Plataforma para la adquisición, procesado y extracción de información de un radar

Jordi Muñoz Artigues

Resumen– La adquisición y tratamiento de datos procedentes de sensores son la base de cualquier sistema de obtención de información. En este trabajo se desarrolla un diseño para una plataforma que realiza el ciclo completo del procesado de datos procedentes de un radar, desde su obtención hasta la extracción de información y almacenamiento de la misma, utilizando herramientas como una base de datos NoSQL y el multithreading. El diseño ideado es implementado en MATLAB y permite tanto el procesado en tiempo real como el postprocesado. Finalmente se realizan pruebas con la plataforma con el fin de caracterizar las principales aplicaciones y limitaciones del sistema en su combinación con la placa Sense2GoL, de Infineon Technologies.

Palabras clave– Procesado de señales digitales, radar de onda continua, radar FM-CW, efecto Doppler, MongoDB, NoSQL, multithreading, Sense2GoL, Infineon Technologies.

Abstract– The recollection and treatment of data from sensors is the basis of any information gathering system. In this work, a design is made for a platform that performs the complete cycle of processing a radar I/Q signal, from obtaining it to extracting information and saving it, using tools such as a NoSQL database and multithreading. The design is implemented in MATLAB and both real-time processing and post-processing are allowed. Finally, tests are carried out with the platform in order to characterize the main applications and limitations of the system in combination with the Sense2GoL board, from Infineon Technologies.

Keywords– Digital signal processing, CW Radar, FM-CW Radar, Doppler effect, MongoDB, NoSQL, multithreading, Sense2GoL, Infineon Technologies

1 Introducción

1.1. Motivación y objetivos

El tratamiento y procesado de los datos obtenidos desde cualquier tipo de sensor, ya sea óptico, térmico o un radar, es igual o más importante que la obtención de datos en si. Una buena plataforma de obtención, almacenamiento y procesado de datos facilita la interpretación de todo tipo de información, mejora la eficiencia en todos los procesos de desarrollo e investigación y otorga una base sólida, robusta y consistente desde la que iniciar cualquier proyecto.

Los radares, utilizados en infinidad de entornos que abarcan desde la automoción hasta material médico pasando por tecnología militar, son el tipo de sensores en el que nos centraremos en este estudio. Estos sensores generan enormes cantidades de datos por segundo y, en la mayoría de aplicaciones, el tiempo de procesado de los datos e interpretación de la información resulta critico.

Pensemos en un detector de colisión de un coche que circula a 120km/h. Si nuestro radar está produciendo unas 6000 muestras por segundo (muestreo a 3000Hz de una señal compleja), sería fácil que una mala plataforma no se adecuase a este volumen de información y esta podría descartar datos con el fin de no sobrecargar el sistema, o incluso colgarse. Obviar un par de segundos implicaría recorrer más de 50 metros a ciegas y causar un accidente.

En este trabajo se detallará la implementación de una plataforma que realiza el tratamiento integral de los datos procedentes de un radar, es decir, los obtiene, los procesa y los guarda. Además, el proyecto se complementa con una serie de pruebas de campo que sirven tanto para caracterizar el rendimiento de la plataforma como del radar utilizado, el Sense2GoL de Infineon Technologies.

Los principales objetivos de esta plataforma son obtener y procesar datos RAW extraídos directamente del radar, im-

• E-mail de contacto: jordi.munozar@e-campus.uab.cat
• Mención realizada: Tecnologies de la Informació
• Trabajo tutorizado por: Pedro De Paco Sánchez (Departamento de telecomunicación e ingeniería de sistemas)
• Curso 2019/20

portar y exportar todo tipo de datos e información de manera rápida y eficiente de una base de datos, y proporcionar las herramientas necesarias para el post-procesado de los datos. Además el diseño ha sido ideado para que se pueda adaptar la plataforma de manera simple y directa a múltiples entornos que requieran cambios en el lenguaje de programación o en la placa desde la que se obtienen los datos. A la hora de cerrar el proyecto, uno de los objetivos adicionales ha sido el de caracterizar por completo el sistema implementado, aportando datos tanto cuantitativos como cualitativos del rendimiento de la plataforma.

1.2. Estructura del documento

El documento se ha estructurado en cuatro partes principales. En la primera parte se hace una introducción a la teoría de radares, comentando cuales son los principales tipos, sus usos y las técnicas en las que basan su funcionamiento. Seguidamente, en la segunda parte, se presenta el diseño teórico de la plataforma, detallando sus funcionalidades y estructura. Al final se comentan todos los cambios que se han tenido que realizar a causa de la pandemia Covid-19.

En tercer lugar se detalla la implementación de la plataforma, explicando sus distintas herramientas, como se han implementado y se hace un profiling de la aplicación para analizar el rendimiento de la plataforma. Finalmente, en la última parte, se hacen una serie de pruebas en un entorno doméstico para caracterizar el rendimiento del radar Sense2GoL de Infineon Technologies e identificar posibles limitaciones y/o usos de la placa y la plataforma.

2 TEORÍA DE RADARES

Un radar [1], del acrónimo inglés *radio detection and ranging*, es un sistema que utiliza la transmisión y recepción de ondas electromagnéticas para obtener información de un objeto como su posición o velocidad.

El principio básico detrás del funcionamiento del radar es la emisión de una señal cuyas características son conocidas. Esta señal, irradiada al espacio por una antena, rebota cuando alcanza un objeto y, mediante antenas receptoras, se capta este rebote o eco. Mediante el análisis de la señal recibida se puede obtener información sobre dicho objeto, como puede ser su distancia al radar, su velocidad, su dirección de movimiento o su posición relativa.

Con el objetivo de obtener dicha información se utilizan técnicas de modulación o de pulsación de señales, y también el Efecto Doppler. Según el tipo de onda, podemos distinguir los siguientes tipos de radares:

- De onda continua: el más básico de todos. Consiste en la transmisión continua de una onda a una frecuencia determinada y la recepción de esa misma onda. Permite calcular la velocidad y la moción de movimiento de un blanco (si se acerca o se aleja).

- De onda continua modulada o FM-CW: añade la modulación de la frecuencia de transmisión para permitir el calculo de la distancia del blanco.

- Pulsado: se basa en la transmisión de trenes de pulsos que permiten hacer marcas temporales y calcular únicamente el rango de un blanco.

- Pulsado Doppler: combina la capacidad de cálculo del rango mediante pulsos y la capacidad de cálculo de la velocidad a partir del Efecto Doppler.

2.1. Radar de onda continua

Los radares de onda continua, tal y como su nombre indica, transmiten una señal sinusoidal de alta frecuencia sin modular de manera ininterrumpida, de forma que se recibe continuamente un eco y este se procesa para extraer información. Este tipo de radares no tienen la capacidad de calcular el rango al que se encuentra un blanco y, por tanto, sus aplicaciones se limitan a la detección de movimiento y cálculo de la velocidad mediante el uso del Efecto Doppler.

Las ventajas de este tipo de radar radican en su sencillez. La implementación de los radares de onda continua se puede realizar en placas pequeñas, ligeras y baratas. Si lo comparamos con los radares pulsados, extraer información a partir de los datos recibidos es más sencillo y funciona mejor en rangos cortos. Además, utilizar una sola frecuencia de transmisión permite tener un receptor con un ancho de banda pequeño y la capacidad de detectar blancos esta determinada por la potencia media del eco recibido.

Las principales desventajas, como ya se puede intuir por su sencillez, es que no es posible obtener datos sobre blancos múltiples o calcular el rango. Además es común que el aislamiento entre las antenas de transmisión y recepción sea bastante pobre.

2.1.1. Efecto Doppler

El efecto Doppler [2] fue definido por primera vez en 1842 por el físico austriaco Christian Doppler. Este fenómeno, utilizado a día de hoy en una gran de cantidad entornos como la astronomía o la navegación, se basa en el cambio en la frecuencia de una señal transmitida de una fuente al rebotar en un blanco en movimiento.

La situación más común de observación del Efecto Doppler es el ruido de un coche o de una sirena al pasar por nuestro lado. Cuando una sirena se acerca hacia nosotros, el sonido de la misma se escucha de manera aguda y, a medida que se vuelve a alejar, el sonido cambia a un tono más grave.

El ejemplo del cambio de tono de una sirena se traduce, de manera teórica, en que si el blanco se acerca hacia el radar, la frecuencia de la señal recibida será superior a la transmitida y si el blanco se aleja del radar, la frecuencia de la señal será inferior a la transmitida.

La formula para calcular la frecuencia Doppler es:

$$f_d = \frac{2 \cdot f_{TX} \cdot v}{c} \tag{1}$$

dónde f_d es la frecuencia Doppler, f_{TX} la frecuencia de transmisión, v la velocidad del blanco y c la velocidad de la luz.

2.1.2. Calculo de la velocidad y moción de movimiento

El radar captura las componentes en fase y cuadratura de la señal que se ha reflejado en el objetivo y, para obtener la frecuencia Doppler, debemos realizar un análisis frecuencial de dichas componentes. El cálculo es tan sencillo como convertir dichas componentes a números complejos, eliminar la componente continua de la señal y, finalmente, realizar la FFT.

Si visualizamos la señal en el dominio frecuencial observaremos un pico a una determinada frecuencia y, teniendo en cuenta que conocemos la frecuencia de transmisión, basta con comparar ambas frecuencias para obtener el *Doppler Shift* o cambio de frecuencia.

A partir del cambio de frecuencia y utilizando la Ecuación (1), el calculo de la velocidad es prácticamente trivial. Así mismo, como ya hemos mencionado con anterioridad, si el cambio de frecuencia ha supuesto una disminución de la misma, significará que el blanco se aleja del observador y viceversa.

En la Fig.1 observamos la relación existente entre el cambio de frecuencia y la velocidad de un blanco.

Velocidad (km/h)	0.5	1	5	10	15	20	25	30	35
Doppler shift (Hz)	22	44	223	445	668	891	1114	1337	1559

Fig. 1: Doppler shift.

2.1.3. Aplicaciones

Las aplicaciones de este tipo de radar son múltiples, y muy probablemente la más común y conocida de todas ellas sea su utilización en los radares de velocidad policiales, inventados por John L. Barker Sr. y Ben Midlock [3] durante la Segunda Guerra Mundial. La sencillez del diseño permite crear radares suficientemente pequeños y ligeros como para ser manejados por una sola persona utilizando algún tipo de soporte como una pistola o un trípode. Dichos radares suelen operar en la banda K (18 a 27 GHz) y K_a (27 a 40 GHz).

Cabe destacar las aplicaciones no policiales de la medición de la velocidad de vehículos. Son muchos los núcleos urbanos en cuyas entradas o cerca de pasos de peatones se colocan señales luminosas que indican la velocidad en tiempo real de los vehículos. Un claro ejemplo es el Radar Aforador DTS 600 de DENSL [4]. Este dispositivo es un radar de onda continua que utiliza el efecto Doppler para realizar los cálculos y trabaja en la banda de los 24 GHz. A parte de actuar como disuasorio de la velocidad en zonas sensibles, el radar permite el almacenamiento de los datos de paso para así facilitar la toma de decisiones sobre la gestión del trafico.

Los radares de onda continua también son utilizados como sensores de movimiento para aplicaciones de seguridad e iluminación y, haciendo gala de esa sencillez, también son usados como sensores para puertas automáticas. La capacidad de obtener la moción del movimiento también es útil para aplicaciones médicas como monitorizar de manera no invasiva la actividad respiratoria o el pulso de los pacientes.

Finalmente, los radares CW también son utilizados comúnmente en aplicaciones militares y actúan como espoletas de proximidad. La amplitud de la señal recibida aumenta a medida que el dispositivo se acerca al blanco y la frecuencia Doppler cambia notablemente una vez sobrepasado el objetivo, momento en el que el misil o proyectil se activa y detona.

2.2. Radar FM-CW

El radar FM-CW [5] es una evolución del radar de onda continua, y añade una modulación a la señal transmitida con el fin de poder realizar una marca temporal y así poder calcular el rango al que se encuentra uno o más blancos. La modulación de la señal, es decir, la manera en la que varía la frecuencia durante un periodo de tiempo, puede producirse con diferentes formas de onda, como por ejemplo sinusoidales, triangulares, cuadradas o de dientes de sierra, siendo esta última la más utilizada.

Entre sus principales ventajas están el bajo consumo de potencia para la transmisión, su arquitectura superheterodina que proporciona buena sensibilidad y estabilidad, y ofrece un mayor ancho de banda que el radar de onda continua sin modular. Sin embargo, la construcción de este radar es más cara que la de uno pulsado, la atenuación por propagación es un factor critico, y puede sufrir de interferencias de otros sistemas de radio cercanos.

Los radares FM-CW también son conocidos como altímetros, ya que son utilizados en aeronaves para medir la distancia relativa al suelo durante las maniobras de aterrizaje. A parte de sus aplicaciones en aeronáutica, este tipo de radar es muy común en soluciones del sector de la automoción e industria.

2.2.1. Cálculo del rango

El cálculo del rango en un radar FM-CW se realiza a partir del delay entre la señal transmitida y la señal recibida. Aun así, no es el delay temporal el utilizado en los sistemas, sino que se mezcla la señal recibida con la transmitida para generar una señal a frecuencia intermedia. Sobre esta nueva señal se realiza la Fast Fourier Transform (FFT) para obtener la llamada *beat frequency*, que no es mas que la diferencia entre frecuencias (ver Fig.2).

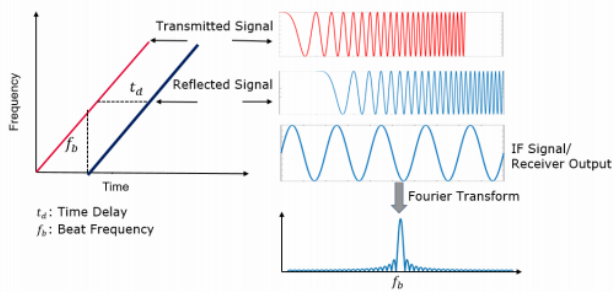

Fig. 2: Principio del radar FM-CW [7].

A partir de esta frecuencia y la ecuación (2) es posible calcular el rango de uno o varios blancos.

$$R = \frac{c \cdot T_c \cdot f_b}{2 \cdot BW} \tag{2}$$

dónde R es el rango (m), C la velocidad de la luz, T_c es el tiempo de chirp (s), f_b es la *beat frequency* (Hz) y BW el ancho de banda (Hz).

3 DISEÑO DEL SISTEMA

El diseño del sistema ha ido variando a medida que se ha ido desarrollando el proyecto, y se ha adaptado a las circunstancias provocadas por la Covid-19. En esta sección se pretende mostrar el concepto inicial del sistema, ideado para el Position2Go (ver Apéndice 4), la base de datos que se ha escogido y la propuesta para desarrollar la plataforma en C/C++. Además, una vez presentado el concepto, se identifican los aspectos que han sido imposibles de realizar y los cambios aplicados para permitir el desarrollando el proyecto.

3.1. Concepto del sistema y diseño en C/C++

El objetivo de la plataforma era claro: adquirir y ordenar adecuadamente los datos para que su procesado tanto en tiempo real como posterior fuera una tarea sencilla. Para afrontar el problema se optó por el conocido paradigma de divide y vencerás, desglosado en este apartado.

La parte de adquirir los datos fue prácticamente trivial gracias a las herramientas proporcionadas por Infineon Technologies. Junto a la placa, el fabricante proporciona una librería de comunicación que permite obtener datos de la placa mediante funciones. En el caso del Position2Go, la comunicación es específica para cada módulo de la placa, es decir, si queremos modificar el numero de chirps que recibimos en cada paquete deberemos comunicarnos con el modulo de configuración y si queremos recibir información sobre los objetivos detectados deberemos comunicarnos con otro modulo. Toda la información, tanto de las funciones como de las estructuras para recibir los datos, está detallada en el manual de la API de Infineon [6].

Solucionada de manera sencilla la parte de la adquisición, había que centrarse en el procesado de los datos, un problema más complejo de lo que aparentaba. A la hora de idear una solución para el procesado se plantearon una serie de cuestiones relevantes para el diseño: ¿Queríamos una aplicación estrictamente en tiempo real o se permitía un pequeño delay? ¿Usaríamos los datos en otras plataformas y/o aplicaciones? ¿La carga computacional sería demasiado alta? En caso de que fuese necesario, ¿cómo se guardarían los datos para su uso *a posteriori*?

La respuesta a la gran mayoría de esas cuestiones era, y sigue siendo, incierta, ya que esta plataforma esta ideada para múltiples usos. Es evidente entonces que la plataforma debía perseguir la flexibilidad y adaptabilidad a los múltiples usos que se le deseasen en el futuro. Para ello, se ideó un sistema que permite el procesado a tiempo real de los datos y, a su vez, guarda la información en una base de datos para su uso posterior a su adquisición.

A continuación se listan las principales características que debía tener el sistema, planteadas desde el punto de vista teórico ya que nunca llegó a ser implementada como tal:

- Procesado en tiempo real: Los datos obtenidos de la placa se iban a guardar en variables locales del código y para extraer información a través de funciones que implementarían los cálculos presentados en la sección 2. También se habilitaría el almacenaje de esas variables locales en la base de datos, además de cualquier dato o información que pueda ser representada en formato JSON.
- Procesado en MATLAB: MATLAB, la placa y la base de datos utilizada son compatibles, tanto a nivel de obtención de datos como de procesado. MATLAB surgió como alternativa para obtener datos de la placa, procesarlos y guardarlos, además de como herramienta de análisis.
- Procesado en otras aplicaciones: la base de datos es compatible con una gran cantidad de lenguajes de programación, y las velocidades de escritura y lectura son muy altas, de manera que se abrieron múltiples posibilidades de crear una interfaz con un lenguaje de programación que permita la visualización de gráficos como, por ejemplo, Python.

El uso de una base de datos solucionaba gran parte de las cuestiones planteadas anteriormente, o como mínimo daba opción a que la plataforma sea adaptable y escalable. Aún así, su uso planteó nuevas preguntas. Es ampliamente conocido que en bases de datos tradicionales los tiempos de lectura y escritura pueden provocar cierto delay, especialmente si el volumen de datos es grande. Si a ello le sumamos el procesado en tiempo real realizando operaciones de gran carga computacional como, por ejemplo, la Transformada Discreta de Fourier, el sistema propuesto podía sufrir una sobrecarga. Es por ello que se eligió una base de datos NoSQL (ver Apéndice 1).

La solución para poder realizar un procesado en tiempo real y guardar esa misma información en una base de datos de manera eficiente venía dada por una herramienta del propio lenguaje de programación: los *threads* o hilos. Programar con hilos indica al procesador que trozos de código debe ejecutar de manera simultanea o paralela, utiliza una memoria compartida entre hilos y, en lineas generales, utiliza menos recursos que otros sistemas de computo paralelo como el *fork()* (ver Apéndice 3).

Para ayudar a comprender el diseño, en la Fig.3 se puede observar un diagrama que representa la idea del sistema.

Con el propósito de facilitar su comprensión, a continuación se explica como se debía llevar a cabo la adquisición, procesado y obtención de información del radar.

Hilo 1:

- En primer lugar se solicitan los datos RAW (componentes en fase y cuadratura de la señal recibida por el radar, de ambas antenas) mediante las funciones de la librería de comunicación en C.
- En segundo lugar se organizan los datos recibidos en estructuras preestablecidas. Se crean variables y vectores en los cuales se registran la antena de recepción, el numero de muestras, el numero de chirps por dataframe, numero de muestras por chirp y componentes I/Q.
- En tercer lugar se añade el dataframe creado anteriormente en una cola FIFO.
- En último lugar, si se ha definido, se procesan los datos recibidos con el fin de obtener información en tiempo real de la señal.

Hilo 2: El segundo hilo es un bucle infinito en el cual se comprueba si hay algún elemento en la cola FIFO mencionada en el hilo 1. En el caso de encontrar un elemento, este se identifica mediante un ID y se guarda en la base de datos.

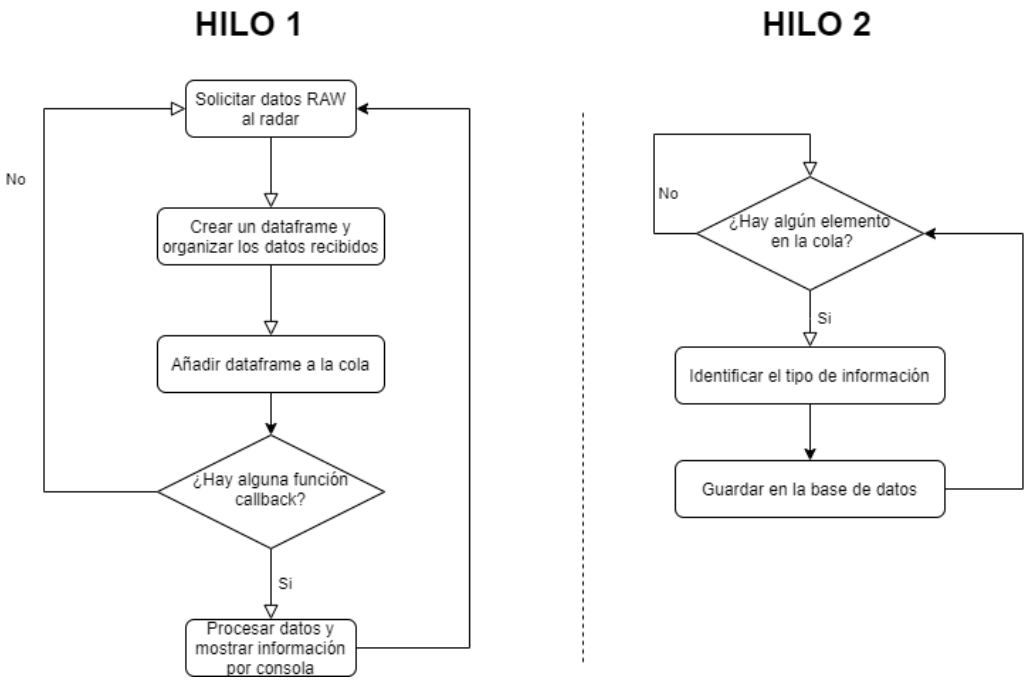

Fig. 3: Diagrama de flujo del núcleo del procesado de datos.

3.2. MongoDB

MongoDB es un sistema de base de datos NoSQL [8] que se aleja del sistema tradicional de las tablas relacionales y guarda los datos en documentos o archivos. MongoDB es considerado como uno de los líderes en el entorno de las bases de datos NoSQL y empresas como Ebay, Adobe, Google, Sega, EA, Verizon o el gobierno de Reino Unido utilizan este sistema. Debido a ello, MongoDB ofrece un amplio servicio de atención al cliente, documentación y foros en linea.

Esta base de datos reúne todas las ventajas de las bases de datos NoSQL, de manera que no es necesario definir ningún esquema previo, la estructura de los objetos es muy clara, la capacidad de escalar es grande y está preparada para que los accesos a los datos sean extremadamente rápidos. Además MongoDB permite indexar a partir de cualquier atributo, las actualizaciones son rápidas y permite realizar consultas complejas.

En cuanto a su uso en nuestra plataforma, esta base de datos permite guardar los datos obtenidos del radar de manera ordenada y rápida para un posterior procesado o análisis. Solo por mencionar algunos ejemplos, MongoDB es compatible con Python o Javascript para realizar un entorno visual con gráficos de los datos obtenidos, y también es compatible con MATLAB para realizar un análisis de la señal registrada.

Además, gracias a la rapidez de lectura y escritura, se podría realizar un pequeño sistema en el que se obtienen y guardan los datos obtenidos del radar utilizando esta plataforma y posteriormente se podrían representar gráficamente utilizando Python con un pequeño retraso prácticamente imperceptible, lo que daría la sensación de un GUI que muestra información en tiempo real. Finalmente cabe destacar que MongoDB implementa un entorno gráfico desde el que ver y gestionar las diferentes bases de datos y la información contenida en ellas (ver Apéndice 2).

Como ya se ha mencionado, el formato de los datos en MongoDB es el BSON o Binary JSON [9]. Este formato consiste en una lista de datos formada por parejas nombre-valor y soporta una gran cantidad de tipos de datos como cadenas de caracteres, enteros, fechas, booleanos, objetos BSON o vectores. En comparación con el formato JSON, el BSON tiene una velocidad de almacenamiento superior y, en lineas generales, es más eficiente. Además, no es necesario predefinir estas estructuras. Esto significa que podemos guardar cualquier tipo de dato estructurado en formato BSON en una colección de datos.

3.2.1. Importar y exportar datos

Importar y exportar datos de MongoDB depende del tipo de lenguaje de programación que se utilice. En este caso se utilizará MATLAB como ejemplo, pero los pasos son extrapolables a cualquier otro entorno. En la documentación de MongoDB [10] se especifican las funciones utilizadas en cada lenguaje de programación.

En primer lugar deberemos establecer una conexión con la base de datos utilizando la función *conn()*.

```
conn = conn(server,port,dbname)
```

. En la función especificaremos el servidor, el puerto y el nombre de la base de datos. En el caso de tener MongoDB instalado en el mismo dispositivo en el que tenemos la plataforma, los valores del servidor y el puerto serán *localhost:27017*.

Una vez establecida la conexión, podemos guardar estructuras con la siguiente función:

```
ntdcs = insert(conn,collection,tdcs)
```

. En los parámetros de la función especificaremos el nombre de la conexión que devuelve la función *conn()*, el nombre de la colección y los datos.

Finalmente, si queremos importar objetos al entorno, utilizaremos la siguiente función:

```
documents = find(conn,collection)
```

En ella especificaremos la conexión y el nombre de la colección a importar y recibiremos todos los documentos que se contengan en ella.

3.3. Cambios a causa del Covid-19

A causa de la pandemia del SARS-COV-2 o Covid-19 que afectó a España especialmente entre los meses de marzo y mayo de 2020, el desarrollo de este trabajo se vio afectado en diversos aspectos que obligaron a realizar cambios y adaptaciones en el diseño ideado.

En este apartado se pretende ilustrar los efectos de la pandemia sobre el proyecto y los cambios que se propusieron para asegurar su viabilidad.

3.3.1. Cambios de hardware

La limitación de la libre circulación de personas establecida por el Real Decreto 463/2020, de 14 de marzo, supuso el cierre de las instituciones universitarias y sus laboratorios. Los cambios en el hardware, motivados por la imposibilidad de acceder al material de trabajo que se encontraba en los laboratorios, fueron significativos.

El kit de desarrollo Position2Go, un radar FM-CW, para el que se había ideado principalmente el sistema, tuvo que cambiar al kit de desarrollo Sense2GoL, un radar Doppler (CW), al cual sí se tuvo acceso durante el periodo de confinamiento. Dichos radares tienen funciones y niveles de complejidad muy diferentes (ver Apéndice 4) y, por tanto, el proyecto tuvo que ser adaptado a las nuevas características de la placa.

3.3.2. Cambios de software

Los cambios de software, motivados por la sustitución del kit de desarrollo, implicaron un modificación tanto en el lenguaje de programación como en la plataforma de desarrollo.

El kit de desarrollo Position2Go incorpora librerías de comunicación para C/C++ y MATLAB mientras que el kit de desarrollo Sense2GoL solo incorpora una librería de comunicación para MATLAB. En consecuencia, el lenguaje de programación cambió de C/C++ a MATLAB y la plataforma de desarrollo cambió de Visual Studio 2019 al propio MATLAB.

La utilización de MATLAB como lenguaje de programación, a parte de introducir cambios en el diseño del sistema, introdujo una serie de limitaciones inherentes del lenguaje. El lenguaje C/C++, compilado y orientado a objeto, es de un nivel más bajo que MATLAB y, por tanto, operaciones sencillas en C como el multithreading podían resultar mucho más complicadas o incluso imposibles de realizar.

3.3.3. Cambios en el concepto del sistema y objetivos

Pasar de un radar FM-CW a uno CW disminuyó considerablemente la complejidad del sistema tanto a nivel de datos como de cómputo y, a su vez, el cambio de C/C++ a MATLAB limitó mucho las opciones de desarrollo.

La hipótesis que se propuso es que todos los cambios obligados hacían que la plataforma inicialmente ideada fuese en gran parte inviable o, como mínimo, muy ineficiente. Por tanto, el principal objetivo del proyecto pasó a desarrollar una plataforma que se asemejase lo máximo posible al diseño inicial y caracterizar por completo las capacidades de la misma con el fin de valorar realmente su utilidad para futuros trabajos.

4 PLATAFORMA MATLAB

Como ya se ha mencionado, tras la pandemia, el objetivo real de esta plataforma realmente no radicaba en que fuese plenamente funcional para futuros proyectos sino que se utilizaría para demostrar y analizar las principales limitaciones y desventajas de la combinación de MATLAB y la placa Sense2GoL.

La plataforma realizada implementa las siguientes funcionalidades:

- Visualización en tiempo real de la velocidad y la moción de movimiento de un blanco.
- Almacenamiento de datos en tiempo real a MongoDB.
- Capacidad de importar grabaciones a MATLAB.
- Análisis de grabaciones completas en MATLAB.
- Capacidad de aislar y analizar *dataframes* de una grabación.

Para implementar dichas funcionalidades se optó por separar el procesado en tiempo real del postprocesado, detallados en los siguientes apartados.

4.1. Procesado en tiempo real

A través de un script se gestiona todo el procesado en tiempo real. El código habilita dos aspectos fundamentales de la plataforma e implementa todas las funcionalidades consideradas como procesado en tiempo real.

El script arranca por completo el sistema del radar, realizando todos los pasos necesarios para establecer una conexión estable con la placa. Durante el arranque también establece una conexión con la base de datos MongoDB y declara las variables en las cuales se guardarán los datos obtenidos. Finalmente el código entra dentro de un bucle infinito que contiene realmente el cuerpo de la plataforma y que se detallará más adelante.

Las variables mencionadas son arrays de estructura, un tipo de datos que agrupa los datos relacionados mediante contenedores denominados campos. La utilización de este tipo de datos es importante ya que su conversión a formato BSON es directa.

Existen dos variables para controlar el flujo del programa. Dichas variables son del tipo booleano y controlan los siguientes aspectos:

- RECORDING_ENABLED - Controla el acceso a la parte del código que almacena la información en MongoDB para su posterior análisis.
- LIVE_VIEW_ENABLED - Controla el acceso a la parte del código que representa en tiempo real y de forma visual los datos obtenidos del radar.

Como ya se ha mencionado, una vez establecidas todas las conexiones necesarias, el programa entra en un bucle infinito. Dentro de cada iteración del mismo se solicita al radar un *dataframe* mediante la función *radar_get_frame* y se obtienen dos vectores de 128 muestras que contienen las componentes en fase y cuadratura de la señal respectivamente.

Si la variable RECORDING_ENABLED está activada, los datos obtenidos se almacenan en los arrays de estructura y se guardan en MongoDB mediante la función *insert()*. Cada sesión de grabación (cada vez que se ejecuta el script) crea una nueva colección que tiene como nombre la fecha exacta en la que empieza la obtención de datos.

Si la variable LIVE_VIEW_ENABLED está activada, se realiza la Transformada de Fourier de la señal y se obtienen datos como el pico de frecuencia, la velocidad del blanco o su dirección de movimiento (si se aleja o si se acerca).

4.2. Postprocesado de señales

La parte del postprocesado de la señal ha sido implementada también con un script pero, a diferencia del procesado en tiempo real, el código está más orientado a proporcionar las herramientas necesarias al usuario que a realizar una función en concreto.

MATLAB es una herramienta de análisis muy potente y, por tanto, limitar la capacidad de la plataforma a unas pocas funciones no se ha considerado apropiado. Por tanto, el objetivo de esta parte del código se fundamenta en incorporar toda la información registrada durante las grabaciones al entorno MATLAB.

Por un lado se ha habilitado un menú que permite realizar tres tipos de acciones:

- Importar una señal al entorno MATLAB
- Analizar una señal del entorno MATLAB identificando la frecuencia del tono, la velocidad y la moción de movimiento.
- Representar gráficamente la velocidad calculada durante una grabación

Por otro lado también hay algunas funciones que sirven como herramienta para el usuario, como por ejemplo una función para analizar de manera individual un *dataframe* de una grabación previamente importada. Todo ello está detallado dentro del propio código en MATLAB.

4.3. Rendimiento y profiling de la aplicación

A lo largo de las distintas pruebas realizadas con las funciones desarrolladas, se observó que la visualización de la señal no se actualizaba a la velocidad a la que llegaban los *dataframes* y era imposible conseguir una sensación de fluidez. Además la solicitud de *dataframes* a la placa se veía ralentizada tanto por el almacenamiento de los datos como por el procesado para mostrar la señal en tiempo real.

Para caracterizar dicho retraso tanto en la visualización de la información por pantalla como el tiempo que pasa entre solicitudes de *dataframes* se realizó un *profiling* de la aplicación, concretamente del *live recording*.

Profiling [11] es una manera de medir el tiempo que tarda el código en ejecutarse de manera que resulta más sencillo encontrar cuellos de botella o partes de código que deben optimizarse. Esta herramienta también es útil para identificar que trozos de código nunca son ejecutados. MATLAB ofrece un Profiler dentro de su plataforma muy intuitivo y gráfico, de manera que la interpretación de los resultados es muy sencilla.

En total se ha realizado una simulación de 24.665 segundos, en los cuales se han realizado 47 iteraciones del bucle de ejecución. En la Fig.4 se pueden observar algunos de los tiempos de ejecución más relevantes.

Función	Llamadas	Tiempo total (s)	Tiempo propio (s)
live_recording	1	24.665	0.672
live_view	47	20.891	3.091
mongo>mongo.insert	47	0.846	0.163
radar_get_frame	48	0.713	0.085

Fig. 4: Profiling de la aplicación.

La visualización de la señal por pantalla mediante una figura de MATLAB ocupa un 84 % del tiempo de ejecución mientras que el almacenamiento de datos en MongoDB y la solicitud de datos a la placa solo ocupan un 3.4 % y un 2.8 % respectivamente.

Si tenemos en cuenta el numero de iteraciones, guardar un *dataframe* en la base de datos tarda de media 18ms mientras que mostrar la información introduce un delay de 444ms. El radar Sense2GoL muestrea a una frecuencia de 3000Hz y, por tanto, podemos concluir que el *live_view* provoca que se pierdan más de 1300 muestras por iteración mientras que guardar en MongoDB solo provoca la pérdida de 54 muestras, una cifra mucho más aceptable.

5 Caracterización de la placa Sense2GoL

La utilización de la placa Sense2GoL en el lugar de la placa Position2Go supuso una gran cantidad de limitaciones, tanto funcionales como operacionales. Si en la sección anterior hemos visto las limitaciones de utilizar MATLAB, en este capítulo se pretende ilustrar las principales limitaciones de la placa Sense2GoL en la detección de blancos de especial interés, como personas o vehículos.

5.1. Metodología y entorno

Dadas las limitaciones de acceso a laboratorios científicos o entornos controlados a causa de la pandemia del SARS-COV-2, el entorno para todas las pruebas de detección y seguimiento de blancos humanos fue casero: un pasillo de 6 metros de largo ubicado en una hogar.

La metodología seguida fue la misma para las distintas pruebas:

- En primer lugar se realizó una descripción concreta entorno, especificando la altura y ángulo del radar, el numero de blancos que estaban en el entorno y su dirección de movimiento. Todo ello se acompañó en la medida de lo posible de diagramas, fotografías y/o esquemas que ayuden a comprender el entorno.
- En segundo lugar se realizó una grabación completa de la señal durante la realización de la prueba, utilizando el algoritmo descrito en sección 4.1

con las opciones RECORDING_ENABLED y LIVE_VIEW_ENABLED activadas.

- Durante la grabación se tomaron notas de las observaciones realizadas tanto en pantalla como en los LEDs del Sense2GoL.
- Finalmente, se importaron las sesiones registradas a MATLAB con el fin de hallar los *dataframes* más importantes y analizar con más detenimiento las observaciones realizadas en tiempo real.

5.2. Detección de un blanco humano

La detección de un blanco humano en las condiciones que se plantean con el radar Sense2GoL era una tarea extremadamente sencilla, pero podía resultar útil para evaluar el rendimiento básico de la parte de postprocesado de la plataforma y también proporcionaba una base desde la que comparar los resultados.

El radar se situó centrado a una altura de 100cm. A la misma altura horizontal, separado por unos 50cm, se situó el blanco humano, que realizó un recorrido de 6 metros en paralelo al radar (ver Apéndice 5, Fig.13).

En la Fig.5 podemos ver la representación de un *dataframe* antes de empezar la prueba, sin ningún objeto en movimiento delante del radar. No se observa ningún pico de frecuencia, de manera que el resultado es el esperado.

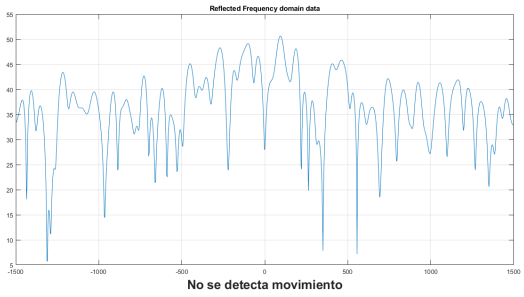

Fig. 5: Dataframe obtenido sin ningún blanco en movimiento.

En la Fig.6 observamos un *dataframe* capturado durante el movimiento de ida del blanco. En él se observa un pico de frecuencia a -180 Hz, ello se traduce a una velocidad de 1.13 m/s o 4.06 km/h, y que se aleja de la posición del radar.

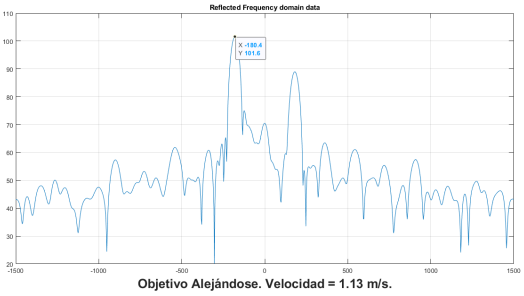

Fig. 6: Dataframe obtenido a partir de un blanco alejándose.

La detección y seguimiento del blanco se realizó correctamente tanto para el viaje de ida como para el viaje de vuelta, y a lo largo de los 6 metros de recorrido, sin llegar a perder en ningún momento el blanco.

5.3. Detección de múltiples blancos humanos

Como ya se ha tratado brevemente en la sección 2, un radar Doppler de onda continua no tiene la capacidad de detectar múltiples blancos a la vez. Esta limitación hace que la placa Sense2GoL no sea útil para aplicaciones que requieran poder diferenciar situaciones en las que se involucran múltiples blancos.

Esta limitación es más que evidente si pensamos en el caso de una habitación inteligente en la que se encienden las luces si se detecta que entra alguien y se apagan si se detecta que sale alguien. En el caso de tener el radar Position2Go, que sí soporta la detección y seguimiento de múltiples blancos, podríamos tener un simple contador de las personas que entran y salen de la habitación y apagar o encender las luces en función de ello. Aparentemente con el Sense2GoL podríamos realizar algo similar simplemente detectando si el blanco se acerca o se aleja de la puerta pero, ¿podremos detectar si entran dos personas a la vez y luego solo sale uno? ¿Seremos capaces de detectar a una persona que sale y a otra que entra en un breve espacio de tiempo? La respuesta es clara: las probabilidades de dejar a alguien a oscuras son muy altas.

Para demostrar esta limitación se realizó de nuevo la prueba anterior pero con dos blancos en lugar de uno. En primer lugar se realizó una prueba con ambos blancos caminando paralelamente en la misma dirección y distanciados 70cm entre si (ver Apéndice 5, Fig. 14). Durante la prueba, a través de los LEDs se observó que, tal y como era de esperar, la placa detectó bien la moción de movimiento e indicaba que no había movimiento cuando ambos blancos se mantenían estáticos. Sin embargo, en ningún momento se pudo saber a que blanco correspondía la velocidad indicada.

Podemos considerar que si ambos blancos caminan de forma paralela en la misma dirección se pueden interpretar como un solo blanco. De ser así, al ser un blanco más grande que un solo humano, la señal recibida será mas clara y los resultados serán incluso mejores que con un solo blanco.

En la Fig.7 observamos uno de los *dataframes* obtenidos durante el transcurso de la prueba, concretamente en el viaje de vuelta hacia el radar. Se observa que se recibe una señal con un *shift* de 177 Hz, que se traduce en una velocidad de 1.11 m/s acercándose del observador.

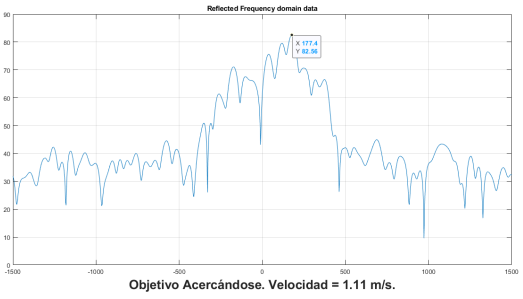

Fig. 7: Dataframe obtenido a partir de dos blancos en paralelo

En segundo lugar se realizó la misma prueba pero con los blancos empezando a caminar desde direcciones opuestas (ver Apéndice 5, Fig.15). Esta vez los resultados son más confusos, tanto los obtenidos por pantalla como la informa-

ción proporcionada por los LEDs.

Por norma general, el radar detecta la velocidad y la moción de movimiento del blanco que se encuentra más cerca del radar, pero tras la repetición de las pruebas se ha comprobado que no siempre se obtienen los mismos resultados. Al obtener resultados inconsistentes, se ha optado por no aportar ningún *dataframe*. En su lugar se aporta la Fig.8, en la que se observa la velocidad de movimiento detectada a lo largo de una de las pruebas. En ella podemos observar como en la primera mitad del movimiento se detecta bien la moción de movimiento y la velocidad del blanco más cercano al radar, pero a partir del cruce entre ambos blancos, los resultados se vuelven inconsistentes.

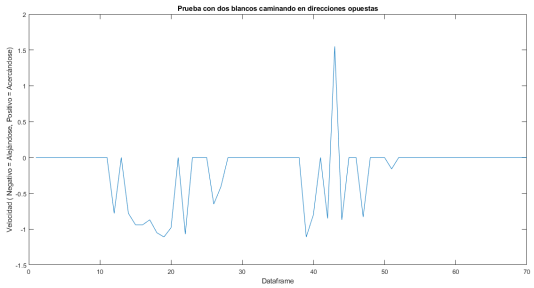

Fig. 8: Velocidad detectada a lo largo de la prueba 3.

5.4. Detección de blancos humanos en ángulo

Tal y como se indica en las especificaciones de la placa Sense2GoL (ver Apéndice 4), esta tiene un ancho de haz de 80ºx29º. La capacidad de detectar blancos en diferentes ángulos, tanto en el plano horizontal como en el vertical, es especialmente importante en aplicaciones practicas. Conseguir que un blanco tenga un ángulo de aproximación cercano a 0º es prácticamente imposible en cualquier aplicación que involucre seres humanos, animales, vehículos, etc.

En las dos primeras pruebas llevadas a cabo en esta sección hemos visto como el experimento incluía un pequeño ángulo horizontal, pero al ser prácticamente imperceptible no es útil para extraer resultados respecto a la detección de blancos en ángulo.

Para la realización de esta prueba, el radar se situó encima de una puerta, a una altura de 220cm y en un ángulo de 45º, y el experimento se realizó de la misma manera que los anteriores: un blanco caminando un total de 6 metros empezando un metro por detrás de la posición del radar (ver Apéndice 5, Fig.16). En total se han hecho 25 iteraciones de esta prueba. Cabe destacar que la realización de esta prueba fue realizada con la opción LIVE_VIEW_ENABLED desactivada.

En 24 de las 25 iteraciones el radar fue incapaz de detectar el blanco a distancias superiores a 400cm, y solo en 2 de las 25 iteraciones el radar no detectó el blanco a menos de 360cm. Es decir, con una probabilidad aproximada del 88 % se detecta un blanco que se acerca a una distancia de entre 360 y 400 centímetros.

5.5. Detección de humanos en perpendicular

Finalmente abordamos la detección de humanos si la placa se encuentra en perpendicular al plano de movimiento del blanco. Dicho ángulo puede ser de utilidad para funcionalidades como puertas automáticas, así que resulta interesante ver el comportamiento y la capacidad de detección de la placa en este entorno.

De la misma manera que en la prueba anterior, se situó el radar a una altura de 220cm y en un ángulo de 90º (ver Apéndice 5, Fig.17 y Fig.18). El experimento se realizó con el blanco empezando a caminar desde una distancia de 6 metros y acercándose al radar desde diferentes ángulos.

Los resultados, resumidos en la siguiente tabla, han sido obtenidos tras repetir la prueba un total de 20 iteraciones para cada ángulo de aproximación y se ha indicado la distancia de detección más repetida.

Ángulo de aproximación	0º	15º	30º	45º
Distancia de detección (cm)	230	140	100	50

Analizando los resultados, vemos que este sensor sería óptimo para una puerta automática de las que se pueden encontrar en las entradas de grandes superficies o en algunos comercios. A partir de un ángulo de aproximación de 35º-40º podríamos empezar a considerar que la distancia de detección no es suficiente para que la puerta se abra sin que la persona tenga que frenarse. Aun así, ¿quién no se ha peleado alguna vez con un sensor porque la puerta no se abría? Obviamente no estamos delante de un dispositivo infalible, pero tras obtener estos resultados podemos considerar que los posibles errores de la placa en un sistema de estas características estarían dentro de lo aceptable.

6 CONCLUSIONES

Pese a la complejidad de adaptar y rediseñar el proyecto a mitad de su realización a causa de la pandemia del SARS-COV-2, las conclusiones que se pueden extraer de todo este trabajo son amplias y especialmente útiles para el desarrollo de futuros trabajos sobre la materia. En esta sección se analizará tanto el rendimiento de la plataforma, los inconvenientes y problemas de su utilización y, finalmente, se repasarán algunos de los posibles usos de esta plataforma.

6.1. Rendimiento del sistema

El rendimiento del sistema implementado ha dado una de cal y otra de arena respecto a lo esperado.

Por un lado, MATLAB ha sido un verdadero quebradero de cabeza. El intento de implementar algo similar al *multithreading* fue un fracaso absoluto dada la complejidad del mismo, la ejecución del programa era lenta y cualquier detalle que en C/C++ se solucionaba con una linea de código fue toda una aventura. La eficiencia de implementar una plataforma de este estilo en MATLAB es prácticamente nula si lo comparamos con una implementación en C/C++.

Por experiencia propia, en C/C++ se puede procesar una señal de cuatro canales muestreada a 16KHz, realizar la FFT de *frames* de 2048 muestras, implementar sistemas de filtrado y procesado extra de la señal para eliminar interferencias no deseadas en un simple módulo de 2 cores y 512MB de RAM y, con todo esto, obtener una frecuencia

de actualización superior a 30 FPS. En MATLAB, con una carga de trabajo muchísimo menor y un equipo mucho más potente, hemos obtenido una frecuencia de actualización sobre los 2/3 FPS.

Por otro lado, la utilización de MongoDB como base de datos ha sido un rayo de luz sobre todo el proyecto. Su utilización es extremadamente sencilla, intuitiva y la documentación es extensa. Había ciertas dudas sobre la utilización de cualquier tipo de base de datos en el sistema dada la percepción que ello añadiría retrasos a múltiples niveles. Sin embargo, las pruebas realizadas en MATLAB han demostrado que los tiempos de lectura y escritura de datos son suficientemente rápidos como para considerar el uso de MongoDB en cualquier implementación futura de una plataforma con similares objetivos.

MongoDB ha demostrado ser lo suficientemente rápida como para servir como punto intermedio entre la obtención y procesado de datos en C/C++, utilizando las librerias de comunicación de Infineon, y la visualización de la información obtenida de manera visual utilizando cualquier lenguaje que lo permita, como puede ser Python.

6.2. Utilidades, inconvenientes y problemas del sistema

Si el cambio de MATLAB a C/C++ ha supuesto muchos problemas a nivel de implementación, el cambio de la placa Position2Go a la placa Sense2GoL ha supuesto todavía más problemas a nivel de concepto. Dichos problemas no provenían de la complejidad de la placa, al contrario, provenían de su simpleza. El diseño inicial fue ideado para gestionar todo tipo de datos provenientes de los múltiples módulos de la placa Position2Go. Posiciones relativas de blancos, velocidades, ángulos, una señal I/Q más rica en información, etc. Todo en su conjunto daba para una gran variedad de pruebas y experimentos a realizar durante el desarrollo del trabajo. Sin embargo, todo esto se quedaba demasiado grande para la placa Sense2GoL.

Simplificar el diseño del sistema ha supuesto tener que explorar otras opciones para completar el trabajo y al final las aportaciones teóricas han cobrado protagonismo frente a las pruebas prácticas. La plataforma ha quedado en un mero concepto de lo que pudo ser aunque sí se han llegado a buenas conclusiones sobre su rendimiento.

Pese a su sencillez, la placa Sense2GoL puede ser útil en múltiples entornos. Se ha demostrado que la detección de blancos en ángulos pronunciados es buena, de manera que esta placa puede actuar como sensor para abrir puertas. Con una placa Arduino y comunicación via UART se podría fabricar un pequeño sensor que al detectar movimiento, sea del tipo que sea, abra automáticamente una puerta. De la misma manera, este dispositivo puede funcionar correctamente en cualquier tarea basada en la detección del nivel más básico (hay movimiento o no). Estamos hablando de aplicaciones des seguridad, iluminación simple, casas inteligentes, etc.

Aunque el trabajo realizado no ha sido el que se había planteado en un principio, dadas las circunstancias, los resultados y conclusiones obtenidas son una buena base desde la que empezar futuros proyectos y llevar a cabo la implementación del diseño inicial.

AGRADECIMIENTOS

En primer lugar, agradecimiento obligado a Lluís Gesa. Sin ser alumno suyo ni formar parte de este trabajo, él me hizo un hueco y se sentó conmigo para discutir como realizar el diseño de la plataforma. Él propuso la utilización de MongoDB y me dio varias ideas que han resultado claves en todo el proyecto. Es difícil encontrar un momento que lo defina mejor. También agradecer a Pedro De Paco la confianza y el trato recibido ante cualquier duda sobre la elaboración del trabajo. Y como no, gracias a mis padres, especialmente a mi madre, por caminar una infinidad de veces por delante del radar para llevar a cabo las pruebas de detección y seguimiento. Bendita paciencia.

REFERENCIAS

[1] “RADAR - Introduction of RADAR Systems, Types and Applications,” ElProCus, 25-Nov-2019. [Online]. Available: https://www.elprocus.com/radar-basics-types-and-applications/.

[2] “Physics Tutorial: The Doppler Effect,” The Physics Classroom. [Online]. Available: https://www.physicsclassroom.com/class/waves/Lesson-3/The-Doppler-Effect.

[3] P. Kennedy, “Who Made That Traffic Radar?,” The New York Times, 2013.

[4] Densl, “RADAR AFORADOR DTS 600,” DENSL. [Online]. Available: http://densl.com/productos/control-policial/radar/.

[5] G. M. Brooker, “Understanding Millimetre Wave FMCW Radars ,” 1st International Conference on Sensing Technology, 2005.

[6] “24 GHz radar tools and development environment user manual,” Infineon Technologies, 2019. [Online]. Available: https://www.infineon.com/dgdl/Infineon-24GHz_Radar_Tools_and_Development_Environment_User_Manual-ApplicationNotes-v01_00-EN.pdf.

[7] “Position2Go software user manual” Infineon Technologies, 2019. [Online]. Available:https://www.infineon.com/dgdl/Infineon-P2G_Software_User_Manual-ApplicationNotes-v01_01-EN.pdf

[8] “What is NoSQL? NoSQL Databases Explained,” MongoDB. [Online]. Available: https://www.mongodb.com/nosql-explained.

[9] “BSON,” BSON (Binary JSON) Serialization. [Online]. Available: http://bsonspec.org/.

[10] “Get started with MongoDB,” MongoDB Documentation. [Online]. Available: https://docs.mongodb.com/.

[11] “MATLAB Profiling,” Profile Your Code to Improve Performance - MATLAB amp; Simulink - MathWorks España. [Online]. Available: https://es.mathworks.com/help/matlab/matlab_prog/profiling-for-improving-performance.html.

APÉNDICE

A.1. Base de datos

La elección de la base de datos no ha sido algo trivial. A día de hoy existen una gran variedad de tipos de bases de datos que se adaptan mejor o peor a distintos entornos. Dichos tipos incluyen bases de datos relacionales, jerárquicas, de red, orientadas a objetos, NoSQL, etc. A día de hoy las dos bases de datos más comunes y utilizadas son las bases de datos relacionales y los sistemas NoSQL.

Las bases de datos relacionales están basadas en modelos que representan datos en tablas y establecen vínculos entre distintos campos. En una base de datos relacional cada fila de una columna es identificada de manera única utilizando una llave primaria y las distintas columnas guardan los atributos. Este tipo de base de datos, gracias a las relaciones que se establecen, es muy intuitivo y directo.

Las bases de datos NOSQL no utilizan tablas relacionales para guardar los datos sino que utilizan otros métodos como documentos, parejas de clave-valor o gráficos. Estas bases de datos proveen una gran flexibilidad y escalan muy bien con grandes cantidades de datos. Sus principales ventajas son las altas velocidades de lectura y escritura, no necesitan definir ningún esquema previo, escalan bien horizontalmente y la mayoría son *open source*. Por contra, la consistencia e integridad de los datos no está garantizada y su indexado es limitado.

Las bases de datos relacionales son fáciles de comprender y manejar, soportan ACID, preservan la integridad de los datos y su indexado es limitado. Sin embargo se deben normalizar los datos, no escalan bien horizontalmente y en determinadas ocasiones su rendimiento es lento, especialmente con el uso de JOIN.

Teniendo esto en cuenta, podemos llegar a la conclusión que las bases de datos NoSQL trabajan mejor en entornos que no requieren soporte ACID, en los que la velocidad es una prioridad sobre la seguridad frente a errores, los datos son dinámicos y se realizan escrituras constantes, y en el que los datos pueden expresarse sin relaciones.

Sin embargo las bases de datos relacionales se adaptan mejor a entornos en los que el volumen de datos es constante, en el que la consistencia y persistencia son importantes, los datos son predecibles y estructurados, la solicitud de datos es compleja y la seguridad frente a errores es más importante que la velocidad de lectura y escritura.

Con toda esta información, ¿qué base de datos se adapta mejor a nuestros requerimientos? En nuestra aplicación cumplir con las características ACID no es algo imprescindible ya que estamos trabajando simplemente con vectores de valores muestreados a frecuencias suficientemente altas como para que el error de un dato no cambie absolutamente nada del resultado final. En nuestra base de datos no se guardarán elementos críticos como pueden ser DNIs, números de teléfono o correos electrónicos.

La flexibilidad también es uno de los principales requerimientos. Al tratarse de una plataforma para la que todavía desconocemos su propósito final, se antoja complicado definir por completo el esquema de una base de datos relacional.

Otro de los puntos críticos es la velocidad de lectura y escritura. En nuestra plataforma estaremos recibiendo *dataframes* a velocidades altas (recordemos que el muestreo de la placa Sense2Go es de 3000Hz) y los *dataframes* rondan las 128/256 muestras.

Con todo esto resulta bastante obvia la elección de una base NoSQL para nuestro sistema. En concreto utilizaremos una base de datos NoSQL basada en documentos o archivos: MongoDB.

A.2. MongoDB Compass Community

MongoDB se instala a través de un simple instalador y, junto con la base de datos, se proporciona una interfaz gráfica llamada MongoDB Compass Community. Dicha interfaz es muy intuitiva y desde ella se pueden gestionar todos los datos y documentos que conforman la base de datos.

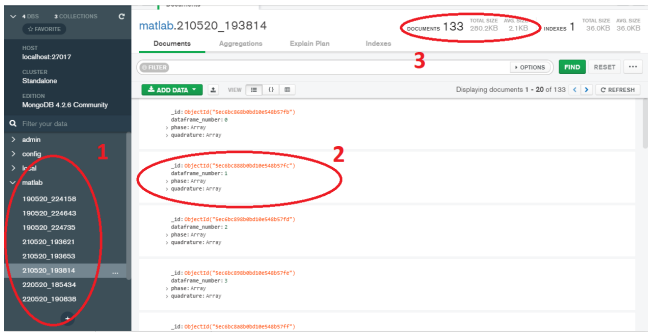

Fig. 9: Interfaz MongoDB Compass Community.

En la Fig.9 podemos observar la interfaz gráfica de MongoDB Compass Community y se han señalado 3 aspectos importantes, detallados a continuación:

- 1 - Conjunto de colecciones. En la barra lateral observamos el conjunto de bases de datos y colecciones. Cada colección tiene como nombre la fecha de su creación.
- 2 - Archivos. En el centro de la interfaz se observa el conjunto de archivos que conforman una colección.
- 3 - Información sobre la colección. En la parte superior se puede observar información general sobre la colección, como el numero total de archivos o su tamaño.

A.3. Computación en paralelo

En la Fig.2 se proponía un sistema de dos hilos pero ni en la parte del procesado en tiempo real ni en la parte del postprocesado ha sido implementado. Su implementación fue descartada por razones de complejidad, ya que en MATLAB no existan las herramientas que si encontramos en C/C++, y la creación de varios hilos de trabajo suponía la utilización de varias extensiones cuyo coste superaba sus beneficios. Aún así, la información recogida en este apéndice puede ser de utilidad en futuras investigaciones.

A.3.1. Función fork()

La función *fork()* es una llamada al sistema que se utiliza para crear un nuevo proceso y desde ese punto el proceso actúa como hijo del proceso principal. A partir de la llamada al *fork()*, los dos procesos ejecutan de manera paralela la siguiente instrucción del código.

Para distinguir el proceso padre del proceso hijo podemos utilizar el valor que devuelve la llamada al *fork()*:

- Si la llamada devuelve un valor negativo, la creación del proceso hijo no ha funcionado.
- Si la llamada devuelve 0, estamos en el proceso hijo.
- Si la llamada no devuelve 0, el valor que devuelve es el PID del proceso hijo.

Como ya se ha mencionado, el valor que devuelve la llamada se utiliza para diferenciar entre procesos, y eso nos permite ejecutar trozos de código diferentes de manera paralela, tal y como podemos ver en la Fig.10.

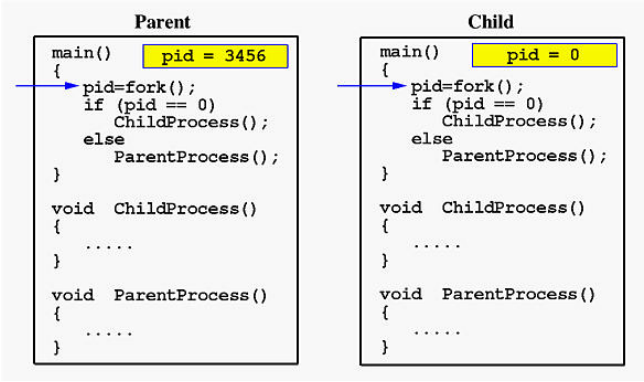

Fig. 10: Ejemplo de utilización del fork(). Fuente: Michigan Tech University

El *fork()* hace dos copias idénticas del espacio de direcciones y, por tanto, ambos procesos tienen las mismas variables que habían sido declaradas antes de la llamada. Sin embargo, como cada proceso tiene su propio espacio de direcciones, los cambios, creación o eliminación de variables son únicos para cada proceso.

Compartir datos entre procesos es posible, pero es necesario la utilización de memoria compartida, algo que eleva la complejidad del código.

A.3.2. Utilización de hilos

El concepto de la utilización de hilos es algo más abstracto que el de *fork()*. Un thread o hilo es un camino de ejecución dentro de un proceso, y pueden existir múltiples de ellos a la vez. Para poner un ejemplo, la ventana de un navegador sería el proceso y las múltiples pestañas serían los hilos.

Al dividir un mismo proceso en múltiples hilos, todos ellos comparten el mismo espacio de memoria y son dependientes entre si. Sin embargo, de la misma manera que los procesos, cada hilo tiene su propio Program Counter, registro y espacio en el stack.

La utilización de hilos supone algunas ventajas bastante relevantes en el contexto que nos ocupa respecto al *fork()*. La comunicación entre piezas de código es más sencilla gracias a la compartición de recursos, se mejora el *throughput* del diseño o sistema y se utiliza de manera más eficiente la arquitectura de múltiples cores que incorporan la gran mayoría de ordenadores de hoy en día.

A continuación se muestra un ejemplo para lanzar un hilo que ejecute una función, extraído de *Geeks for Geeks*.

```
void foo(param)
{
    // Do something
}

// The parameters to the function are put
// after the comma
std::thread thread_obj(foo, params);
```

Esta solución de computación en paralelo era la ideada en el diseño de la Fig.3.

A.4. Especificaciones del hardware utilizado

A.4.1. Infineon Sense2GoL

El kit Sense2GoL es una plataforma de demostración que opera en la banda de 24 GHz y utiliza el efecto Doppler para la detección de movimiento, su dirección y la velocidad, de manera que es un radar de onda continua sin modular. Las características de la placa, detalladas más adelante en esta sección, hacen del Sense2GoL una útil herramienta para aplicaciones relacionadas con la seguridad, sensores de luz, casas inteligentes y, en general, aplicaciones basadas en la detección de blancos.

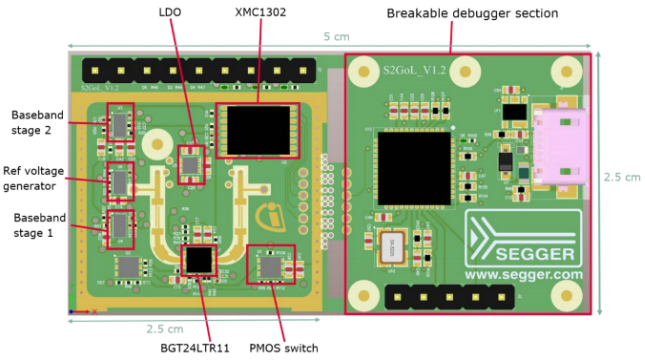

Fig. 11: Placa Sense2GoL. Fuente: Infineon Technologies

La arquitectura del Sense2GoL consiste de los siguientes componentes:

- El chip BGT24LTR11 24 GHz RF con un transmisor y un receptor.
- El microcontrolador XMC1302 Cortex-M0 con el que se realizan las tareas de muestreo y el procesado de la señal analógica. El XMC1302 Cortex-M0 trabaja a una frecuencia de CPU de 32 MHz, tiene una memoria Flash de 16 kB y una memoria RAM de la misma capacidad. El conversor analógico-digital es de 12 bits y cuenta con 16 canales.
- Un *debugger* con soporte para comunicación UART.
- Un amplificador para el canal de recepción.
- Antenas tipo parche microstrip con una ganancia de 10 dBi

Las principales características del kit de desarrollo son la capacidad de detectar la velocidad, el movimiento y la

dirección de dicho movimiento, una alta sensibilidad de detección, capacidad de medir la velocidad de blancos humanos, capacidad de operar en diferentes condiciones meteorológicas y capacidad de detectar a través de materiales no metálicos.

En la Tabla 1 se observan las principales especificaciones del módulo Sense2GoL.

Además de detalles sobre el hardware, Infineon Technologies proporciona datos sobre la precisión del Sense2GoL. A través de pruebas realizadas mediante un simulador Doppler, el fabricante ofrece un margen de error de 0.3 km/h en la medida de la velocidad, además de las limitaciones de distancias especificados en la Tabla 1.

A.4.2. Infineon Position2Go

El kit de desarrollo Position2Go es un radar FM-CW con capacidad para medir distancia, velocidad, dirección de movimiento y ángulo de múltiples blancos de manera simultanea. Las aplicaciones de esta placa, mucho más diversas que las del Sense2GoL, incluyen el seguimiento de blancos humanos, sensores de colisión, reconocimiento de obstáculos, etc.

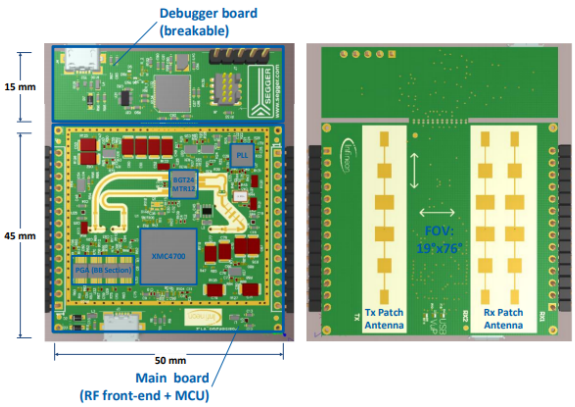

Fig. 12: Placa Position2Go. Fuente: Infineon Technologies

La arquitectura del Position2Go está compuesta por cuatro secciones:

- Parte RF. Consiste del chip BGT24MTR12 de 24 GHz y las antenas tipo parche microstrip encargadas de la transmisión y recepción.
- Parte analógica. Provee una interfaz entre las partes de RF y las partes digitales de la placa. Contiene varios PGAs.
- Control de la frecuencia. Contiene un PLL de N fraccional.
- Parte digital. Consiste del microcontrolador XMC4700 Cortex-M4 que muestrea y procesa los datos analógicos recibidos. También configura el BGT24MTR12, el PLL y los PGAs.

Las principales características de la placa Position2Go son la detección y seguimiento de múltiples blancos en exteriores, la medición de rangos entre 1 y 50 metros, capacidad de detección de movimiento y presencia de blancos humanos, es operacional en condiciones climatológicas adversas y puede detectar blancos a través de materiales no metálicos.

En la Tabla 2 se observan las principales especificaciones de la placa Position2Go.

A.5. Entornos de pruebas realizadas

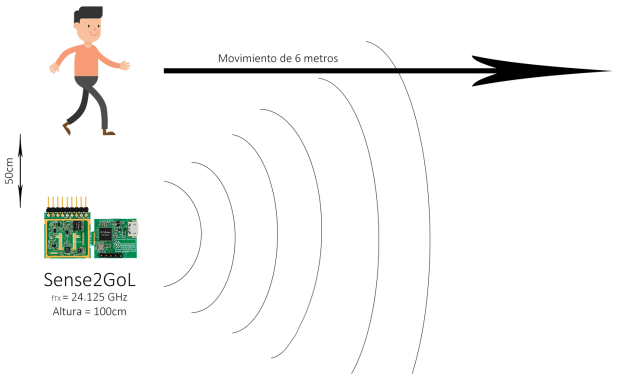

Fig. 13: Entorno prueba 1.

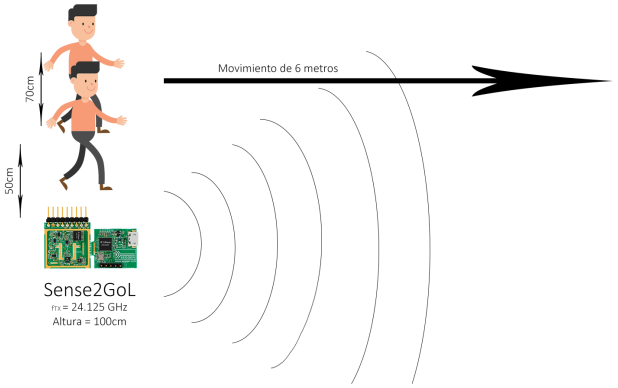

Fig. 14: Entorno prueba 2.

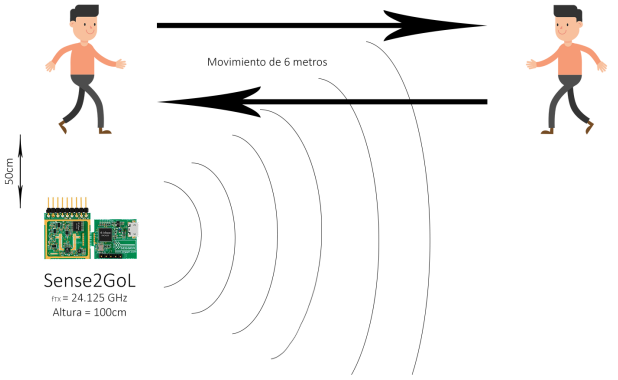

Fig. 15: Entorno prueba 3.

Parámetro	**Unidad**	**Typ.**	**Comentarios**
Velocidad mínima	km/h	0.5	
Velocidad máxima	km/h	30	
Distancia máxima	m	15	Blanco humano
Frecuencia de transmisión	GHz	24.125	
Frecuencia de recepción	GHz	24.125	
Tensión de alimentación	V	5	Min. 3.3, Max. 5.5
Tipo de antena		1 x 4	
Ancho de haz horizontal (3 dB)	Grados	80	
Ancho de haz vertical (3 dB)	Grados	29	

Tabla 1: ESPECIFICACIONES DE LA PLACA SENSE2GOL. FUENTE: INFINEON TECHNOLOGIES

Parámetro	**Unidad**	**Typ.**	**Comentarios**
Velocidad mínima	km/h	0	Depende de la configuración de la placa
Velocidad máxima	km/h	36	Depende de la configuración de la placa
Distancia mínima	cm	60	
Distancia máxima	m	15	
Precisión del rango	cm	± 20	
Resolución del rango	cm	90	
Precisión del ángulo	Grados	<5	FoV $\pm 30^o$
	Grados	<10	FoV $\pm 65^o$
Frecuencia de transmisión	GHz	Min. 24.0 Max 24.25	
Frecuencia de recepción	GHZ	Min. 24.0 Max 24.25	
Ancho de haz horizontal (3 dB)	Grados	76	
Ancho de haz vertical (3 dB)	Grados	19	

Tabla 2: ESPECIFICACIONES DE LA PLACA POSITION2GO. FUENTE: INFINEON TECHNOLOGIES

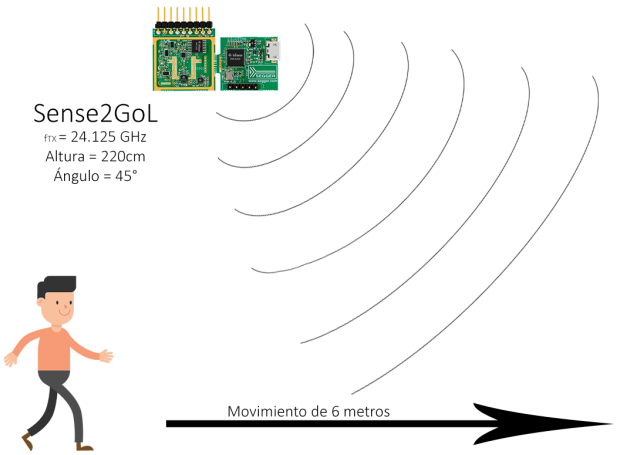

Fig. 16: Entorno prueba 4.

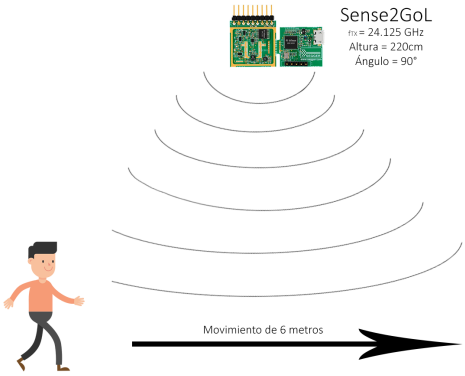

Fig. 17: Entorno prueba 5.

Fig. 18: Entorno real prueba 5.