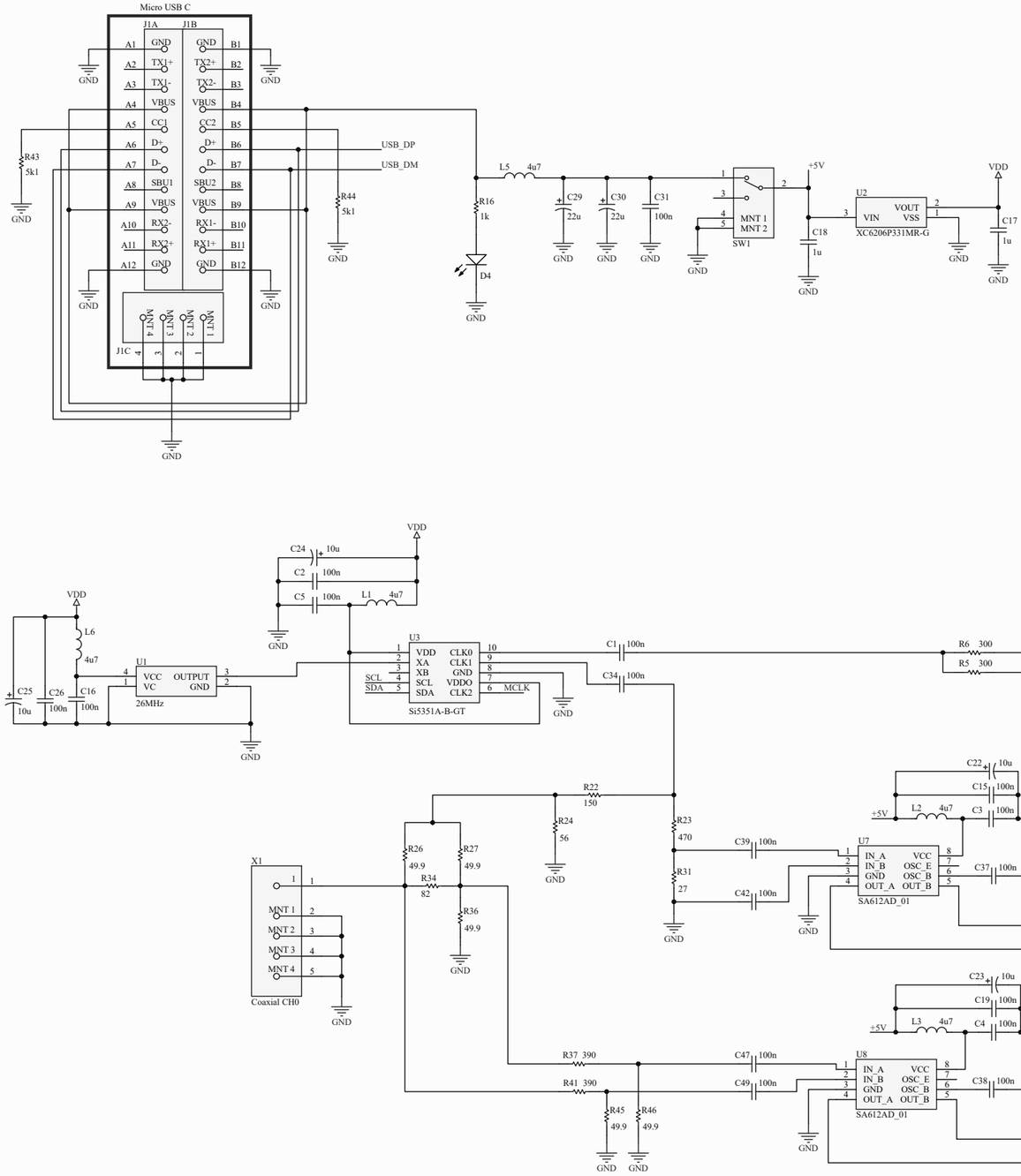


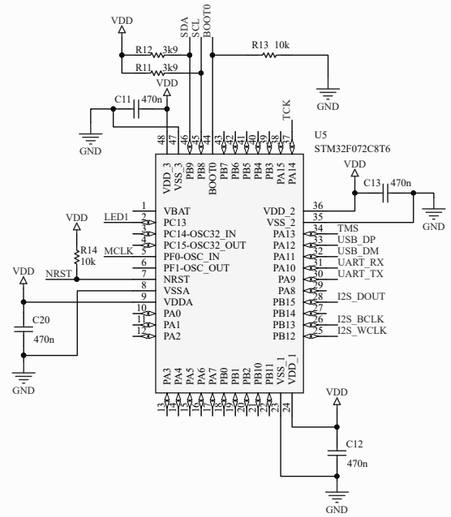




Title	VNA		
Size	Number		Revision
A3	1		1
Date:	03/04/2021	Sheet 1 of 1	
File:	C:\Users\...\VNA_SCH.SchDoc	Drawn By:	Javier Garrido

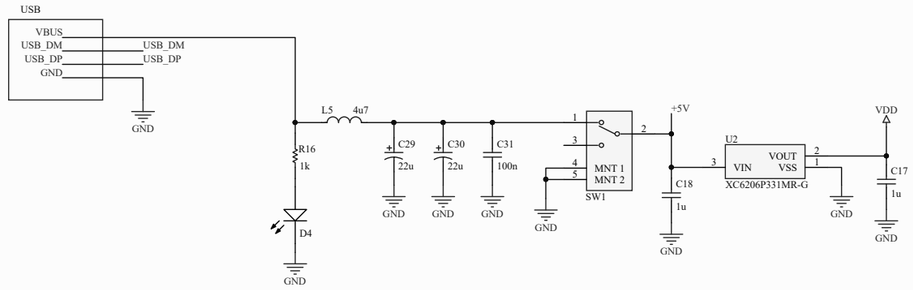
III.2. Esquemático del VNA de 1 puerto. Primera versión

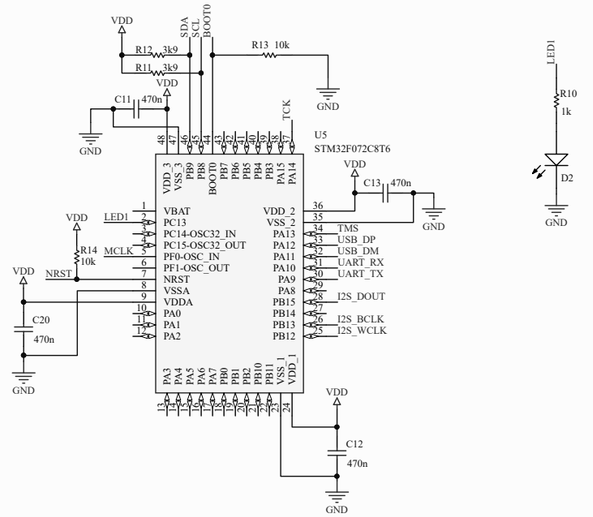


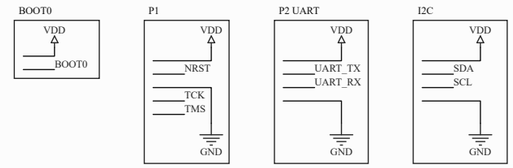


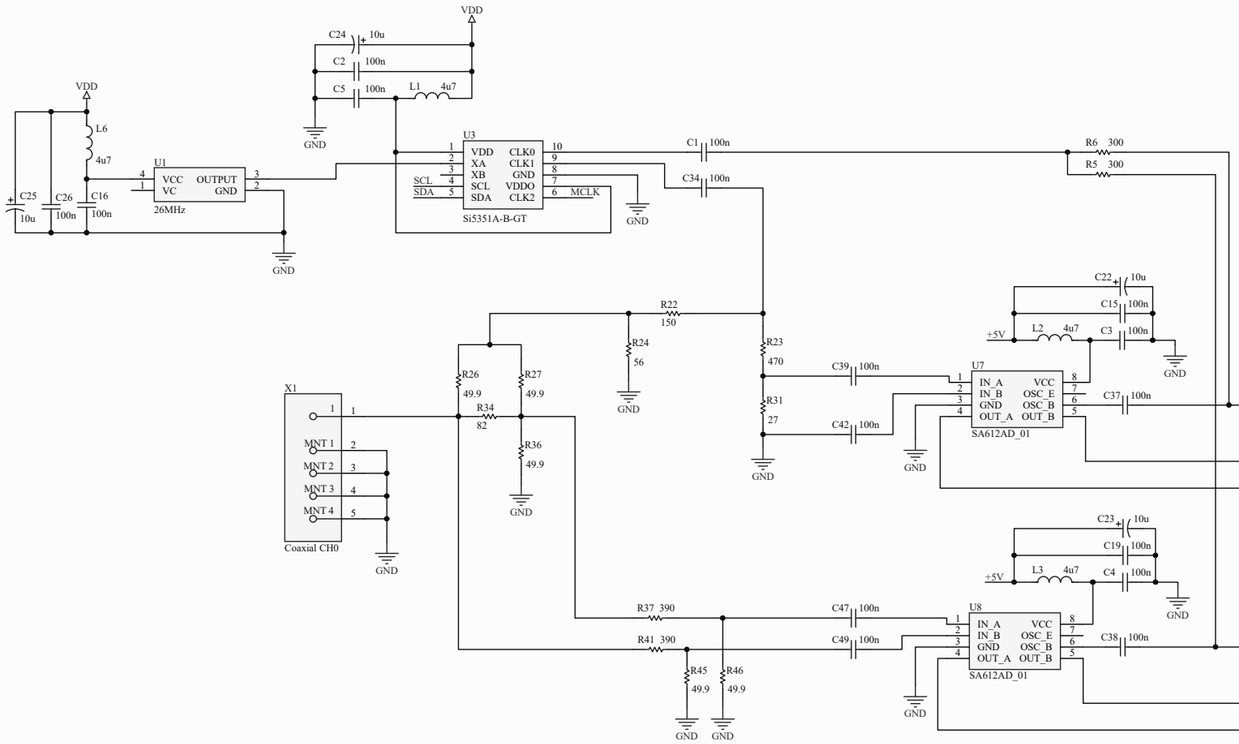
Title	VNA Mini
Size	A3
Number	1
Revision	1
Date:	11/04/2021
Sheet	1 of 1
File:	C:\Users\...\VNA_mini_SCH.SchDoc
Drawn By:	Javier Garrido

III.3. Esquemático del VNA de 1 puerto. Última versión



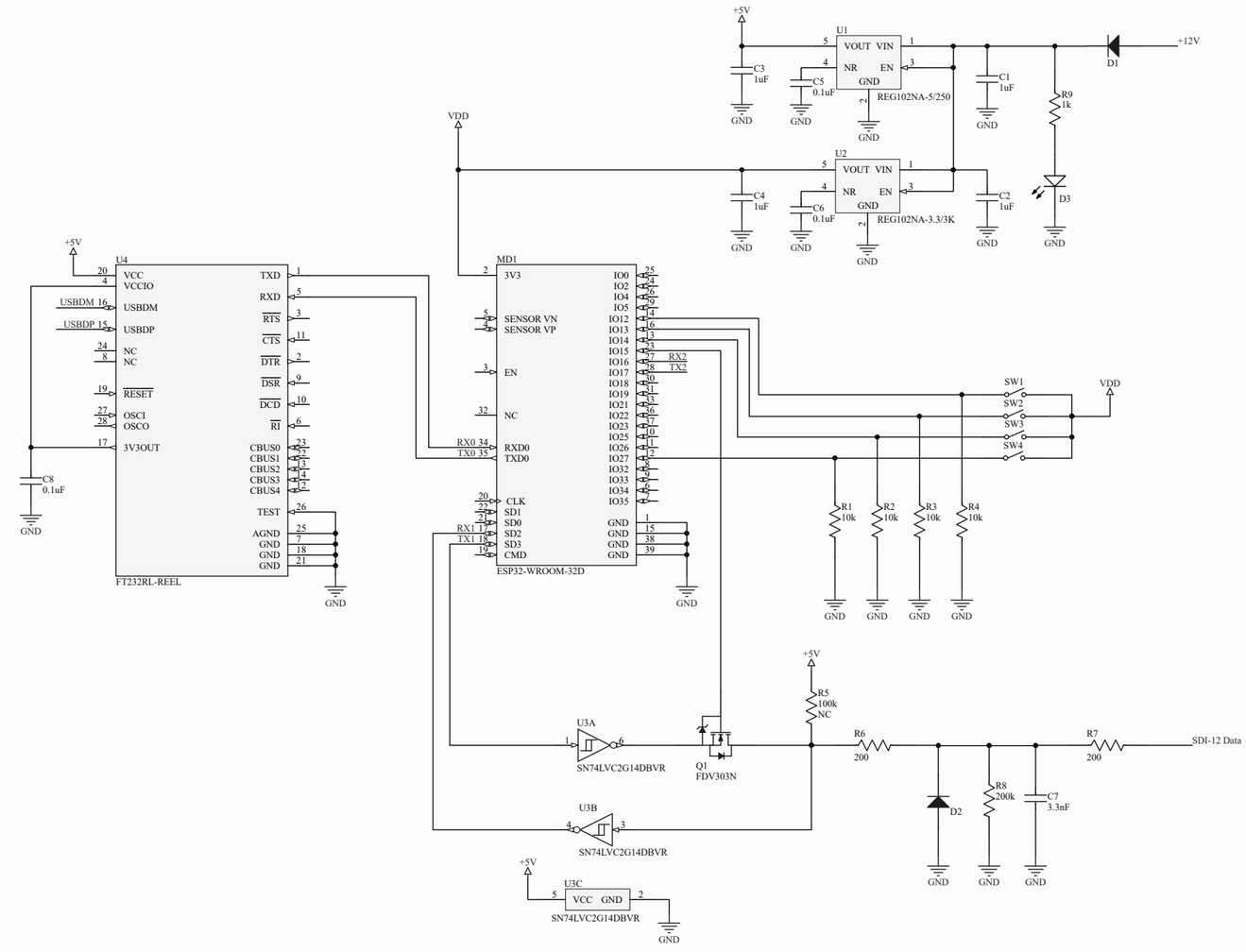






Title	VNA Mini
Size	A3
Number	1
Revision	1
Date:	25/06/2021
Sheet	1 of 1
File:	C:\Users\...\VNA_mini_SCH.SchDoc
Drawn By:	Javier Garrido

III.4. Esquemático del Datalogger

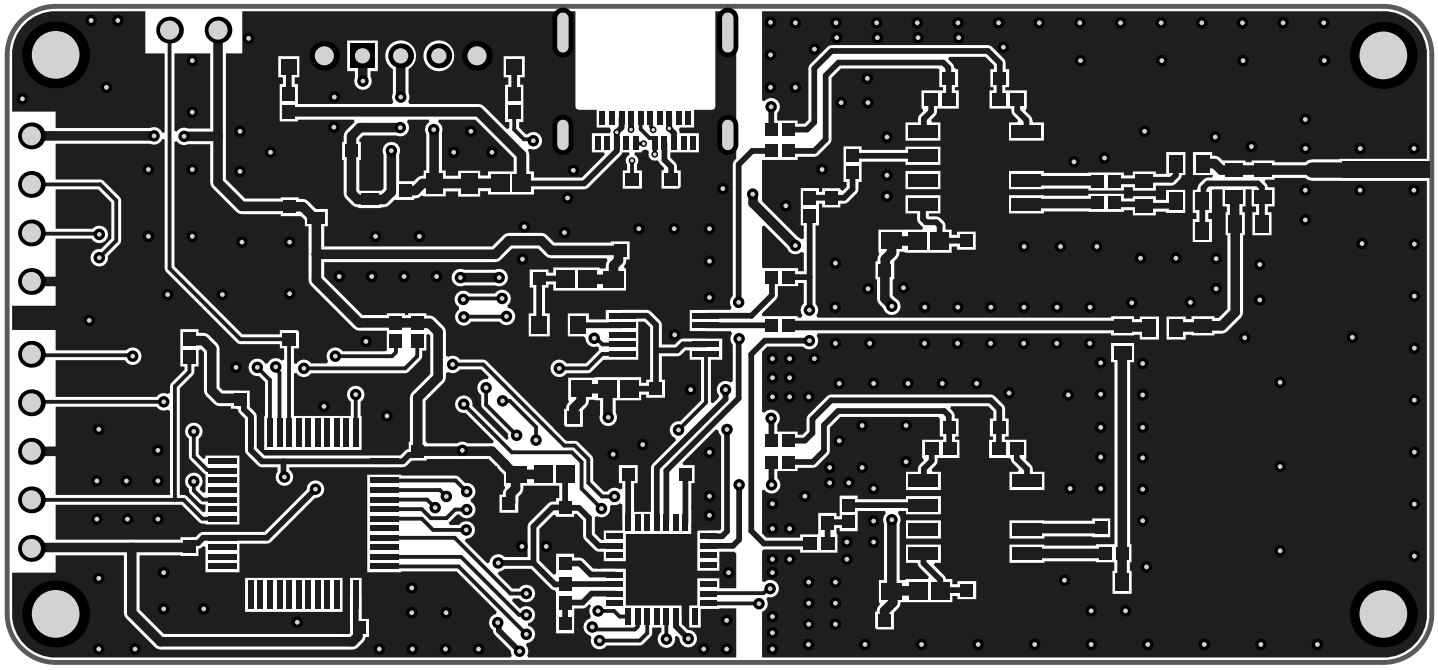


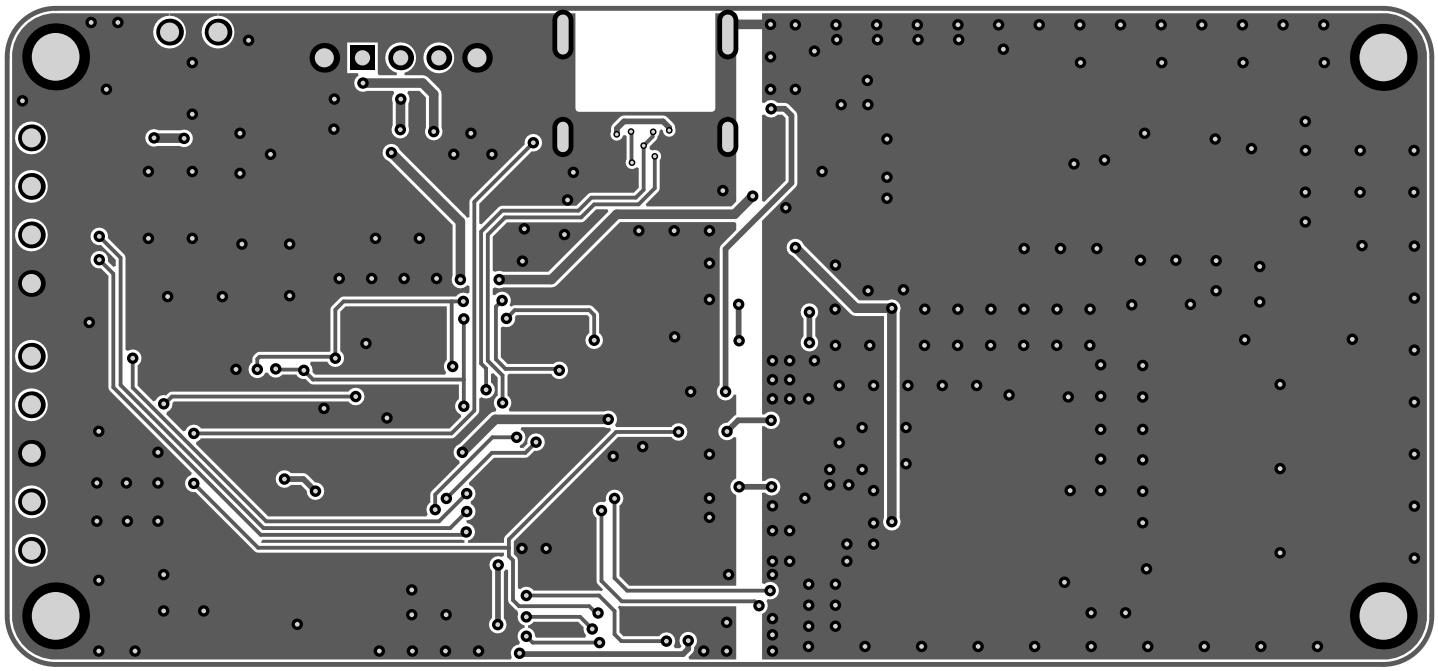
Title	Datalogger VNA
Size	A3
Number	1
Revision	1
Date:	04/07/2021
Sheet	1 of 1
File:	C:\Users\...\ESP32_Datalogger_SCH.SchDoc
Drawn By:	Javier Garrido

Anexo IV

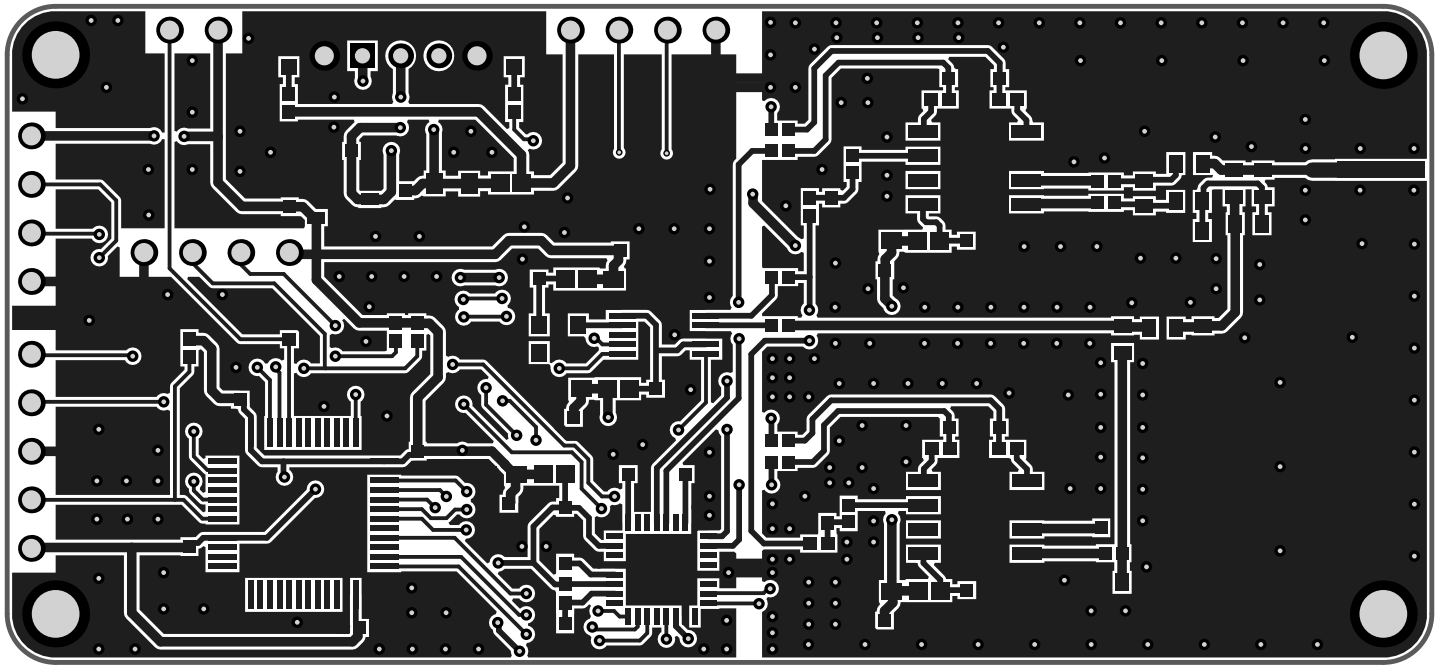
Placas de circuito impreso (PCB)

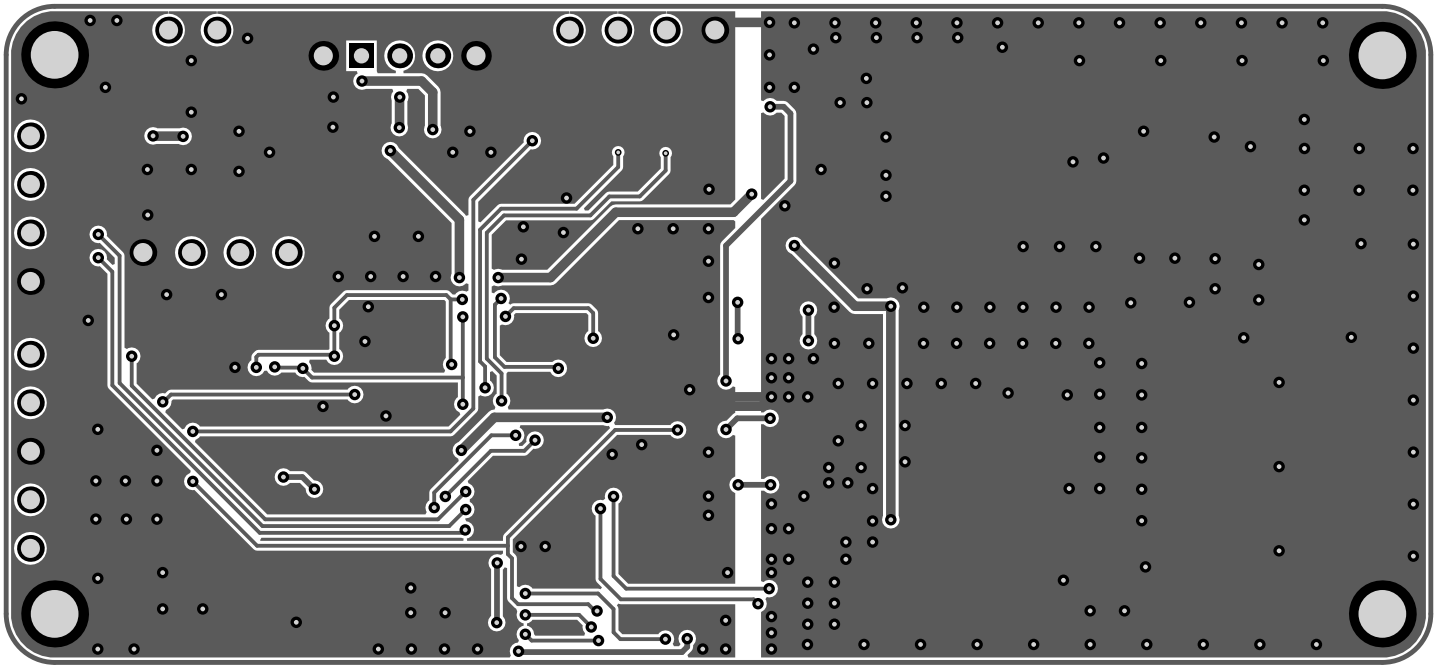
IV.1. PCB del VNA. Primera versión



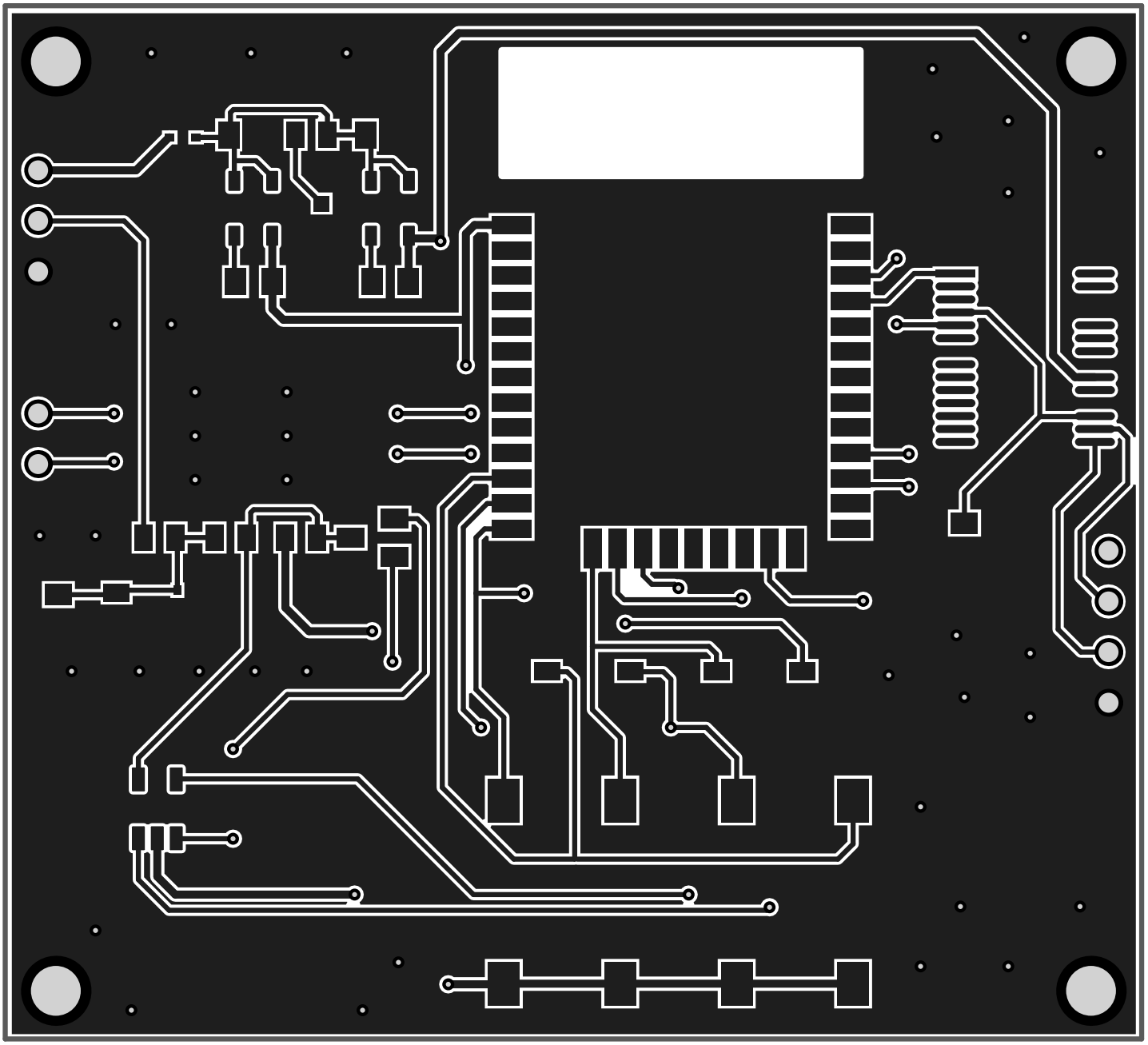


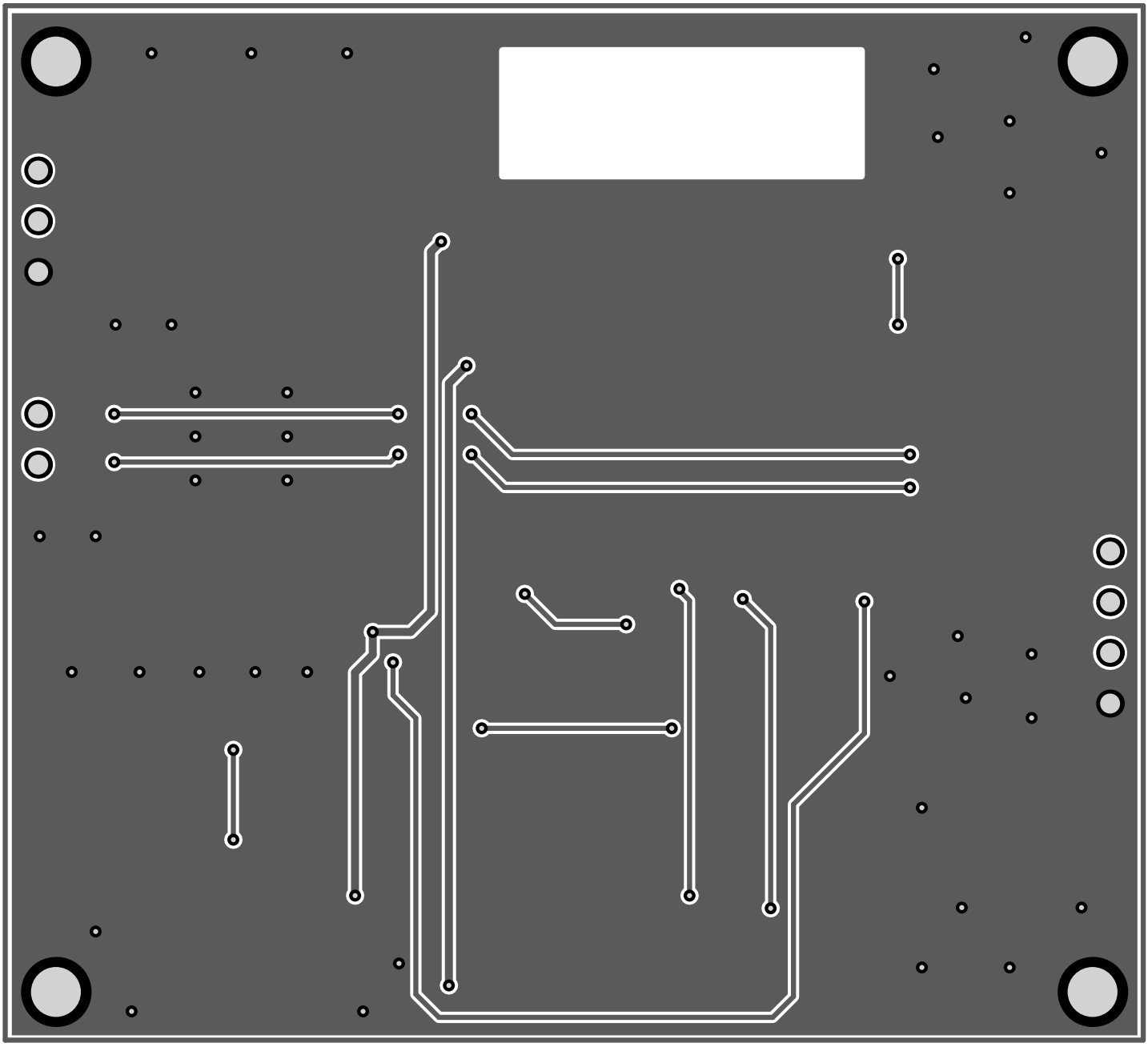
IV.2. PCB del VNA. Última versión



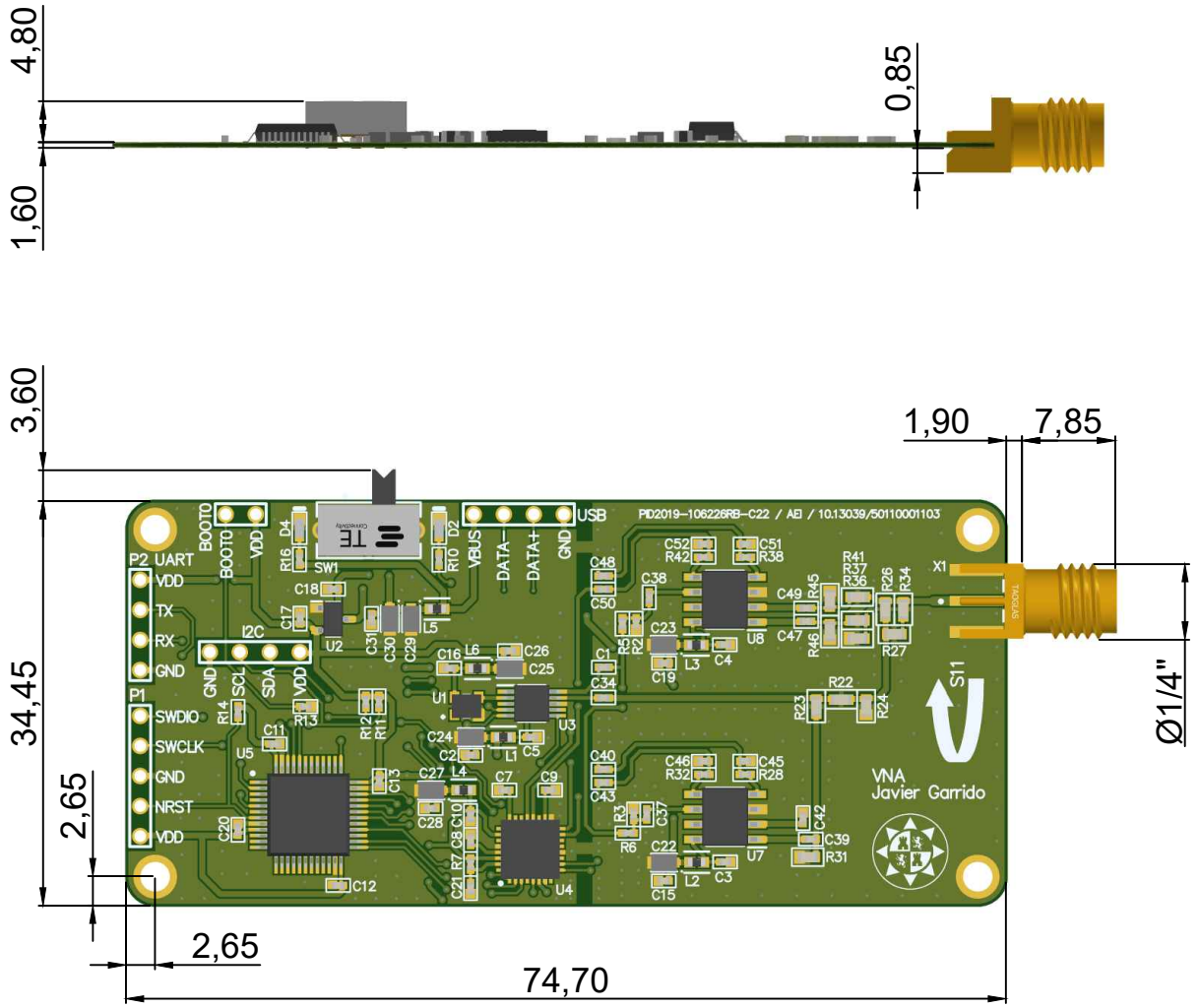


IV.3. PCB del Datalogger





IV.4. Dimensiones del VNA (mm)


4,80
1,60
0,85
3,60
34,45
2,65
2,65
74,70
1,90
7,85
Ø1/4"
P2 UART
VDD
TX
RX
GND
P1
SWDIO
SWCLK
GND
NRST
VDD
I2C
GND
SCL
SDA
VDD
BOOT0
BOOT0
VDD
SW1
USB
VBUS
DATA-
DATA+
GND
X1
S11
VNA
Javier Garrido
PID2019-106226RB-C22 / AEI / 10.13039/50110001103

Anexo V

Códigos

V.1. Código C del *main.c* del microcontrolador del VNA

```
/*
 * VNA
 */

#include "ch.h"
#include "hal.h"
#include "usbcfg.h"
#include "si5351.h"
#include "nanovna.h"
#include "fft.h"

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


static BaseSequentialStream *shell_stream;
bool enable_usart_data = false;

#define ENABLE_TRANSFORM_COMMAND
#define ENABLE_SCANBIN_COMMAND
#define ENABLE_USART_COMMAND

#define VNA_SHELL_FUNCTION(command_name) static void ...
    command_name(int argc, char *argv[])

static void apply_CH0_error_term_at(int i);
static void apply_edelay(void);

static uint16_t get_sweep_mask(void);
static void cal_interpolate(void);
static void update_frequencies(bool interpolate);
static int  set_frequency(uint32_t freq);
static void set_frequencies(uint32_t start, uint32_t stop, ...
    uint16_t points);
static bool sweep(bool break_on_operation, uint16_t ch_mask);
static void transform_domain(void);

uint8_t sweep_mode = SWEEP_ENABLE;

volatile uint16_t wait_count = 0;
// Punto de barrido actual
static uint16_t p_sweep = 0;
```

```
42 // Buffer I2S
43 static int16_t rx_buffer[AUDIO_BUFFER_LEN * 2];
44 // Datos medidos en el barrido
45 float measured[1][POINTS_COUNT][2];
46 uint32_t frequencies[POINTS_COUNT];
47
48
49 // Hilo para el sweep
50 static THD_WORKING_AREA(waThread1, 768);
51 static THD_FUNCTION(Thread1, arg)
52 {
53   while (1) {
54     wdgReset(&WDGD1);
55     bool completed = false;
56     if (sweep_mode&(SWEEP_ENABLE|SWEEP_ONCE)) {
57       completed = sweep(true, get_sweep_mask());
58       sweep_mode&=~SWEEP_ONCE;
59     } else {
60       __WFI();
61     }
62     // Process collected data only if scan completed
63     if ((sweep_mode & SWEEP_ENABLE) && completed) {
64       if (electrical_delay != 0) apply_edelay();
65       if ((domain_mode & DOMAIN_MODE) == DOMAIN_TIME) ...
           transform_domain();
66     }
67   }
68 }
69
70
71 static inline void
72 pause_sweep(void)
73 {
74   sweep_mode &= ~SWEEP_ENABLE;
75 }
76
77 static inline void
78 resume_sweep(void)
79 {
80   sweep_mode |= SWEEP_ENABLE;
81 }
82
83 void
84 toggle_sweep(void)
85 {
86   sweep_mode ^= SWEEP_ENABLE;
87 }
88
89
```

```
90  // Dominio del tiempo
91  static float
92  bessel0(float x)
93  {
94    const float eps = 0.0001;
95
96    float ret = 0;
97    float term = 1;
98    float m = 0;
99
100   while (term  > eps * ret) {
101     ret += term;
102     ++m;
103     term *= (x*x) / (4*m*m);
104   }
105   return ret;
106 }
107
108 static float
109 kaiser_window(float k, float n, float beta)
110 {
111   if (beta == 0.0) return 1.0;
112   float r = (2 * k) / (n - 1) - 1;
113   return bessel0(beta * sqrt(1 - r * r)) / bessel0(beta);
114 }
115
116 static void
117 transform_domain(void)
118 {
119   float* tmp = (float*)spi_buffer;
120
121   uint16_t window_size = sweep_points, offset = 0;
122   uint8_t is_lowpass = FALSE;
123   uint8_t td_func = domain_mode & TD_FUNC;
124   switch (td_func) {
125     case TD_FUNC_BANDPASS:
126       offset = 0;
127       window_size = sweep_points;
128       break;
129     case TD_FUNC_LOWPASS_IMPULSE:
130     case TD_FUNC_LOWPASS_STEP:
131       is_lowpass = TRUE;
132       offset = sweep_points;
133       window_size = sweep_points * 2;
134       break;
135   }
136
137   float beta = 0.0f;
138   switch (domain_mode & TD_WINDOW) {
```

```
139     case TD_WINDOW_MINIMUM:
140     //beta = 0.0f;
141       break;
142     case TD_WINDOW_NORMAL:
143       beta = 6.0f;
144       break;
145     case TD_WINDOW_MAXIMUM:
146       beta = 13.0f;
147       break;
148   }
149
150   static float window_scale = 1.0f;
151   static uint16_t td_cache = 0;
152   uint16_t td_check = (domain_mode & ...
        (TD_WINDOW|TD_FUNC))|(sweep_points<<5);
153   if (td_cache!=td_check){
154     td_cache=td_check;
155     if (td_func == TD_FUNC_LOWPASS_STEP)
156       window_scale = 1.0f;
157     else {
158       window_scale = 0.0f;
159       for (int i = 0; i < sweep_points; i++)
160         window_scale += kaiser_window(i + offset, ...
              window_size, beta);
161       window_scale = (FFT_SIZE/2) / window_scale;
162       if (td_func == TD_FUNC_BANDPASS)
163         window_scale *= 2;
164     }
165   }
166
167   uint16_t ch_mask = get_sweep_mask();
168   for (int ch = 0; ch < 1; ch++,ch_mask>>=1) {
169     if ((ch_mask&1)==0) continue;
170     memcpy(tmp, measured[ch], sizeof(measured[0]));
171     for (int i = 0; i < sweep_points; i++) {
172       float w = kaiser_window(i + offset, window_size, beta) ...
            * window_scale;
173       tmp[i * 2 + 0] *= w;
174       tmp[i * 2 + 1] *= w;
175     }
176     for (int i = sweep_points; i < FFT_SIZE; i++) {
177       tmp[i * 2 + 0] = 0.0;
178       tmp[i * 2 + 1] = 0.0;
179     }
180     if (is_lowpass) {
181       for (int i = 1; i < sweep_points; i++) {
182         tmp[(FFT_SIZE - i) * 2 + 0] = tmp[i * 2 + 0];
183         tmp[(FFT_SIZE - i) * 2 + 1] = -tmp[i * 2 + 1];
184       }
```

```
185     }
186
187     fft_inverse((float(*)[2])tmp);
188     memcpy(measured[ch], tmp, sizeof(measured[0]));
189     for (int i = 0; i < sweep_points; i++) {
190       measured[ch][i][0] /= (float)FFT_SIZE;
191       if (is_lowpass) {
192         measured[ch][i][1] = 0.0;
193       } else {
194         measured[ch][i][1] /= (float)FFT_SIZE;
195       }
196     }
197     if ((domain_mode & TD_FUNC) == TD_FUNC_LOWPASS_STEP) {
198       for (int i = 1; i < sweep_points; i++) {
199         measured[ch][i][0] += measured[ch][i - 1][0];
200       }
201     }
202   }
203 }
204
205
206 void set_power(uint8_t value){
207   if (value > SI5351_CLK_DRIVE_STRENGTH_8MA) value = ...
       SI5351_CLK_DRIVE_STRENGTH_AUTO;
208   if (current_props._power == value) return;
209   current_props._power = value;
210   // Actualizar la potencia
211   if (!(sweep_mode&SWEEP_ENABLE)) si5351_set_power(value);
212 }
213
214
215 static void (*sample_func)(float *gamma) = calculate_gamma;
216
217 config_t config = {
218   .magic       = CONFIG_MAGIC,
219   .dac_value   = 1922,
220   ._mode       = VNA_MODE_START_STOP,
221   .harmonic_freq_threshold = FREQUENCY_THRESHOLD,
222   ._serial_speed = 1000000,
223   .bandwidth = BANDWIDTH_1000
224 };
225
226 properties_t current_props;
227
228 // Configuración por defecto del VNA
229 static const trace_t def_trace[TRACES_MAX] = {  //enable, ...
      type, channel, reserved, scale, refpos
230   { 1, TRC_LOGMAG, 0, 0, 10.0, NGRIDY-1 },
231   { 1, TRC_LOGMAG, 1, 0, 10.0, NGRIDY-1 },
```

```
232   { 1, TRC_SMITH,  0, 0, 1.0, 0 },
233   { 1, TRC_PHASE,  1, 0, 90.0, NGRIDY/2 }
234 };
235
236 static const marker_t def_markers[MARKERS_MAX] = {
237   { 1, 0, 30, 0 }, { 0, 0, 40, 0 }, { 0, 0, 60, 0 }, { 0, 0, ...
        80, 0 }
238 };
239
240 // Cargar las propiedades por defecto
241 void load_default_properties(void)
242 {
243   current_props._frequency0   =     50000;    // inicio =  50kHz
244   current_props._frequency1   = 900000000;    // fin    = 900MHz
245   current_props._sweep_points = POINTS_COUNT_DEFAULT; // ...
        Puntos por defecto
246   current_props._cal_status   = 0;
247   current_props._electrical_delay = 0.0;
248   memcpy(current_props._trace, def_trace, sizeof(def_trace));
249   memcpy(current_props._markers, def_markers, ...
        sizeof(def_markers));
250   current_props._velocity_factor = 0.7;
251   current_props._active_marker   = 0;
252   current_props._domain_mode     = 0;
253   current_props._marker_smith_format = MS_RLC;
254   current_props._power = SI5351_CLK_DRIVE_STRENGTH_AUTO;
255 }
256
257 int load_properties(uint32_t id)
258 {
259   int r = caldata_recall(id);
260   update_frequencies(false);
261   return r;
262 }
263
264
265 // Función de callback I2S DMA
266 static volatile systime_t ready_time = 0;
267 void i2s_end_callback(I2SDriver *i2sp, size_t offset, size_t n)
268 {
269   int16_t *p = &rx_buffer[offset];
270   (void)i2sp;
271   if (wait_count == 0 || chVTGetSystemTimeX() < ready_time) ...
        return;
272   if (wait_count == config.bandwidth+2)      // Resetear y ...
        vaciar buffer
273     reset_dsp_accumerator();
274   else if (wait_count <= config.bandwidth+1) // ...
        Procesamiento de los datos
```

```
275     dsp_process(p, n);
276
277   --wait_count;
278 }
279
280 static const I2SConfig i2sconfig = {
281   NULL,                     // TX Buffer
282   rx_buffer,                // RX Buffer
283   AUDIO_BUFFER_LEN * 2,     // RX Buffer size
284   NULL,                     // tx callback
285   i2s_end_callback,         // rx callback
286   0,                        // i2scfgr
287   0                         // i2spr
288 };
289
290 #define DELAY_CHANNEL_CHANGE   3
291 #define DELAY_SWEEP_START     50    // Delay para empezar el ...
      barrido
292
293 #define DSP_START(delay) {ready_time = chVTGetSystemTimeX() ...
      + delay; wait_count = config.bandwidth+2;}
294 #define DSP_WAIT         while (wait_count) {__WFI();}
295 #define RESET_SWEEP      {p_sweep = 0;}
296
297 #define SWEEP_CH0_MEASURE   1
298
299 static uint16_t get_sweep_mask(void){
300   uint16_t ch_mask = 0;
301   int t;
302   for (t = 0; t < TRACES_MAX; t++) {
303     if (!trace[t].enabled)
304       continue;
305     if (trace[t].channel == 0) ch_mask|=SWEEP_CH0_MEASURE;
306   }
307   return ch_mask;
308 }
309
310
311 // Bucle principal para medir el S11
312 static bool sweep(bool break_on_operation, uint16_t ch_mask)
313 {
314   int delay;
315   if (p_sweep>=sweep_points || break_on_operation == false) ...
        RESET_SWEEP;
316   if (break_on_operation && ch_mask == 0)
317     return false;
318
319   // Parpadeo del LED mientras escanea
320   palClearPad(GPIOC, GPIOC_LED);
```

```
321
322   // Esperar un tiempo para potencia estable
323   int st_delay = DELAY_SWEEP_START;
324   for (; p_sweep < sweep_points; p_sweep++) {
325     delay = set_frequency(frequencies[p_sweep]);
326
327     // CH0: REFLEXIÓN, resetear y empezar la medida
328     if (ch_mask & SWEEP_CH0_MEASURE) {
329       tlv320aic3204_select(0);
330       DSP_START(delay+st_delay);
331       delay = DELAY_CHANNEL_CHANGE;
332       DSP_WAIT;
333       (*sample_func)(measured[0][p_sweep]);      // Calcula ...
             el coeficiente de reflexión
334       if(enable_usart_data) {
335         streamWrite((BaseSequentialStream *)&SD1, (uint8_t ...
               *)measured[0][p_sweep], sizeof(float)*2);
336         shell_printf("%f %f\r\n", measured[0][p_sweep][0], ...
               measured[0][p_sweep][1]);
337       }
338       if (APPLY_CALIBRATION_AFTER_SWEEP == 0 && (cal_status ...
             & CALSTAT_APPLY))
339         apply_CH0_error_term_at(p_sweep);
340     }
341
342     if (operation_requested && break_on_operation) break;
343     st_delay = 0;
344   }
345
346   if(enable_usart_data) {
347         streamWrite((BaseSequentialStream *)&SD1, (uint8_t ...
               *)"\r\n\r\n\r\n\r\n", 8);
348         shell_printf("\r\n\r\n\r\n\r\n");
349     }
350
351   // Aplica la calibración al final
352   if (APPLY_CALIBRATION_AFTER_SWEEP && (cal_status & ...
        CALSTAT_APPLY) && p_sweep == sweep_points){
353     uint16_t start_sweep;
354     for (start_sweep = 0; start_sweep < p_sweep; start_sweep++){
355       if (ch_mask & SWEEP_CH0_MEASURE) ...
             apply_CH0_error_term_at(start_sweep);
356     }
357   }
358
359   // Parpadeo del LED mientras escanea
360   palSetPad(GPIOC, GPIOC_LED);
361   return p_sweep == sweep_points;
362 }
```

```
363
364
365 static int set_frequency(uint32_t freq)
366 {
367   return si5351_set_frequency(freq, current_props._power);
368 }
369
370 void set_bandwidth(uint16_t bw_count){
371   config.bandwidth = bw_count&0x1FF;
372 }
373
374 uint32_t get_bandwidth_frequency(uint16_t bw_freq){
375   return (AUDIO_ADC_FREQ/AUDIO_SAMPLES_COUNT)/(bw_freq+1);
376 }
377
378 #define MAX_BANDWIDTH      (AUDIO_ADC_FREQ/AUDIO_SAMPLES_COUNT)
379 #define MIN_BANDWIDTH      ...
    ((AUDIO_ADC_FREQ/AUDIO_SAMPLES_COUNT)/512 + 1)
380
381
382 void set_sweep_points(uint16_t points){
383   if (points == sweep_points || points > POINTS_COUNT)
384     return;
385
386   sweep_points = points;
387   update_frequencies(cal_status & CALSTAT_APPLY);
388
389 }
390
391 #define SCAN_MASK_OUT_FREQ       0b00000001
392 #define SCAN_MASK_OUT_DATA0      0b00000010
393 #define SCAN_MASK_OUT_DATA1      0b00000100
394 #define SCAN_MASK_NO_CALIBRATION 0b00001000
395 #define SCAN_MASK_BINARY         0b10000000
396
397
398 static void
399 set_frequencies(uint32_t start, uint32_t stop, uint16_t points)
400 {
401   uint32_t i;
402   uint32_t step = (points - 1);
403   uint32_t span = stop - start;
404   uint32_t delta = span / step;
405   uint32_t error = span % step;
406   uint32_t f = start, df = step>>1;
407   for (i = 0; i <= step; i++, f+=delta) {
408     frequencies[i] = f;
409     df+=error;
410     if (df >=step) {
```

```
411       f++;
412       df -= step;
413     }
414   }
415   // Fuera del rango de barrido
416   for (; i < POINTS_COUNT; i++)
417     frequencies[i] = 0;
418 }
419
420 static void
421 update_frequencies(bool interpolate)
422 {
423   uint32_t start, stop;
424   start = get_sweep_frequency(ST_START);
425   stop  = get_sweep_frequency(ST_STOP);
426
427   set_frequencies(start, stop, sweep_points);
428   if (interpolate)
429     cal_interpolate();
430   RESET_SWEEP;
431 }
432
433 void
434 set_sweep_frequency(int type, uint32_t freq)
435 {
436   int cal_applied = cal_status & CALSTAT_APPLY;
437
438   // Comprobar si la frecuencia se sale de los límites
439   if (type != ST_SPAN && freq < START_MIN)
440     freq = START_MIN;
441   if (freq > STOP_MAX)
442     freq = STOP_MAX;
443   uint32_t center, span;
444   switch (type) {
445     case ST_START:
446       config._mode &= ~VNA_MODE_CENTER_SPAN;
447       frequency0 = freq;
448       if (frequency1 < freq) frequency1 = freq;
449       break;
450     case ST_STOP:
451       config._mode &= ~VNA_MODE_CENTER_SPAN;
452       frequency1 = freq;
453       if (frequency0 > freq) frequency0 = freq;
454       break;
455     case ST_CENTER:
456       config._mode |= VNA_MODE_CENTER_SPAN;
457       center = freq;
458       span   = (frequency1 - frequency0)>>1;
459       if (span > center - START_MIN)
```

```
460         span = (center - START_MIN);
461       if (span > STOP_MAX - center)
462         span = (STOP_MAX - center);
463       frequency0 = center - span;
464       frequency1 = center + span;
465       break;
466     case ST_SPAN:
467       config._mode |= VNA_MODE_CENTER_SPAN;
468       center = (frequency0>>1) + (frequency1>>1);
469       span = freq>>1;
470       if (center < START_MIN + span)
471         center = START_MIN + span;
472       if (center > STOP_MAX - span)
473         center = STOP_MAX - span;
474       frequency0 = center - span;
475       frequency1 = center + span;
476       break;
477     case ST_CW:
478       config._mode |= VNA_MODE_CENTER_SPAN;
479       frequency0 = freq;
480       frequency1 = freq;
481       break;
482   }
483   update_frequencies(cal_applied);
484 }
485
486 uint32_t
487 get_sweep_frequency(int type)
488 {
489   if (frequency0 > frequency1) {
490     uint32_t t = frequency0;
491     frequency0 = frequency1;
492     frequency1 = t;
493   }
494   switch (type) {
495     case ST_START:  return frequency0;
496     case ST_STOP:   return frequency1;
497     case ST_CENTER: return frequency0/2 + frequency1/2;
498     case ST_SPAN:   return frequency1 - frequency0;
499     case ST_CW:     return frequency0;
500   }
501   return 0;
502 }
503
504
505 // Corrección de errores
506
507 static void
508 eterm_set(int term, float re, float im)
```

```
509 {
510   int i;
511   for (i = 0; i < sweep_points; i++) {
512     cal_data[term][i][0] = re;
513     cal_data[term][i][1] = im;
514   }
515 }
516
517 static void
518 eterm_copy(int dst, int src)
519 {
520   memcpy(cal_data[dst], cal_data[src], sizeof cal_data[dst]);
521 }
522
523
524 static void
525 eterm_calc_es(void)
526 {
527   int i;
528   for (i = 0; i < sweep_points; i++) {
529     // z=1/(jwc*z0) = 1/(2*pi*f*c*z0)
530     // s11ao = (z-1)/(z+1) = (1-1/z)/(1+1/z) = ...
          (1-jwcz0)/(1+jwcz0)
531     // Preparo 1/s11ao para mayor eficiencia
532     float c = 50e-15;
533     float z0 = 50;
534     float z = 2 * VNA_PI * frequencies[i] * c * z0;
535     float sq = 1 + z*z;
536     float s11aor = (1 - z*z) / sq;
537     float s11aoi = 2*z / sq;
538
539     // S11mo'= S11mo - Ed
540     // S11ms'= S11ms - Ed
541     float s11or = cal_data[CAL_OPEN][i][0] - ...
          cal_data[ETERM_ED][i][0];
542     float s11oi = cal_data[CAL_OPEN][i][1] - ...
          cal_data[ETERM_ED][i][1];
543     float s11sr = cal_data[CAL_SHORT][i][0] - ...
          cal_data[ETERM_ED][i][0];
544     float s11si = cal_data[CAL_SHORT][i][1] - ...
          cal_data[ETERM_ED][i][1];
545
546     // Es = (S11mo'/s11ao + S11ms')/(S11mo' - S11ms')
547     float numr = s11sr + s11or * s11aor - s11oi * s11aoi;
548     float numi = s11si + s11oi * s11aor + s11or * s11aoi;
549     float denomr = s11or - s11sr;
550     float denomi = s11oi - s11si;
551     sq = denomr*denomr+denomi*denomi;
552     cal_data[ETERM_ES][i][0] = (numr*denomr + numi*denomi)/sq;
```

```
553     cal_data[ETERM_ES][i][1] = (numi*denomr - numr*denomi)/sq;
554   }
555   cal_status &= ~CALSTAT_OPEN;
556   cal_status |= CALSTAT_ES;
557 }
558
559 static void
560 eterm_calc_er(int sign)
561 {
562   int i;
563   for (i = 0; i < sweep_points; i++) {
564     // Er = sign*(1-sign*Es)S11ms'
565     float s11sr = cal_data[CAL_SHORT][i][0] - ...
          cal_data[ETERM_ED][i][0];
566     float s11si = cal_data[CAL_SHORT][i][1] - ...
          cal_data[ETERM_ED][i][1];
567     float esr = cal_data[ETERM_ES][i][0];
568     float esi = cal_data[ETERM_ES][i][1];
569     if (sign > 0) {
570       esr = -esr;
571       esi = -esi;
572     }
573     esr = 1 + esr;
574     float err = esr * s11sr - esi * s11si;
575     float eri = esr * s11si + esi * s11sr;
576     if (sign < 0) {
577       err = -err;
578       eri = -eri;
579     }
580     cal_data[ETERM_ER][i][0] = err;
581     cal_data[ETERM_ER][i][1] = eri;
582   }
583   cal_status &= ~CALSTAT_SHORT;
584   cal_status |= CALSTAT_ER;
585 }
586
587 // Et está invertido para mayor eficiencia
588 static void
589 eterm_calc_et(void)
590 {
591   int i;
592   for (i = 0; i < sweep_points; i++) {
593     // Et = 1/(S21mt - Ex)
594     float etr = cal_data[CAL_THRU][i][0] - ...
          cal_data[CAL_ISOLN][i][0];
595     float eti = cal_data[CAL_THRU][i][1] - ...
          cal_data[CAL_ISOLN][i][1];
596     float sq = etr*etr + eti*eti;
597     float invr = etr / sq;
```

```
598     float invi = -eti / sq;
599     cal_data[ETERM_ET][i][0] = invr;
600     cal_data[ETERM_ET][i][1] = invi;
601   }
602   cal_status &= ~CALSTAT_THRU;
603   cal_status |= CALSTAT_ET;
604 }
605
606
607 static void apply_CH0_error_term_at(int i)
608 {
609     // S11m' = S11m - Ed
610     // S11a = S11m' / (Er + Es S11m')
611     float s11mr = measured[0][i][0] - cal_data[ETERM_ED][i][0];
612     float s11mi = measured[0][i][1] - cal_data[ETERM_ED][i][1];
613     float err = cal_data[ETERM_ER][i][0] + s11mr * ...
            cal_data[ETERM_ES][i][0] - s11mi * ...
            cal_data[ETERM_ES][i][1];
614     float eri = cal_data[ETERM_ER][i][1] + s11mr * ...
            cal_data[ETERM_ES][i][1] + s11mi * ...
            cal_data[ETERM_ES][i][0];
615     float sq = err*err + eri*eri;
616     float s11ar = (s11mr * err + s11mi * eri) / sq;
617     float s11ai = (s11mi * err - s11mr * eri) / sq;
618     measured[0][i][0] = s11ar;
619     measured[0][i][1] = s11ai;
620 }
621
622 static void apply_edelay(void)
623 {
624   int i;
625   float real, imag;
626   float s, c;
627   uint16_t ch_mask = get_sweep_mask();
628   for (i=0;i<sweep_points;i++){
629     vna_sin_cos(electrical_delay * frequencies[i] * 1E-12, ...
            &s, &c);
630     if (ch_mask & SWEEP_CH0_MEASURE){
631       real = measured[0][i][0];
632       imag = measured[0][i][1];
633       measured[0][i][0] = real * c - imag * s;
634       measured[0][i][1] = imag * c + real * s;
635     }
636   }
637 }
638
639 void
640 cal_collect(uint16_t type)
641 {
```

```
642   uint16_t dst, src;
643
644   static const struct {
645     uint16_t set_flag;
646     uint16_t clr_flag;
647     uint8_t dst;
648     uint8_t src;
649  } calibration_set[]={
650 //    type       set data flag       reset flag                ...
     destination source
651     [CAL_LOAD] = {CALSTAT_LOAD,  ~(           ...
          CALSTAT_APPLY), CAL_LOAD,  0},
652     [CAL_OPEN] = {CALSTAT_OPEN,  ...
          ~(CALSTAT_ES|CALSTAT_APPLY), CAL_OPEN,  0},
653     [CAL_SHORT]= {CALSTAT_SHORT, ...
          ~(CALSTAT_ER|CALSTAT_APPLY), CAL_SHORT, 0},
654     [CAL_THRU] = {CALSTAT_THRU,  ...
          ~(CALSTAT_ET|CALSTAT_APPLY), CAL_THRU,  1},
655     [CAL_ISOLN]= {CALSTAT_ISOLN, ~(           ...
          CALSTAT_APPLY), CAL_ISOLN, 1},
656   };
657   if (type >= ARRAY_COUNT(calibration_set)) return;
658   cal_status|=calibration_set[type].set_flag;
659   cal_status&=calibration_set[type].clr_flag;
660   dst = calibration_set[type].dst;
661   src = calibration_set[type].src;
662
663   // Hacer un sweep para registrar datos
664   uint8_t bw = config.bandwidth;
665   if (bw < BANDWIDTH_100)
666     config.bandwidth = BANDWIDTH_100;
667
668   sweep(false, SWEEP_CH0_MEASURE);
669   config.bandwidth = bw;
670
671   // Copiar datos de calibración
672   memcpy(cal_data[dst], measured[src], sizeof measured[0]);
673 }
674
675 void
676 cal_done(void)
677 {
678   if (!(cal_status & CALSTAT_LOAD))
679     eterm_set(ETERM_ED, 0.0, 0.0);
680   if ((cal_status & CALSTAT_SHORT) && (cal_status & ...
       CALSTAT_OPEN)) {
681     eterm_calc_es();
682     eterm_calc_er(-1);
683   } else if (cal_status & CALSTAT_OPEN) {
```

```
684       eterm_copy(CAL_SHORT, CAL_OPEN);
685       eterm_set(ETERM_ES, 0.0, 0.0);
686       eterm_calc_er(1);
687     } else if (cal_status & CALSTAT_SHORT) {
688       eterm_set(ETERM_ES, 0.0, 0.0);
689       cal_status &= ~CALSTAT_SHORT;
690       eterm_calc_er(-1);
691     } else if (!(cal_status & CALSTAT_ER)){
692       eterm_set(ETERM_ER, 1.0, 0.0);
693     } else if (!(cal_status & CALSTAT_ES)) {
694       eterm_set(ETERM_ES, 0.0, 0.0);
695     }
696
697     if (!(cal_status & CALSTAT_ISOLN))
698       eterm_set(ETERM_EX, 0.0, 0.0);
699     if (cal_status & CALSTAT_THRU) {
700       eterm_calc_et();
701     } else if (!(cal_status & CALSTAT_ET)) {
702       eterm_set(ETERM_ET, 1.0, 0.0);
703     }
704
705     cal_status |= CALSTAT_APPLY;
706   }
707
708   static void
709   cal_interpolate(void)
710   {
711     const properties_t *src = caldata_reference();
712     uint32_t i, j;
713     int eterm;
714     if (src == NULL)
715       return;
716
717     // Subir no interpolados si hay
718     if (frequencies[0] == src->_frequency0 && ...
          frequencies[src->_sweep_points-1] == src->_frequency1){
719       memcpy(current_props._cal_data, src->_cal_data, ...
            sizeof(src->_cal_data));
720       cal_status = src->_cal_status;
721       return;
722     }
723
724     uint32_t src_f = src->_frequency0;
725     for (i = 0; i < sweep_points; i++) {
726       if (frequencies[i] >= src_f)
727         break;
728
729       // Rellenar cal_data
730       for (eterm = 0; eterm < 5; eterm++) {
```

```
731       cal_data[eterm][i][0] = src->_cal_data[eterm][0][0];
732       cal_data[eterm][i][1] = src->_cal_data[eterm][0][1];
733     }
734   }
735
736   // Reconstruiur la lista de frecuencias de la fuente
737   uint32_t src_points = (src->_sweep_points - 1);
738   uint32_t span = src->_frequency1 - src->_frequency0;
739   uint32_t delta = span / src_points;
740   uint32_t error = span % src_points;
741   uint32_t df = src_points>>1;
742   j = 0;
743   for (; i < sweep_points; i++) {
744     uint32_t f = frequencies[i];
745     if (f == 0) goto interpolate_finish;
746     for (; j < src_points; j++) {
747       if (src_f <= f && f < src_f + delta) {
748         // Encontrado f entre las frecuencias, en j y j+1
749         float k1 = (delta == 0) ? 0.0 : (float)(f - src_f) / ...
              delta;
750         uint32_t idx = j;
751         if (si5351_get_harmonic_lvl(src_f) != ...
              si5351_get_harmonic_lvl(src_f+delta)) {
752           // f en el armónico anterior, extrapolar de los 2 ...
                puntos anteriores
753           if (si5351_get_harmonic_lvl(f) == ...
                si5351_get_harmonic_lvl(src_f)){
754             if (idx >= 1){
755               idx--; k1+=1.0;
756             }
757             else // Límite de puntos
758               k1 = 0.0;
759           }
760           // f en el siguiente armónico, extrapolar de los 2 ...
                puntos siguientes
761           else {
762             if (idx < src_points){
763               idx++; k1-=1.0;
764             }
765             else // Límite de puntos
766               k1 = 1.0;
767           }
768         }
769         float k0 = 1.0 - k1;
770         for (eterm = 0; eterm < 5; eterm++) {
771           cal_data[eterm][i][0] = ...
                src->_cal_data[eterm][idx][0] * k0 + ...
                src->_cal_data[eterm][idx+1][0] * k1;
772           cal_data[eterm][i][1] = ...
```

```
                src->_cal_data[eterm][idx][1] * k0 + ...
                src->_cal_data[eterm][idx+1][1] * k1;
773           }
774           break;
775         }
776         df+=error;if (df >=src_points) {src_f++;df -= src_points;}
777         src_f+=delta;
778       }
779       if (j == src_points)
780         break;
781     }
782
783     // Mayor que la última frecuencia del rango de la fuente
784     for (; i < sweep_points; i++) {
785       // Rellenar cal_data al final de la fuente
786       for (eterm = 0; eterm < 5; eterm++) {
787         cal_data[eterm][i][0] = ...
              src->_cal_data[eterm][src_points][0];
788         cal_data[eterm][i][1] = ...
              src->_cal_data[eterm][src_points][1];
789       }
790     }
791 interpolate_finish:
792     cal_status = src->_cal_status | CALSTAT_INTERPOLATED;
793 }
794
795
796 void set_electrical_delay(float picoseconds)
797 {
798     if (electrical_delay != picoseconds) {
799       electrical_delay = picoseconds;
800     }
801 }
802
803
804 #ifdef ENABLE_TRANSFORM_COMMAND
805 static void
806 set_domain_mode(int mode) // DOMAIN_FREQ o DOMAIN_TIME
807 {
808     if (mode != (domain_mode & DOMAIN_MODE)) {
809       domain_mode = (domain_mode & ~DOMAIN_MODE) | (mode & ...
          DOMAIN_MODE);
810       uistat.lever_mode = LM_MARKER;
811     }
812 }
813
814 static void
815 set_timedomain_func(int func) // TD_FUNC_LOWPASS_IMPULSE, ...
      TD_FUNC_LOWPASS_STEP o TD_FUNC_BANDPASS
```

```
816 {
817   domain_mode = (domain_mode & ~TD_FUNC) | (func & TD_FUNC);
818 }
819
820 static void
821 set_timedomain_window(int func) // ...
      TD_WINDOW_MINIMUM/TD_WINDOW_NORMAL/TD_WINDOW_MAXIMUM
822 {
823   domain_mode = (domain_mode & ~TD_WINDOW) | (func & TD_WINDOW);
824 }
825 #endif
826
827
828 /*
829  * Shell
830  */
831
832 #ifdef ENABLE_TRANSFORM_COMMAND
833 VNA_SHELL_FUNCTION(cmd_transform)
834 {
835   int i;
836   if (argc == 0) {
837     goto usage;
838   }
839   //                                              0   1       2 ...
          3        4       5      6       7
840   static const char cmd_transform_list[] = ...
        "on|off|impulse|step|bandpass|minimum|normal|maximum";
841   for (i = 0; i < argc; i++) {
842     switch (get_str_index(argv[i], cmd_transform_list)) {
843       case 0:
844         set_domain_mode(DOMAIN_TIME);
845         return;
846       case 1:
847         set_domain_mode(DOMAIN_FREQ);
848         return;
849       case 2:
850         set_timedomain_func(TD_FUNC_LOWPASS_IMPULSE);
851         return;
852       case 3:
853         set_timedomain_func(TD_FUNC_LOWPASS_STEP);
854         return;
855       case 4:
856         set_timedomain_func(TD_FUNC_BANDPASS);
857         return;
858       case 5:
859         set_timedomain_window(TD_WINDOW_MINIMUM);
860         return;
861       case 6:
```

```
862           set_timedomain_window(TD_WINDOW_NORMAL);
863           return;
864         case 7:
865           set_timedomain_window(TD_WINDOW_MAXIMUM);
866           return;
867         default:
868           goto usage;
869       }
870     }
871     return;
872   usage:
873     shell_printf("usage: transform {%s} [...]\r\n", ...
          cmd_transform_list);
874   }
875   #endif
876
877   VNA_SHELL_FUNCTION(cmd_freq)
878   {
879     if (argc != 1) {
880       goto usage;
881     }
882     uint32_t freq = my_atoui(argv[0]);
883
884     pause_sweep();
885     set_frequency(freq);
886     return;
887   usage:
888     shell_printf("usage: freq {frequency(Hz)}\r\n");
889   }
890
891   VNA_SHELL_FUNCTION(cmd_power)
892   {
893     if (argc == 0) {
894       shell_printf("power: %d\r\n", current_props._power);
895       return;
896     }
897     if (argc != 1) {
898       shell_printf("usage: power {0-3}|{255 - auto}\r\n");
899       return;
900     }
901     set_power(my_atoi(argv[0]));
902   }
903
904   VNA_SHELL_FUNCTION(cmd_threshold)
905   {
906     uint32_t value;
907     if (argc != 1) {
908       shell_printf("usage: threshold {frequency in harmonic ...
            mode}\r\n"\
```

```
909                     "current: %d\r\n", ...
                            config.harmonic_freq_threshold);
910       return;
911     }
912     value = my_atoui(argv[0]);
913     config.harmonic_freq_threshold = value;
914   }
915
916   VNA_SHELL_FUNCTION(cmd_saveconfig)
917   {
918     (void)argc;
919     (void)argv;
920     config_save();
921     shell_printf("Config saved.\r\n");
922   }
923
924   VNA_SHELL_FUNCTION(cmd_clearconfig)
925   {
926     if (argc != 1) {
927       shell_printf("usage: clearconfig {protection key}\r\n");
928       return;
929     }
930
931     if (strcmp(argv[0], "1234") != 0) {
932       shell_printf("Key unmatched.\r\n");
933       return;
934     }
935
936     clear_all_config_prop_data();
937     shell_printf("Config and all cal data cleared.\r\n"\
938                  "Do reset manually to take effect.\r\n");
939   }
940
941   VNA_SHELL_FUNCTION(cmd_pause)
942   {
943     (void)argc;
944     (void)argv;
945     pause_sweep();
946   }
947
948   VNA_SHELL_FUNCTION(cmd_resume)
949   {
950     (void)argc;
951     (void)argv;
952     update_frequencies(cal_status & CALSTAT_APPLY);
953     resume_sweep();
954   }
955
956   VNA_SHELL_FUNCTION(cmd_reset)
```

```
957 {
958   (void)argc;
959   (void)argv;
960   if (argc == 1) {
961     if (strcmp(argv[0], "dfu") == 0) {
962       shell_printf("Performing reset to DFU mode\r\n");
963       enter_dfu();
964       return;
965     }
966   }
967   shell_printf("Performing reset\r\n");
968   rccEnableWWDG(FALSE);
969   WWDG->CFR = 0x60;
970   WWDG->CR = 0xff;
971   while (1);
972 }
973
974 VNA_SHELL_FUNCTION(cmd_data)
975 {
976   int i;
977   int sel = 0;
978   float (*array)[2];
979   if (argc == 1)
980     sel = my_atoi(argv[0]);
981   if (sel < 0 || sel >=7)
982     goto usage;
983
984   array = sel < 2 ? measured[sel] : cal_data[sel-2];
985
986   for (i = 0; i < sweep_points; i++)
987     shell_printf("%f %f\r\n", array[i][0], array[i][1]);
988   return;
989 usage:
990   shell_printf("usage: data [array]\r\n");
991 }
992
993 VNA_SHELL_FUNCTION(cmd_bandwidth)
994 {
995   uint16_t user_bw;
996   if (argc == 1)
997     user_bw = my_atoui(argv[0]);
998   else if (argc == 2){
999     uint16_t f = my_atoui(argv[0]);
1000         if (f > MAX_BANDWIDTH) user_bw = 0;
1001    else if (f < MIN_BANDWIDTH) user_bw = 511;
1002    else user_bw = ...
         ((AUDIO_ADC_FREQ+AUDIO_SAMPLES_COUNT/2)/AUDIO_SAMPLES_COUNT)/f ...
         - 1;
1003  }
```

```
1004    else
1005      goto result;
1006    set_bandwidth(user_bw);
1007  result:
1008    shell_printf("bandwidth %d (%uHz)\r\n", config.bandwidth, ...
          get_bandwidth_frequency(config.bandwidth));
1009  }
1010
1011  VNA_SHELL_FUNCTION(cmd_scan)
1012  {
1013    uint32_t start, stop;
1014    uint16_t points = sweep_points;
1015    int i;
1016    if (argc < 2 || argc > 4) {
1017      shell_printf("usage: scan {start(Hz)} {stop(Hz)} ...
            [points] [outmask]\r\n");
1018      return;
1019    }
1020
1021    start = my_atoui(argv[0]);
1022    stop = my_atoui(argv[1]);
1023    if (start == 0 || stop == 0 || start > stop) {
1024        shell_printf("frequency range is invalid\r\n");
1025        return;
1026    }
1027    if (argc >= 3) {
1028      points = my_atoui(argv[2]);
1029      if (points == 0 || points > POINTS_COUNT) {
1030        shell_printf("sweep points exceeds range ...
              "define_to_STR(POINTS_COUNT)"\r\n");
1031        return;
1032      }
1033      sweep_points = points;
1034    }
1035    uint16_t mask = 0;
1036    uint16_t sweep_ch = SWEEP_CH0_MEASURE;
1037
1038  #ifdef ENABLE_SCANBIN_COMMAND
1039    if (argc == 4) {
1040      mask = my_atoui(argv[3]);
1041      if (sweep_mode&SWEEP_BINARY) mask|=SCAN_MASK_BINARY;
1042      sweep_ch = (mask>>1)&3;
1043    }
1044    sweep_mode&=~(SWEEP_BINARY);
1045  #endif
1046
1047    uint32_t old_cal_status = cal_status;
1048    if (mask&SCAN_MASK_NO_CALIBRATION) cal_status&=~CALSTAT_APPLY;
1049    // Reconstruir la tabla de frecuencias si es necesario
```

```
1050   if (frequencies[0]!=start || frequencies[points-1]!=stop){
1051     set_frequencies(start, stop, points);
1052     if (cal_status & CALSTAT_APPLY)
1053       cal_interpolate();
1054   }
1055
1056   if (sweep_ch & SWEEP_CH0_MEASURE)
1057     sweep(false, sweep_ch);
1058
1059   cal_status = old_cal_status;
1060
1061   pause_sweep();
1062   // Datos de salida
1063   if (mask) {
1064     if (mask&SCAN_MASK_BINARY){
1065       streamWrite(shell_stream, (void *)&mask, ...
             sizeof(uint16_t));
1066       streamWrite(shell_stream, (void *)&points, ...
             sizeof(uint16_t));
1067       for (i = 0; i < points; i++) {
1068         if (mask & SCAN_MASK_OUT_FREQ ) ...
               streamWrite(shell_stream, (void ...
               *)&frequencies[i],    sizeof(uint32_t));  // 4 ...
               bytes .. frecuencia
1069         if (mask & SCAN_MASK_OUT_DATA0) ...
               streamWrite(shell_stream, (void ...
               *)&measured[0][i][0], sizeof(float)* 2);  // 4+4 ...
               bytes .. S11 real/imag
1070       }
1071     }
1072     else{
1073       for (i = 0; i < points; i++) {
1074         if (mask & SCAN_MASK_OUT_FREQ ) shell_printf("%u ", ...
               frequencies[i]);
1075         if (mask & SCAN_MASK_OUT_DATA0) shell_printf("%f %f ...
               ", measured[0][i][0], measured[0][i][1]);
1076         shell_printf("\r\n");
1077       }
1078     }
1079   }
1080 }
1081
1082 #ifdef ENABLE_SCANBIN_COMMAND
1083 VNA_SHELL_FUNCTION(cmd_scan_bin)
1084 {
1085   sweep_mode|= SWEEP_BINARY;
1086   cmd_scan(argc, argv);
1087   sweep_mode&=~(SWEEP_BINARY);
1088 }
```

```
1089 #endif
1090
1091 VNA_SHELL_FUNCTION(cmd_sweep)
1092 {
1093   if (argc == 0) {
1094     shell_printf("%u %u %d\r\n", ...
          get_sweep_frequency(ST_START), ...
          get_sweep_frequency(ST_STOP), sweep_points);
1095     return;
1096   } else if (argc > 3) {
1097     goto usage;
1098   }
1099   uint32_t value0 = 0;
1100   uint32_t value1 = 0;
1101   uint32_t value2 = 0;
1102   if (argc >= 1) value0 = my_atoui(argv[0]);
1103   if (argc >= 2) value1 = my_atoui(argv[1]);
1104   if (argc >= 3) value2 = my_atoui(argv[2]);
1105 #if MAX_FREQ_TYPE != 5
1106 #error "Sweep mode possibly changed, check cmd_sweep function"
1107 #endif
1108   // Comprobar sweep {start|stop|center|span|cw} {freq(Hz)}
1109   // Obtengo ST_START, ST_STOP, ST_CENTER, ST_SPAN, ST_CW
1110   static const char sweep_cmd[] = "start|stop|center|span|cw";
1111   if (argc == 2 && value0 == 0) {
1112     int type = get_str_index(argv[0], sweep_cmd);
1113     if (type == -1)
1114       goto usage;
1115     set_sweep_frequency(type, value1);
1116     return;
1117   }
1118   //  Comprobar sweep {start(Hz)} [stop(Hz)]
1119   if (value0)
1120     set_sweep_frequency(ST_START, value0);
1121   if (value1)
1122     set_sweep_frequency(ST_STOP, value1);
1123   if (value2)
1124     set_sweep_points(value2);
1125   return;
1126 usage:
1127   shell_printf("usage: sweep {start(Hz)} [stop(Hz)] ...
        [points]\r\n"\
1128                "\tsweep {%s} {freq(Hz)}\r\n", sweep_cmd);
1129 }
1130
1131 VNA_SHELL_FUNCTION(cmd_cal)
1132 {
1133   static const char *items[] = { "load", "open", "short", ...
        "thru", "isoln", "Es", "Er", "Et", "cal'ed" };
```

```
1134
1135   if (argc == 0) {
1136     int i;
1137     for (i = 0; i < 9; i++) {
1138       if (cal_status & (1<<i))
1139         shell_printf("%s ", items[i]);
1140     }
1141     shell_printf("\r\n");
1142     return;
1143   }
1144   //                                        0    1     2    3 ...
          4    5  6   7     8
1145   static const char cmd_cal_list[] = ...
       "load|open|short|thru|isoln|done|on|off|reset";
1146   switch (get_str_index(argv[0], cmd_cal_list)) {
1147     case 0:
1148       cal_collect(CAL_LOAD);
1149       return;
1150     case 1:
1151       cal_collect(CAL_OPEN);
1152       return;
1153     case 2:
1154       cal_collect(CAL_SHORT);
1155       return;
1156     case 3:
1157       cal_collect(CAL_THRU);
1158       return;
1159     case 4:
1160       cal_collect(CAL_ISOLN);
1161       return;
1162     case 5:
1163       cal_done();
1164       return;
1165     case 6:
1166       cal_status |= CALSTAT_APPLY;
1167       return;
1168     case 7:
1169       cal_status &= ~CALSTAT_APPLY;
1170       return;
1171     case 8:
1172       cal_status = 0;
1173       return;
1174     default:
1175       break;
1176   }
1177   shell_printf("usage: cal [%s]\r\n", cmd_cal_list);
1178 }
1179
1180 VNA_SHELL_FUNCTION(cmd_save)
```

```
1181 {
1182   if (argc != 1)
1183     goto usage;
1184
1185   int id = my_atoi(argv[0]);
1186   if (id < 0 || id >= SAVEAREA_MAX)
1187     goto usage;
1188   caldata_save(id);
1189   return;
1190
1191  usage:
1192   shell_printf("save {id}\r\n");
1193 }
1194
1195 VNA_SHELL_FUNCTION(cmd_recall)
1196 {
1197   if (argc != 1)
1198     goto usage;
1199
1200   int id = my_atoi(argv[0]);
1201   if (id < 0 || id >= SAVEAREA_MAX)
1202     goto usage;
1203   if (load_properties(id))
1204     shell_printf("Err, default load\r\n");
1205   return;
1206  usage:
1207   shell_printf("recall {id}\r\n");
1208 }
1209
1210 VNA_SHELL_FUNCTION(cmd_edelay)
1211 {
1212   if (argc != 1) {
1213     shell_printf("%f\r\n", electrical_delay);
1214     return;
1215   }
1216   set_electrical_delay(my_atof(argv[0]));
1217 }
1218
1219 VNA_SHELL_FUNCTION(cmd_frequencies)
1220 {
1221   int i;
1222   (void)argc;
1223   (void)argv;
1224   for (i = 0; i < sweep_points; i++) {
1225     if (frequencies[i] == 0) break;
1226     shell_printf("%u\r\n", frequencies[i]);
1227   }
1228 }
1229
```

```
1230 #ifdef __USE_SERIAL_CONSOLE__
1231 #ifdef ENABLE_USART_COMMAND
1232 VNA_SHELL_FUNCTION(cmd_usart_cfg)
1233 {
1234   if (argc != 1) goto result;
1235   uint32_t speed = my_atoui(argv[0]);
1236   if (speed < 300) speed = 300;
1237   config._serial_speed = speed;
1238   shell_update_speed();
1239 result:
1240   shell_printf("Serial: %u baud\r\n", config._serial_speed);
1241 }
1242
1243 VNA_SHELL_FUNCTION(cmd_usart)
1244 {
1245   uint32_t time = 2000; // 200ms de espera por defecto
1246   if (argc == 0 || argc > 2 || (config._mode & ...
        VNA_MODE_SERIAL)) return;
1247   if (argc == 2) time = my_atoui(argv[1])*10;
1248   sdWriteTimeout(&SD1, (uint8_t *)argv[0], strlen(argv[0]), ...
        time);
1249   sdWriteTimeout(&SD1, (uint8_t *)VNA_SHELL_NEWLINE_STR, ...
        sizeof(VNA_SHELL_NEWLINE_STR)-1, time);
1250   uint32_t size;
1251   uint8_t buffer[64];
1252   while ((size = sdReadTimeout(&SD1, buffer, sizeof(buffer), ...
        time)))
1253     streamWrite(&SDU1, buffer, size);
1254 }
1255
1256 VNA_SHELL_FUNCTION(cmd_usart_data)
1257 {
1258   if(argc == 1) {
1259   uint8_t resp = my_atoui(argv[0]);
1260     if(resp == 1) {
1261       enable_usart_data = true;
1262       return;
1263     }
1264     if(resp == 0) {
1265       enable_usart_data = false;
1266       return;
1267     }
1268   }
1269   shell_printf("usage: usart_data {1/0}\r\n");
1270 }
1271 #endif
1272 #endif
1273
1274 VNA_SHELL_FUNCTION(cmd_help)
```

```
1275 {
1276   (void)argc;
1277   (void)argv;
1278   const VNAShellCommand *scp = commands;
1279   shell_printf("Commands:");
1280   while (scp->sc_name != NULL) {
1281     shell_printf(" %s", scp->sc_name);
1282     scp++;
1283   }
1284   shell_printf(VNA_SHELL_NEWLINE_STR);
1285   return;
1286 }
1287
1288
1289 #pragma pack(push, 2)
1290 typedef struct {
1291   const char           *sc_name;
1292   vna_shellcmd_t    sc_function;
1293   uint16_t flags;
1294 } VNAShellCommand;
1295 #pragma pack(pop)
1296
1297 // Algunos comandos solo se pueden ejecutar en el hilo del ...
       sweep, no en el main
1298 #define CMD_WAIT_MUTEX  1
1299 #define CMD_BREAK_SWEEP 2
1300
1301 // Lista de comandos
1302 static const VNAShellCommand commands[] =
1303 {
1304     {"scan"        , cmd_scan        , ...
          CMD_WAIT_MUTEX|CMD_BREAK_SWEEP},
1305 #ifdef ENABLE_SCANBIN_COMMAND
1306     {"scan_bin"    , cmd_scan_bin    , ...
          CMD_WAIT_MUTEX|CMD_BREAK_SWEEP},
1307 #endif
1308     {"data"        , cmd_data        , 0},
1309     {"frequencies" , cmd_frequencies , 0},
1310     {"freq"        , cmd_freq        , ...
          CMD_WAIT_MUTEX|CMD_BREAK_SWEEP},
1311     {"sweep"       , cmd_sweep       , ...
          CMD_WAIT_MUTEX|CMD_BREAK_SWEEP},
1312     {"power"       , cmd_power       , 0},
1313     {"bandwidth"   , cmd_bandwidth   , 0},
1314     {"saveconfig"  , cmd_saveconfig  , 0},
1315     {"clearconfig" , cmd_clearconfig , 0},
1316     {"pause"       , cmd_pause       , ...
          CMD_WAIT_MUTEX|CMD_BREAK_SWEEP},
1317     {"resume"      , cmd_resume      , ...
```

```
        CMD_WAIT_MUTEX|CMD_BREAK_SWEEP},
1318    {"cal"          , cmd_cal          , CMD_WAIT_MUTEX},
1319    {"save"         , cmd_save         , 0},
1320    {"recall"       , cmd_recall       , ...
        CMD_WAIT_MUTEX|CMD_BREAK_SWEEP},
1321    {"edelay"       , cmd_edelay       , 0},
1322    {"reset"        , cmd_reset        , 0},
1323 #ifdef __USE_SERIAL_CONSOLE__
1324 #ifdef ENABLE_USART_COMMAND
1325    {"usart_cfg"    , cmd_usart_cfg    , ...
        CMD_WAIT_MUTEX|CMD_BREAK_SWEEP},
1326    {"usart"        , cmd_usart        , ...
        CMD_WAIT_MUTEX|CMD_BREAK_SWEEP},
1327    {"usart_data"   , cmd_usart_data   , 0},
1328 #endif
1329 #endif
1330 #ifdef ENABLE_TRANSFORM_COMMAND
1331    {"transform"    , cmd_transform    , 0},
1332 #endif
1333    {"threshold"    , cmd_threshold    , 0},
1334    {"help"         , cmd_help         , 0},
1335    {NULL           , NULL             , 0}
1336 };
1337
1338
1339 // Comprobar conexión Serial
1340 #ifdef __USE_SERIAL_CONSOLE__
1341 #if HAL_USE_SERIAL == FALSE
1342 #error "For serial console need HAL_USE_SERIAL as TRUE in ...
    halconf.h"
1343 #endif
1344
1345 // Seleccionar stream de entrada para comandos de la Shell
1346 #define PREPARE_STREAM shell_stream = ...
    (config._mode&VNA_MODE_SERIAL) ? (BaseSequentialStream ...
    *)&SD1 : (BaseSequentialStream *)&SDU1;
1347
1348 // Actualizar la velocidad y configuración de la conexión Serial
1349 void shell_update_speed(void){
1350   SerialConfig s_config = {config._serial_speed, 0, ...
        USART_CR2_STOP1_BITS, 0 };
1351   sdStop(&SD1);
1352   sdStart(&SD1, &s_config);  // USART
1353 }
1354
1355 // Comprobar el estado de la conexión USB
1356 static bool usb_IsActive(void){
1357   return usbGetDriverStateI(&USBD1) == USB_ACTIVE;
1358 }
```

```
1359
1360 // Resetear la cola I/O de la Shell
1361 void shell_reset_console(void){
1362   // USB
1363   if (usb_IsActive()){
1364     if (config._mode & VNA_MODE_SERIAL)
1365       sduDisconnectI(&SDU1);
1366     else
1367       sduConfigureHookI(&SDU1);
1368   }
1369   // Serial
1370   oqResetI(&SD1.oqueue);
1371   iqResetI(&SD1.iqueue);
1372 }
1373
1374 // Comprobar conexión activa para la Shell
1375 static bool shell_check_connect(void){
1376   // Serial
1377   if (config._mode & VNA_MODE_SERIAL)
1378     return true;
1379   // USB
1380   return usb_IsActive();
1381 }
1382
1383 static void shell_init_connection(void){
1384   // Inicializa y comienza el driver serial-over-USB CDC ...
        SDU1, conectado a USBD1
1385   sduObjectInit(&SDU1);
1386   sduStart(&SDU1, &serusbcfg);
1387
1388   // Establecer la configuración de la velocidad Serial para SD1
1389   shell_update_speed();
1390
1391   // Activa el driver del USB y luego el pull-up del bus del ...
        USB en D+
1392   usbDisconnectBus(&USBD1);
1393   chThdSleepMilliseconds(100);
1394   usbStart(&USBD1, &usbcfg);
1395   usbConnectBus(&USBD1);
1396
1397   // Establecer el stream I/O (SDU1 o SD1) para la Shell
1398   PREPARE_STREAM;
1399 }
1400
1401 #else
1402
1403 // Solo consola por USB, shell_stream siempre sobre el USB
1404 #define PREPARE_STREAM
1405
```

```
1406 // Comprobar si la conexión está activa
1407 static bool shell_check_connect(void){
1408   return SDU1.config->usbp->state == USB_ACTIVE;
1409 }
1410
1411 // Inicializar la conexión I/O con la Shell sobre el USB
1412 static void shell_init_connection(void){
1413   // Inicializa y comienza el driver serial-over-USB CDC ...
         SDU1, conectado a USBD1
1414   sduObjectInit(&SDU1);
1415   sduStart(&SDU1, &serusbcfg);
1416
1417   // Activa el driver del USB y luego el pull-up del bus del ...
         USB en D+
1418   usbDisconnectBus(&USBD1);
1419   chThdSleepMilliseconds(100);
1420   usbStart(&USBD1, &usbcfg);
1421   usbConnectBus(&USBD1);
1422
1423   // Establecer el stream I/O SDU1 para la Shell
1424   shell_stream = (BaseSequentialStream *)&SDU1;
1425 }
1426 #endif
1427
1428
1429 /*
1430  * Leer comando de shell_stream
1431  */
1432 static int VNAShell_readLine(char *line, int max_size)
1433 {
1434   // Leer la línea del stream de entrada
1435   uint8_t c;
1436   // Preparar I/O para shell_stream
1437   PREPARE_STREAM;
1438   char *ptr = line;
1439   while (1) {
1440     // Devuelve 0 solo si el stream no está activo
1441     if (streamRead((BaseSequentialStream *)&SD1, &c, 1) == 0)
1442       return 0;
1443     // Borrar
1444     if (c == 8 || c == 0x7f) {
1445       if (ptr != line) {
1446         static const char backspace[] = {0x08, 0x20, 0x08, ...
              0x00};
1447         shell_printf(backspace);
1448         ptr--;
1449       }
1450       continue;
1451     }
```

```
1452      // Nueva línea (Enter)
1453      if (c == '\r') {
1454        shell_printf(VNA_SHELL_NEWLINE_STR);
1455        *ptr = 0;
1456        return 1;
1457      }
1458      // Otros (los salta)
1459      if (c < 0x20)
1460        continue;
1461      // Almacenar
1462      if (ptr < line + max_size - 1) {
1463        streamPut(shell_stream, c);       // Echo
1464        *ptr++ = (char)c;
1465      }
1466    }
1467    return 0;
1468  }
1469
1470  /*
1471   * Comprobar y ejecutar el comando
1472   */
1473  static void VNAShell_executeLine(char *line)
1474  {
1475    // Comprueba y ejecuta la línea
1476    char *lp = line, *ep;
1477    shell_nargs = 0;
1478
1479    while (*lp != 0) {
1480      // Se salta los espacios en blanco y tabulaciones al ...
           comienzo del string
1481      while (*lp == ' ' || *lp == '\t') lp++;
1482      // Si un argumento comienza con comillas, entonces su ...
           delimitador son otras comillas,
1483      // si no, el delimitador es un espacio en blanco
1484      ep = (*lp == '"') ? strpbrk(++lp, "\"") : strpbrk(lp, " ...
           \t");
1485      // Almacenar en la string de args
1486      shell_args[shell_nargs++] = lp;
1487      // Final de la string de entrada
1488      if ((lp = ep) == NULL) break;
1489      // Comprobación de los límites del argumento
1490      if (shell_nargs > VNA_SHELL_MAX_ARGUMENTS) {
1491        shell_printf("too many arguments, max " define_to_STR(
1492            VNA_SHELL_MAX_ARGUMENTS) "" VNA_SHELL_NEWLINE_STR);
1493        return;
1494      }
1495      // Establecer cero al final de la string y continuar ...
           comprobando
1496      *lp++ = 0;
```

```
1497    }
1498    if (shell_nargs == 0) return;
1499    // Ejecutar línea
1500    const VNAShellCommand *scp;
1501    for (scp = commands; scp->sc_name != NULL; scp++) {
1502      if (strcmp(scp->sc_name, shell_args[0]) == 0) {
1503        if (scp->flags & CMD_WAIT_MUTEX) {
1504          shell_function = scp->sc_function;
1505          if (scp->flags & CMD_BREAK_SWEEP) ...
                operation_requested|=OP_CONSOLE;
1506          // Espera a ejecutar el comando en el hilo del sweep
1507          do {
1508            chThdSleepMilliseconds(100);
1509          } while (shell_function);
1510        } else {
1511          scp->sc_function(shell_nargs - 1, &shell_args[1]);
1512        }
1513        return;
1514      }
1515    }
1516    shell_printf("%s?" VNA_SHELL_NEWLINE_STR, shell_args[0]);
1517  }
1518
1519
1520  /*
1521   * Configuración del bus I2C
1522   */
1523
1524  // Define la velocidad del bus I2C
1525  #define STM32_I2C_SPEED                     600
1526
1527  #if STM32_I2C1_CLOCK == 8    // STM32_I2C1SW == ...
       STM32_I2C1SW_HSI     (HSI=8MHz)
1528  #if   STM32_I2C_SPEED == 400 // 400kHz @ HSI 8MHz
1529   #define STM32_I2C_TIMINGR  STM32_TIMINGR_PRESC(0U)  |\
1530                              STM32_TIMINGR_SCLDEL(3U) | ...
                                   STM32_TIMINGR_SDADEL(1U) |\
1531                              STM32_TIMINGR_SCLH(3U)   | ...
                                   STM32_TIMINGR_SCLL(9U)
1532  #endif
1533  #elif  STM32_I2C1_CLOCK == 48 // STM32_I2C1SW == ...
       STM32_I2C1SW_SYSCLK  (SYSCLK = 48MHz)
1534   #if   STM32_I2C_SPEED == 400 // 400kHz @ SYSCLK 48MHz
1535   #define STM32_I2C_TIMINGR  STM32_TIMINGR_PRESC(5U)  |\
1536                              STM32_TIMINGR_SCLDEL(3U) | ...
                                   STM32_TIMINGR_SDADEL(3U) |\
1537                              STM32_TIMINGR_SCLH(3U)   | ...
                                   STM32_TIMINGR_SCLL(9U)
1538   #elif STM32_I2C_SPEED == 600 // 600kHz @ SYSCLK 48MHz, ...
```

```
    obtener valores manualmente, x1.5 velocidad I2C
1539  #define STM32_I2C_TIMINGR  STM32_TIMINGR_PRESC(0U)    |\
1540                             STM32_TIMINGR_SCLDEL(10U) | ...
                                    STM32_TIMINGR_SDADEL(10U) |\
1541                             STM32_TIMINGR_SCLH(30U)    | ...
                                    STM32_TIMINGR_SCLL(50U)
1542  #elif STM32_I2C_SPEED == 900 // 900kHz @ SYSCLK 48MHz, ...
    obtener valores manualmente, x2 velocidad I2C
1543  #define STM32_I2C_TIMINGR  STM32_TIMINGR_PRESC(0U)    |\
1544                             STM32_TIMINGR_SCLDEL(10U) | ...
                                    STM32_TIMINGR_SDADEL(10U) |\
1545                             STM32_TIMINGR_SCLH(23U)    | ...
                                    STM32_TIMINGR_SCLL(30U)
1546  #endif
1547 #elif  STM32_I2C1_CLOCK == 72 // STM32_I2C1SW == ...
    STM32_I2C1SW_SYSCLK  (SYSCLK = 72MHz)
1548  #if   STM32_I2C_SPEED == 400 // ~400kHz @ SYSCLK 72MHz
1549  #define STM32_I2C_TIMINGR  STM32_TIMINGR_PRESC(0U)    |\
1550                             STM32_TIMINGR_SCLDEL(10U) | ...
                                    STM32_TIMINGR_SDADEL(10U) |\
1551                             STM32_TIMINGR_SCLH(80U)    | ...
                                    STM32_TIMINGR_SCLL(100U)
1552  #elif STM32_I2C_SPEED == 600 // ~600kHz @ SYSCLK 72MHz, ...
    obtener valores manualmente, x1.5 velocidad I2C
1553  #define STM32_I2C_TIMINGR  STM32_TIMINGR_PRESC(0U)    |\
1554                             STM32_TIMINGR_SCLDEL(10U) | ...
                                    STM32_TIMINGR_SDADEL(10U) |\
1555                             STM32_TIMINGR_SCLH(40U)    | ...
                                    STM32_TIMINGR_SCLL(80U)
1556  #elif STM32_I2C_SPEED == 900 // ~900kHz @ SYSCLK 72MHz, ...
    obtener valores manualmente, x2 velocidad I2C
1557  #define STM32_I2C_TIMINGR  STM32_TIMINGR_PRESC(0U)    |\
1558                             STM32_TIMINGR_SCLDEL(10U) | ...
                                    STM32_TIMINGR_SDADEL(10U) |\
1559                             STM32_TIMINGR_SCLH(30U)    | ...
                                    STM32_TIMINGR_SCLL(40U)
1560  #endif
1561 #endif
1562
1563
1564 // Configuración del reloj I2C (depende de STM32_I2C1SW en ...
    mcuconf.h)
1565 static const I2CConfig i2ccfg = {
1566   .timingr = STM32_I2C_TIMINGR,  // Inicialización del ...
       registro TIMINGR
1567   .cr1 = 0,                      // Inicialización del ...
       registro CR1
1568   .cr2 = 0                       // Inicialización del ...
       registro CR2
```

```
1569 };
1570
1571
1572 /*
1573  * Watchdog frecuencia (más de un segundo): LSI = 40000 / ...
        (64 * 1000)
1574  */
1575 static const WDGConfig wdgcfg = {
1576     STM32_IWDG_PR_64,
1577     STM32_IWDG_RL(1250),
1578     STM32_IWDG_WIN_DISABLED
1579 };
1580
1581
1582 /*
1583  * Main
1584  */
1585 // El tamaño del stack del hilo del main está definido en el ...
       Makefile, USE_PROCESS_STACKSIZE = 0x200
1586 int main(void)
1587 {
1588 /*
1589  * Inicializar el sistema ChibiOS
1590  */
1591   halInit();
1592   chSysInit();
1593
1594 /*
1595  * Restaurar la configuración
1596  */
1597   config_recall();
1598
1599 /*
1600  * Restaurar las frecuencias y la calibración del slot 0 de ...
        la memoria flash
1601  */
1602   load_properties(0);
1603
1604 /*
1605  * Inicializar la conexión con la Shell
1606  */
1607   shell_init_connection();
1608
1609 /*
1610  * Inicializar el I2C
1611  */
1612   i2cStart(&I2CD1, &i2ccfg);
1613
1614 /*
```

```
1615   * Inicializar el Si5351
1616   */
1617    si5351_init();
1618
1619  /*
1620   * Inicializar el códec TLV320AIC3204
1621   */
1622    tlv320aic3204_init();
1623
1624  /*
1625   * Inicializar el I2S
1626   */
1627    i2sInit();
1628    i2sObjectInit(&I2SD2);
1629    i2sStart(&I2SD2, &i2sconfig);
1630    i2sStartExchange(&I2SD2);
1631
1632   /*
1633   * Inicializar el Watchdog
1634   */
1635    wdgStart(&WDGD1, &wdgcfg);
1636
1637  /*
1638   * Comienzo del hilo sweep
1639   */
1640    chThdCreateStatic(waThread1, sizeof(waThread1), ...
          NORMALPRIO-1, Thread1, NULL);
1641
1642  /*
1643   * Lectura y ejecución de comandos de la Shell
1644   */
1645    while (1) {
1646      if (shell_check_connect()) {
1647        shell_printf(VNA_SHELL_NEWLINE_STR"NanoVNA ...
             Shell"VNA_SHELL_NEWLINE_STR);
1648
1649        do {
1650          shell_printf(VNA_SHELL_PROMPT_STR);
1651          if (VNAShell_readLine(shell_line, VNA_SHELL_MAX_LENGTH))
1652            VNAShell_executeLine(shell_line);
1653          else
1654            chThdSleepMilliseconds(200);
1655        } while (shell_check_connect());
1656
1657      }
1658      chThdSleepMilliseconds(1000);
1659    }
1660  }
```

V.2. Código MicroPython del Datalogger

```
from machine import Pin, UART
import time
import struct
import math
import cmath
import os

DEBUG = False

# Comandos para guardar un barrido por UART
mensajeON = 'usart_data 1\r\n'
mensajeOFF = 'usart_data 0\r\n'

# Comandos para calibrar
mensajeCALL = 'cal load\r\n'
mensajeCALO = 'cal open\r\n'
mensajeCALS = 'cal short\r\n'

# Dirección del VNA para SDI-12
ID = '1'

# UART para la comunicación con el VNA
uart = UART(2, 1000000)
uart.init(1000000, bits=8, parity=None, stop=1)

# UART para SDI-12
uart_sdi12 = UART(1, 1200)
uart_sdi12.init(1200, bits=7, parity=0, stop=1)
buffer_sdi12 = b''

botonON = Pin(12, Pin.IN)
botonCALL = Pin(13, Pin.IN)
botonCALO = Pin(14, Pin.IN)
botonCALS = Pin(27, Pin.IN)
direccion_sdi12 = Pin(15, Pin.OUT)

# Comenzar barrido
def pulsarON(p):
    for i in range(len(mensajeON)):
        if DEBUG:
            print(mensajeON[i], end='')
        uart.write(str(mensajeON[i]))
        if DEBUG:
            time.sleep(0.1)
```

```
45
46 # Calibración "load"
47 def pulsarCALL(p):
48     for i in range(len(mensajeCALL)):
49         if DEBUG:
50             print(mensajeCALL[i], end='')
51         uart.write(str(mensajeCALL[i]))
52         if DEBUG:
53             time.sleep(0.1)
54
55 # Calibración "open"
56 def pulsarCALO(p):
57     for i in range(len(mensajeCALO)):
58         if DEBUG:
59             print(mensajeCALO[i], end='')
60         uart.write(str(mensajeCALO[i]))
61         if DEBUG:
62             time.sleep(0.1)
63
64 # Calibración "short"
65 def pulsarCALS(p):
66     for i in range(len(mensajeCALS)):
67         if DEBUG:
68             print(mensajeCALS[i], end='')
69         uart.write(str(mensajeCALS[i]))
70         if DEBUG:
71             time.sleep(0.1)
72
73 botonON.irq(trigger=Pin.IRQ_RISING, handler=pulsarON)
74 botonCALL.irq(trigger=Pin.IRQ_RISING, handler=pulsarCALL)
75 botonCALO.irq(trigger=Pin.IRQ_RISING, handler=pulsarCALO)
76 botonCALS.irq(trigger=Pin.IRQ_RISING, handler=pulsarCALS)
77
78 measured = [[None for i in range(2)] for j in range(101)]
79 permitividad_compleja = [None for i in range(101)]
80 permitividad_re_im = [[None for i in range(2)] for j in ...
       range(101)]
81 i = 0
82 medido = 0
83 num_fichero = '1'
84 n_puntos = 5
85
86 # Constantes para la permitividad
87 c =  [[complex(-1.69, -7.32), complex(-1.79, -7.49), ...
       complex(1.00, -0.01)], [complex(-83.93, -867.51), ...
       complex(-137.77, -860.57), complex(-0.85, 0.23)],
88        [complex(13.16, -460.70), complex(-44.05, -458.85), ...
             complex(-1.00, 0.07)], [complex(17.03, -313.73), ...
             complex(-41.43, -311.76), complex(-1.06, 0.11)],
```

```
89        [complex(2.09, -234.08), complex(-55.36, -228.01), ...
             complex(-0.99, 0.25)], [complex(-1.70, -183.19), ...
             complex(-57.20, -174.37), complex(-0.92, 0.32)],
90        [complex(0.92, -153.68), complex(-55.50, -143.64), ...
             complex(-0.95, 0.37)], [complex(1.61, -131.26), ...
             complex(-54.23, -120.17), complex(-0.93, 0.42)],
91        [complex(0.85, -114.67), complex(-54.09, -101.54), ...
             complex(-0.89, 0.48)], [complex(0.06, -102.54), ...
             complex(-54.40, -87.25), complex(-0.85, 0.55)],
92        [complex(-0.13, -92.08), complex(-53.80, -75.13), ...
             complex(-0.81, 0.60)], [complex(-0.51, -83.48), ...
             complex(-53.21, -64.79), complex(-0.76, 0.66)],
93        [complex(-0.61, -75.83), complex(-51.74, -55.61), ...
             complex(-0.71, 0.69)], [complex(-0.49, -70.63), ...
             complex(-51.14, -48.77), complex(-0.67, 0.75)],
94        [complex(-0.73, -65.68), complex(-50.50, -42.19), ...
             complex(-0.62, 0.79)], [complex(-1.14, -60.98), ...
             complex(-49.41, -35.69), complex(-0.56, 0.82)],
95        [complex(-0.84, -57.32), complex(-48.42, -30.81), ...
             complex(-0.51, 0.86)], [complex(-1.03, -54.11), ...
             complex(-47.55, -25.94), complex(-0.45, 0.89)],
96        [complex(-1.19, -50.87), complex(-46.21, -21.34), ...
             complex(-0.39, 0.91)], [complex(-1.00, -48.25), ...
             complex(-45.01, -17.70), complex(-0.33, 0.94)],
97        [complex(0.14, -46.72), complex(-43.34, -16.10), ...
             complex(-0.29, 0.97)], [complex(-0.93, -43.56), ...
             complex(-41.54, -11.44), complex(-0.21, 0.94)],
98        [complex(-0.85, -41.59), complex(-40.22, -8.71), ...
             complex(-0.15, 0.97)], [complex(-1.94, -39.46), ...
             complex(-38.69, -4.92), complex(-0.07, 0.95)],
99        [complex(-1.59, -38.02), complex(-37.23, -2.71), ...
             complex(-0.02, 0.97)], [complex(-1.67, -36.22), ...
             complex(-35.54, -0.26), complex(0.04, 0.96)],
100       [complex(2.04, -32.01), complex(-31.02, -2.89), ...
             complex(-0.04, 1.01)], [complex(-2.20, -33.34), ...
             complex(-32.42, 3.99), complex(0.16, 0.95)],
101       [complex(-2.62, -32.20), complex(-30.78, 5.94), ...
             complex(0.22, 0.93)], [complex(-2.59, -31.35), ...
             complex(-29.64, 7.61), complex(0.28, 0.91)],
102       [complex(-2.29, -30.20), complex(-28.12, 8.64), ...
             complex(0.33, 0.92)], [complex(-2.57, -28.82), ...
             complex(-26.48, 10.12), complex(0.40, 0.89)],
103       [complex(-3.05, -28.21), complex(-25.14, 11.76), ...
             complex(0.47, 0.86)], [complex(-3.00, -27.01), ...
             complex(-23.50, 12.70), complex(0.52, 0.82)],
104       [complex(-3.13, -25.98), complex(-21.75, 13.85), ...
             complex(0.57, 0.78)], [complex(-3.40, -24.52), ...
             complex(-19.46, 14.57), complex(0.61, 0.72)],
105       [complex(-3.42, -23.66), complex(-17.89, 15.33), ...
```

```
            complex(0.66, 0.68)], [complex(-3.49, -22.83), ...
            complex(-16.27, 15.88), complex(0.71, 0.62)],
106         [complex(-3.43, -22.19), complex(-14.78, 16.67), ...
            complex(0.75, 0.57)], [complex(-3.70, -21.77), ...
            complex(-13.36, 17.55), complex(0.80, 0.47)],
107         [complex(-3.62, -21.27), complex(-11.81, 17.93), ...
            complex(0.82, 0.44)], [complex(-3.49, -20.66), ...
            complex(-10.46, 18.05), complex(0.84, 0.39)],
108         [complex(-3.50, -20.29), complex(-9.05, 18.43), ...
            complex(0.86, 0.32)], [complex(-3.53, -19.76), ...
            complex(-7.65, 18.54), complex(0.88, 0.26)],
109         [complex(-3.82, -19.24), complex(-6.01, 18.69), ...
            complex(0.89, 0.18)], [complex(-4.15, -19.10), ...
            complex(-4.43, 19.05), complex(0.89, 0.10)],
110         [complex(-4.18, -18.58), complex(-2.96, 18.78), ...
            complex(0.89, 0.04)], [complex(-4.14, -18.03), ...
            complex(-1.70, 18.42), complex(0.87, -0.01)],
111         [complex(-4.35, -17.39), complex(-0.16, 17.97), ...
            complex(0.83, -0.06)], [complex(-4.55, -17.21), ...
            complex(1.21, 17.76), complex(0.83, -0.13)],
112         [complex(-4.57, -16.62), complex(2.25, 17.15), ...
            complex(0.81, -0.16)], [complex(-4.58, -16.05), ...
            complex(3.32, 16.44), complex(0.80, -0.21)],
113         [complex(-4.86, -16.37), complex(4.79, 16.26), ...
            complex(0.77, -0.27)], [complex(-4.60, -15.65), ...
            complex(5.57, 15.53), complex(0.74, -0.33)],
114         [complex(-4.57, -15.34), complex(6.26, 14.88), ...
            complex(0.73, -0.37)], [complex(-3.96, -15.53), ...
            complex(6.76, 15.28), complex(0.74, -0.42)],
115         [complex(-3.69, -15.41), complex(7.81, 14.72), ...
            complex(0.73, -0.46)], [complex(-2.91, -14.52), ...
            complex(7.41, 13.68), complex(0.75, -0.46)],
116         [complex(-2.66, -12.36), complex(5.84, 12.69), ...
            complex(0.76, -0.45)], [complex(-2.51, -12.71), ...
            complex(6.62, 12.59), complex(0.73, -0.49)],
117         [complex(-2.75, -12.00), complex(7.15, 11.56), ...
            complex(0.70, -0.52)], [complex(-2.56, -11.54), ...
            complex(7.33, 10.71), complex(0.68, -0.53)],
118         [complex(-3.10, -10.94), complex(7.95, 9.41), ...
            complex(0.61, -0.54)], [complex(-2.73, -10.60), ...
            complex(7.67, 9.08), complex(0.61, -0.57)],
119         [complex(-2.96, -10.01), complex(7.99, 7.86), ...
            complex(0.58, -0.59)], [complex(-2.22, -9.60), ...
            complex(7.68, 7.35), complex(0.57, -0.61)],
120         [complex(-2.08, -9.80), complex(8.04, 6.72), ...
            complex(0.54, -0.66)], [complex(-2.21, -8.82), ...
            complex(7.29, 5.50), complex(0.51, -0.64)],
121         [complex(-2.60, -8.83), complex(8.12, 4.66), ...
            complex(0.43, -0.66)], [complex(-2.75, -9.06), ...
```

```
            complex(8.59, 3.79), complex(0.34, -0.71)],
122         [complex(-2.77, -8.79), complex(8.49, 3.12), ...
            complex(0.30, -0.73)], [complex(-2.18, -8.96), ...
            complex(8.49, 3.11), complex(0.26, -0.77)],
123         [complex(-2.53, -8.63), complex(8.16, 1.86), ...
            complex(0.23, -0.76)], [complex(-1.80, -8.79), ...
            complex(8.13, 1.66), complex(0.20, -0.80)],
124         [complex(-2.32, -8.12), complex(7.54, 0.56), ...
            complex(0.16, -0.74)], [complex(-2.21, -8.04), ...
            complex(7.48, -0.19), complex(0.09, -0.75)],
125         [complex(-2.90, -8.10), complex(7.16, -1.13), ...
            complex(0.02, -0.77)], [complex(-2.67, -8.44), ...
            complex(7.32, -1.49), complex(-0.05, -0.78)],
126         [complex(-3.26, -7.92), complex(6.73, -2.51), ...
            complex(-0.14, -0.73)], [complex(-3.13, -7.85), ...
            complex(6.23, -2.75), complex(-0.12, -0.74)],
127         [complex(-3.11, -6.58), complex(4.86, -3.03), ...
            complex(-0.10, -0.69)], [complex(-2.11, -5.47), ...
            complex(4.03, -2.18), complex(-0.06, -0.78)],
128         [complex(-1.98, -5.12), complex(3.60, -2.08), ...
            complex(-0.10, -0.82)], [complex(-0.97, -5.45), ...
            complex(4.32, -1.63), complex(-0.21, -0.85)],
129         [complex(-1.02, -4.54), complex(3.50, -1.62), ...
            complex(-0.26, -0.84)], [complex(-0.81, -3.71), ...
            complex(2.74, -1.30), complex(-0.24, -0.77)],
130         [complex(-1.17, -3.22), complex(2.29, -1.49), ...
            complex(-0.32, -0.67)], [complex(-0.92, -3.49), ...
            complex(2.33, -1.78), complex(-0.32, -0.63)],
131         [complex(-0.95, -1.81), complex(1.33, -0.62), ...
            complex(-0.33, -0.64)], [complex(0.27, 0.51), ...
            complex(0.90, 1.80), complex(-0.23, -0.74)],
132         [complex(2.17, 2.33), complex(1.73, 4.05), ...
            complex(-0.23, -0.77)], [complex(3.86, 1.32), ...
            complex(3.72, 3.78), complex(-0.36, -0.69)],
133         [complex(7.39, 0.86), complex(7.16, 4.60), ...
            complex(-0.45, -0.74)], [complex(9.18, -0.80), ...
            complex(9.73, 3.47), complex(-0.56, -0.77)],
134         [complex(7.12, -3.85), complex(8.87, -0.35), ...
            complex(-0.44, -0.59)], [complex(4.76, -4.82), ...
            complex(7.14, -2.02), complex(-0.55, -0.56)],
135         [complex(7.00, -1.61), complex(8.43, 1.01), ...
            complex(-0.48, -0.62)], [complex(0.46, -0.26), ...
            complex(1.94, 0.49), complex(-0.41, -0.48)],
136         [complex(-1.65, -1.43), complex(0.00, -1.11), ...
            complex(-0.39, -0.45)], [complex(-0.91, -7.67), ...
            complex(1.37, -7.28), complex(-0.50, -0.16)],
137         [complex(-2.19, -3.73), complex(-0.34, -3.37), ...
            complex(-0.65, -0.25)]]
138
```

```
139
140 # Bucle principal
141 while True:
142     if uart.any() >= 8:
143         buffer = uart.read(4)
144         [re] = struct.unpack('f', buffer)
145         if DEBUG:
146             print(re, end='')
147             print(' ', end='')
148         buffer = uart.read(4)
149         [im] = struct.unpack('f', buffer)
150         if DEBUG:
151             print(im)
152
153         reb = struct.pack('f', re)
154         imb = struct.pack('f', im)
155
156         if reb + imb == b'\r\n\r\n\r\n\r\n':
157             if DEBUG:
158                 print('\nFin barrido\n')
159                 print(measured)
160                 print()
161             i = 0
162             medido = medido + 1
163
164             if medido == 2:    # Me quedo con el segundo barrido
165                 medido = 0
166                 print("S11")
167                 print(measured)
168                 print("Permitividad")
169                 print(permitividad_re_im)
170
171                 # Guardar los datos en un .txt
172                 f_s11 = open('s11_' + num_fichero + '.txt', 'w')
173                 for k in range(len(measured)):
174                     f_s11.write(str(measured[k][0]) + '\t' + ...
                            str(measured[k][1]) + '\n')
175                 f_s11.close()
176
177                 f_perm = open('perm_' + num_fichero + ...
                        '.txt', 'w')
178                 for k in range(len(permitividad_re_im)):
179                     f_perm.write(str(permitividad_re_im[k][0]) ...
                            + '\t' + ...
                            str(permitividad_re_im[k][1]) + '\n')
180                 f_perm.close()
181
182                 num_fichero = str(int(num_fichero) + 1)
183
```

```
184                 for i in range(len(mensajeOFF)):
185                     if DEBUG:
186                         print(mensajeOFF[i], end='')
187                     uart.write(str(mensajeOFF[i]))
188                     if DEBUG:
189                         time.sleep(0.1)
190 
191         else:
192             # Relleno de matriz con los valores de S11
193             measured[i][0] = re
194             measured[i][1] = im
195 
196             # Relleno de matriz con los valores de la ...
                    permitividad
197             S11 = complex(re, im)
198             permitividad_compleja[i] = (c[i][0]*S11 - ...
                    c[i][1])/(c[i][2] - S11)
199             r, phi = cmath.polar(permitividad_compleja[i])
200             real, imag = r*math.cos(phi), r*math.sin(phi)
201             permitividad_re_im[i][0] = real
202             permitividad_re_im[i][1] = imag
203 
204             if DEBUG:
205                 print(measured)
206 
207             i = i + 1
208 
209 
210     # SDI-12
211     direccion_sdi12.value(0)
212     if uart_sdi12.any():
213         ch_sdi12 = uart_sdi12.read(1)
214         buffer_sdi12 += ch_sdi12
215         if DEBUG:
216             print('Buffer: ', end='')
217             print(buffer_sdi12)
218         if ch_sdi12 == b'!':
219             direccion_sdi12.value(1)
220             if b'?!' in buffer_sdi12: # ?!
221                 uart_sdi12.write(ID + '\r\n')
222                 time.sleep(0.04)
223 
224             elif bytes(ID, 'utf-8') + b'I!' in buffer_sdi12: ...
                    # aI!
225                 uart_sdi12.write(ID + ...
                        '14jgarrido0000001001VNA\r\n')
226                 time.sleep(0.26)
227 
228             elif bytes(ID, 'utf-8') + b'A' in buffer_sdi12: ...
```

```
                # aAb!
229             index = buffer_sdi12.index(b'A')
230             ID = chr(buffer_sdi12[index + 1])
231             uart_sdi12.write(ID + '\r\n')
232             time.sleep(0.04)
233
234         elif bytes(ID, 'utf-8') + b'M!' in buffer_sdi12: ...
                # aM!
235             uart_sdi12.write(ID + '0011\r\n')
236             time.sleep(0.08)
237
238         elif bytes(ID, 'utf-8') + b'D0!' in ...
                buffer_sdi12: # aD0! -> S11
239             mensaje = ID
240             for i in range(n_puntos):
241                 if measured[i][0] >= 0:
242                     mensaje += '+'
243                     mensaje += str(measured[i][0])
244                 else:
245                     mensaje += str(measured[i][0])
246                 if measured[i][1] >= 0:
247                     mensaje += '+'
248                     mensaje += str(measured[i][1])
249                 else:
250                     mensaje += str(measured[i][1])
251             mensaje += '\r\n'
252             uart_sdi12.write(mensaje)
253             time.sleep(0.30)
254
255         elif bytes(ID, 'utf-8') + b'D1!' in ...
                buffer_sdi12: # aD1! -> Permitividad
256             mensaje = ID
257             for i in range(n_puntos):
258                 if permitividad_re_im[i][0] >= 0:
259                     mensaje += '+'
260                     mensaje += str(permitividad_re_im[i][0])
261                 else:
262                     mensaje += str(permitividad_re_im[i][0])
263                 if permitividad_re_im[i][1] >= 0:
264                     mensaje += '+'
265                     mensaje += str(permitividad_re_im[i][1])
266                 else:
267                     mensaje += str(permitividad_re_im[i][1])
268             mensaje += '\r\n'
269             uart_sdi12.write(mensaje)
270             time.sleep(0.30)
271
272         elif bytes(ID, 'utf-8') + b'!' in buffer_sdi12: ...
                # a!
```

```
273                 uart_sdi12.write(ID + '\r\n')
274                 time.sleep(0.04)
275
276             buffer_sdi12 = b''
```

V.3. Código MATLAB para la calibración OWL

```
close all
clear all
clc
set(0, 'DefaultLineLineWidth', 1.25, ...
    'DefaultAxesTitleFontWeight', ...
    'bold','DefaultAxesFontSize', 11);

%% Cargar el workspace

% [num, txt, raw] = xlsread('MEDIDAS_VNA.xlsx');
% VNArawS11_2021 = cell2table(raw(2:end, :));
% VNArawS11_2021.Properties.VariableNames = raw(1, :);
% save VNArawS11_2021.mat VNArawS11_2021
load 'VNArawS11_2021.mat'

% Temperatura de las muestras (°C)
Temp.VNA.Air = 26.2;
Temp.VNA.DistWater = 25.9;
Temp.VNA.Acetone = 26.2;
Temp.VNA.Isopropanol = 26.2;
Temp.VNA.Methanol = 25.9;
Temp.VNA.EthyleneGlycol = 26.5;

% Gráficas del S11
col = javicolormap(12);
figure
hold on
p11 = plot(VNArawS11_2021.Frequency, VNArawS11_2021.RealAir, ...
    'Color', col(1,:), 'DisplayName', 'Real Aire', ...
    'LineWidth', 2)
p12 = plot(VNArawS11_2021.Frequency, VNArawS11_2021.ImagAir, ...
    'Color', col(2,:), 'DisplayName', 'Imag Aire', ...
    'LineWidth', 2)
p21 = plot(VNArawS11_2021.Frequency, ...
    VNArawS11_2021.RealIsopropanol, 'Color', col(3,:), ...
    'DisplayName', 'Real Isopropanol', 'LineWidth', 2)
p22 = plot(VNArawS11_2021.Frequency, ...
    VNArawS11_2021.ImagIsopropanol, 'Color', col(4,:), ...
    'DisplayName', 'Imag Isopropanol', 'LineWidth', 2)
p31 = plot(VNArawS11_2021.Frequency, ...
    VNArawS11_2021.RealAcetone, 'Color', col(5,:), ...
    'DisplayName', 'Real Acetona', 'LineWidth', 2)
p32 = plot(VNArawS11_2021.Frequency, ...
```

```
    VNArawS11_2021.ImagAcetone, 'Color', col(6,:), ...
    'DisplayName', 'Imag Acetona', 'LineWidth', 2)
p41 = plot(VNArawS11_2021.Frequency, ...
    VNArawS11_2021.RealMethanol, 'Color', col(7,:), ...
    'DisplayName', 'Real Metanol', 'LineWidth', 2)
p42 = plot(VNArawS11_2021.Frequency, ...
    VNArawS11_2021.ImagMethanol, 'Color', col(8,:), ...
    'DisplayName', 'Imag Metanol', 'LineWidth', 2)
p51 = plot(VNArawS11_2021.Frequency, ...
    VNArawS11_2021.RealEthyleneglycol, 'Color', col(9,:), ...
    'DisplayName', 'Real Etilenglicol', 'LineWidth', 2)
p52 = plot(VNArawS11_2021.Frequency, ...
    VNArawS11_2021.ImagEthyleneglycol, 'Color', col(10,:), ...
    'DisplayName', 'Imag Etilenglicol', 'LineWidth', 2)
p61 = plot(VNArawS11_2021.Frequency, ...
    VNArawS11_2021.RealDistWater, 'Color', col(11,:), ...
    'DisplayName', 'Real Agua destilada', 'LineWidth', 2)
p62 = plot(VNArawS11_2021.Frequency, ...
    VNArawS11_2021.ImagDistWater, 'Color', col(12,:), ...
    'DisplayName', 'Imag Agua destilada', 'LineWidth', 2)
legend([p11 p12 p21 p22 p31 p32 p41 p42 p51 p52 p61 p62], ...
    {'Real Aire', 'Imag Aire', 'Real Isopropanol', 'Imag ...
    Isopropanol', 'Real Acetona', 'Imag Acetona', 'Real ...
    Metanol', 'Imag Metanol', 'Real Etilenglicol', 'Imag ...
    Etilenglicol', 'Real Agua destilada', 'Imag Agua destilada'})
ylabel('S11'), xlabel('Frecuencia (Hz)'), set(gca, ...
    'FontSize',13)

%% Interpolación para normalizar las frecuencias de los ...
    datos del S11 del VNA

Frequency = VNArawS11_2021.Frequency';
VNAS11 = table(Frequency', 'VariableNames', {'Frequency'});

% Aire
[xData, yData] = prepareCurveData(VNArawS11_2021.Frequency, ...
    VNArawS11_2021.RealAir);
ft = 'pchipinterp';
[fitresult, gof] = fit(xData, yData, ft, 'Normalize', 'on');
realAir = fitresult(VNAS11.Frequency);
[xData, yData] = prepareCurveData(VNArawS11_2021.Frequency, ...
    VNArawS11_2021.ImagAir);
ft = 'pchipinterp';
[fitresult, gof] = fit(xData, yData, ft, 'Normalize', 'on');
imagAir = fitresult(VNAS11.Frequency);
VNAS11.Air = complex(realAir, imagAir);

% Isopropanol
[xData, yData] = prepareCurveData(VNArawS11_2021.Frequency, ...
```

```
    VNArawS11_2021.RealIsopropanol);
ft = 'pchipinterp';
[fitresult, gof] = fit(xData, yData, ft, 'Normalize', 'on');
realIsopropanol = fitresult(VNAS11.Frequency);
[xData, yData] = prepareCurveData(VNArawS11_2021.Frequency, ...
    VNArawS11_2021.ImagIsopropanol);
ft = 'pchipinterp';
[fitresult, gof] = fit(xData, yData, ft, 'Normalize', 'on');
imagIsopropanol = fitresult(VNAS11.Frequency);
VNAS11.Isopropanol = complex(realIsopropanol, imagIsopropanol);

% Acetona
[xData, yData] = prepareCurveData(VNArawS11_2021.Frequency, ...
    VNArawS11_2021.RealAcetone);
ft = 'pchipinterp';
[fitresult, gof] = fit(xData, yData, ft, 'Normalize', 'on');
realAcetone = fitresult(VNAS11.Frequency);
[xData, yData] = prepareCurveData(VNArawS11_2021.Frequency, ...
    VNArawS11_2021.ImagAcetone);
ft = 'pchipinterp';
[fitresult, gof] = fit(xData, yData, ft, 'Normalize', 'on');
imagAcetone = fitresult(VNAS11.Frequency);
VNAS11.Acetone = complex(realAcetone, imagAcetone);

% Metanol
[xData, yData] = prepareCurveData(VNArawS11_2021.Frequency, ...
    VNArawS11_2021.RealMethanol);
ft = 'pchipinterp';
[fitresult, gof] = fit(xData, yData, ft, 'Normalize', 'on');
realMethanol = fitresult(VNAS11.Frequency);
[xData, yData] = prepareCurveData(VNArawS11_2021.Frequency, ...
    VNArawS11_2021.ImagMethanol);
ft = 'pchipinterp';
[fitresult, gof] = fit(xData, yData, ft, 'Normalize', 'on');
imagMethanol = fitresult(VNAS11.Frequency);
VNAS11.Methanol = complex(realMethanol, imagMethanol);

% Etilenglicol
[xData, yData] = prepareCurveData(VNArawS11_2021.Frequency, ...
    VNArawS11_2021.RealEthyleneglycol);
ft = 'pchipinterp';
[fitresult, gof] = fit(xData, yData, ft, 'Normalize', 'on');
realEthyleneGlycol = fitresult(VNAS11.Frequency);
[xData, yData] = prepareCurveData(VNArawS11_2021.Frequency, ...
    VNArawS11_2021.ImagEthyleneglycol);
ft = 'pchipinterp';
[fitresult, gof] = fit(xData, yData, ft, 'Normalize', 'on');
imagEthyleneGlycol = fitresult(VNAS11.Frequency);
VNAS11.EthyleneGlycol = complex(realEthyleneGlycol, ...
```

```
    imagEthyleneGlycol);
100
101 % Agua destilada
102 [xData, yData] = prepareCurveData(VNArawS11_2021.Frequency, ...
    VNArawS11_2021.RealDistWater);
103 ft = 'pchipinterp';
104 [fitresult, gof] = fit(xData, yData, ft, 'Normalize', 'on');
105 realDistWater = fitresult(VNAS11.Frequency);
106 [xData, yData] = prepareCurveData(VNArawS11_2021.Frequency, ...
    VNArawS11_2021.ImagDistWater);
107 ft = 'pchipinterp';
108 [fitresult, gof] = fit(xData, yData, ft, 'Normalize', 'on');
109 imagDistWater = fitresult(VNAS11.Frequency);
110 VNAS11.DistWater = complex(realDistWater, imagDistWater);
111
112 clear xData yData fitresult ft gof realAir realIsopropanol ...
    realAcetone realDistWater realEthanol realMethanol ...
    imagAir imagIsopropanol imagAcetone imagDistWater ...
    imagEthanol imagMethanol realEthyleneGlycol ...
    imagEthyleneGlycol
113
114 %% Modelos para definir la permitividad de los materiales
115 % En la bibliografía se proporcionan datos a temperaturas ...
    redondeadas.
116 % Necesitamos calcular los valores correspondientes para la ...
    temperatura
117 % de nuestras muestras, por lo que aplicaremos una ...
    interpolación lineal
118 % en cada caso.
119
120 % Aire
121 Perm_Air.VNA = ones(1, length(VNAS11.Frequency));
122
123
124 % Agua destilada (Debye, con datos de Kaatze (1989); ...
    interpolación entre 25°C y 30°C)
125 Eps_0_DistWater.VNA = interp1([25,30], [78.36,76.56], ...
    Temp.VNA.DistWater);
126 RelaxT_DistWater.VNA = interp1([25,30], ...
    [8.27*10^-12,7.28*10^-12], Temp.VNA.DistWater);
127 Eps_inf_DistWater.VNA = interp1([25,30], [5.2,5.2], ...
    Temp.VNA.DistWater);
128
129 for ii=1:length(VNAS11.Frequency)
130     Perm_DistWater.VNA(ii) = Debye(Eps_0_DistWater.VNA, ...
        Eps_inf_DistWater.VNA, ...
        RelaxT_DistWater.VNA,VNAS11.Frequency(ii));
131 end
132
```

```
133
134 % Acetona (Cole-Cole, con datos de Buckley and Maryott ...
      (1958) (página 8); interpolación entre 20°C y 40°C)
135 Eps_0_Acetone.VNA = interp1([1,20,40], [23.29,21.2,19.29], ...
      Temp.VNA.Acetone);
136 Eps_inf_Acetone.VNA = interp1([1,20,40], [1.93,1.9,0.87], ...
      Temp.VNA.Acetone);
137 RelaxT_Acetone.VNA = interp1([1,20,40], [0.75,0.63,0.52], ...
      Temp.VNA.Acetone)*10^-2/(2*pi*physconst('LightSpeed'));
138
139 for ii=1:length(VNAS11.Frequency)
140     Perm_Acetone.VNA(ii) = ColeCole(Eps_0_Acetone.VNA, ...
          Eps_inf_Acetone.VNA, RelaxT_Acetone.VNA, 1, ...
          VNAS11.Frequency(ii));
141 end
142
143
144 % Isopropanol (Double Debye, con datos de Gregory and Clarke ...
      (2012); interpolación entre 25°C y 30°C)
145 Eps_s_Isopropanol.VNA = interp1([25,30], [19.3,18.5], ...
      Temp.VNA.Isopropanol);
146 fr1_Isopropanol.VNA = interp1([25,30], [0.443E9,0.557E9], ...
      Temp.VNA.Isopropanol);
147 Eps_h_Isopropanol.VNA = interp1([25,30], [3.551,3.538], ...
      Temp.VNA.Isopropanol);
148 fr2_Isopropanol.VNA = interp1([25,30], [5.999E9,6.655E9], ...
      Temp.VNA.Isopropanol);
149 Eps_inf_Isopropanol.VNA = interp1([25,30], [3.065,3.056], ...
      Temp.VNA.Isopropanol);
150
151 for ii=1:length(VNAS11.Frequency)
152     Perm_Isopropanol.VNA(ii) = Eps_inf_Isopropanol.VNA + ...
          (Eps_s_Isopropanol.VNA-Eps_h_Isopropanol.VNA) / ...
          (1+j*VNAS11.Frequency(ii)/fr1_Isopropanol.VNA) + ...
          (Eps_h_Isopropanol.VNA-Eps_inf_Isopropanol.VNA) / ...
          (1+j*VNAS11.Frequency(ii)/fr2_Isopropanol.VNA);
153 end
154
155
156 % Metanol (Single Debye, con datos de Gregory and Clarke ...
      (2012); interpolación entre 25°C y 30°C)
157 Eps_s_Methanol.VNA = interp1([25,30], [32.66,31.69], ...
      Temp.VNA.Methanol);
158 fr_Methanol.VNA = interp1([25,30], [3.141E9,3.49E9], ...
      Temp.VNA.Methanol);
159 Eps_inf_Methanol.VNA = interp1([25,30], [5.563,5.45], ...
      Temp.VNA.Methanol);
160
161 for ii=1:length(VNAS11.Frequency)
```

```
162     Perm_Methanol.VNA(ii) = Eps_inf_Methanol.VNA + ...
            (Eps_s_Methanol.VNA-Eps_inf_Methanol.VNA) / ...
            (1+j*VNAS11.Frequency(ii) / fr_Methanol.VNA);
163 end
164
165 % Etilenglicol (Davidson-Cole, con datos de Gregory and ...
       Clarke (2012); interpolación entre 25°C y 30°C)
166 Eps_0_EthyleneGlycol.VNA = interp1([25,30], [40.75,39.66], ...
       Temp.VNA.EthyleneGlycol);
167 Eps_inf_EthyleneGlycol.VNA = interp1([25,30], [4.7,4.712], ...
       Temp.VNA.EthyleneGlycol);
168 Relaxfreq_EthyleneGlycol.VNA = interp1([25,30], ...
       [1.19E9,1.451E9], Temp.VNA.EthyleneGlycol);
169 Beta.VNA = interp1([25,30], [0.859,0.864], ...
       Temp.VNA.EthyleneGlycol);
170
171 for ii=1:length(VNAS11.Frequency)
172     Perm_EthyleneGlycol.VNA(ii) = ...
            ColeDavidson(Eps_0_EthyleneGlycol.VNA, ...
            Eps_inf_EthyleneGlycol.VNA, ...
            Relaxfreq_EthyleneGlycol.VNA,Beta.VNA, ...
            VNAS11.Frequency(ii));
173 end
174
175 % Comparativa de las permitividades teóricas para la temperatura
176 % de las muestras
177 col = javicolormap(7);
178
179 figure
180 hold on
181 p1 = plot(VNAS11.Frequency, real(Perm_Air.VNA), ...
       'Color',col(1,:), 'DisplayName', 'Aire', 'LineWidth', 2)
182 scatter(VNAS11.Frequency, real(Perm_Air.VNA), 10, '*', ...
       'MarkerEdgeColor', col(1,:))
183 p2 = plot(VNAS11.Frequency, real(Perm_Isopropanol.VNA), ...
       'Color', col(2,:), 'DisplayName', 'Isopropanol', ...
       'LineWidth', 2)
184 scatter(VNAS11.Frequency, real(Perm_Isopropanol.VNA), 10, ...
       '*', 'MarkerEdgeColor', col(2,:))
185 p3 = plot(VNAS11.Frequency, real(Perm_Acetone.VNA), 'Color', ...
       col(3,:), 'DisplayName', 'Acetona', 'LineWidth', 2)
186 scatter(VNAS11.Frequency, real(Perm_Acetone.VNA), 10, '*', ...
       'MarkerEdgeColor', col(3,:))
187 p4 = plot(VNAS11.Frequency, real(Perm_Methanol.VNA), ...
       'Color', col(5,:), 'DisplayName', 'Metanol', 'LineWidth', 2)
188 scatter(VNAS11.Frequency, real(Perm_Methanol.VNA), 10, '*', ...
       'MarkerEdgeColor', col(5,:))
189 p5 = plot(VNAS11.Frequency, real(Perm_EthyleneGlycol.VNA), ...
       'Color', col(6,:), 'DisplayName', 'Etilenglicol', ...
```

```
    'LineWidth', 2)
190 scatter(VNAS11.Frequency, real(Perm_EthyleneGlycol.VNA), 10, ...
    '*', 'MarkerEdgeColor', col(6,:))
191 p6 = plot(VNAS11.Frequency, real(Perm_DistWater.VNA), ...
    'Color', col(7,:), 'DisplayName', 'Agua destilada', ...
    'LineWidth', 2)
192 scatter(VNAS11.Frequency, real(Perm_DistWater.VNA), 10, '*', ...
    'MarkerEdgeColor', col(7,:))
193 legend([p1 p2 p3 p4 p5 p6], {'Aire', 'Isopropanol', ...
    'Acetona', 'Metanol', 'Etilenglicol', 'Agua destilada'})
194 ylabel('\epsilon'), xlabel('Frecuencia (Hz)'), set(gca, ...
    'FontSize',13)
195
196 figure
197 hold on
198 p1 = plot(VNAS11.Frequency, -imag(Perm_Air.VNA), 'Color', ...
    col(1,:), 'DisplayName', 'Aire', 'LineWidth', 2)
199 scatter(VNAS11.Frequency, -imag(Perm_Air.VNA), 10, '*', ...
    'MarkerEdgeColor', col(1,:))
200 p2 = plot(VNAS11.Frequency, -imag(Perm_Isopropanol.VNA), ...
    'Color', col(2,:), 'DisplayName', 'Isopropanol', ...
    'LineWidth', 2)
201 scatter(VNAS11.Frequency, -imag(Perm_Isopropanol.VNA), 10, ...
    '*', 'MarkerEdgeColor', col(2,:))
202 p3 = plot(VNAS11.Frequency, -imag(Perm_Acetone.VNA), ...
    'Color', col(3,:), 'DisplayName', 'Acetona', 'LineWidth', 2)
203 scatter(VNAS11.Frequency, -imag(Perm_Acetone.VNA), 10, '*', ...
    'MarkerEdgeColor', col(3,:))
204 p4 = plot(VNAS11.Frequency, -imag(Perm_Methanol.VNA), ...
    'Color', col(5,:), 'DisplayName', 'Metanol', 'LineWidth', 2)
205 scatter(VNAS11.Frequency, -imag(Perm_Methanol.VNA), 10, '*', ...
    'MarkerEdgeColor', col(5,:))
206 p5 = plot(VNAS11.Frequency, -imag(Perm_EthyleneGlycol.VNA), ...
    'Color', col(6,:), 'DisplayName', 'Etilenglicol', ...
    'LineWidth', 2)
207 scatter(VNAS11.Frequency, -imag(Perm_EthyleneGlycol.VNA), ...
    10, '*', 'MarkerEdgeColor', col(6,:))
208 p6 = plot(VNAS11.Frequency, -imag(Perm_DistWater.VNA), ...
    'Color', col(7,:), 'DisplayName', 'Agua destilada', ...
    'LineWidth', 2)
209 scatter(VNAS11.Frequency, -imag(Perm_DistWater.VNA), 10, ...
    '*', 'MarkerEdgeColor', col(7,:))
210 legend([p1 p2 p3 p4 p5 p6], {'Aire', 'Isopropanol', ...
    'Acetona', 'Metanol', 'Etilenglicol', 'Agua destilada'})
211 ylabel('\epsilon'), xlabel('Frecuencia (Hz)'), set(gca, ...
    'FontSize',13)
212
213 %% Permitividad a partir del S11 del VNA
214 % Seguimos el procedimiento descrito en "Wagner et al. ...
```

```
    Numerical 3-D FEM
% and Experimental Analysis of the Open-Ended Coaxial Line ...
    Technique for
% Microwave Dielectric Spectroscopy on Soil", 2014.

% -> Calibración Open-Water-Liquid (OWL)


% Calibración con Open(Aire), Water(Agua destilada) y Liquid ...
    (Metanol).
for ii=1:length(VNAS11.Frequency)
    M(:,:,ii) = [VNAS11.Air(ii) -1 ...
        -Perm_Air.VNA(ii);VNAS11.DistWater(ii) -1 ...
        -Perm_DistWater.VNA(ii);VNAS11.Methanol(ii) -1 ...
        -Perm_Methanol.VNA(ii)];
    e(:,ii) = [-Perm_Air.VNA(ii)*VNAS11.Air(ii); ...
        -Perm_DistWater.VNA(ii)*VNAS11.DistWater(ii); ...
        -Perm_Methanol.VNA(ii)*VNAS11.Methanol(ii)];
    c(:,ii) = inv(M(:,:,ii))*e(:,ii);
end

for ii=1:length(VNAS11.Frequency)
    Estim_VNA_Perm_Air(ii) = ...
        transpose((c(1,ii)*VNAS11.Air(ii)-c(2,ii)) / ...
        (c(3,ii)-VNAS11.Air(ii)));
    Estim_VNA_Perm_Isopropanol(ii) = ...
        transpose((c(1,ii)*VNAS11.Isopropanol(ii)-c(2,ii)) / ...
        (c(3,ii)-VNAS11.Isopropanol(ii)));
    Estim_VNA_Perm_Acetone(ii) = ...
        transpose((c(1,ii)*VNAS11.Acetone(ii)-c(2,ii)) / ...
        (c(3,ii)-VNAS11.Acetone(ii)));
    Estim_VNA_Perm_Methanol(ii) = ...
        transpose((c(1,ii)*VNAS11.Methanol(ii)-c(2,ii)) / ...
        (c(3,ii)-VNAS11.Methanol(ii)));
    Estim_VNA_Perm_EthyleneGlycol(ii) = ...
        transpose((c(1,ii)*VNAS11.EthyleneGlycol(ii)-c(2,ii)) ...
        / (c(3,ii)-VNAS11.EthyleneGlycol(ii)));
    Estim_VNA_Perm_DistWater(ii) = ...
        transpose((c(1,ii)*VNAS11.DistWater(ii)-c(2,ii)) / ...
        (c(3,ii)-VNAS11.DistWater(ii)));
end

SE_Methanol_VNA = sum([abs(real(Estim_VNA_Perm_Air) - ...
    real(Perm_Air.VNA)),abs(real(Estim_VNA_Perm_Isopropanol) ...
    - real(Perm_Isopropanol.VNA)),...
    abs(real(Estim_VNA_Perm_Acetone) - ...
        real(Perm_Acetone.VNA)),...
    abs(real(Estim_VNA_Perm_Methanol) - ...
        real(Perm_Methanol.VNA)), ...
```

```
        abs(real(Estim_VNA_Perm_EthyleneGlycol) - ...
        real(Perm_EthyleneGlycol.VNA)),...
240     abs(real(Estim_VNA_Perm_DistWater) - ...
        real(Perm_DistWater.VNA))]);
241
242 RMSE_Methanol_VNA = ...
    sqrt((sum([(real(Estim_VNA_Perm_Isopropanol) - ...
    real(Perm_Isopropanol.VNA)).^2,...
243     (real(Estim_VNA_Perm_Acetone) - ...
        real(Perm_Acetone.VNA)).^2,...
244     (real(Estim_VNA_Perm_EthyleneGlycol) - ...
        real(Perm_EthyleneGlycol.VNA)).^2]))/(3*length(Estim_VNA_Perm_Air)));
245
246 figure
247 hold on
248 plot(VNAS11.Frequency,real(Perm_Air.VNA), 'Color', col(1,:))
249 scatter(VNAS11.Frequency, real(Estim_VNA_Perm_Air), 10, '*', ...
    'MarkerEdgeColor', col(1,:))
250 plot(VNAS11.Frequency, real(Perm_Isopropanol.VNA), 'Color', ...
    col(2,:))
251 scatter(VNAS11.Frequency, real(Estim_VNA_Perm_Isopropanol), ...
    10, '*', 'MarkerEdgeColor', col(2,:))
252 plot(VNAS11.Frequency, real(Perm_Acetone.VNA), 'Color', ...
    col(3,:))
253 scatter(VNAS11.Frequency, real(Estim_VNA_Perm_Acetone), 10, ...
    '*', 'MarkerEdgeColor', col(3,:))
254 plot(VNAS11.Frequency, real(Perm_Methanol.VNA), 'Color', ...
    col(5,:))
255 scatter(VNAS11.Frequency, real(Estim_VNA_Perm_Methanol), 10, ...
    '*', 'MarkerEdgeColor', col(5,:))
256 plot(VNAS11.Frequency, real(Perm_EthyleneGlycol.VNA), ...
    'Color', col(6,:))
257 scatter(VNAS11.Frequency, ...
    real(Estim_VNA_Perm_EthyleneGlycol), 10, '*', ...
    'MarkerEdgeColor', col(6,:))
258 plot(VNAS11.Frequency, real(Perm_DistWater.VNA), 'Color', ...
    col(7,:))
259 scatter(VNAS11.Frequency, real(Estim_VNA_Perm_DistWater), ...
    10, '*', 'MarkerEdgeColor', col(7,:))
260 ylim([0 80])
261 %xlim([0 900e6])
262 title('Calibración OWL con Metanol')
263 ylabel('Re(\epsilon)')
264 xlabel('Frecuencia (Hz)')
265 legend('Aire modelo', 'Aire estimada', 'Isopropanol modelo', ...
    'Isopropanol estimada', 'Acetona modelo', 'Acetona ...
    estimada', 'Metanol modelo', 'Metanol estimada', ...
    'Etilenglicol modelo', 'Etilenglicol estimada', 'Agua ...
    destilada modelo', 'Agua destilada estimada')
```

```
266
267
268 % Calibración con Open(Aire), Water(Agua destilada) y Liquid ...
       (Isopropanol)
269 for ii=1:length(VNAS11.Frequency)
270     M(:,:,ii) = [VNAS11.Air(ii) -1 ...
            -Perm_Air.VNA(ii);VNAS11.DistWater(ii) -1 ...
            -Perm_DistWater.VNA(ii);VNAS11.Isopropanol(ii) -1 ...
            -Perm_Isopropanol.VNA(ii)];
271     e(:,ii) = [-Perm_Air.VNA(ii)*VNAS11.Air(ii); ...
            -Perm_DistWater.VNA(ii)*VNAS11.DistWater(ii); ...
            -Perm_Isopropanol.VNA(ii)*VNAS11.Isopropanol(ii);];
272     c(:,ii) = inv(M(:,:,ii))*e(:,ii);
273 end
274
275 for ii=1:length(VNAS11.Frequency)
276     Estim_VNA_Perm_Air(ii) = ...
            transpose((c(1,ii)*VNAS11.Air(ii)-c(2,ii)) / ...
            (c(3,ii)-VNAS11.Air(ii)));
277     Estim_VNA_Perm_Isopropanol(ii) = ...
            transpose((c(1,ii)*VNAS11.Isopropanol(ii)-c(2,ii)) / ...
            (c(3,ii)-VNAS11.Isopropanol(ii)));
278     Estim_VNA_Perm_Acetone(ii) = ...
            transpose((c(1,ii)*VNAS11.Acetone(ii)-c(2,ii)) / ...
            (c(3,ii)-VNAS11.Acetone(ii)));
279     Estim_VNA_Perm_Methanol(ii) = ...
            transpose((c(1,ii)*VNAS11.Methanol(ii)-c(2,ii)) / ...
            (c(3,ii)-VNAS11.Methanol(ii)));
280     Estim_VNA_Perm_EthyleneGlycol(ii) = ...
            transpose((c(1,ii)*VNAS11.EthyleneGlycol(ii)-c(2,ii)) ...
            / (c(3,ii)-VNAS11.EthyleneGlycol(ii)));
281     Estim_VNA_Perm_DistWater(ii) = ...
            transpose((c(1,ii)*VNAS11.DistWater(ii)-c(2,ii)) / ...
            (c(3,ii)-VNAS11.DistWater(ii)));
282 end
283 SE_Isopropanol_VNA = sum([abs(real(Estim_VNA_Perm_Air) - ...
       real(Perm_Air.VNA)),abs(real(Estim_VNA_Perm_Isopropanol) ...
       - real(Perm_Isopropanol.VNA)),...
284     abs(real(Estim_VNA_Perm_Acetone) - ...
            real(Perm_Acetone.VNA)),...
285     abs(real(Estim_VNA_Perm_Methanol) - ...
            real(Perm_Methanol.VNA)), ...
            abs(real(Estim_VNA_Perm_EthyleneGlycol) - ...
            real(Perm_EthyleneGlycol.VNA)),...
286     abs(real(Estim_VNA_Perm_DistWater) - ...
            real(Perm_DistWater.VNA))]);
287 RMSE_Isopropanol_VNA = ...
       sqrt((sum([(real(Estim_VNA_Perm_Methanol) - ...
       real(Perm_Methanol.VNA)).^2,...
```

```
288      (real(Estim_VNA_Perm_Acetone) - ...
            real(Perm_Acetone.VNA)).^2,...
289      (real(Estim_VNA_Perm_EthyleneGlycol) - ...
            real(Perm_EthyleneGlycol.VNA)).^2]))/(3*length(Estim_VNA_Perm_Air)));
290
291  figure
292  hold on
293  plot(VNAS11.Frequency, real(Perm_Air.VNA), 'Color', col(1,:))
294  scatter(VNAS11.Frequency, real(Estim_VNA_Perm_Air), 10, '*', ...
        'MarkerEdgeColor', col(1,:))
295  plot(VNAS11.Frequency, real(Perm_Isopropanol.VNA), 'Color', ...
        col(2,:))
296  scatter(VNAS11.Frequency, real(Estim_VNA_Perm_Isopropanol), ...
        10, '*', 'MarkerEdgeColor', col(2,:))
297  plot(VNAS11.Frequency, real(Perm_Acetone.VNA), 'Color', ...
        col(3,:))
298  scatter(VNAS11.Frequency, real(Estim_VNA_Perm_Acetone), 10, ...
        '*', 'MarkerEdgeColor', col(3,:))
299  plot(VNAS11.Frequency, real(Perm_Methanol.VNA), 'Color', ...
        col(5,:))
300  scatter(VNAS11.Frequency, real(Estim_VNA_Perm_Methanol), 10, ...
        '*', 'MarkerEdgeColor', col(5,:))
301  plot(VNAS11.Frequency, real(Perm_EthyleneGlycol.VNA), ...
        'Color', col(6,:))
302  scatter(VNAS11.Frequency, ...
        real(Estim_VNA_Perm_EthyleneGlycol), 10, '*', ...
        'MarkerEdgeColor', col(6,:))
303  plot(VNAS11.Frequency, real(Perm_DistWater.VNA), 'Color', ...
        col(7,:))
304  scatter(VNAS11.Frequency, real(Estim_VNA_Perm_DistWater), ...
        10, '*', 'MarkerEdgeColor', col(7,:))
305  ylim([0 80])
306  %xlim([0 900e6])
307  title('Calibración OWL con Isopropanol')
308  ylabel('Re(\epsilon)')
309  xlabel('Frecuencia (Hz)')
310  legend('Aire modelo', 'Aire estimada', 'Isopropanol modelo', ...
        'Isopropanol estimada', 'Acetona modelo', 'Acetona ...
        estimada', 'Metanol modelo', 'Metanol estimada', ...
        'Etilenglicol modelo', 'Etilenglicol estimada', 'Agua ...
        destilada modelo', 'Agua destilada estimada')
311
312
313  % Calibración con Open(Aire), Water(Agua destilada) y Liquid ...
        (Acetona)
314  % Este es el mejor (error más bajo)
315  for ii=1:length(VNAS11.Frequency)
316      M(:,:,ii) = [VNAS11.Air(ii) -1 ...
            -Perm_Air.VNA(ii);VNAS11.DistWater(ii) -1 ...
```

```
        -Perm_DistWater.VNA(ii);VNAS11.Acetone(ii) -1 ...
        -Perm_Acetone.VNA(ii)];
317     e(:,ii) = [-Perm_Air.VNA(ii)*VNAS11.Air(ii); ...
        -Perm_DistWater.VNA(ii)*VNAS11.DistWater(ii); ...
        -Perm_Acetone.VNA(ii)*VNAS11.Acetone(ii);];
318     c(:,ii) = inv(M(:,:,ii))*e(:,ii);
319 end
320
321 for ii=1:length(VNAS11.Frequency)
322     Estim_VNA_Perm_Air(ii) = ...
        transpose((c(1,ii)*VNAS11.Air(ii)-c(2,ii)) / ...
        (c(3,ii)-VNAS11.Air(ii)));
323     Estim_VNA_Perm_Isopropanol(ii) = ...
        transpose((c(1,ii)*VNAS11.Isopropanol(ii)-c(2,ii)) / ...
        (c(3,ii)-VNAS11.Isopropanol(ii)));
324     Estim_VNA_Perm_Acetone(ii) = ...
        transpose((c(1,ii)*VNAS11.Acetone(ii)-c(2,ii)) / ...
        (c(3,ii)-VNAS11.Acetone(ii)));
325     Estim_VNA_Perm_Methanol(ii) = ...
        transpose((c(1,ii)*VNAS11.Methanol(ii)-c(2,ii)) / ...
        (c(3,ii)-VNAS11.Methanol(ii)));
326     Estim_VNA_Perm_EthyleneGlycol(ii) = ...
        transpose((c(1,ii)*VNAS11.EthyleneGlycol(ii)-c(2,ii)) ...
        / (c(3,ii)-VNAS11.EthyleneGlycol(ii)));
327     Estim_VNA_Perm_DistWater(ii) = ...
        transpose((c(1,ii)*VNAS11.DistWater(ii)-c(2,ii)) / ...
        (c(3,ii)-VNAS11.DistWater(ii)));
328 end
329 SE_Acetone_VNA = sum([abs(real(Estim_VNA_Perm_Air) - ...
    real(Perm_Air.VNA)),abs(real(Estim_VNA_Perm_Isopropanol) ...
    - real(Perm_Isopropanol.VNA)),...
330     abs(real(Estim_VNA_Perm_Acetone) - ...
        real(Perm_Acetone.VNA)),...
331     abs(real(Estim_VNA_Perm_Methanol) - ...
        real(Perm_Methanol.VNA)),abs(real(Estim_VNA_Perm_EthyleneGlycol) ...
        - real(Perm_EthyleneGlycol.VNA)),...
332     abs(real(Estim_VNA_Perm_DistWater) - ...
        real(Perm_DistWater.VNA))]);
333 RMSE_Acetone_VNA = ...
    sqrt((sum([(real(Estim_VNA_Perm_Isopropanol) - ...
    real(Perm_Isopropanol.VNA)).^2,...
334     (real(Estim_VNA_Perm_Methanol) - ...
        real(Perm_Methanol.VNA)).^2,...
335     (real(Estim_VNA_Perm_EthyleneGlycol) - ...
        real(Perm_EthyleneGlycol.VNA)).^2]))/(3*length(Estim_VNA_Perm_Air)));
336
337 figure
338 hold on
339 plot(VNAS11.Frequency, real(Perm_Air.VNA), 'Color', col(1,:))
```

```
340 scatter(VNAS11.Frequency, real(Estim_VNA_Perm_Air), 10, '*', ...
        'MarkerEdgeColor', col(1,:))
341 plot(VNAS11.Frequency, real(Perm_Isopropanol.VNA), 'Color', ...
        col(2,:))
342 scatter(VNAS11.Frequency, real(Estim_VNA_Perm_Isopropanol), ...
        10, '*', 'MarkerEdgeColor', col(2,:))
343 plot(VNAS11.Frequency, real(Perm_Acetone.VNA), 'Color', ...
        col(3,:))
344 scatter(VNAS11.Frequency, real(Estim_VNA_Perm_Acetone), 10, ...
        '*', 'MarkerEdgeColor', col(3,:))
345 plot(VNAS11.Frequency, real(Perm_Methanol.VNA), 'Color', ...
        col(5,:))
346 scatter(VNAS11.Frequency, real(Estim_VNA_Perm_Methanol), 10, ...
        '*', 'MarkerEdgeColor', col(5,:))
347 plot(VNAS11.Frequency, real(Perm_EthyleneGlycol.VNA), ...
        'Color', col(6,:))
348 scatter(VNAS11.Frequency, ...
        real(Estim_VNA_Perm_EthyleneGlycol), 10, '*', ...
        'MarkerEdgeColor', col(6,:))
349 plot(VNAS11.Frequency, real(Perm_DistWater.VNA), 'Color', ...
        col(7,:))
350 scatter(VNAS11.Frequency, real(Estim_VNA_Perm_DistWater), ...
        10, '*', 'MarkerEdgeColor', col(7,:))
351 ylim([0 80])
352 %xlim([0 900e6])
353 title('Calibración OWL con Acetona')
354 ylabel('Re(\epsilon)')
355 xlabel('Frecuencia (Hz)')
356 legend('Aire modelo', 'Aire estimada', 'Isopropanol modelo', ...
        'Isopropanol estimada', 'Acetona modelo', 'Acetona ...
        estimada', 'Metanol modelo', 'Metanol estimada', ...
        'Etilenglicol modelo', 'Etilenglicol estimada', 'Agua ...
        destilada modelo', 'Agua destilada estimada')
357
358
359 % Calibración con Open(Aire), Water(Agua destilada) y Liquid ...
        (Etilenglicol)
360 for ii=1:length(VNAS11.Frequency)
361     M(:,:,ii) = [VNAS11.Air(ii) -1 ...
            -Perm_Air.VNA(ii);VNAS11.DistWater(ii) -1 ...
            -Perm_DistWater.VNA(ii);VNAS11.EthyleneGlycol(ii) -1 ...
            -Perm_EthyleneGlycol.VNA(ii)];
362     e(:,ii) = [-Perm_Air.VNA(ii)*VNAS11.Air(ii); ...
            -Perm_DistWater.VNA(ii)*VNAS11.DistWater(ii); ...
            -Perm_EthyleneGlycol.VNA(ii)*VNAS11.EthyleneGlycol(ii);];
363     c(:,ii) = inv(M(:,:,ii))*e(:,ii);
364 end
365
366 for ii=1:length(VNAS11.Frequency)
```

```
367     Estim_VNA_Perm_Air(ii) = ...
            transpose((c(1,ii)*VNAS11.Air(ii)-c(2,ii)) / ...
            (c(3,ii)-VNAS11.Air(ii)));
368     Estim_VNA_Perm_Isopropanol(ii) = ...
            transpose((c(1,ii)*VNAS11.Isopropanol(ii)-c(2,ii)) / ...
            (c(3,ii)-VNAS11.Isopropanol(ii)));
369     Estim_VNA_Perm_Acetone(ii) = ...
            transpose((c(1,ii)*VNAS11.Acetone(ii)-c(2,ii)) / ...
            (c(3,ii)-VNAS11.Acetone(ii)));
370     Estim_VNA_Perm_Methanol(ii) = ...
            transpose((c(1,ii)*VNAS11.Methanol(ii)-c(2,ii)) / ...
            (c(3,ii)-VNAS11.Methanol(ii)));
371     Estim_VNA_Perm_EthyleneGlycol(ii) = ...
            transpose((c(1,ii)*VNAS11.EthyleneGlycol(ii)-c(2,ii)) ...
            / (c(3,ii)-VNAS11.EthyleneGlycol(ii)));
372     Estim_VNA_Perm_DistWater(ii) = ...
            transpose((c(1,ii)*VNAS11.DistWater(ii)-c(2,ii)) / ...
            (c(3,ii)-VNAS11.DistWater(ii)));
373 end
374 SE_EthyleneGlycol_VNA = sum([abs(real(Estim_VNA_Perm_Air) - ...
        real(Perm_Air.VNA)),abs(real(Estim_VNA_Perm_Isopropanol) ...
        - real(Perm_Isopropanol.VNA)),...
375      abs(real(Estim_VNA_Perm_Acetone) - ...
            real(Perm_Acetone.VNA)),...
376      abs(real(Estim_VNA_Perm_Methanol) - ...
            real(Perm_Methanol.VNA)), ...
            abs(real(Estim_VNA_Perm_EthyleneGlycol) - ...
            real(Perm_EthyleneGlycol.VNA)),...
377      abs(real(Estim_VNA_Perm_DistWater) - ...
            real(Perm_DistWater.VNA))]);
378 RMSE_EthyleneGlycol_VNA = ...
        sqrt((sum([(real(Estim_VNA_Perm_Isopropanol) - ...
        real(Perm_Isopropanol.VNA)).^2,...
379      (real(Estim_VNA_Perm_Acetone) - ...
            real(Perm_Acetone.VNA)).^2,...
380      (real(Estim_VNA_Perm_Methanol) - ...
            real(Perm_Methanol.VNA)).^2]))/(3*length(Estim_VNA_Perm_Air)));
381
382 figure
383 hold on
384 plot(VNAS11.Frequency, real(Perm_Air.VNA), 'Color', col(1,:))
385 scatter(VNAS11.Frequency, real(Estim_VNA_Perm_Air), 10, '*', ...
        'MarkerEdgeColor', col(1,:))
386 plot(VNAS11.Frequency, real(Perm_Isopropanol.VNA), 'Color', ...
        col(2,:))
387 scatter(VNAS11.Frequency, real(Estim_VNA_Perm_Isopropanol), ...
        10, '*', 'MarkerEdgeColor', col(2,:))
388 plot(VNAS11.Frequency, real(Perm_Acetone.VNA), 'Color', ...
        col(3,:))
```

```
389 scatter(VNAS11.Frequency, real(Estim_VNA_Perm_Acetone), 10, ...
        '*', 'MarkerEdgeColor', col(3,:))
390 plot(VNAS11.Frequency, real(Perm_Methanol.VNA), 'Color', ...
        col(5,:))
391 scatter(VNAS11.Frequency, real(Estim_VNA_Perm_Methanol), 10, ...
        '*', 'MarkerEdgeColor', col(5,:))
392 plot(VNAS11.Frequency, real(Perm_EthyleneGlycol.VNA), ...
        'Color', col(6,:))
393 scatter(VNAS11.Frequency, ...
        real(Estim_VNA_Perm_EthyleneGlycol), 10, '*', ...
        'MarkerEdgeColor', col(6,:))
394 plot(VNAS11.Frequency, real(Perm_DistWater.VNA), 'Color', ...
        col(7,:))
395 scatter(VNAS11.Frequency, real(Estim_VNA_Perm_DistWater), ...
        10, '*', 'MarkerEdgeColor', col(7,:))
396 ylim([0 80])
397 %xlim([0 900e6])
398 title('Calibración OWL con Etilenglicol')
399 ylabel('Re(\epsilon)')
400 xlabel('Frecuencia (Hz)')
401 legend('Aire modelo', 'Aire estimada', 'Isopropanol modelo', ...
        'Isopropanol estimada', 'Acetona modelo', 'Acetona ...
        estimada', 'Metanol modelo', 'Metanol estimada', ...
        'Etilenglicol modelo', 'Etilenglicol estimada', 'Agua ...
        destilada modelo', 'Agua destilada estimada')
402
403
404
405 %% Funciones
406 function [Debye_Perm] = Debye(eps_0, eps_inf, relaxT, freq)
407 % Función que calcula el modelo de Debye de medios dieléctricos
408 % La función ofrece como salida el valor complejo de la ...
        permitividad a partir de las entradas: 1) 'eps_0' = ...
        Permitividad estática; 2) 'eps_inf' = Permitividad en el ...
        infinito; 3) 'relaxT' = Tiempo de relajación (s); 4) ...
        'freq' = Frecuencia (Hz)
409
410     Debye_Perm = eps_inf+(eps_0 - eps_inf) / (1 + ...
            j*2*pi*freq*relaxT);
411 end
412
413 function [ColeCole_Perm] = ColeCole(eps_0, eps_inf, relaxT, ...
        beta, freq)
414 % Función que calcula el modelo de Cole-Cole de medios ...
        dieléctricos
415 % La función ofrece como salida el valor complejo de la ...
        permitividad a partir de las entradas: 1) 'eps_0' = ...
        Permitividad estática; 2) 'eps_inf' = Permitividad en el ...
        infinito; 3) 'relaxT' = Tiempo de relajación (s); 4) ...
```

```
    'beta' = Parámetro que caracteriza los tiempos de la ...
    distribución de relajación; 5) 'freq' = Frecuencia (Hz)
416
417     ColeCole_Perm = eps_inf+(eps_0 - eps_inf) / (1 + ...
            (j*2*pi*freq*relaxT)^beta);
418 end
419
420 function [ColeDavidson_Perm] = ColeDavidson(eps_0, eps_inf, ...
    relaxfreq, beta, freq)
421 % Función que calcula el modelo de Davidson-Cole de medios ...
    dieléctricos
422 % La función ofrece como salida el valor complejo de la ...
    permitividad a partir de las entradas: 1) 'eps_0' = ...
    Permitividad estática; 2) 'eps_inf' = Permitividad en el ...
    infinito; 3) 'relaxT' = Tiempo de relajación (s); 4) ...
    'beta' = Parámetro que caracteriza los tiempos de la ...
    distribución de relajación; 5) 'freq' = Frecuencia (Hz)
423
424     ColeDavidson_Perm = eps_inf+(eps_0 - eps_inf) / (1 + ...
            j*freq/relaxfreq)^beta;
425 end
426
427
428
429
430
431
432
433 function J = javicolormap(m)
434     % Mapa de colores
435
436     c = [1, 60, 244;
437         239, 1, 51;
438         48, 239, 36;
439         194, 85, 230;
440         252, 166, 26;
441         29, 214, 255;
442         254, 139, 255;
443         41, 238, 162;
444         141, 24, 249;
445         247, 193, 34;
446         20, 128, 253;
447         253, 44, 253;
448         161, 252, 55;
449         96, 4, 199;
450         168, 104, 0];
451
452     if m <= length(c)
453         J = zeros(m,3);
```

```
454         for ii=1:m
455             J(ii,:) = c(ii,:)./255;
456         end
457     else
458         J = zeros(m,3);
459         positions = linspace(0,1,length(c));
460         J = customcolormap(positions,c,m);
461         J = J./255;
462     end
463 end
```

Bibliografía

[1] M. Hiebel, *Fundamentals of Vector Network Analysis*, 2007.

[2] N. Wagner et al., *Numerical 3-D FEM and Experimental Analysis of the Open-Ended Coaxial Line Technique for Microwave Dielectric Spectroscopy on Soil*, 2014.

[3] Juan D. et al., *Evaluating a low-cost self-manufactured Coaxial Open-Ended Probe for the Measurement of the Complex Permittivity of Granular Media*, 2021.

[4] U. Kaatze, *Complex Permittivity of Water as a Function of Frequency and Temperature*, 1989.

[5] F. Buckley, A. A. Maryott, *Circular of the Bureau of Standards no. 589: Tables of Dielectric Dispersion Data for Pure Liquids and Dilute Solutions*, 1958.

[6] A. P. Gregory, R. N. Clarke, *Tables of the Complex Permittivity of Dielectric Reference Liquids at Frequencies up to 5 GHz*, 2012.

[7] S. Havriliak, S. Negami, *A Complex Plane Representation of Dielectric and Mechanical Relaxation Processes in Some Polymers*, 1967.

[8] U. Kaatze, *Measuring the dielectric properties ofmaterials. Ninety-year development fromlow-frequency techniques to broadbandspectroscopy and high-frequency imaging*, 2012.

[9] T. A. Belyaeva et al., *Complex Dielectric Permittivity of Saline Soils and Rocks at Frequencies from 10 kHz to 8 GHz*, 2017.

[10] F. M. Francisca, V. A. Rinaldi, *Complex Dielectric Permittivity of Soil–Organic Mixtures (20 MHz–1.3 GHz)*, 2003.

[11] Juan D. et al., *Dielectric Spectroscopy and Application of Mixing Models Describing Dielectric Dispersion in Clay Minerals and Clayey Soils*, 2020.

[12] J. M. Miranda, J. L. Sebastián, M. Sierra, J. Margineda, *Ingeniería de Microondas, Técnicas experimentales*, 2002.

[13] L. F. Chen, C. K. Ong, C. P. Neo, *Microwave Electronics: Measurement and Materials Characterization*, 2004.

[14] W. H. Hayt, J. A. Buck, *Engineering Electromagnetics*, 2000.

[15] Agilent Technologies, *Agilent Network Analyzer Basics*, 2004.

[16] Agilent Technologies, *Agilent Basics of Measuring the Dielectric Properties of Materials. Application Note*, 2006.

[17] M. W. Hyde, M. J. Havrilla, A. E. Bogle, N. J. Lehman, *Broadband Characterization of Materials Using a Dual-Rigded Waveguide*, 2013.

[18] K. Saeed, R. D. Pollard, C. Hunter, *Substrate Integrated Waveguide Cavity Resonators for Complex Permitivity Characterizaion of Materials*, 2008.

[19] H. Esteban, J. M. Catala-Civera, S. Cogollos, V. E. Boria, *Characterization of Complex Permitivity Propieties of Materials in Rectangular Waveguides using Hybrid Iterative Method*, 2000.

[20] H. Kikuyama, *Fundamentals of Vector Signal Analysis*, 2010.

[21] Ballo D., *Network Analyzer Basics*, 1998.

[22] F. Caspers (CERN), *RF engineering basic concepts: S-parameters and the Smith chart*, 2007.

[23] Anritsu, *Application Note. Time Domain Measurements Using Vector Network Analyzers*, 2013.

[24] R. M. Guglielmi, *Using STM32 USART with ChibiOS Serial Driver*, 2019.

[25] H. F. Cancino de Greiff, *Circuitos de RF y las Comunicaciones Analógicas*, 2011.

[26] Surface mount process, *A guide to effective stencil design*, 2015.

[27] D. Lee, *Smith Chart Tuning*, 2013.

[28] G. Radha, Dr. G. S. N. Raju, *Transmission and Reflection Characteristics of Electromagnetic Energy in Biological Tissues*, 2013.

[29] Electronics Notes, *What is a Vector Network Analyzer, VNA: the basics*, 2010.

[30] HP Memory Project, *Network Analyzer - Scalar & Vector*, 2010.

[31] Keysight Technologies, *The Evolution of RF/Microwave Network Analyzers*, 2014.

[32] Giovanni Di Sirio, *ChibiOS/RT. Reference Manual*, 2021.

[33] Giovanni Di Sirio, *ChibiOS/HAL. Reference Manual*, 2021.

[34] J. M. Rodríguez et al., *Medidas de propiedades dieléctricas de materiales de construcción utilizando una guía rectangular*, 2008.

[35] K. Y. You, *Materials Characterization Using Microwave Waveguide Systems*, 2017.

[36] SDI-12 Support Group, *SDI-12. A Serial-Digital Interface Standard for Microprocessor-Based Sensors*, 2021.

[37] A. Saari et al., *Using SDI-12 with ST Microelectronics MCU's*, 2015.

[38] Juan C., *Estudio del efecto de la cristalización fría y del envejecimiento físico en las relajaciones de los polímeros mediante la técnica de corrientes estimuladas térmicamente*, 1999.

[39] Sergio R. et al., *Estudio de la dinámica de relajación en fase amorfa de un fármaco en alta presión*, 2018.

[40] Javier C. et al., *Caracterización de materiales para sensores de Radiofrecuencia*, 2015.

[41] Daniel I., José María R., *Estudio de las propiedades dieléctricas de materiales en Radiofrecuencia*, 2014.

[42] T. Takahashi, *NanoVNA*, 2020.

Agradecimientos

Quisiera agradecer, en primer lugar, a mis directores de proyecto Manuel Jiménez Buendía, Ana Toledo Moreo y Juan Domingo González Teruel, por la oportunidad que me han otorgado de poder realizar el presente Trabajo Fin de Grado en participación con uno de sus proyectos de doctorado, y por la confianza que me han transmitido en todo momento. El proyecto me ha permitido enfrentarme al campo de las Telecomunicaciones, así como profundizar en el campo de la electrónica y automática, de forma que me ha posibilitado ampliar mis conocimientos en ambas áreas de trabajo.

Además, quiero agradecer a mi familia su constante sacrificio para llegar hasta aquí, por esperar siempre algo grande de mí, por haberme ayudado y apoyado en todo momento, y por su apuesta incondicional por mi futuro.

También, quisiera agradecer a todos los profesores que he tenido a lo largo de mi paso por la universidad su esfuerzo por mostrarme y enseñarme todo lo que saben, y por facilitarme el camino durante mi aprendizaje. Esto es el resultado de su buen trabajo.

Por último, quiero agradecer a mis amigos, compañeros de carrera y personas que he conocido en la comunidad universitaria por los grandes momentos vividos durante estos cuatro años y por los recuerdos que nos unirán para siempre, aunque el tiempo a veces nos separe.