IMSE -cnm Instituto de Microelectrónica de Sevilla

TRABAJO DE FIN DE GRADO

DESARROLLO DE SETUP EXPERIMENTAL Y CONTROL AUTOMÁTICO DE INSTRUMENTOS PARA OPTIMIZAR ATAQUES DE CANAL LATERAL

Alejandro Casado Galán

Tutor: Antonio José Acosta Jiménez
Co-tutora: Erica Tena Sánchez

Gracias a Erica por todo lo que me ha enseñado

Gracias a Antonio y al IMSE por darme la oportunidad de trabajar con ellos

Gracias a mi familia y a mis amigos por creer en mí y estar ahí siempre

ÍNDICE

1. RESUMEN

Un ataque de canal lateral explota un observable físico proveniente de un dispositivo criptográfico con el fin de extraer sus secretos [2]. El objetivo de este proyecto es optimizar y automatizar los ataques de canal lateral en el ámbito de la lectura de trazas y la medida de emisión electromagnética de un circuito criptográfico objeto de estudio, en concreto un cifrador AES-128 operando en modo continuo de cifrado. Normalmente, a la hora de realizar un ataque de canal lateral, se suele medir el consumo de potencia o la emisión de radiación electromagnética del dispositivo electrónico con el fin de obtener la clave secreta de encriptación que usa. Si optamos por medir el campo electromagnético se pueden usar sondas de alta precisión (unos 0.2 mm) pero al posicionar la sonda a mano es complicado tener un método riguroso de encontrar siempre la zona de interés en la que queremos analizar la emisión electromagnética del dispositivo.

Debido a estas razones, en este trabajo se expone un método automatizado y que podemos usar en prácticamente cualquier dispositivo de hardware encriptado con la finalidad de identificar qué zonas de éste nos interesan para realizar un ataque electromagnético de canal lateral. Para ilustrar este método, llevaremos a cabo un trabajo práctico donde haremos un mapa cartográfico de emisión electromagnética de un cifrador AES implementado en una FPGA, que estará realizando operaciones de encriptado en bucle.

2. INTRODUCCIÓN

2.1. Criptografía, algoritmo de encriptación AES y ataques de canal lateral

La palabra criptografía proviene en un sentido etimológico del griego *Kriptos* (ocultar) y *Graphos* (escritura) y consiste en aplicar alguna técnica para hacer ininteligible un mensaje. Su finalidad principal es garantizar la confidencialidad (que la información sea accesible únicamente para los autorizados), integridad (que la información solo pueda ser modificada por los autorizados) y disponibilidad (que la información sea accesible para su consulta o modificación cuando se requiera) [3]. Aunque la criptografía pueda aplicarse a cualquier tipo de comunicación, nuestro trabajo está situado en el contexto de la criptografía informática.

En el contexto de la informática, un algoritmo de encriptación transforma una secuencia de bits, el mensaje que se desea encriptar, a otra distinta a partir de una clave secreta, el mensaje encriptado. Un buen algoritmo de encriptación opera de tal forma que imposibilita el acceso al mensaje original a partir del mensaje encriptado sin conocer la clave, al menos teóricamente. AES[1] es un algoritmo simétrico (utiliza la misma clave para encriptar que para desencriptar) por bloques de longitud finita e igual a la longitud de la clave, en nuestro caso trataremos con longitudes de 128 bits. Este algoritmo se caracteriza por aplicar reiteradamente funciones de permutación y sustitución sobre cada bloque de bits.

Hoy en día toda la informática y la electrónica se fundamenta en los circuitos integrados de tecnología CMOS[2]. Todas las operaciones que realiza un circuito de esta tecnología se basan en la implementación de puertas lógicas, construidas a partir de transistores n-MOS y p-MOS. La puerta lógica más simple es el inversor de corriente y durante su funcionamiento podemos distinguir entre 3 tipos de consumo de potencia. La corriente de fuga, la corriente que fluye desde la fuente de alimentación hasta la tierra cuando el bit cambia de 0 a 1 (o de 1 a 0) y la corriente usada para cargar y descargar las capacidades del circuito. Durante el cambio de estado de la salida de una puerta lógica CMOS se produce un pulso de corriente repentino que conlleva una variación del campo electromagnético cercano. Los ataques de canal lateral que explotan esta emisión electromagnética se denominan SEMA (*Simple Electromagnetic*

[1] Del inglés *Advanced Encryption Estandard* también conocido como *Rijndael*. Es un esquema de cifrado por bloques adoptado como un estándar de cifrado por el gobierno de los EEUU, creado en Bélgica. Se transformó en un estándar efectivo el 26 de mayo de 2002 y es uno de los algoritmos más populares usados en criptografía simétrica (*https://es.wikipedia.org/wiki/Advanced_Encryption_Standard*).
[2] Del inglés *Complementary Metal-Oxide Semiconductor*.

Analysis) o DEMA (*Differential Electromagnetic Analysis*). En un ataque SEMA se usa la información de una medida del campo electromagnético para la determinación de la clave y en un ataque DEMA se usan muchas medidas para determinar la clave a partir del análisis estadístico. En un análisis DEMA el atacante dispone de un modelo matemático hipotético del dispositivo para predecir los valores de radiación electromagnética y compararlos posteriormente con las medidas realizadas. Estudiando la correlación entre las medidas realizadas y las predichas por el modelo se puede determinar la clave del encriptador, dependiendo de la precisión del modelo que dependerá del conocimiento previo del atacante sobre el dispositivo [5].

2.2. Cartografía de emisión electromagnética

El método que se propone en este trabajo consiste en tomar un dispositivo de hardware encriptado, seleccionar una serie de puntos en este y realizar una medida de la emisión electromagnética en cada uno de ellos. Después se procesarán los datos tomados sobre el campo electromagnético radiado y se realizará un mapa cartográfico del dispositivo marcando las zonas de mayor interés. Así, alguien que desee hacer un ataque de canal lateral de análisis electromagnético al dispositivo tendrá señaladas a su disposición las zonas donde le conviene medir, reduciendo el trabajo y el tiempo requeridos.

El dispositivo de hardware que utilizaremos para poner en práctica este método será una FPGA[3], una *Nexys A7* de la compañía *Digilent*. La programaremos mediante el software *ISE* de *Xilinx* para que realice encriptaciones en bucle según el estándar de encriptación AES. Posteriormente realizaremos medidas del campo electromagnético emitido por el dispositivo en una cuadrícula de puntos que representaremos en un mapa cartográfico coordenado en función de la intensidad de campo radiada. El aparato que nos permitirá realizar este mapa cartográfico y gracias al cual podremos tener un método riguroso, automatizado y preciso, será la mesa XY. Este dispositivo nos permitirá mover la FPGA en un plano de forma segura y con una precisión muy alta (156 nm) y es prácticamente la motivación principal de este proyecto.

Este experimento se llevó a cabo en el laboratorio de ciberseguridad del *Instituto de Microelectrónica de Sevilla* (IMSE).

[3] FPGA viene del acrónimo inglés *Field Programmable Gate Arrays* y es un dispositivo de hardware programable.

3. INSTRUMENTACIÓN

3.1. Configuración del setup experimental

Los componentes del experimento que vamos a realizar en el laboratorio son los siguientes:

- Mesa XY
- Sonda de medida de campo magnético y brazo articulado
- Osciloscopio
- FPGA y software para su programación
- Ordenador y software de control y procesado de datos

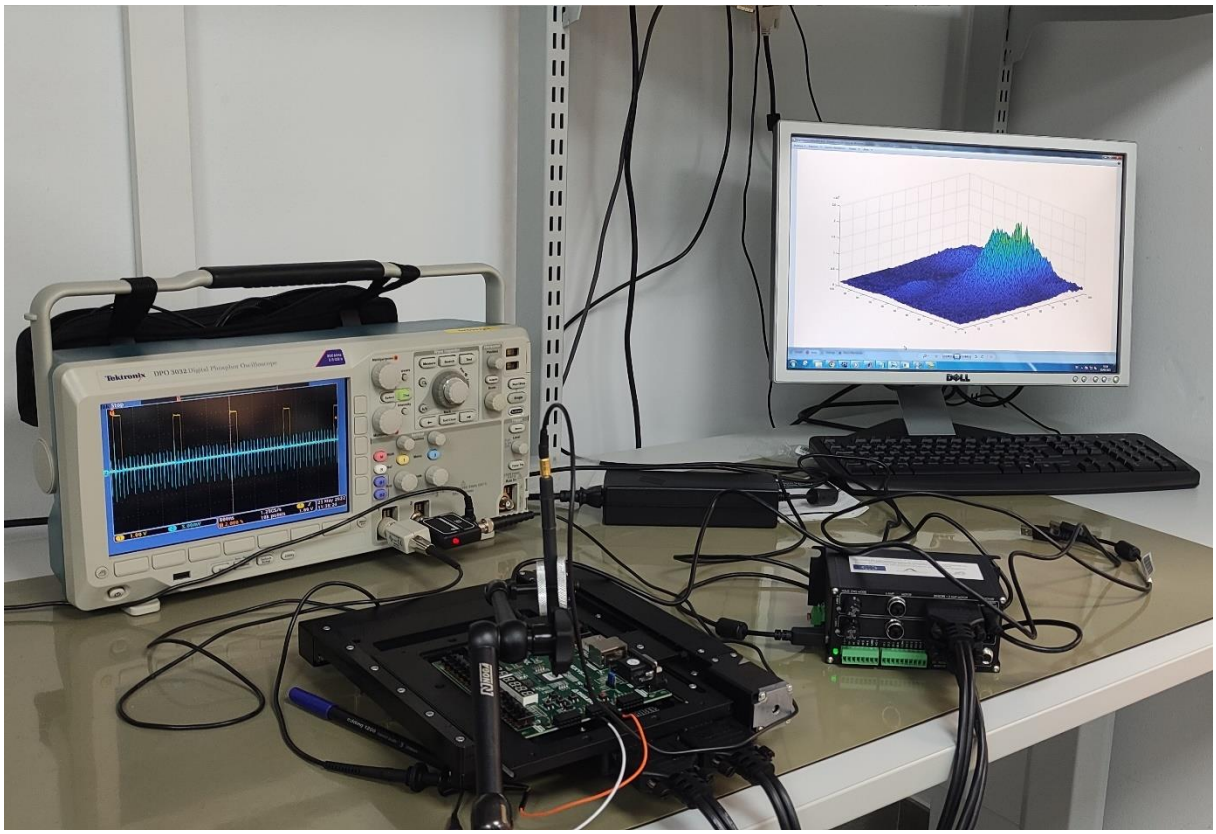

Figura 1. Setup experimental.

Cada uno de los componentes nombrados anteriormente juega su papel fundamental en el proyecto. La mesa XY nos permitirá realizar desplazamientos en el plano a la FPGA muy precisos para tener un mapa cartográfico detallado, la sonda de medida de campo magnético será el aparato que usaremos para medir las emisiones, el osciloscopio nos servirá para visualizar la medida de la sonda en un intervalo de tiempo que podremos seleccionar, la FPGA será el dispositivo de hardware del cual escanearemos su emisión electromagnética al estar realizando encriptaciones en bucle y, finalmente, usaremos un ordenador con el software

necesario (*MATLAB* y la suite *ISE* de *Xilinx*) que nos servirá de nexo para el control de todos los instrumentos simultáneamente y para procesar los datos que tomemos.

3.2. Mesa XY (*Zaber ASR100B120B*)

Este dispositivo será el más importante en cuanto a la automatización de los ataques. Es el que nos permite realizar desplazamientos en el plano en dos dimensiones y con una precisión relativamente alta (156 nm). Será donde situemos la placa base de la FPGA para moverla y escanear todos sus puntos.

El dispositivo está compuesto por dos partes principales: Una, que es la mesa XY como tal, compuesta por dos plataformas y dos servomotores que facilitan el movimiento en las dos direcciones del plano, y la otra, que es el controlador que permite la comunicación entre el ordenador y la propia mesa XY. También dispondremos de una fuente de alimentación para suministrar potencia al aparato y cables para establecer las conexiones controlador-mesa y PC-controlador.

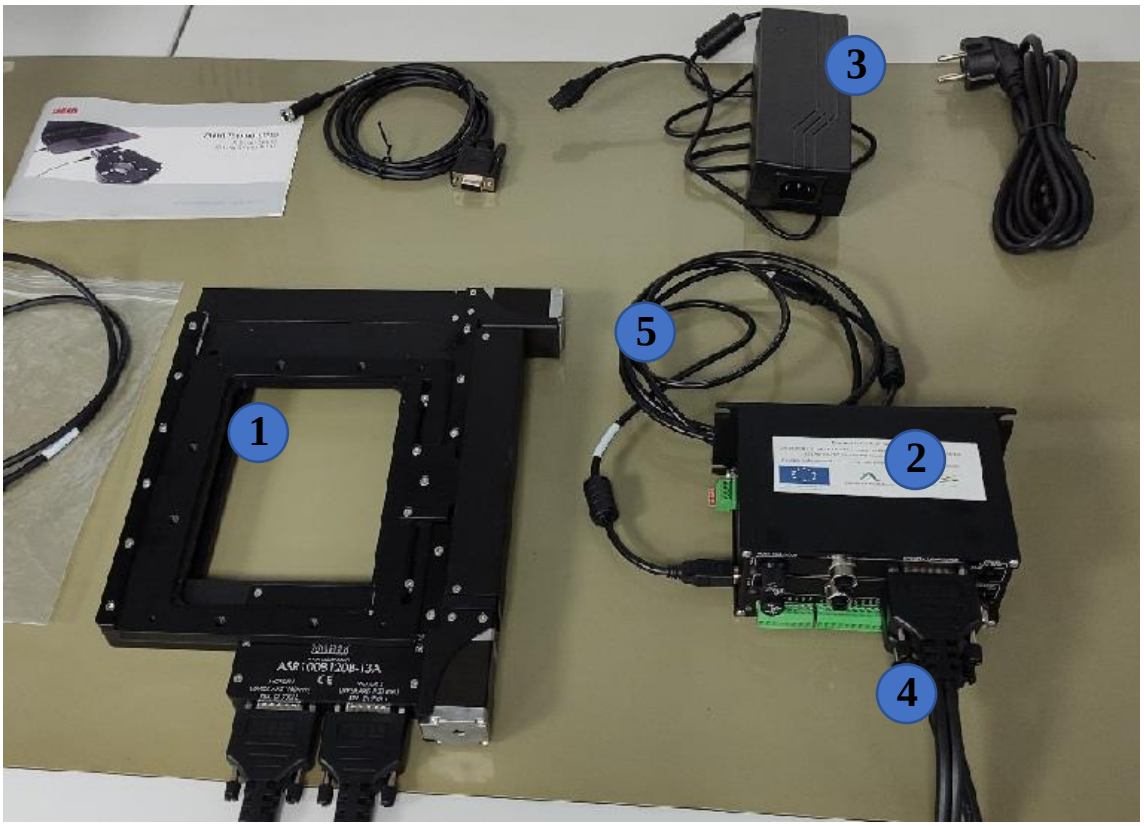

Figura 2. Componentes de la Mesa XY. 1 – Mesa XY. 2 – Controlador. 3 – Fuente de alimentación y cable. 4 – Cables MC10T3. 5 – Cable USB

El controlador consta de varias entradas y salidas. Las más importantes y las que utilizaremos son: las salidas de los cables *MC10T3*, que son los que transmiten la señal a los motores de la mesa; dos entradas de alimentación, aunque solo necesita corriente por una de ellas, y una entrada/salida USB que va conectada al PC. También consta de dos ruedas que controlan los motores de la mesa.

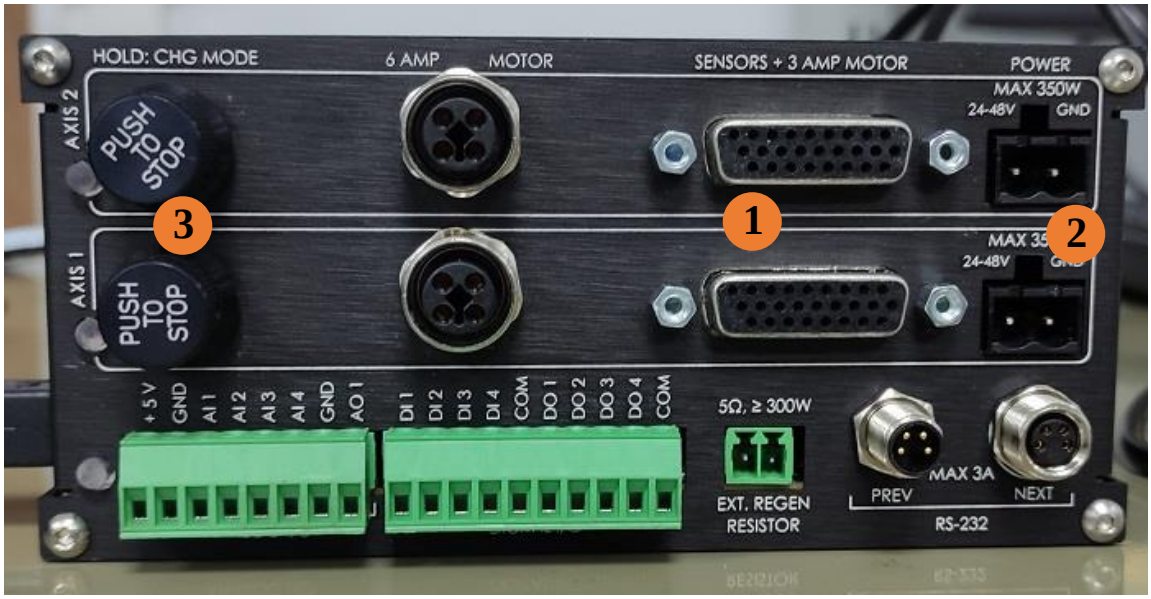

Figura 3. Vista frontal del controlador de la mesa XY. A) Salida de los cables MC10T3. B) Entradas del cable de alimentación. C) Ruedas de control manual de los motores.

La salida del cable USB para la conexión con el ordenador está a uno de los lados del controlador (por eso no se aprecia en la *Figura 3*). Como se puede ver en la *Figura 3*, los componentes están duplicados. Esto se debe a que cada una de las salidas va dirigida a uno de los dos motores de la mesa XY. No entraremos en detalle con el resto de las entradas/salidas que aparecen en la *Figura 3* puesto que con las descritas nos basta para para realizar nuestro objetivo.

Las ruedas de control (*Figura 3, C*) nos proporcionan el método más directo para controlar los motores de la mesa. Tan solo necesitamos conectar los cables de comunicación y alimentación y al girarlas, en un sentido u otro, haremos que los motores se muevan a una velocidad constante ,en un sentido u otro, en función de cuanto las giremos. La cantidad que podemos girar las ruedas está discretizada en pasos de ángulo constante a modo de selector, cada una de las posiciones de la rueda aplica una velocidad distinta en cada motor. A pesar de que los barridos

son automáticos, estas ruedas nos serán de gran utilidad a la hora de colocar la mesa una determinada posición inicial.

3.3. Sonda de medida de campo electromagnético (*Rohde&Schwarz HZ-15*)

Disponemos de un kit de sondas diseñado para tomar medidas de radiación de campo cercano. Tenemos varias de ellas, algunas miden campo eléctrico, otras campo magnético, y las formas y tamaños varían en cada una de ellas (ver *Figura 4*).

Figura 4. Kit de sondas de electromagnéticas.

Podemos notar en la *Figura 4* que falta una de las sondas, y es la que aparece en la *Figura 5*, que será la que utilizaremos para medir la emisión de la FPGA. Empezamos haciendo medidas con la sonda de campo eléctrico de menor tamaño (en la *Figura 4* la cuarta en orden descendente). Tiene una precisión de 0.2 mm, ideal para hacer medidas en la FPGA que es un cuadrado de 15x15 mm^2). Sin embargo el campo eléctrico que radiaba la FPGA cuando estaba encriptando en bucle era muy débil (incluso usando la escala más pequeña del osciloscopio era complicado diferenciar el ruido de las medidas útiles) por ello la elección fue usar una sonda de campo magnético de igual tamaño y precisión (ver *Figura 5*).

Figura 5. Sonda de medida de campo magnético cercano.

Esta sonda tiene forma de lápiz, siendo la punta la zona sensible al campo magnético. El tamaño de la zona sensible es prácticamente igual al de la sonda de campo eléctrico con la que hicimos las medidas preliminares, con lo cual podemos suponer que la precisión será también del mismo orden. En la punta tiene una línea blanca que nos indica en qué dirección tiene que ir la intensidad de corriente eléctrica para que la medida de campo magnético sea la más eficaz. Si nos fijamos en la *Figura 6*, en el dibujo central, que es la sonda que vamos a utilizar, podemos ver una representación gráfica de cuál es la componente del campo magnético que vamos a medir con esta sonda. Ahí se indica que la medida es proporcional a la intensidad de corriente y podemos ver como la marca blanca de la que hablábamos está orientada en el sentido de su circulación. A partir de las ecuaciones de Maxwell podemos deducir el campo magnético que genera una intensidad de corriente que circula por un cable recto en dirección azimutal (1).

$$B = \frac{\mu_0 I}{2\pi r} \tag{1}$$

Donde μ_0 es la permeabilidad magnética del vacío, I la intensidad de corriente que circula por el cable y r la distancia a la que estaremos midiendo. Podemos ver que efectivamente el campo magnético medido es directamente proporcional a la intensidad y además es inversamente proporcional a la distancia al cable.

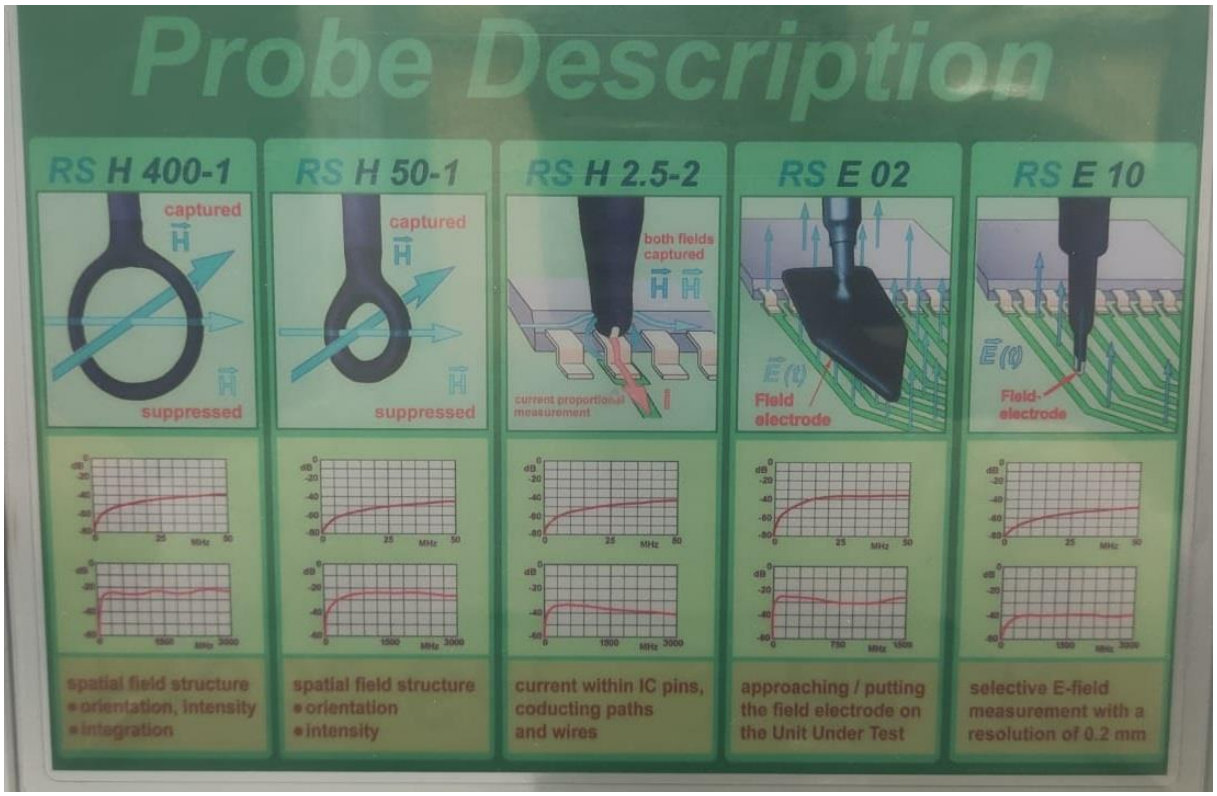

Figura 6. Descripción de las sondas del kit.

Para sujetar la sonda y que se quedase fija en una posición determinada utilizamos un brazo articulado (*Figura 7*) sujeto a la mesa donde estaba todo el setup. Esto nos permitía acercar la sonda a una distancia muy cercana a la FPGA de forma segura y tener así una medida del campo magnético con una intensidad suficiente para diferenciarla del ruido de fondo.

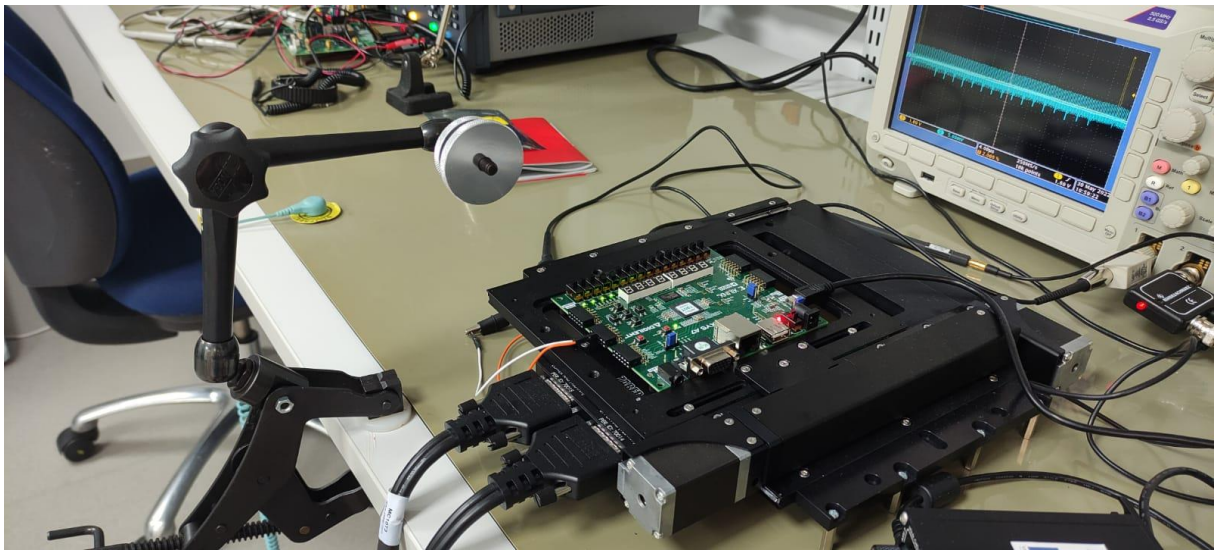

Figura 7. A la izquierda, brazo articulado; en el centro, mesa XY con la FPGA; a la derecha, osciloscopio con la medida de una traza.

Para tener una mejor medida usaremos también un preamplificador para la sonda de 20 dB (*Rohde&Schwarz HZ-16*) que irá conectado en serie entre la sonda y el osciloscopio. De esta forma la señal que obtendremos será lo suficientemente grande como para ser legible en el osciloscopio y poder así procesarla debidamente.

Figura 8. Amplificador para la sonda de 20 dB.

El amplificador tiene una entrada para la sonda (es el conector de la parte de abajo en la *Figura 8*), un conector para la entrada de uno de los canales del osciloscopio (conector de la izquierda) y una entrada de alimentación de 12 V para realizar la amplificación.

3.4. Osciloscopio (*Tektronix DPO 3032*)

A la hora de realizar las medidas haremos uso de un osciloscopio (*Tektronix DPO 3032*) para recopilar las trazas[4] de emisión electromagnética de la FPGA (ver *Figura 9*).

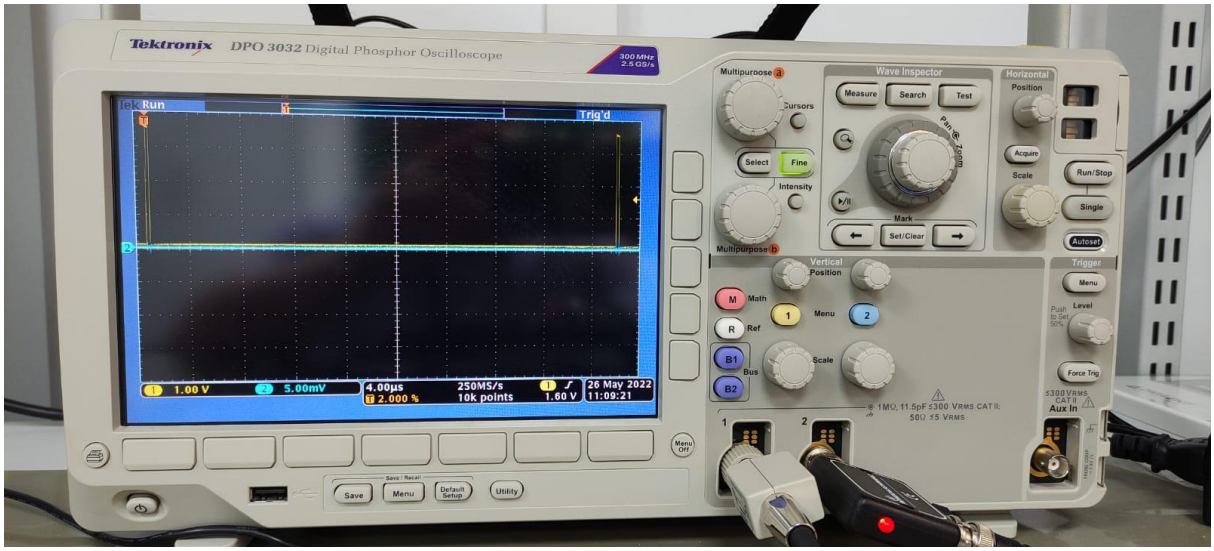

Figura 9. Osciloscopio.

Este dispositivo nos permite visualizar la medida que hace la sonda de campo electromagnético en función del tiempo. Estas trazas de emisión serán las que analizaremos para determinar si la zona de la FPGA en la que estamos midiendo es la que nos interesa para hacer un ataque electromagnético, esto es, la zona donde se está realizando la encriptación.

Un osciloscopio es el dispositivo más indicado para esta tarea debido a diversas razones. Para empezar nos permite medir la intensidad del campo electromagnético no en un solo instante de tiempo, sino en un intervalo que podemos elegir. Esto es muy útil ya que nos conviene ver la forma de la traza electromagnética en distintos instantes para diferenciar las que nos interesan de las que no. A parte de esto también podemos escoger una señal externa para el disparo, esto es, el momento en el que se empieza a capturar la señal en el osciloscopio, como, por ejemplo, el reloj interno de la FPGA, aunque ya veremos más adelante que nos conviene más usar otras señales.

Este osciloscopio tiene un total de dos entradas (dos canales) de cable coaxial. Nos es suficiente con dos entradas, ya que una de ellas será la de la sonda (con el amplificador) y la otra la

[4] Dentro del contexto de los ataques de canal lateral, una traza es el elemento de información del observable físico que queremos explotar.

usaremos para el disparo. En el panel de control tenemos muchas opciones de configuración. Por ejemplo, podemos cambiar la escala de tiempos, desde el orden de los segundos hasta el orden de las decenas de nanosegundos. También tenemos opción a elegir la magnitud de la escala vertical, desde el orden las decenas de voltios por división hasta el orden de los milivoltios por división. Asimismo podemos modificar el nivel y el tipo de disparo y a que canal queremos asociarlo. Igualmente podremos elegir la resolución de la medida, o sea, cuantos puntos del intervalo de tiempo queremos tomar, y el tipo de adquisición (simple, promediada, detección de picos, etc.). A parte de poder modificar todos estos parámetros a mano desde el panel de control del osciloscopio, también podremos hacerlo desde el ordenador cuando establezcamos una comunicación entre el dispositivo y el PC a través de un cable USB.

3.5. FPGA (*Nexys A7*) y software de programación (*ISE*)

Una FPGA es un dispositivo de hardware programable. Este dispositivo está formado por una matriz de bloques lógicos configurables interconectados entre sí mediante una red de conexiones que es completamente reprogramable (ver *Figura 10*) [6]. No tiene tan buen rendimiento como un ASIC[5], pero el hecho de que se pueda reprogramar para realizar distintas tareas lo hace muy útil.

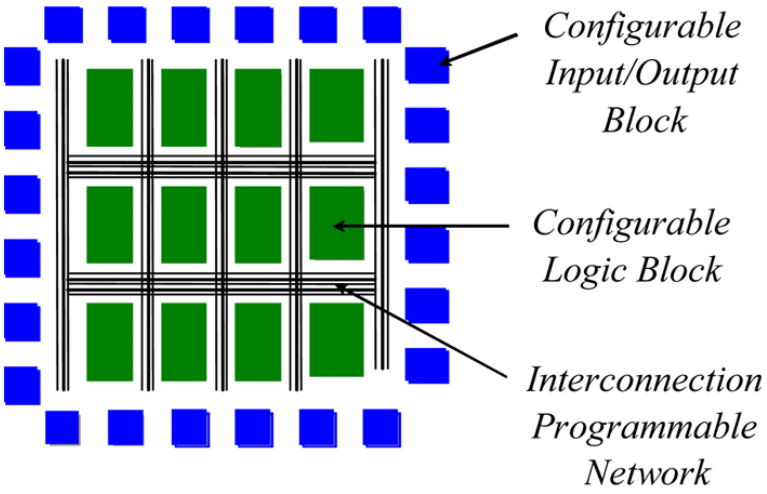

Figura 10. Esquema de una FPGA, extraído de [6].

[5] Del inglés *Application Specific Integrated Circuit*, es un circuito integrado diseñado para el cumplimiento de una función específica con el mayor rendimiento posible.

Para este proyecto usaremos una FPGA modelo *Nexys A7* del fabricante *Digilent* y para programarla haremos uso del programa *ISE* de *Xilinx*. Las características completas de la FPGA están en su manual [8].

Como podemos ver en la *Figura 11*, la placa base de la FPGA tiene una gran variedad de componentes. El principal y el que más nos interesa es la FPGA en sí, en la *Figura 11* es el chip cuadrado negro con una pegatina blanca que se sitúa justo en el centro de la placa. En la parte noroeste podemos ver una luz roja y cerca un cable USB conectado. La luz roja nos indica que llega potencia al sistema y el cable es el que la suministra, a parte de ser el canal por el que se transmiten los datos a la hora de programar la FPGA. En la parte sur tenemos una serie de interruptores y leds de color verde, ambos son programables para que realicen distintas funciones y hablaremos de ellos más adelante.

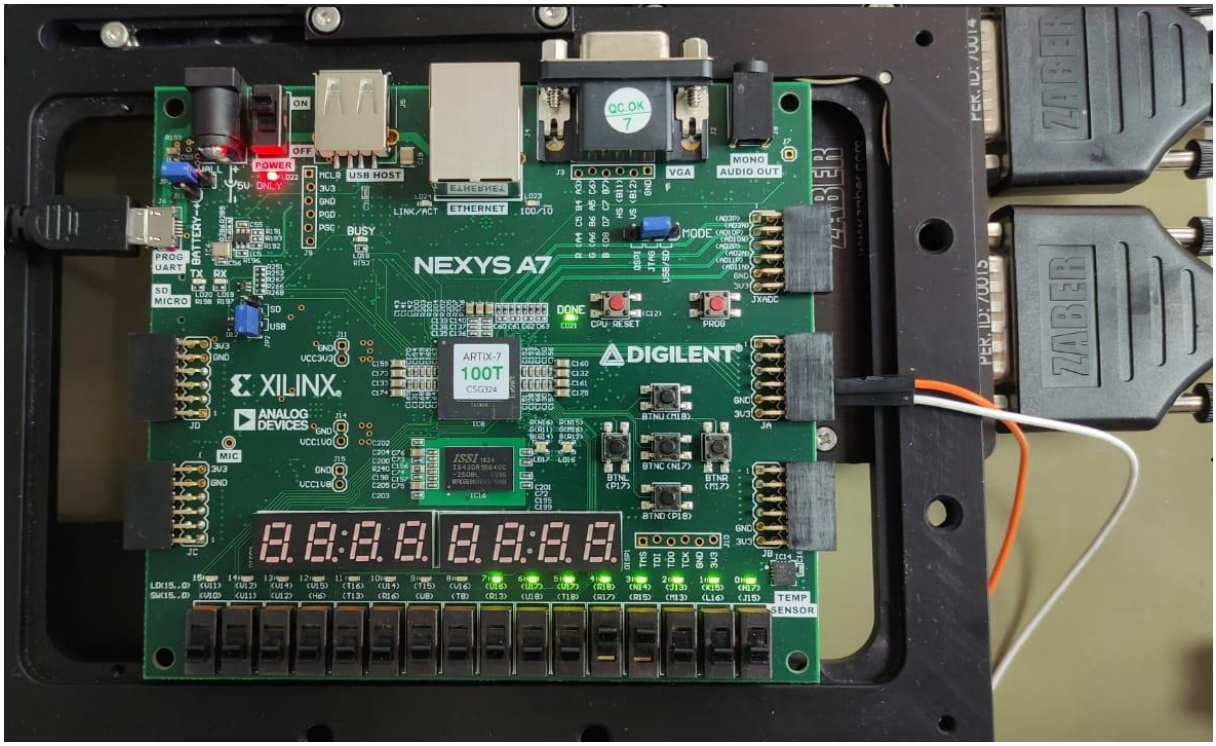

Figura 11. La Nexys A7 montada en la plataforma de la mesa XY.

La propia placa dispone de un oscilador de 100 MHz que usa de reloj interno. También dispone de una memoria flash en la que podemos guardar la información de como programar la FPGA para que no se pierda después de quitar la corriente del sistema. No obstante como el encriptador en bucle que vamos a usar no es un programa excesivamente complejo y además trabajaremos siempre con el ordenador del laboratorio cerca no es necesario usarla. En la *Figura 11* podemos

ver como salen dos cables (naranja y blanco) de la FPGA, son los que llevan la señal del disparo al osciloscopio. Para ello usaremos los conectores *Pmod*, que son unas salidas que tiene la placa de la FPGA y que podemos programar. En la *Figura 12* aparece un esquema y la dirección de los pines de cada *Pmod* que usaremos en el programa *ISE* para transmitir la señal que escojamos.

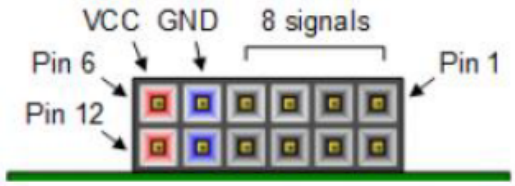

Figure 10.1 Pmod Connectors; Front View, as Loaded on PCB

Pmod JA	Pmod JB	Pmod JC	Pmod JD	Pmod XDAC
JA1: C17	JB1: D14	JC1: K1	JD1: H4	JXADC1: A13 (AD3P)
JA2: D18	JB2: F16	JC2: F6	JD2: H1	JXADC2: A15 (AD10P)
JA3: E18	JB3: G16	JC3: J2	JD3: G1	JXADC3: B16 (AD2P)
JA4: G17	JB4: H14	JC4: G6	JD4: G3	JXADC4: B18 (AD11P)
JA7: D17	JB7: E16	JC7: E7	JD7: H2	JXADC7: A14 (AD3N)
JA8: E17	JB8: F13	JC8: J3	JD8: G4	JXADC8: A16 (AD10N)
JA9: F18	JB9: G13	JC9: J4	JD9: G2	JXADC9: B17 (AD2N)

Figura 12. Conectores Pmod de la FPGA, tabla de pines. Extraído de [8].

El programa *ISE* nos permite diseñar configuraciones de FPGAs mediante lenguajes de descripción de hardware (*Verilog* o *VHDL*). Estos lenguajes simplifican mucho la tarea ya que permiten describir un circuito integrado con la sintaxis de un lenguaje de programación sin tener que entrar en detalles de bajo nivel. Es el propio programa el que, después de haber escrito nuestro código, configura la conexión entre bloques lógicos programables, entradas y salidas de la FPGA que vamos a programar[6]. Para ello se genera un archivo de extensión *.bit* que puede ser descargado e implementado en la FPGA. Además de esto, también nos permite realizar simulaciones de nuestro diseño antes de implementarlo en un circuito real.

En este entorno programaremos el encriptador AES de tal forma que esté realizando operaciones de encriptación en bucle. Partiremos de un texto inicial que será el primero que encripte, a partir de ahí, el texto encriptado lo enviaremos de vuelta a encriptar reiterando el proceso indefinidamente (ver *Figura 13*). No profundizaremos sobre la forma explícita del

[6] A este proceso se le conoce como Place & Route.

código del encriptador, ya que el programa es relativamente complejo y no es el objeto de estudio de este trabajo.

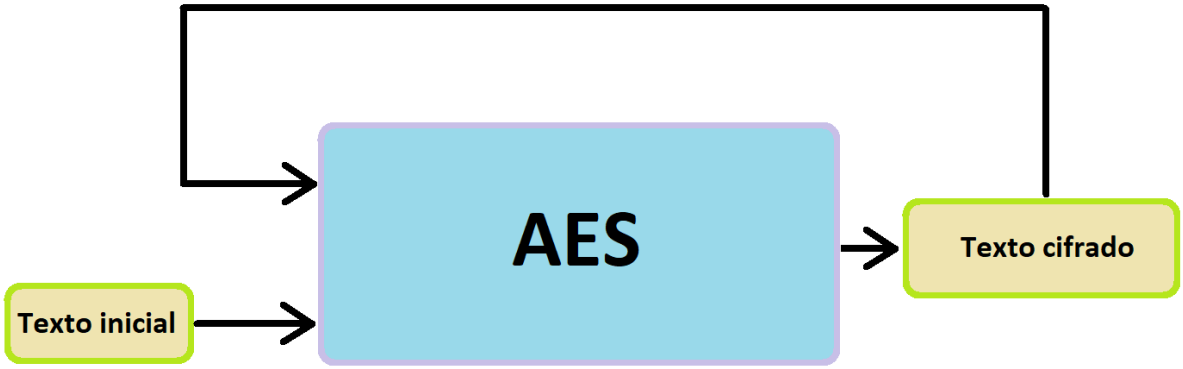

Figura 13. Encriptación en bucle.

Dentro de un proyecto de ISE hay un archivo fundamental de extensión *.ucf* cuya finalidad es indicar al programa a qué puertos de la FPGA queremos asignarle las respectivas entradas y salidas que utilizaremos. Como no es muy complejo hablaremos un poco de su estructura. Como podemos ver en la *Figura 14* tenemos varias líneas de código donde se hacen referencia a salidas y entradas con el nombre entre comillas. Por ejemplo, en la línea 17 vemos una referencia a la variable *"clk"*, está será la entrada corresponde al oscilador interno de 100 MHz de la *Nexys A7*, o sea, el reloj. Justo en las líneas adyacentes arriba y abajo (líneas 16 y 18) tenemos dos entradas asociadas cada una a uno de los interruptores de la placa de la FPGA. La entrada *"reset"* funciona como un interruptor de encendido y apagado, si está activada el programa funciona, si no lo está, no lo hace. La entrada *"tx_ready"* da paso a que se generen encriptaciones en bucle indefinidamente, si está activada el algoritmo estará encriptando texto continuamente, si está desactivada dejará de hacerlo. Deberán estar ambas activas siempre.

```
16  NET "tx_ready" LOC = R15 | IOSTANDARD=LVCMOS33;
17  NET "clk" LOC = E3 | IOSTANDARD=LVCMOS33;
18  NET "reset" LOC = R17 | IOSTANDARD=LVCMOS33;        //
19
20  // CN3 HEADER
21  NET "start_tx" LOC = C17 | IOSTANDARD=LVCMOS33;
22  NET "clk_out" LOC = D18 | IOSTANDARD=LVCMOS33;
23  NET "tx_ready_out" LOC = E18 | IOSTANDARD=LVCMOS33;
24  NET "out_start" LOC = G17 | IOSTANDARD=LVCMOS33;
25
```

Figura 14. Captura del archivo .ucf del proyecto de ISE.

Después, desde la línea 21 hasta la 24 tenemos declaradas distintas salidas que podremos usar como disparo para el osciloscopio. La salida *"start_tx"* se activa cada vez que un nuevo byte va a ser encriptado y se activa varias veces por cada ciclo de encriptación. Las salidas *"clk_out"* y *"tx_ready_out"* son las mismas señales que el reloj (*"clk"*) y *"tx_ready"* de las cuales hablamos antes. Por último, la señal de la línea 24, *"out_start"*, se activa cada vez que empieza un nuevo ciclo de encriptación, esta será la señal que usaremos para el disparo del osciloscopio.

Si nos fijamos en la *Figura 14*, podemos ver que a la derecha de cada asignación de las variables tenemos un parámetro, *LOC*, al cual se le da un valor. Esto indica al programa a que pin de la FPGA queremos asociar la salida o entrada en cuestión. Si nos fijamos en los nombres de este parámetro en las salidas (líneas 21-24) y lo comparamos con la primera columna de la tabla de la *Figura 12*, podemos ver, que cada salida está asociada a uno de los pines del conector *Pmod JA*. En el caso de las entradas (líneas 16-18), *R15* y *R17* corresponden a interruptores de la placa y *E3* es la localización del reloj interno de la *Nexys A7*. Todos estos valores y más vienen tabulados en el manual [8].

3.6. Software de control y procesado de datos

3.6.1. Control de la mesa XY

Este software diseñado por los fabricantes de la mesa nos permite acceder a los dispositivos de esta marca y controlarlos desde el ordenador de una manera más sofisticada que con el control manual. En nuestro caso lo usaremos para controlar la mesa XY, aunque será únicamente con el fin de aprender más sobre su funcionamiento, ya que como veremos más adelante el control mediante MATLAB es el que más nos interesa.

Como podemos ver en la *Figura 15* tenemos una ventana principal donde aparecen los dispositivos que están conectados al PC (en nuestro caso solo vamos a conectar un dispositivo, la mesa XY). El dispositivo *01* es el controlador de la mesa (modelo *X-MCC2*) que, a su vez, nos da acceso a los dos motores que controlan cada uno de los movimientos en el plano XY, *01 axis 1* y *01 axis 2*. El primero es el que controla el eje horizontal y el segundo el que controla el eje vertical.

En la columna *Position* podemos ver un valor numérico asociado al desplazamiento de cada eje en pasos del motor. Sus valores posibles son números enteros de 0 a 640000 en el eje horizontal

y de 0 a 768000 en el eje vertical. Teniendo en cuenta que el eje horizontal se desplaza hasta 100mm y el vertical hasta 120 mm podemos ver la gran precisión que tiene este dispositivo, ya que puede realizar desplazamientos tan pequeños como 156 nm en ambas direcciones.

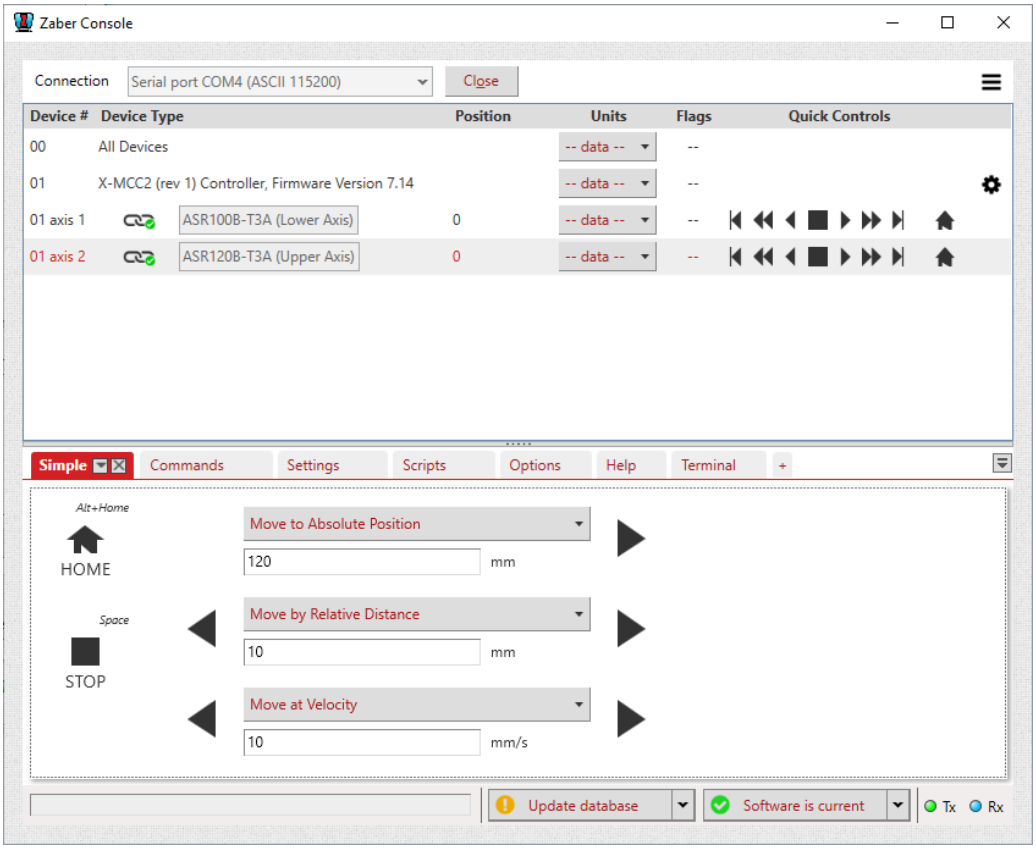

Figura 15. Interfaz de la consola ZABER.

Como podemos ver (*Figura 15*), en la mitad inferior de la consola hay pestañas que nos dan acceso a distintas herramientas. En la figura está seleccionada la pestaña *Simple* que, como su nombre indica, es una interfaz sencilla para realizar operaciones simples. Concretamente nos permite mover las plataformas a una posición absoluta del eje, moverlas una distancia relativa a la posición actual y moverlas a una velocidad constante. También nos permite pararlos (*STOP*) y activar el comando *Home*. En el siguiente párrafo se explican con más detalle estas operaciones básicas.

El comando *Home* es uno de los más importantes, ya que nos permite resetear el aparato a un estado inicial y es fundamental hacerlo al principio de una serie de operaciones ya que le permite al controlador conocer la posición de los ejes. A priori, cuando nosotros encendemos la mesa, los ejes pueden estar en cualquier posición desconocida debido a usos pasados que los hayan dejado en cualquier sitio. El dispositivo carece de sensores que le permitan saber en qué

posición absoluta está cada eje, la única posición que puede detectar es el cero, que correspondería al punto donde el desplazamiento en el sentido negativo ya no es posible porque llegamos al límite de la longitud del eje. Precisamente el comando "Home" busca esta posición para "hacerle saber" al controlador en que posición está el cero. Cuando activamos este comando ambas plataformas se desplazan en sentido negativo de los ejes (el sentido, positivo o negativo, viene determinado por el fabricante) y cuando no pueden desplazarse más se paran y el controlador toma la posición como el cero del eje (una forma de visualizarlo es pensar en dos ejes coordenados en los que las unidades van de 0 a 640000 o 768000 en función de si se trata del eje x o y).

A partir de aquí, cada desplazamiento que ocurra en la mesa, ya sea manual con las ruedas de control, o desde el ordenador, queda registrado y el controlador sabrá en que posición se encuentra la mesa hasta que desconectemos la fuente de alimentación. Desde estas condiciones podemos realizar desplazamientos absolutos, que serían aquellos que llevan a la plataforma a una determinada posición del eje (recordemos, de 0 a 640000 o de 0 a 768000 en función de cuál sea) o desplazamientos relativos respecto a la posición actual (aunque estos también podíamos hacerlos antes sin realizar el comando *Home*, pero es buena praxis activarlo siempre en el comienzo de cada sesión para saber dónde están los límites de cada eje). Otra opción es realizar un desplazamiento a velocidad constante de las plataformas en cualquier dirección o sentido.

Por último, hablemos de las unidades en las que trabaja el dispositivo. La unidad natural de la mesa es aquella en la que están divididos los ejes (una unidad corresponde a un paso del servomotor). El horizontal va de 0 a 640000 unidades y el vertical de 0 a 768000 unidades. No obstante, para facilitar el trabajo con unidades más familiares, los fabricantes nos dan opción de introducir las distancias (o velocidades) en el sistema internacional: metros, centímetros, milímetros, micras o nanómetros (o m/s, cm/s, mm/s, μm/s, nm/s) o en pulgadas (o pulgadas/s).

Veamos que herramientas nos ofrece la siguiente pestaña, *Commands*. Como vemos en la *Figura 16* tenemos a nuestra disposición una lista con comandos que podemos enviar a la mesa, algunos de ellos tienen un espacio para escribir un argumento. En la figura solo vemos una parte de ellos, la lista sigue pero se muestran los más importantes (y los más utilizados). Tenemos el comando "home" del que ya hablamos anteriormente, *move abs* que realiza un desplazamiento absoluto en el eje que seleccionemos (ver *Figura 15* en la lista de dispositivos), *move rel* que

realiza un desplazamiento relativo a la posición actual y *move vel* que ordena el movimiento a una velocidad constante. Todos estos comandos son los mismos que explicamos antes, solo que ahora están en una forma más afín a un programador. Exceptuando *home* todos tienen un argumento en los que introducimos un valor en las unidades que seleccionemos (corresponde al desplazamiento que realizará en esas unidades o el movimiento a velocidad constante).

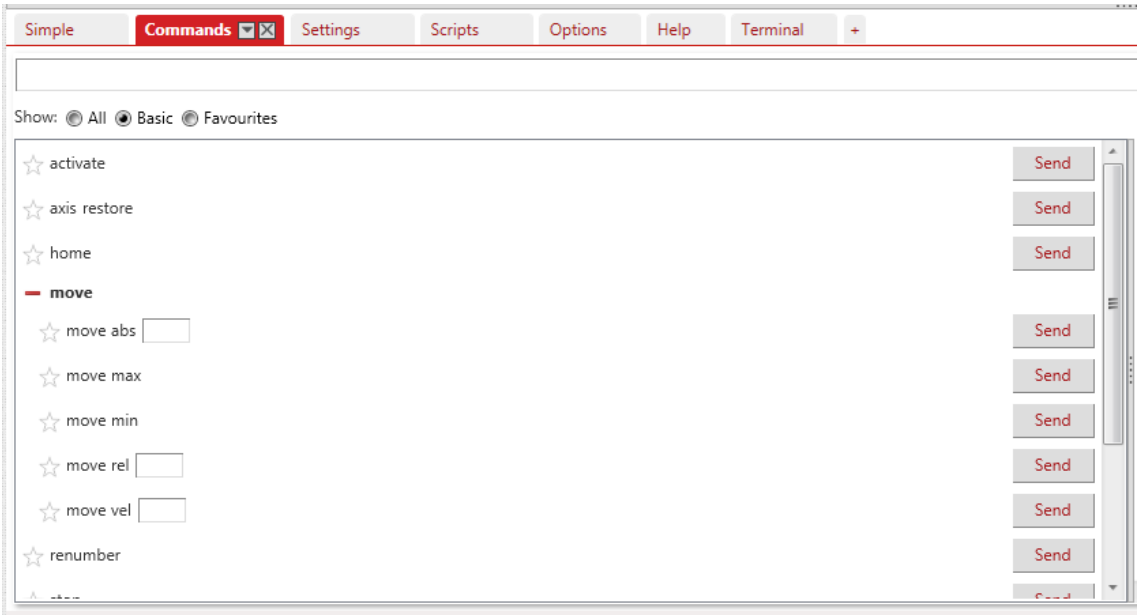

Figura 16. Interfaz de la consola ZABER – Commands.

En la siguiente pestaña, *Settings* (ver *Figura 17*), tenemos una interfaz parecida a la pestaña de comandos, pero hay una diferencia. Mientras que los comandos son instrucciones que damos a la mesa para que realice una operación, en los ajustes estas instrucciones lo que hacen son configurar los parámetros de la mesa para su uso posterior. También podemos solicitar datos sobre el estado actual de la mesa (*Write* envía un dato, *Read* lo solicita). Por ejemplo, la opción *maxspeed* nos permite imponer un límite a la velocidad a la que queremos que se mueva un determinado eje, o conocer que velocidad máxima está configurada en el momento. La opción *pos* nos permite conocer en qué posición está un eje o cambiarla (aunque esto no resultaría en un movimiento, solo cambiaríamos el dato que tiene registrado el controlador sobre la posición de un eje, lo cual en la mayoría de los casos no es recomendable por lo que explicamos sobre el comando *home*).

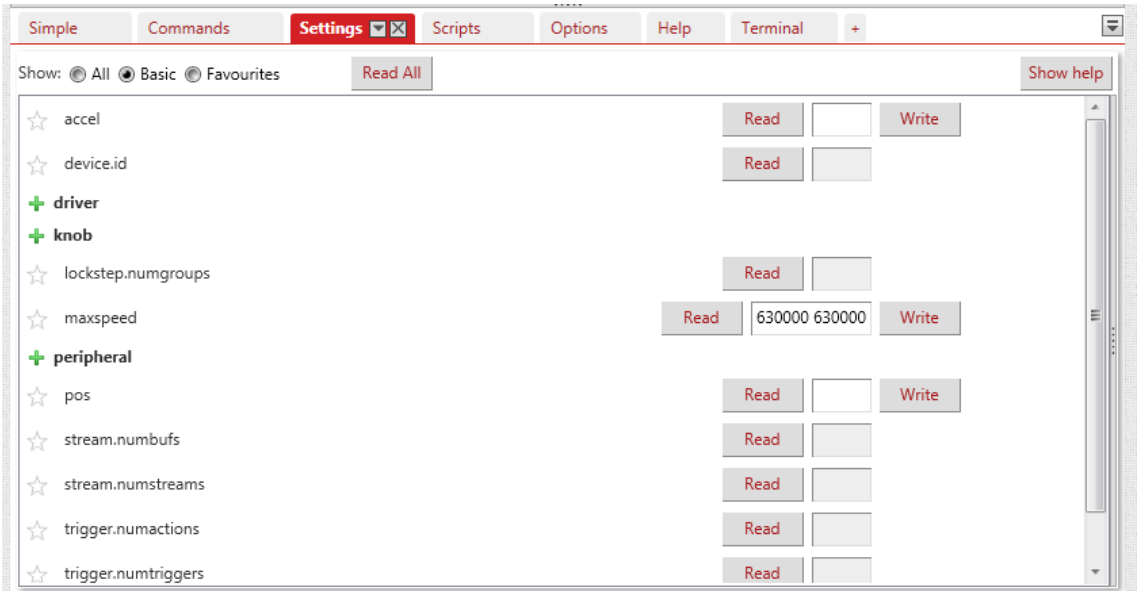

Figura 17. Interfaz de la consola ZABER – Settings.

La siguiente pestaña se llama *Scripts* (*Figura 18*) y nos da acceso a una herramienta para escribir programas en *C#*, *Python* y *JavaScript*, a parte de algunos scripts predeterminados que vienen con la consola a modo de ejemplo. No entraremos en detalle sobre la forma y la sintaxis del código de los scripts ya que no vamos a utilizar *C#*, *Python* o *JavaScript* como lenguajes de programación, utilizaremos MATLAB. Estos scripts utilizan los comandos y ajustes que vimos en las pestañas de *Commands* y *Settings* integrados en un solo programa para realizar una función determinada más compleja, aunque los que vienen de ejemplo realizan instrucciones muy simples.

Si nos fijamos en los nombres de algunos scripts podemos observar que algunos llevan la palabra “ASCII” o “Binary” entre paréntesis. Esto se debe a que podemos comunicarnos con el controlador de la mesa en protocolo ASCII o binario. Hasta entonces hemos utilizado ASCII y lo seguiremos haciendo ya que es el más cómodo para las tareas que vamos a desarrollar en el proyecto.

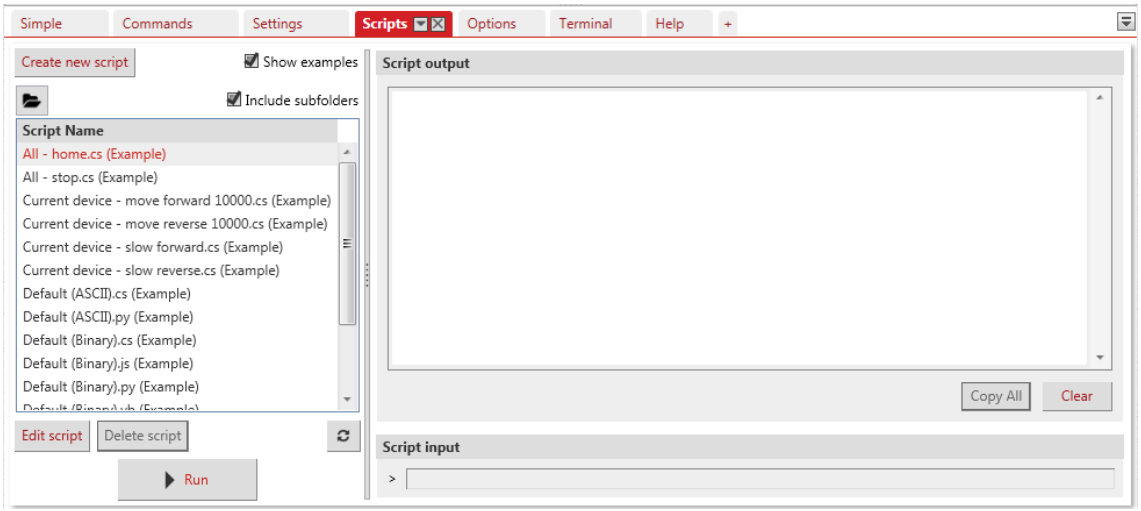

Figura 18. Interfaz de la consola ZABER – Scripts.

Terminemos con la pestaña *Terminal* (*Figura 19*). Se trata de una consola que registra todo lo que hace el dispositivo, ya sea un movimiento, un ajuste de un parámetro o la solicitud de un dato. También tenemos una entrada de texto en la que podemos escribir los comandos directamente para que los ejecute. En la *Figura 19* podemos ver una serie de líneas correspondientes a la inicialización del dispositivo cuando lo conectamos al ordenador. El carácter "/" nos indica que se ha enviado una instrucción a la mesa y el carácter "@" indica lo que el dispositivo nos comunica a nosotros. El "01" hace referencia al controlador, que es el dispositivo conectado (ver *Figura 15*, en la lista de dispositivos, "*X-MCC2 (rev 1) Controller, Firmware Version 7.14*"). Después el número que sigue a continuación, 0, 1 o 2, hace referencia al dispositivo asociado al controlador (0 es el propio controlador, 1 y 2 son los ejes que controla). En la imagen solo aparece el 0 ya que no hemos enviado ninguna instrucción a los ejes ni hemos solicitado ningún dato sobre ellos.

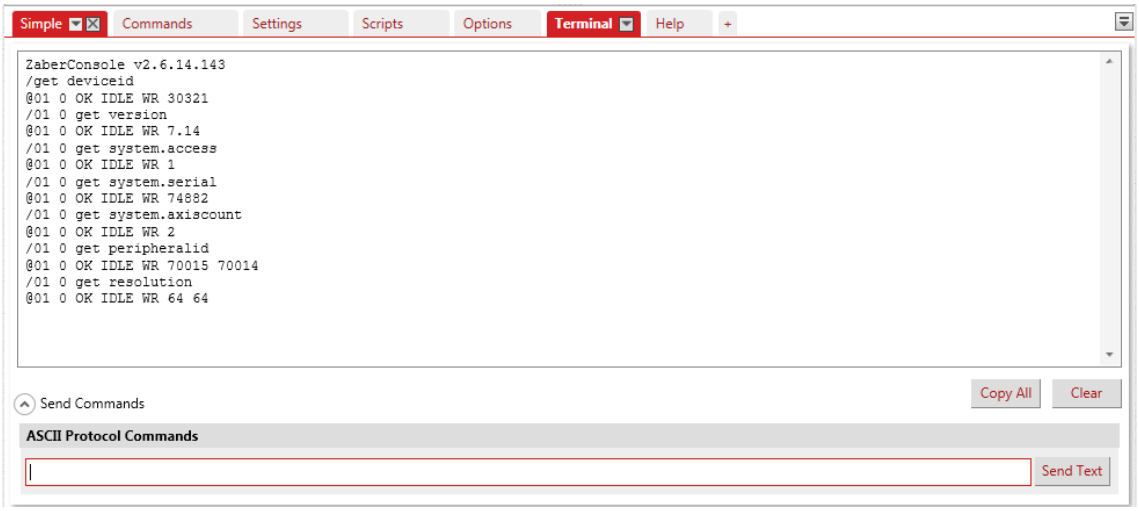

Figura 19. Interfaz de la consola ZABER – Terminal.

Ya hemos visto como son las funciones básicas de la mesa y como implementarlas desde el software de la consola Zaber que nos proporcionan los fabricantes. No obstante, para el trabajo que queremos desarrollar en este proyecto es una forma muy rudimentaria y simple de hacer las cosas. Todo lo expuesto hasta ahora se ha hecho con el objetivo de familiarizarse un poco más con el funcionamiento del dispositivo para facilitar su comprensión a la hora de implementar estas funciones en MATLAB, que será el software que usaremos.

3.6.2. Control de la mesa XY en MATLAB

La consola de Zaber nos permite programar scripts para controlar la mesa XY con las instrucciones que queramos, pero no es lo óptimo. Los fabricantes nos dan acceso a librerías de MATLAB para poder escribir los scripts desde este programa y, como también necesitaremos controlar otro dispositivo simultáneamente (osciloscopio) nos conviene usar MATLAB como nexo entre ambos. Otra razón de peso es la potencia del software de MATLAB y todas las herramientas que nos proporciona, ya que no solo nos limitaremos a controlar la mesa y tomar datos, sino también a procesarlos.

En el siguiente enlace tenemos todo el código que se puede utilizar para controlar la mesa con MATLAB:

https://www.zaber.com/software/docs/motion-library/ascii/references/matlab/

Como se puede ver hay una gran cantidad de comandos que se pueden utilizar para programar el movimiento de la mesa, sin embargo tan solo con el uso de unos pocos comandos básicos

podremos realizar la tarea que nos concierne. A parte de los comandos que invocan las librerías de Zaber a MATLAB y los que establecen la comunicación entre el controlador de la mesa y el ordenado, usaremos las siguientes funciones básicas:

- Comando *home* explicado anteriormente
- Funciones de desplazamiento relativo, para desplazar los ejes la distancia que deseemos con respecto a su posición actual.

Únicamente con estas dos funciones seremos capaces de realizar barridos a la FPGA, ya que el resto de código auxiliar nos lo proporcionará MATLAB.

3.6.3. Control del osciloscopio con MATLAB

Otra de las buenas razones por las que el uso de MATLAB era el más conveniente es que necesitamos también tomar los datos de la medida de la sonda directamente del osciloscopio. El software de MATLAB nos permite establecer una conexión con dispositivos externos de distintas maneras, escogimos comunicarnos con el osciloscopio mediante el estándar VISA[7]. Para ello solo necesitamos saber la dirección del instrumento, que se puede encontrar navegando los propios menús del osciloscopio, y la sintaxis de los comandos e instrucciones que se hallan recopilados en el manual de *Tektronix* [7]. Una vez hecho esto, ya podemos configurar los parámetros de medida del osciloscopio como mejor nos convenga y solicitar los datos de cualquiera de los dos canales para su posterior análisis y procesado.

Solo quedaría integrar en un solo script de MATLAB el control de la mesa XY junto con el del osciloscopio para empezar con las medidas.

[7] Acrónimo del inglés *Virtual Instrument Software Architecture*, es un estándar de comunicación para instrumentos de medida. Extraído de https://www.tek.com

4. EXPERIMENTO Y RESULTADOS

4.1. Barridos

El modus operandi que llevaremos a cabo será el siguiente: Primero, colocar la FPGA programada con el encriptador en bucle en la mesa bien sujeta para que acompañe al movimiento de esta (casualmente el ancho de la placa base coincidía perfectamente con el ancho de la plataforma de la mesa, 100 mm, con lo cual no hicieron falta medidas adicionales de sujeción). Después, situar la sonda de medida campo magnético cercano a poca distancia del dispositivo y en una posición fija haciendo uso del brazo articulado. De esta forma lo que se mueve no es la sonda, sino la FPGA que queremos atacar y esto nos proporciona mucha estabilidad. Así dispuestos los aparatos, dividiremos la FPGA en una cuadrícula de la resolución que determinemos adecuada y tomaremos una medida de la traza de emisión electromagnética en cada sector de la cuadrícula (ver *Figura 20*).

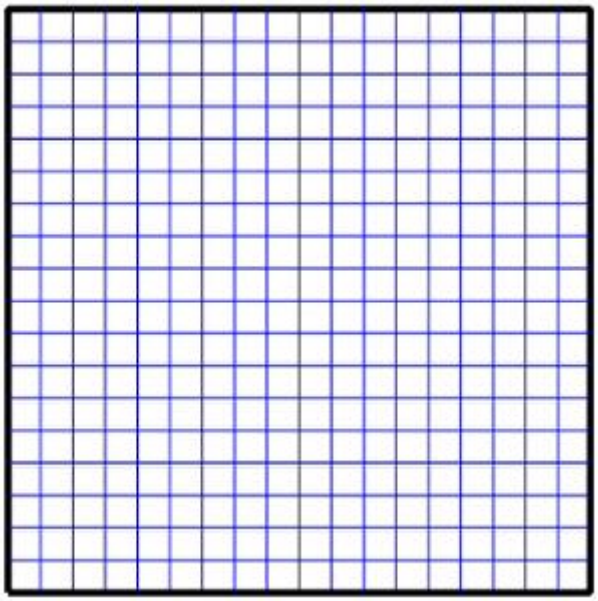

Figura 20. Ejemplo de cuadrícula de 18x18.

La propuesta principal era probar 3 tipos de barridos con los cuales escanear todas las zonas de la cuadrícula: El barrido zigzag, el barrido en espiral y el barrido zigzag diagonal. El barrido zigzag comienza en una de las esquinas y va escaneando por filas. El barrido espiral comienza en el centro de la cuadrícula y va escaneando sectores cada vez más alejados de este describiendo una espiral cuadrada creciente a su paso. El barrido zigzag diagonal es más complejo, comienza en una esquina también, pero va escaneando diagonales de la cuadrícula en lugar de filas. Todo esto viene ilustrado en las *Figuras 21* y *22.*

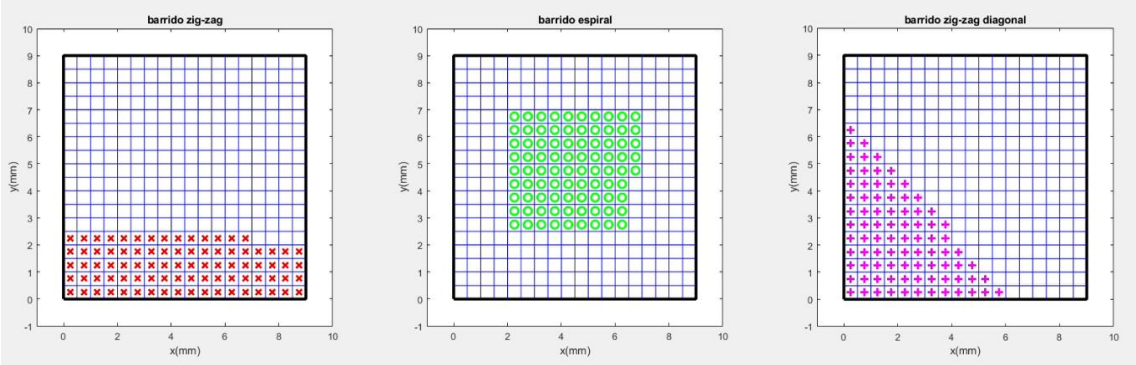

Figura 21. De izquierda a derecha: barrido zigzag, espiral y zigzag diagonal.

En lo que a tiempo se refiere cualquiera de los 3 tardaría lo mismo en recorrer todos los sectores de la cuadrícula, sin embargo la forma de hacerlo no es la misma y en cada caso hay ciertas zonas privilegiadas. Pueden servirnos de utilidad si tenemos alguna sospecha de donde se encuentra la zona de interés que queremos analizar o por si, habiendo hecho un escáner preliminar, deseamos realizar otro con mayor resolución en un determinado subsector de la cuadrícula con cierta forma. En el caso del barrido en espiral, por ejemplo, empezamos el movimiento desde el centro de la espiral y puede sernos de utilidad para ahorrar tiempo si sabemos que la zona de interés está situada cerca del centro. En el caso del barrido en zigzag diagonal, la forma en la que barremos es distinta, ya que vamos desde una esquina hasta la opuesta y la forma que tiene el área escaneada es la de un triángulo isósceles (al menos hasta antes de llegar a la diagonal principal, después la zona barrida se asemeja más a una forma de diamante). Esto nos puede servir de utilidad si queremos escanear un sector triangular de la cuadrícula, como por ejemplo alguna de las esquinas.

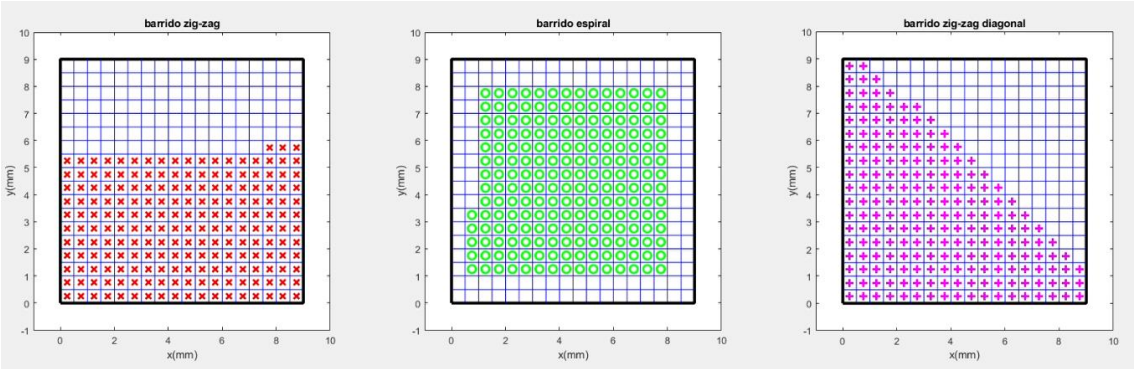

Figura 22. Evolución de los barridos.

Si comparamos la *Figura 21* con la *Figura 22* vemos como cada barrido prioriza una de las zonas de la cuadrícula. También da la sensación de que el barrido espiral en la *Figura 22* está casi terminado en comparación con los otros dos aunque les quede a todos la misma área por escanear. Aunque solo sea un efecto óptico, ilustra lo dicho en el párrafo anterior sobre priorizar, por ejemplo, el centro de la cuadrícula si es nuestra zona de interés. Dicho esto, como nuestro objetivo es escanear la cuadrícula al completo, es decir, el tiempo que tardemos será el mismo en cualquier caso, usaremos el barrido zigzag normal que es el más fácil de programar.

El script de MATLAB, que controla mesa XY y osciloscopio simultáneamente, tenía diversas características. Antes de ejecutarlo debemos configurar los parámetros del barrido, esto es, el tamaño de la cuadrícula que vamos a escanear (en nuestro caso 15 mm de lado, que es lo que mide la FPGA) y la resolución de la cuadrícula, es decir, en cuantas filas y columnas vamos a dividirla para medir la emisión en cada uno de los sectores resultantes. A modo de ejemplo, una cuadrícula de 15 mm de lado con una resolución de 18 tendría $18^2 = 324$ sectores cuadrados cada uno de ellos de 15/18 = 0.833 mm de lado (sirva la *Figura 20* como ilustración). Una vez especificadas las dimensiones podemos ejecutar el script.

Al ejecutar el código, primero se establece la comunicación con la mesa XY, mediante las funciones de las librerías de Zaber, y la comunicación con el osciloscopio vía VISA. Después se configura el osciloscopio con los siguientes parámetros:

- Intervalo de tiempo de medida: 4 μs
- Número total de puntos por traza: 10^4
- Escala vertical del canal de la sonda: 5.00 mV/división
- Escala vertical del canal del disparo: 1.00 V/división
- Tipo de disparo: pendiente ascendente a 1.6 V
- Tipo de adquisición: promediado de 16 adquisiciones

Una vez hecho esto se ejecuta el comando *home* de la mesa XY, del cual hablamos en la sección 3.6, para que el controlador sepa donde está el comienzo de cada eje. En este momento se pausa el código para que el operario, mediante el uso de las ruedas de control manual de la mesa, sitúe una de las esquinas de la FPGA (concretamente la superior izquierda por la forma en la que está programado el barrido zigzag, pero es completamente arbitrario) justo debajo de la sonda. Pulsando cualquier tecla reanudamos la ejecución y comienza el barrido y la toma de datos.

Cada vez que se ha guardado la traza de un sector de la cuadrícula la mesa se desplaza un paso para que la sonda apunte al siguiente sector (siguiendo el orden del barrido zigzag antes explicado), así sucesivamente hasta que tenemos una traza de cada uno de los sectores de la cuadrícula. Una vez terminado el algoritmo podemos pasar al procesado de datos, pero antes hagamos una puntualización sobre la orientación de la sonda.

En la sección 3.3 dijimos que la sonda de medida de campo magnético tenía una marca blanca en la punta que indicaba la dirección en la que debía circular una intensidad de corriente para tener una medida de campo magnético óptima. Como la FPGA tiene simetría cuadrada (ver *Figura 11*) y el cableado interno va en las direcciones de los lados del cuadrado, la elección más sabia para no perder información de ninguna componente del campo magnético es situar la sonda de tal manera que la marca blanca apunte en la dirección de cualquiera de las diagonales del cuadrado. De esta forma mediremos el campo de las intensidades de corriente de los cables que vayan en dirección vertical o en dirección horizontal sin perder información de ninguna de las dos componentes. Para ilustrar esto hicimos un barrido (resolución 10) con la sonda en distintas posiciones y obtuvimos los siguientes resultados:

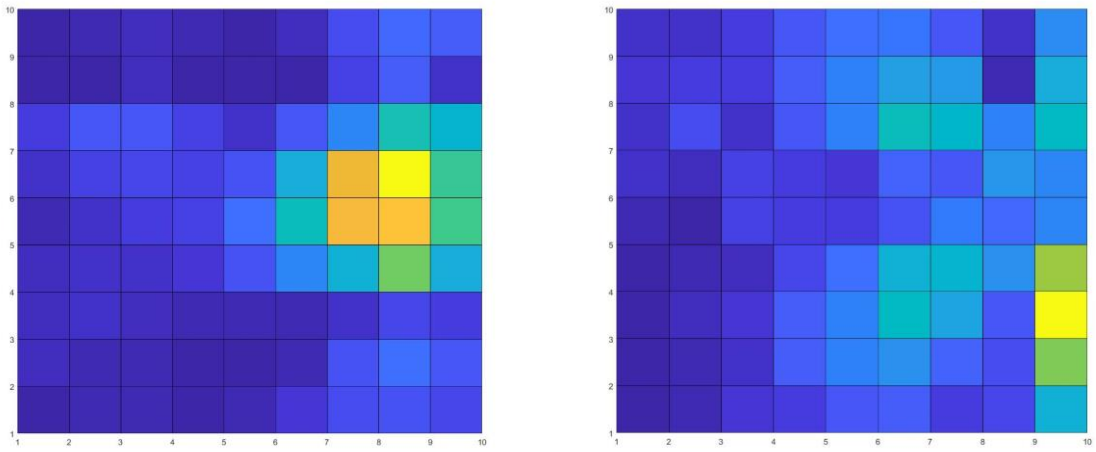

Figura 23. A la izquierda barrido con la marca de la sonda en dirección vertical, a la derecha barrido con la marca de la sonda en dirección horizontal.

Podemos ver en la *Figura 23* un mapa de colores de la cuadrícula del barrido. Lo que se representó en cada sector fue el valor máximo de la traza correspondiente, para tener mayor intuición de la forma de una traza ver *Figura 24*. Los tonos azules representan valores bajos del máximo, los amarillos y naranjas valores altos y los tonos verdes son valores intermedios. En ambos casos la FPGA estaba programada de igual forma y realizando encriptaciones en bucle, sin embargo puede apreciarse una diferencia notable en cada tipo de barrido.

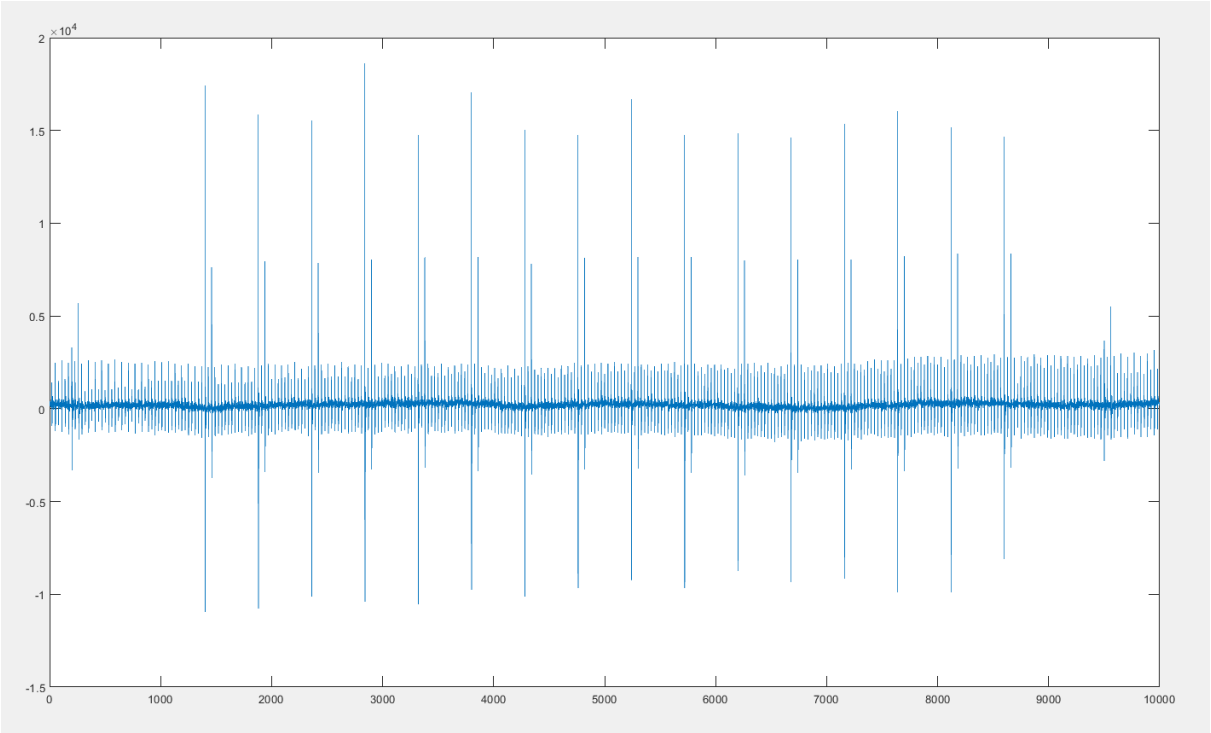

Figura 24. Traza electromagnética en una de las zonas de mayor intensidad. El eje horizontal es el tiempo y el vertical la intensidad de campo magnético.

Tanto en el caso de la sonda en horizontal como en el de la sonda en vertical estaríamos perdiendo información de la emisión de campo magnético de partes de la FPGA, por ello optamos por orientar la sonda con la marca blanca en diagonal y obtuvimos lo representado en la *Figura 25*.

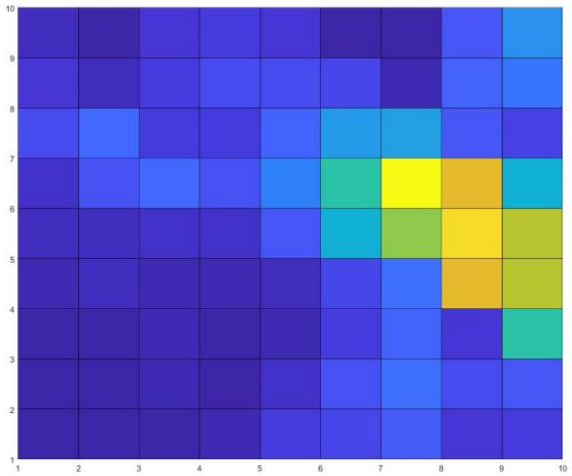

Figura 25. Barrido con la marca de la sonda en diagonal.

En este caso las "zonas calientes" de los barridos vertical y horizontal siguen apareciendo aquí, es decir, no perdemos información de ninguna de las dos, aunque aparecen con menor intensidad debido a que al estar en diagonal la sonda medimos la proyección del vector del campo magnético en esa dirección. Antes de continuar, si observamos con detenimiento los mapas de colores en las figuras, podemos ver que, a pesar de tener una resolución 10, contamos 9 cuadrados por fila o columna. Esto se debe a la forma que tiene MATLAB de representar estas figuras. En realidad, los puntos que medimos son los vértices de estos cuadrados, y MATLAB asigna un valor al área encerrada en cada cuadrado en función del valor de los 4 vértices adyacentes. No obstante, tampoco supone un problema ya que no necesitamos tanto nivel de precisión, además, en las siguientes figuras utilizaremos otras funciones de este programa para representar los mapas de formas más variadas.

4.2. Mapas cartográficos

Una vez estudiada la forma óptima de realizar los barridos, o sea, con la sonda en diagonal, a continuación se presentan mapas cartográficos de distintas resoluciones de la cuadrícula.

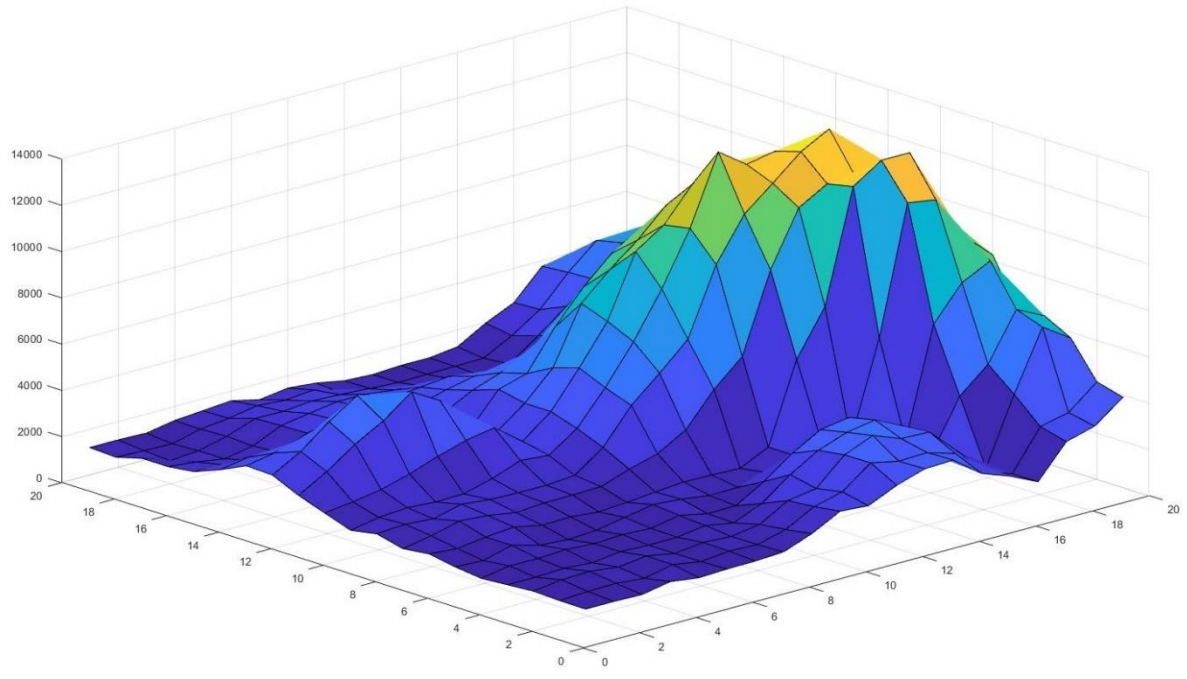

Figura 26. Barrido con resolución 20 – Mapa 3D

En la *Figura 26* está representado un barrido de resolución 20 en formato tridimensional. Ya podemos ver como se dijo antes que los tonos azules son los correspondientes a valores más bajos, los verdes a valores intermedios y los naranjas y amarillos a valores más altos. Y

recordemos que un sector de la cuadrícula no es el cuadrado sino el vértice, y que el color del área es un promediado de los cuatro vértices de cada cuadrado.

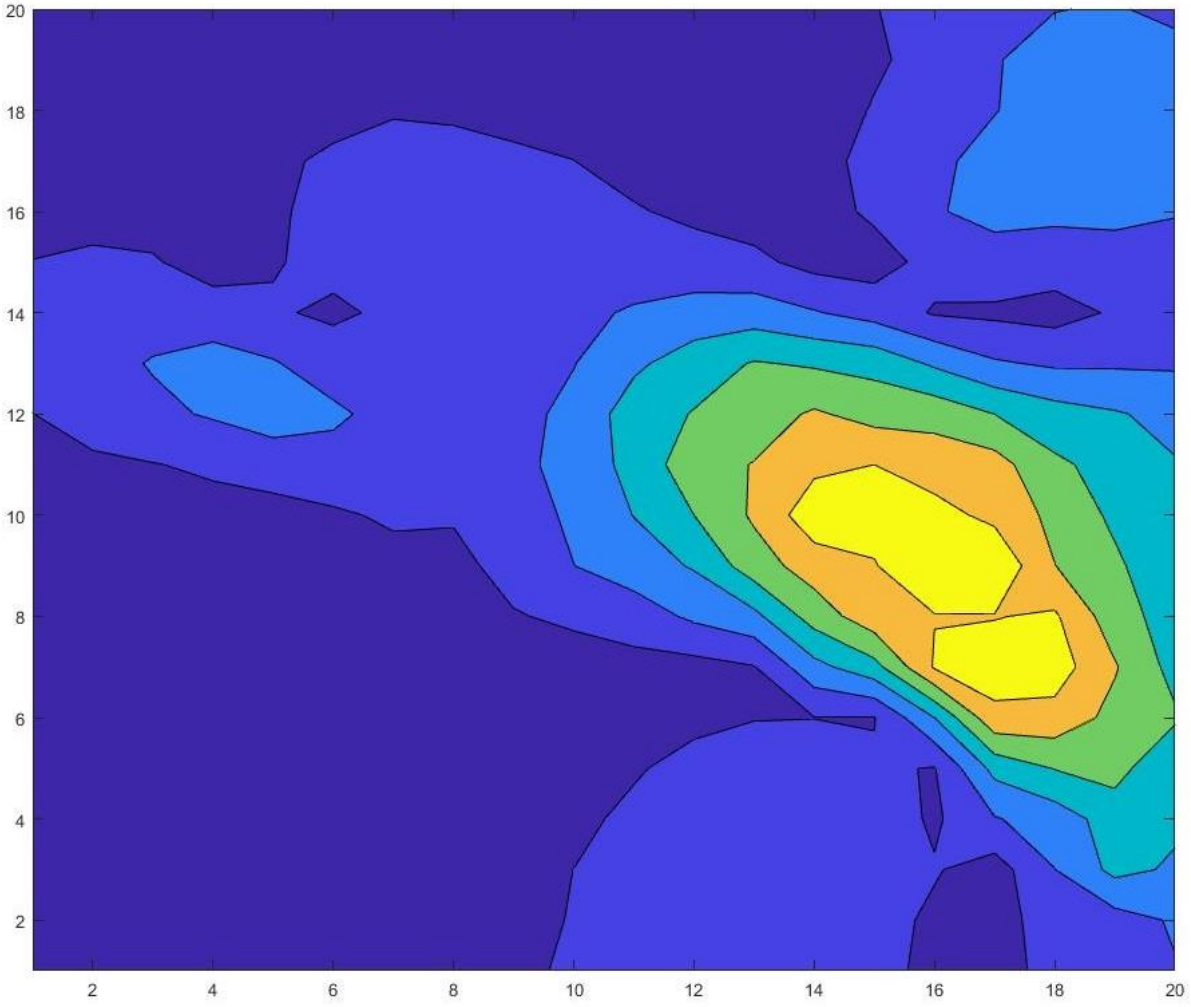

Figura 27. Barrido con resolución 20 – Mapa de superficies

En la *Figura 27* vemos el mismo barrido pero representado con un mapa bidimensional. Esta vez prescindimos del formato "cuadrícula" y vemos una representación con líneas más redondeadas. Aun así se ve que la zona de mayor intensidad sigue estando en el mismo sitio, como era de esperar. Aunque a priori no podemos decir que sea la que nos interesa a la hora de realizar un ataque, esto es solo un análisis preliminar, posteriormente habría que estudiar la forma de las trazas de cada zona, lo cual haremos más adelante.

Veamos que mapas obtenemos si aumentamos la resolución, lo haremos con ejemplos de resoluciones de 40 y de 100.

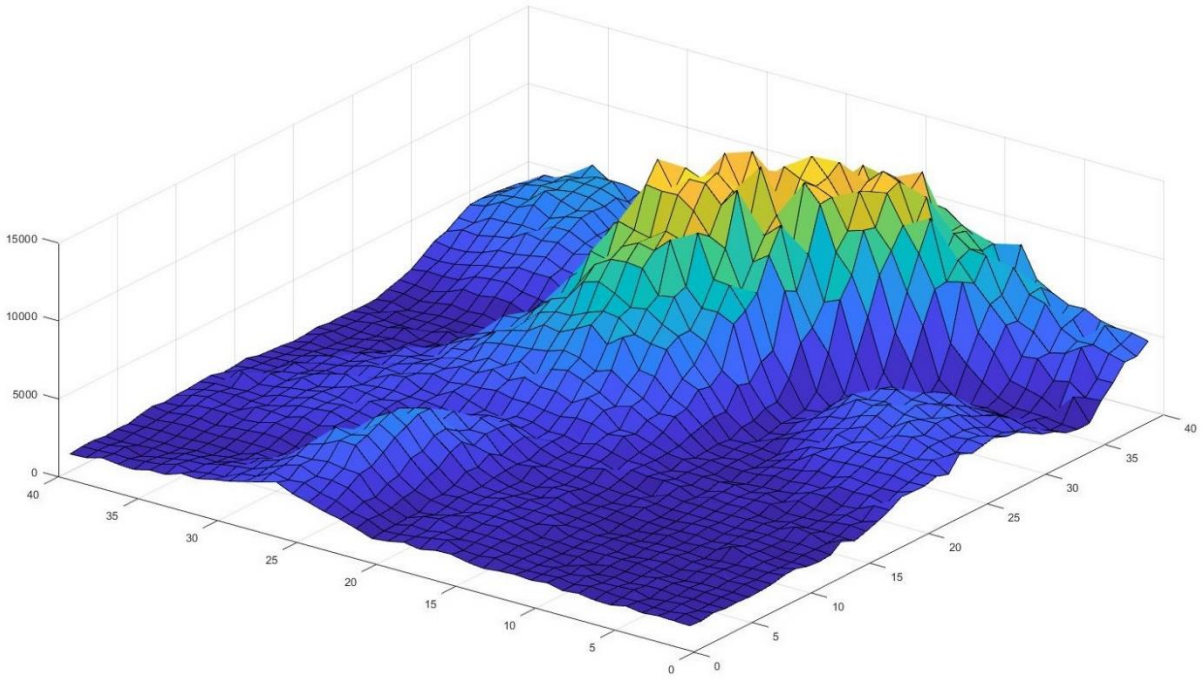

Figura 28. Barrido con resolución 40 – Mapa 3D

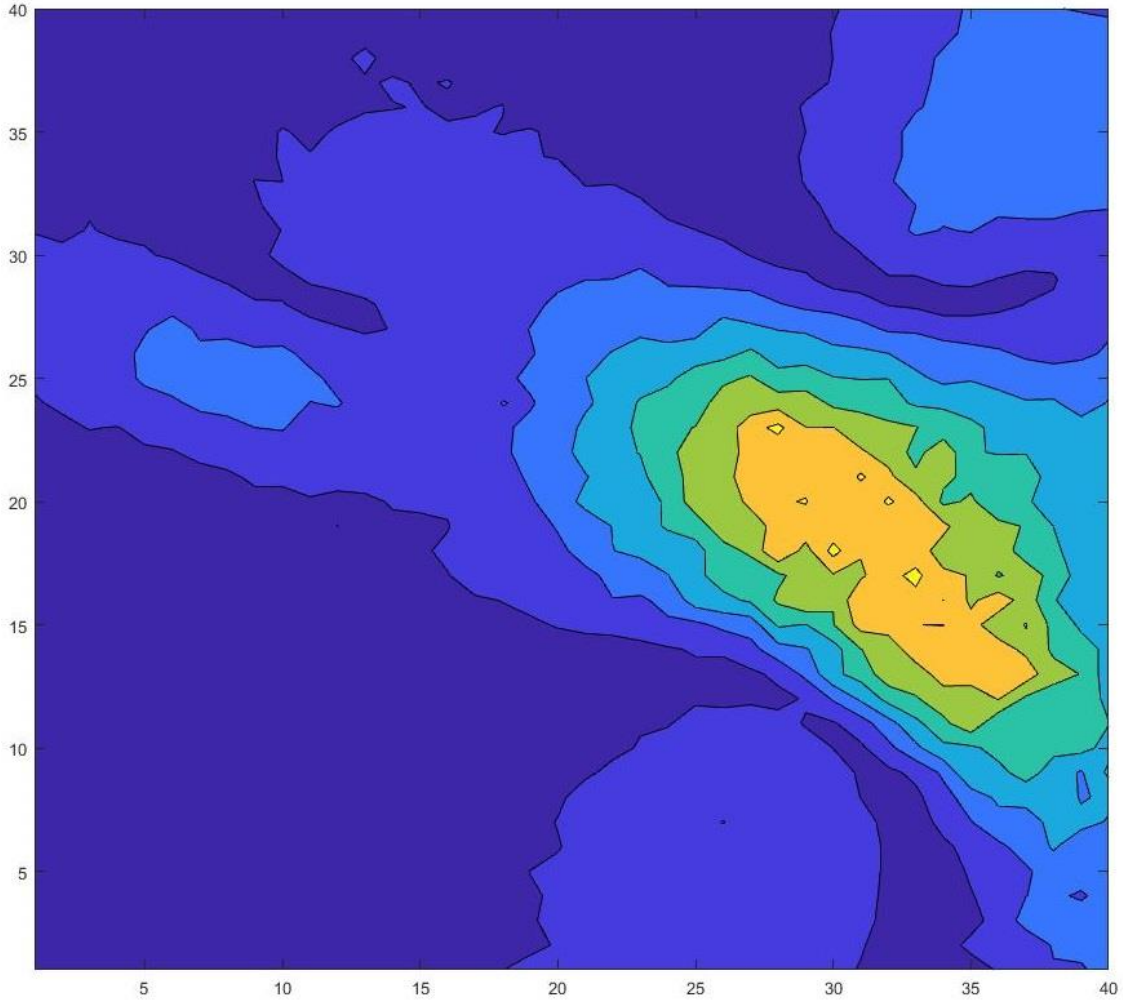

Figura 29. Barrido con resolución 40 – Mapa de superficies

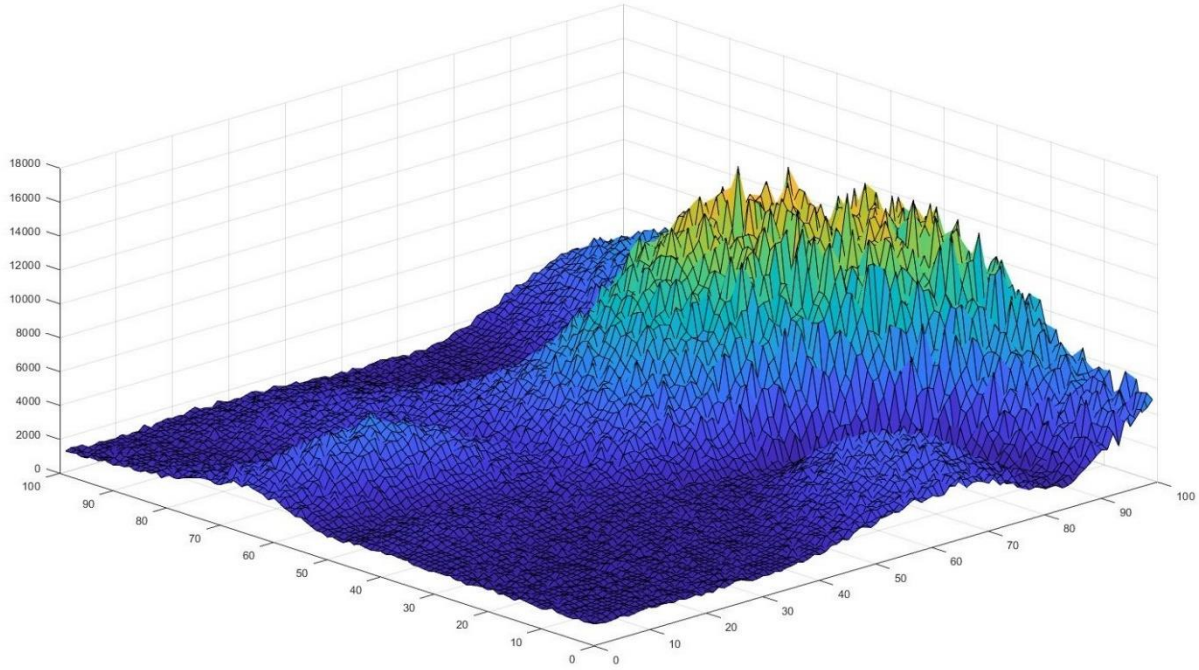

Figura 30. Barrido con resolución 100 – Mapa 3D

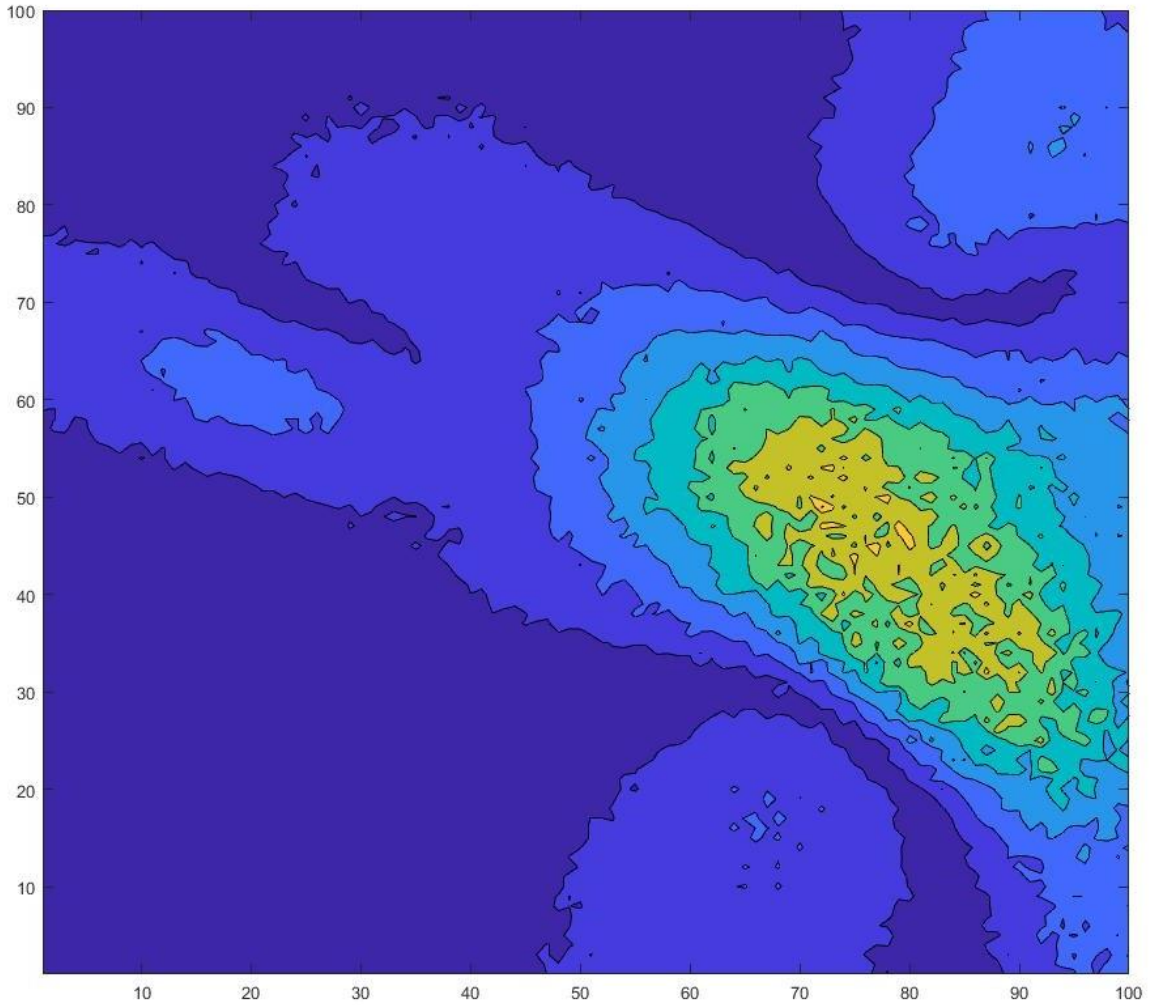

Figura 31. Barrido con resolución 100 – Mapa de superficies

Tanto en las *Figuras 28* y *29*, correspondientes al barrido de resolución 40, como en las *Figuras 30* y *31*, correspondientes al barrido de resolución 100, podemos ver que las "zonas calientes" y las "zonas frías" se sitúan en las mismas coordenadas.

Una pregunta que nos podríamos plantear es hasta que punto nos conviene aumentar la resolución, para ello estudiemos el tiempo que tarda cada barrido en realizarse en función de la resolución de la cuadrícula.

Tiempos frente a resolución					
resolución	5	10	20	40	100
t (s)	14.28	48.16	170.86	604.32	3643.00

Tabla 1. Resolución de la cuadrícula y tiempo transcurrido desde el inicio hasta la finalización del barrido.

Podemos visualizar los puntos de la *Tabla 1* en un gráfico, tal y como aparece en la *Figura 32*.

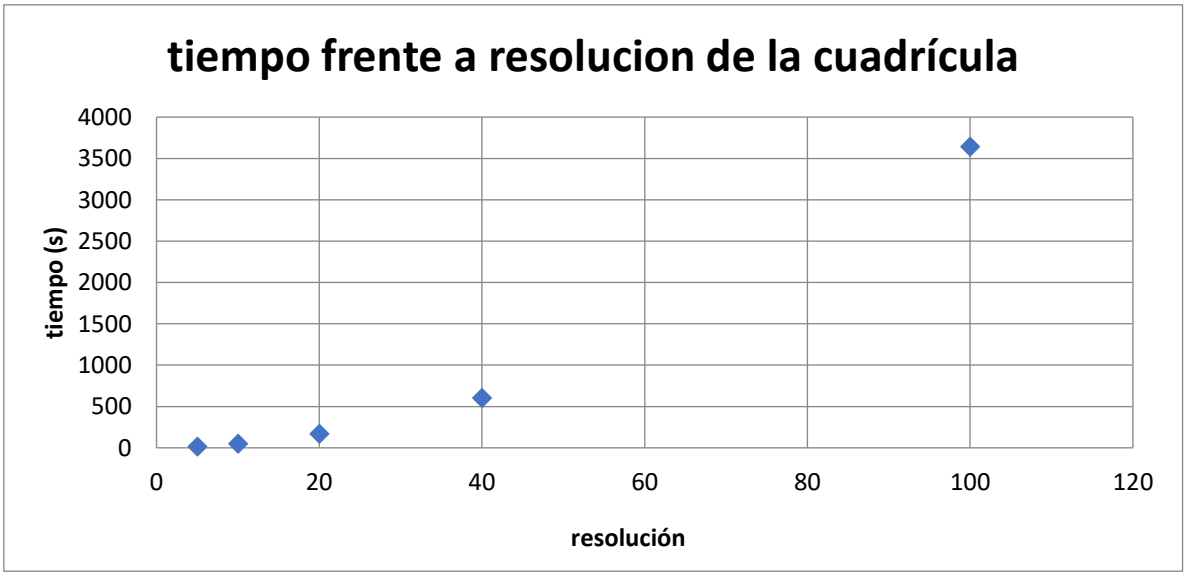

Figura 32. Representación gráfica de los tiempos de barrido frente a la resolución de la cuadrícula.

Se puede apreciar que los puntos describen la forma de una parábola. Veamos que obtenemos si representamos el tiempo de barrido frente al cuadrado de la resolución (*Figura 33*).

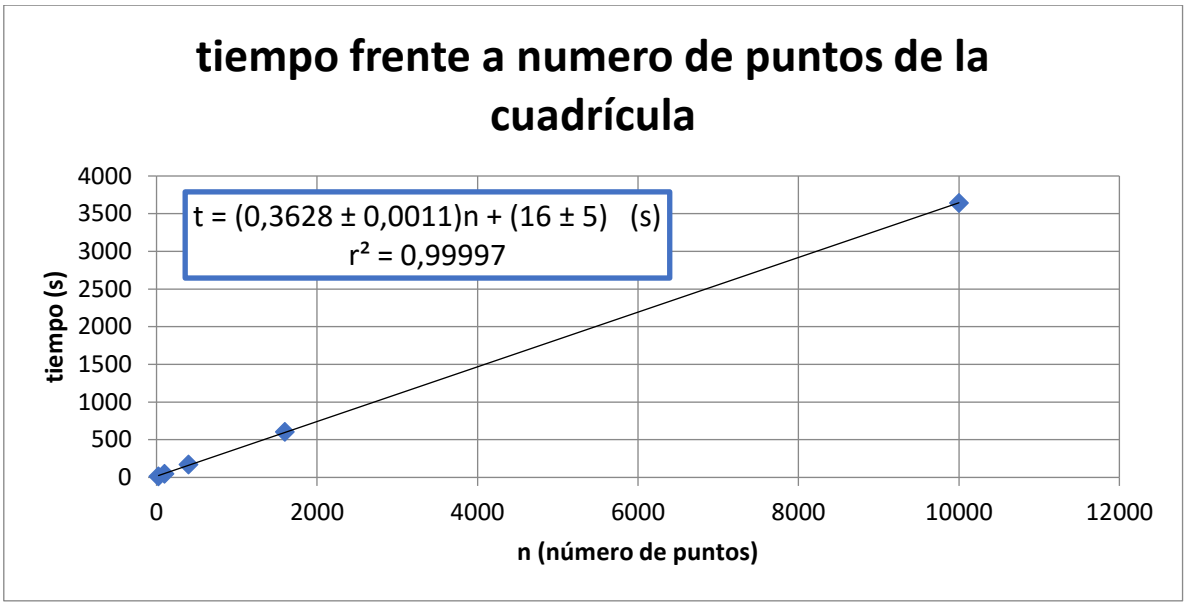

Figura 33. Representación gráfica de los tiempos de barrido frente a la resolución de la cuadrícula al cuadrado, o sea, el número de puntos.

Como era de esperar, ya que el número de puntos que se escanean en un barrido es igual al cuadrado de la resolución, tenemos que el tiempo que tarda el barrido es proporcional a este valor. Si calculamos la ecuación de la recta de regresión lineal, representada en la *Figura 33*, podemos ver el tiempo por sector de la cuadrícula que tarda en tomarse la medida.

Pendiente	(0.3628 ± 0.0011) *s*
Ordenada en el origen	(16 ± 5) *s*
r^2	0.99997

Tabla 2. Parámetros de la recta de regresión lineal de los puntos de la Tabla 1.

4.3. Trazas electromagnéticas

Para finalizar, vamos a ver cómo son las trazas de las regiones de la FPGA. Haremos uso de los datos guardados para el barrido 100x100 y las coordenadas las representaremos en las de la cuadrícula. Si vemos la *Figura 31* que es el mapa cartográfico del barrido 100x100, podemos ver valores en el eje horizontal y vertical, estas serán las coordenadas x e y respectivamente.

En la *Figura 34* se representa la traza que obtenemos en una de las zonas azul oscuro, o sea, poca intensidad, concretamente en las coordenadas (20,20).

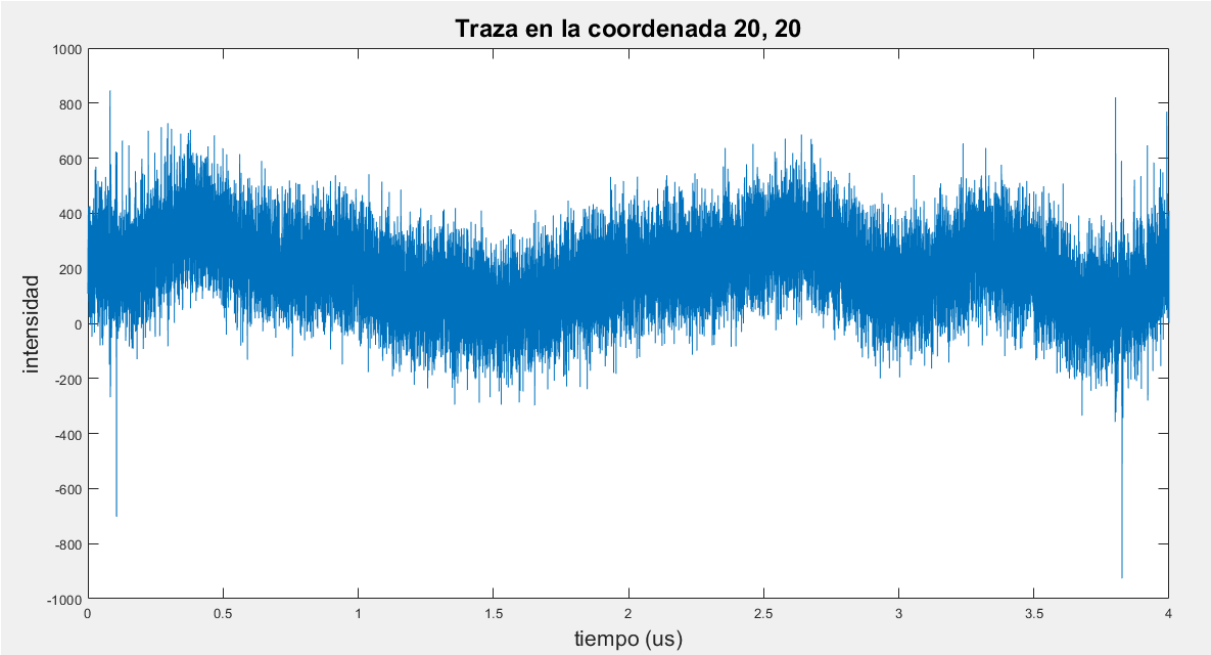

Figura 34. Traza de la coordenada (20,20).

Veamos que aspecto tiene una traza de la zona amarilla, o sea, la de mayor intensidad, concretamente en las coordenadas (83,44).

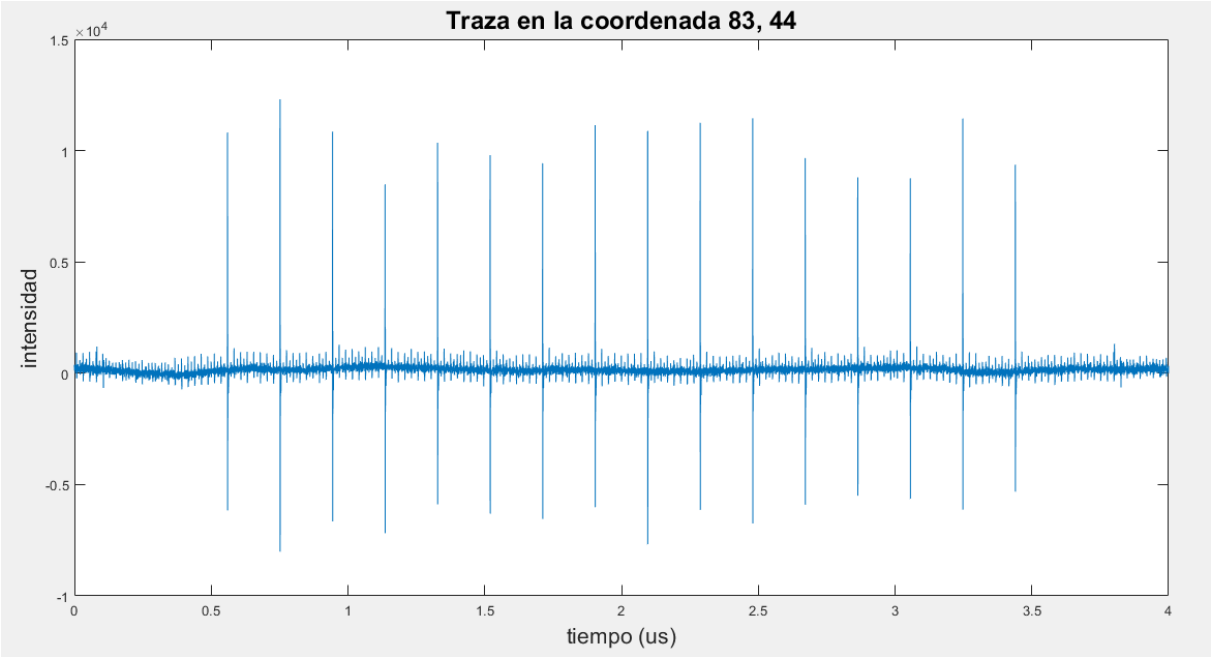

Figura 35. Traza de la coordenada (83,44).

En el mapa cartográfico del barrido de resolución 100 podemos observar otras zonas con cierta intensidad a parte de la amarillo, hablamos de las zonas azul un poco más claro que el de fondo

y se sitúan en torno a las coordenadas (95,87), (65,18) y (20,60). Representaremos las trazas de estas coordenadas en las *Figuras 36, 37* y *38*.

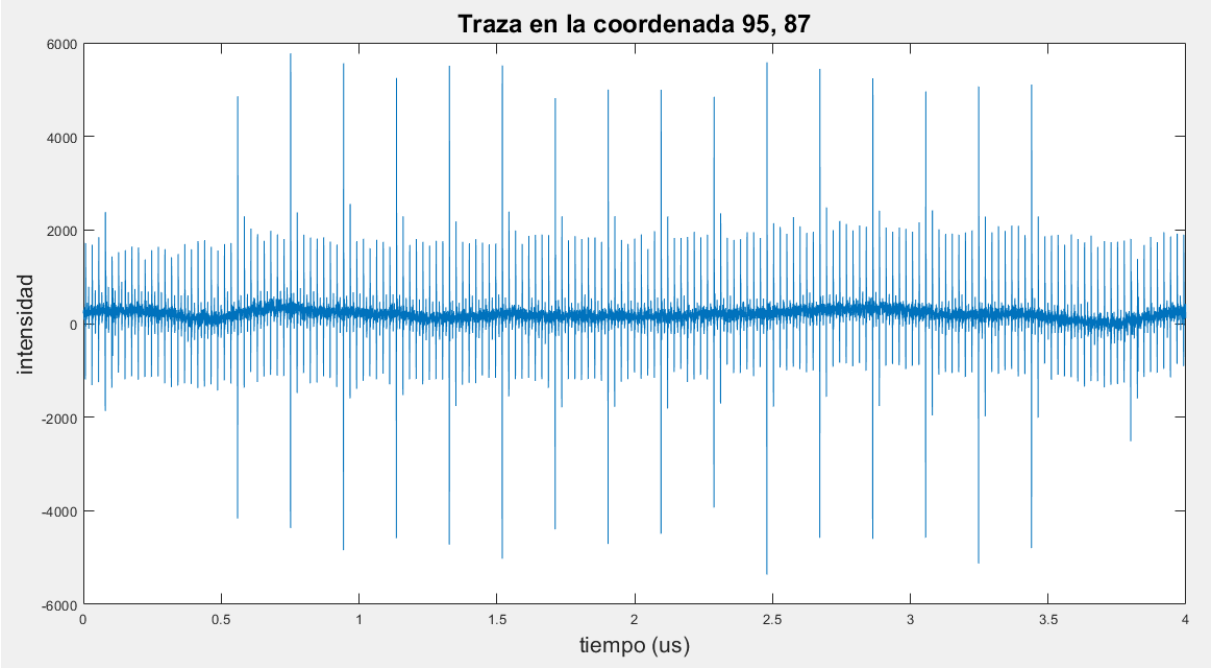

Figura 36. Traza de la coordenada (95,87).

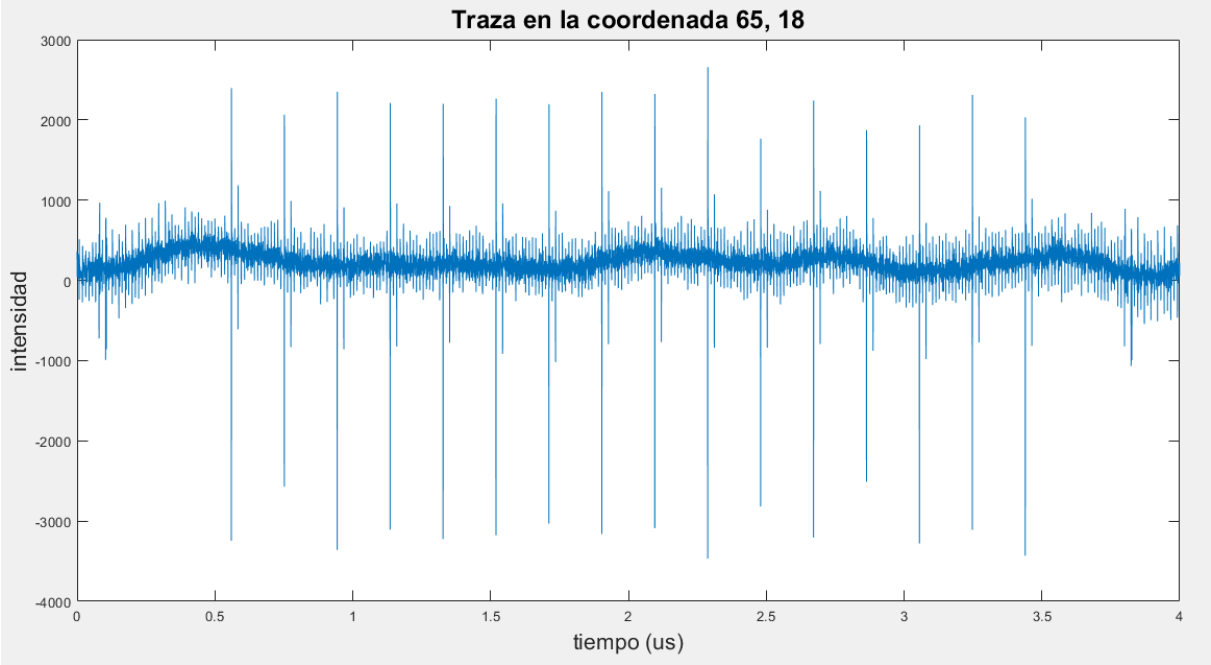

Figura 37. Traza de la coordenada (65,18).

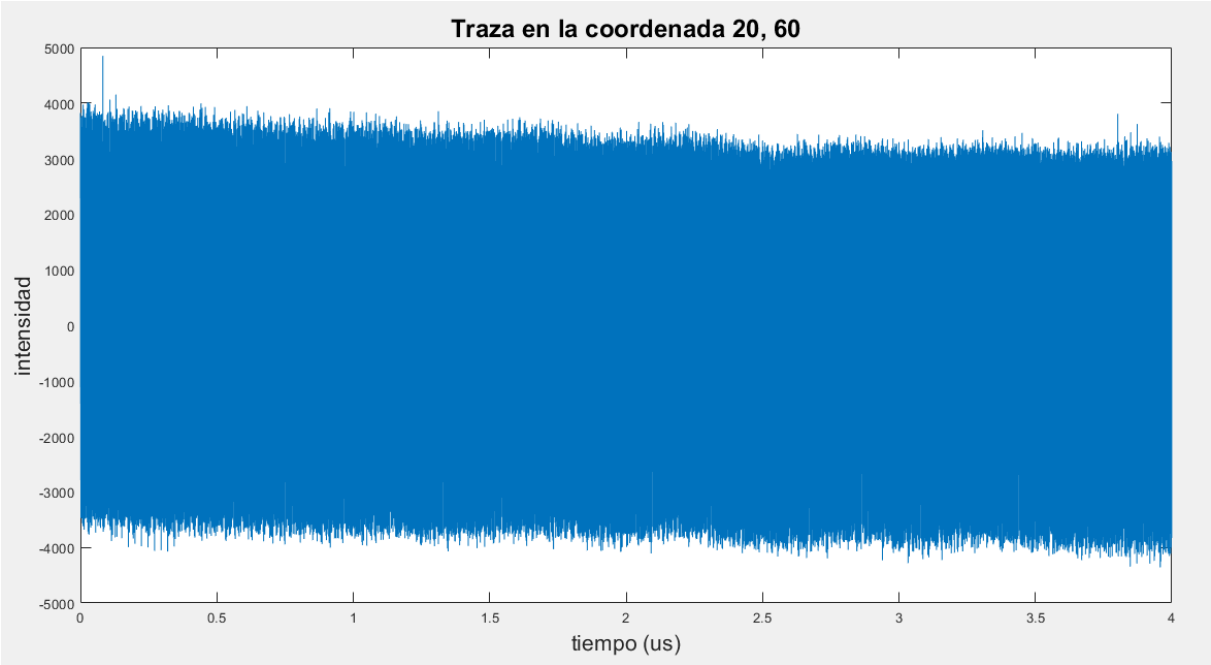

Figura 38. Traza de la coordenada (20,60).

5. DISCUSIÓN

Los resultados de los mapas cartográficos de emisión electromagnética son muy interesantes. Fijándonos en las *Figuras* desde la *26* hasta la *31* podemos ver como en todas ellas la "zona caliente" (refiriéndonos a esto como la zona de mayor intensidad) se sitúa en el mismo sitio siempre. Lo mismo sucede con los demás sectores de menor intensidad. A pesar de que la imagen global sea parecida, obtenemos más detalle a pequeña escala conforme aumentamos la resolución del barrido. Sin embargo, no tiene porque ser el barrido con más resolución el más conveniente.

Recordemos que el objetivo del trabajo es localizar la zona óptima para realizar un ataque electromagnético, no necesitamos tener un análisis con demasiado detalle de la emisión en todos los puntos. Además, viendo las *Tablas 1* y *2* y las *Figuras 32* y *33* es bastante razonable tratar de mantener la resolución del barrido lo más pequeña posible por cuestiones de tiempo. En definitiva, el tamaño de la resolución que escojamos depende del contexto en el que estemos (tamaño del dispositivo que vayamos a atacar, tiempo disponible, importancia del detalle a pequeña escala…). Personalmente creo que en nuestro caso un barrido de 20x20 nos da toda la información que necesitamos sobre la FPGA y lo realizamos en menos de 3 minutos. Comparado con los 10 minutos del barrido 40x40 y de los 60 minutos que tarda el barrido 100x100 vemos que es un ahorro de tiempo muy considerable. Y como ya comentamos en el párrafo anterior, se diferencian con suficiente detalle las zonas que nos interesan, habiéndonos ahorrado tiempo y trabajo a la hora de atacar el dispositivo.

Podría argumentarse que las zonas de mayor intensidad no debieran ser necesariamente las de mayor interés a la hora de realizar un ataque, y es un argumento totalmente válido. Sin embargo, en nuestro caso, si analizamos trazas de muestra de cada zona, podemos concluir que hay una correlación entre el valor máximo de intensidad de una traza y su interés a la hora de realizar un ataque electromagnético. En la *Figura 35* vemos lo que sería una traza óptima a la hora de realizar un ataque. Podemos identificar claramente 16 picos de intensidad y están muy bien diferenciados del ruido de fondo. No entraremos en detalle de como se realiza un ataque electromagnético porque escapa del ámbito de este trabajo, pero sí explicaremos por qué a un atacante le interesa una traza con este aspecto.

Cada uno de los picos que aparece es el resultado de la emisión de campo magnético de un pulso de corriente. El primer pico corresponde a la expansión de la clave del AES, cada uno de los 10 siguientes corresponde a un ciclo de encriptación, donde el algoritmo AES transforma el texto de entrada en un texto encriptado a lo largo de 10 rondas. Los 5 picos siguientes son operaciones auxiliares para realizar la realimentación del texto cifrado de salida del AES y convertirlo de nuevo a texto de entrada para iniciar el siguiente ciclo de encriptado. Por tanto, si vemos este patrón de 16 picos, estaremos cerca de la zona donde se está implementando la encriptación en la FPGA y será la zona óptima para las medidas de un ataque electromagnético.

Si nos vamos a zonas de intensidad baja, como la de la *Figura 34* vemos que la traza de muestra de ese sector no nos aporta ninguna información útil, es solo ruido. Si vemos el aspecto de las trazas de muestra de las zonas de intensidades moderadas (*Figuras 36* y *37*) vemos que tenemos el mismo patrón de picos que en la traza de la zona de mayor intensidad. Además podemos apreciar otros fenómenos, como el flanco de bajada de cada pulso de corriente, que aparece como un pico de menor intensidad justo a la derecha de cada pico (*Figura 37*) o las emisiones de los flancos de reloj (*Figura 38*) que son picos de igual tamaño y menor al de los picos principales y que aparecen con mayor frecuencia. Es importante no confundir los pulsos de reloj con los pulsos de encriptación, y el criterio para diferenciarlos está en que los pulsos de reloj aparecerán siempre con una amplitud prácticamente idéntica y en todo instante de tiempo. Estas dos zonas pueden ser válidas también para realizar un ataque, aunque la relación señal-ruido en ellas es peor, por eso quizás nos convenga más la zona amarilla donde la definición de los picos era muy buena.

Hay otra zona de intensidad moderada, y es la correspondiente a la *Figura 38*. Dentro del contexto del mapa cartográfico no parece haber ninguna diferencia entre esta zona y las dos que se han comentado en el párrafo anterior, sin embargo al estudiar con más detalle el aspecto de las trazas vemos que no tienen nada que ver. Esta traza, a pesar de tener una intensidad moderada no nos aporta ninguna información. No es ruido de fondo porque la intensidad del campo magnético aquí es considerablemente más grande que en las zonas de azul más oscuro en el mapa (ver *Figura 34*) , pero no es para nada lo que buscamos. Por esta razón puede que el criterio de cartografiar el valor más alto de la intensidad de cada traza no sea el método óptimo, aunque en general nos ha dado buenos resultados en el estudio de esta FPGA y este encriptador.

Podrían proponerse otros criterios distintos a clasificar las trazas únicamente en función del valor más alto de la intensidad. Por ejemplo, podríamos programar con MATLAB un script que fuese capaz de contar los picos principales y compararlos con el ruido de fondo y representar en el mapa una "puntuación" proporcional a lo bien definidos que estuvieran estos picos y lo tanto que se pareciesen al patrón de 16 picos que comentamos anteriormente. Habría que tener cuidado con que no confundiese los picos de encriptación con los del reloj, tal y como se ha comentado anteriormente.

6. CONCLUSIONES

Uno de los pasos más importantes a la hora de realizar un ataque de canal lateral electromagnético es el posicionamiento correcto de la sonda de medida. En este trabajo hemos visto como podemos automatizar este proceso con el uso de una mesa XY y su control mediante un ordenador. Gracias a este método podemos diferenciar las zonas que son de interés de las que no lo son a la hora de realizar un ataque electromagnético en tiempos relativamente pequeños ahorrando así trabajo.

Para respaldar este método hemos realizado un mapa cartográfico de emisión electromagnética de una FPGA *Nexys A7* de *Agilent* programada con un encriptador AES de 128 bits realizando operaciones de encriptación en bucle y en menos de 3 minutos hemos obtenido un resultado con suficiente resolución como para localizar las zonas de interés.

Anexo I. Fragmento del código de MATLAB del barrido zigzag.

```
while t

    Y = zeros(1,n);

    % lectura de traza
    fprintf(visaObj,':DATa:SOUrce CH2');
    fprintf(visaObj,':DATa:STARt 1');
    fprintf(visaObj,':DATa:STOP 10000');
    fprintf(visaObj,':DATa:ENCdg ASCIi');
    fprintf(visaObj,':CURVe?');
    traceCH2=fscanf(visaObj);
    Y=str2num(traceCH2(1:end));

    if par
        M(i+1,:,j+1) = Y;
    else
        M(c-i,:,j+1) = Y;
    end


    if i == c - 1
        axisY.moveRelative(ac, Units.LENGTH_MILLIMETRES)
        dir = -dir;
    else
        axisX.moveRelative(dir*ac, Units.LENGTH_MILLIMETRES)
    end
    i = i+1;


    if i >= c
        i = 0;
        j = j+1;
        par = ~par;
    end

    if j >= c
        t = false;
    end

    %pause(0.5)
end
```

7. BIBLIOGRAFÍA

[1] E. Tena-Sánchez, A. Casado-Galán, V. Zuñiga-González, F. E. Potestad-Ordóñez and A. J. Acosta, "Automated experimental setup for EM cartography to enhance EM attacks", Submitted to the International Conference on Design of Circuits and Integrated Systems (DCIS'22), pp. 1-6, 2022.

[2] K. Papagiannopoulos, O. Glamočanin, M. Azouaoui, D. Ros, F. Regazzoni and M. Stojilović, "The Side-Channel Metric Cheat Sheet".

[3] G. Granados Paredes, "Introducción a la Criptografía", Revista Digital Universitaria de la Universidad Nacional Autónoma de México, vol. 7, no. 7, Jul. 2006. https://www.ru.tic.unam.mx/handle/123456789/1105

[4] R. Lumbiarres-López, M López-García and E.F. Cantó-Navarro, "Ataques por canal lateral sobre el algoritmo de encriptación AES implementado en MicroBlaze", Universidad Politécnica de Cataluña, Vilanova i la Geltrú; Universidad Rovira i Virgili, Tarragona.

[5] E. De Mulder *et al*., "Electromagnetic Analysis Attack on an FPGA Implementation of an Elliptic Curve Cryptosystem," *EUROCON 2005 - The International Conference on "Computer as a Tool"*, 2005, pp. 1879-1882, doi: 10.1109/EURCON.2005.1630348. https://ieeexplore.ieee.org/abstract/document/1630348

[6] E. Monmasson and M. N. Cirstea, "FPGA Design Methodology for Industrial Control Systems—A Review," in *IEEE Transactions on Industrial Electronics*, vol. 54, no. 4, pp. 1824-1842, Aug. 2007, doi: 10.1109/TIE.2007.898281. https://ieeexplore.ieee.org/abstract/document/4267891

[7] Tektronix, Inc., "MSO 3000 and DPO 3000 Series Digital Phosphor Oscilloscopes, Programmer Manual". https://download.tek.com/manual/MSO3000-and-DPO3000-Programmer-Manual.pdf

[8] Digilent, "Nexys A7 FPGA Board Reference Manual".

[9] Carlos J. Jiménez Fernández, "Diseño Digital Avanzado". Universidad de Sevilla (España) – [copia privada].

[10] S. Mangard, E. Oswald, and T. Popp, "Power Analysis Attacks: Revealing the Secrets of Smart Cards", Springer, 2007.

[11] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis", in Proc. of International Cryptology Conference (CRYPTO'99), pp. 388-397, 1999.

[12] P. Kocher, ”Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, Other Systems”, in Proc. of International Cryptology Conference (CRYPTO’96), pp. 104-113, 1996.

[13] Y. Hayashi, N. Homma, T. Mizuki, T. Aoki, H. Sone, L. Sauvage, and J.L. Danger, ”Analysis of Electromagnetic Information Leakage From Cryptographic Devices With Different Physical Structures”, IEEE Transactions on Electromagnetic Compatibility, vol. 55, no. 3, pp. 571-580, Jun. 2013.

[14] K. Gandolfi, C. Mourtel, and F. Olivier, ”Electromagnetic Analysis: Concrete Results,” in International Workshop on Cryptographic Hardware and Embedded Systems (CHES’01), vol. 2162, pp.251–261, 2001.

[15] E. Peeters, F.X. Standaert, and J.J. Quisquater, ”Power and electromagnetic analysis:Improved model consequences and comparisons,” in Integration, the VLSI Journal, Special Issue: Embedded Cryptographic Hardware, vol. 40, no. 1, pp. 52–60, 2007.

[16] J.J. Quisquater and D. Samyde, ”Electromagnetic attack,” In: van Tilborg H.C.A., Jajodia S. (eds) Encyclopedia of Cryptography and Security, 2011.

[17] L. Sauvage, S. Guilley, and Y. Mathieu, ”Electromagnetic radiations of fpgas: High spatial resolution cartography and attack on a cryptographic module,” in ACM Transactions on Reconfigurable Technology and Systems (TRETS’09), vol. 2, no 1, pp. 1-24, 2009.

[18] D. R´eal, F. Valette, and M. Drissi, ”Enhancing correlation electromagnetic attack using planar near-field cartography,” in Design, Automation & Test in Europe Conference & Exhibition (DATE’09), pp. 628-633, April 2009.

[19] A. Dehbaoui, V. Lomne, P. Maurine, L. Torres, and M. Robert, ”Enhancing electromagnetic attacks using spectral coherence based cartography,” in IFIP/IEEE International Conference on Very Large Scale Integration- System on a Chip, pp. 135-155, October 2009.

[20] Pub, NIST FIPS. ”197: Advanced encryption standard (AES).” Federal information processing standards publication 197.441 (2001): 0311.