Universidad de Alcalá
Escuela Politécnica Superior

GRADO EN INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL

Trabajo Fin de Grado

DISEÑO, SIMULACIÓN E IMPLEMENTACIÓN, DE UN PREDICTOR DE PULSOS RADAR PARA GUERRA ELECTRÓNICA

Autor: Borja Miguel Peñuelas Morales

Tutor: Ignacio Bravo Muñoz

2015

UNIVERSIDAD DE ALCALÁ
Escuela Politécnica Superior

GRADO EN INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL

Trabajo Fin de Grado

DISEÑO, SIMULACIÓN E IMPLEMENTACIÓN, DE UN PREDICTOR DE PULSOS RADAR PARA GUERRA ELECTRÓNICA

Autor: Borja Miguel Peñuelas Morales

Director: Ignacio Bravo Muñoz

TRIBUNAL:

Presidente: José Luis Lázaro Galilea

Vocal 1º: Ernesto Martín Gorostiza

Vocal 2º: Ignacio Bravo Muñoz

CALIFICACIÓN: .....
FECHA: .....

ÍNDICE DE CONTENIDO

RESUMEN

El constante desarrollo en la Tecnología de Defensa lleva asociado el desarrollo de Contramedidas efectivas para mantener la superioridad tecnológica, con el fin de elevar la seguridad de las operaciones tanto civiles como militares. Cada vez es más decisiva para ello la llamada *Guerra Electrónica*, donde la tecnología RADAR juega un papel esencial permitiendo la monitorización precisa de objetivos y amenazas.

Esto supone que dominará el espacio electromagnético quien sea capaz de operar sus sistemas bajo dos condiciones indispensables: inmunidad al sabotaje, y capacidad para anular las medidas enemigas.

Los sistemas RADAR de detección modernos emplean patrones de pulsos complejos, para impedir la anulación o falseo de sus medidas. En el siguiente trabajo se recopilan técnicas matemáticas útiles para la predicción de estos pulsos, lo que permite de nuevo el uso de *Contramedidas* contra ellos. Se realiza un estudio cuantitativo que permitirá obtener requerimientos técnicos del sistema y se establece una arquitectura escalable para su implementación en un sistema de tiempo real capaz de operar en un escenario con un número en constante crecimiento de amenazas. Así mismo se incluyen simulaciones de las partes esenciales y se detallan las configuraciones necesarias en un sistema electrónico digital para su óptimo funcionamiento de cara una futura implementación en hardware.

Palabras clave: Predictor, RADAR, Contramedidas, Guerra Electrónica

ABSTRACT

The constant development of Defense technology, carries an associated development of effective Countermeasures to maintain technological superiority and raise security in civil and military operations. To that end, the so-called Electronic Warfare (EW) has become an increasingly determining factor. RADAR technology plays an essential role, allowing precise monitoring of targets and threats.

Whoever can operate under the following two conditions will dominate the electromagnetic space. Immunity to hijacking, and the ability of nulling enemy measures.

Modern detection RADAR systems employ complex pulse patterns to prevent their measurements being overridden or deceived by the enemy. The present work collects mathematical techniques which are useful to perform pulse prediction, which re-enables the use of Countermeasures against them. A quantitative study is performed in order to obtain technical requirements for such a system. With scalability in mind, an architecture that allows real-time operation is defined, making it capable of successfully targeting an increasing number of threats. Simulations of the essential parts are included, as well as details for optimum operation in a digital electronic system, targeting a future hardware implementation.

Keywords: Predictor, RADAR, ECM, Electronic Warfare

RESUMEN EXTENDIDO

En el presente trabajo se han explorado los factores que determinan los requisitos técnicos para un sistema de Contramedidas Electrónicas[1]. Dado que se muestra cómo un sistema Predictor mejoraría notablemente las capacidades del conjunto, se estudian las técnicas matemáticas necesarias para abordar el problema de la predicción. Se realiza una simulación de las dos partes esencial del sistema, la de identificación de periodo, y la de estimación. Se analiza su desempeño y se estudia el hardware más apropiado para la futura implementación de un sistema que las incluya.

Primero se hace un recorrido por la historia de la tecnología RADAR las Contramedidas, llegando a ver cómo en la actualidad es necesario un sistema Predictor para contrarrestar los avances en tecnología anti-intercepción.

Planteada la situación, se determina la estructura de un sistema que da respuesta a las necesidades de estimación para asistir en la realización de Contramedidas. Entonces se hace un estudio de las técnicas y algoritmos que permiten solucionar los problemas que presenta la estimación de eventos futuros. Los dos principales problemas a los que se reduce son la identificación del periodo de la secuencia de pulsos recibidos, y posteriormente, conocido el periodo, la estimación de los estados futuros.

Para elaborar un método de identificación de periodo que cumpla los objetivos marcados, se han analizado distintos algoritmos, y tomando elementos de cada uno de ellos, se ha confeccionado el empleado finalmente: el histograma de escalera de comparadores. Con él, se obtienen de manera instantánea las repeticiones de cada periodo y sub-periodo de la secuencia que se encuentra almacenada en el buffer a analizar.

La estimación se realiza con un filtro recursivo Kalman. Se explican los motivos que hacen esta la forma adecuada de conseguir la estimación, y se hace un análisis exhaustivo de la matemática que aplica, para poder inferir las fórmulas que debe aplicar el algoritmo de predicción. Se presentan los valores de las matrices implicadas en el proceso de forma general, extensible al número de estados que se haya de emplear.

Para poder establecer requerimientos técnicos en un sistema de este tipo, se comienza haciendo una descripción de los ECM[2], y explicando los conceptos generales de aplicación.

Los fenómenos físicos que tienen lugar en la operación de sistemas RADAR son analizados, así como las características de sistemas relacionados que influyen en los parámetros de funcionamiento del ECM y el Predictor. Por último se muestran las condiciones de interoperabilidad que tiene que respetar tras su integración.

[1] Contramedidas Electrónicas – Sistemas empleados para aplicar técnicas de anulación o manipulación de las medidas de receptores RADAR.

[2] ECM – Electronic Countermeasures. Traducción Contramedidas Electrónicas.

Se exponen los criterios que se han seguido para confeccionar una arquitectura de Predictor de apoyo a ECM eficaz. Estos son:

El dimensionado parametrizable, que permite tanto la simulación de distintas configuraciones, como la adecuación inmediata de la implementación en hardware de acuerdo a los recursos presentes en el dispositivo que lo alojará.

La multiplexación en el tiempo de los recursos de cálculo, que posibilita que el sistema crezca en capacidades a un ritmo mayor del que crecen los recursos empleados.

Monitorización de cambios en el periodo identificado, para elevar la robustez del sistema frente a imperfecciones de la trama recibida.

Las partes esenciales del estimador son elaboradas siguiendo los criterios marcados para satisfacer los requisitos de operación, versatilidad, exactitud y traducción a elementos de hardware implementables.

Para comprobar sus capacidades de operación, el identificador de periodo y el estimador Kalman son simulados. Puesto que son parametrizables, se simulan bajo distintas configuraciones y con distintas tramas. De esta forma se validan para su uso en el sistema final, y se pueden obtener conclusiones para la mejora de su funcionamiento.

Finalmente, se hace una introducción del hardware que se considera más apropiado para la implementación de la operación descrita y simulada. Se recomienda una pieza comercial en base a sus características y las herramientas software asociadas a ella.

Por último, como trabajo futuro se abre la puerta a una estructura que pueda incorporar varias técnicas de predicción simultáneas y elegir la que esté dando mejores resultados.

CAPÍTULO 1:
HISTORIA Y MOTIVACIÓN, OBJETIVOS DEL TRABAJO Y METODOLOGÍA

CAPÍTULO 1: HISTORIA Y MOTIVACIÓN, OBJETIVOS DEL TRABAJO Y METODOLOGÍA

1.1 HISTORIA Y MOTIVACIÓN

La tecnología RADAR ha encontrado un enorme número de aplicaciones desde que se descubre el fenómeno físico que emplea, y se comienza a aprovechar con utilidad práctica.

Hace más de 100 años Nicola Testa sugirió que la reflexión de ondas electromagnéticas podría usarse para detectar objetos metálicos. Menos de treinta años después se equipa el primer barco con RADAR; sin acabar el decenio siguiente, estaba siendo desarrollado por todas las grandes potencias mundiales. [1]

Posteriormente de desarrollan nuevos radares con un funcionamiento que promete mayor seguridad ante las capacidades de interferencia enemigas: la agilidad en frecuencia. Aunque se habían realizado experimentos desde 1965, el dominio de la técnica hasta poder ofrecer un producto final que funcione bajo este principio, requirió años y un cambio en los materiales y elementos empleados.

En marzo de 1967 se anuncia el desarrollo de un sistema de detección coherente con antena de recepción y de transmisión separadas. La capacidad de estos radares para diferenciar elementos móviles de los fijos hizo pensar que podría utilizarse en medicina, para detectar si el corazón de un paciente se había parado. [2]

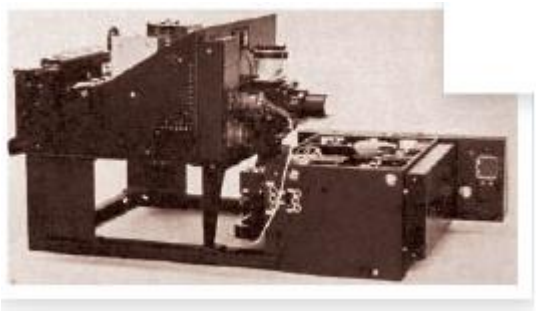


Ilustración 1. Durante la exploración de la técnica, se proponían magnetrones "spin-tuned" para adaptar modelos de RADAR antiguos a la tecnología ágil en frecuencia *[2]*.

Al problema de la capacidad de interferencia enemiga, se añade el de los rangos ambiguos para blancos en movimiento, que se detallará posteriormente. Como respuesta a ambos, se llega a la conclusión de que frente a los radares de onda continua, o *Intervalo de Repetición de Pulso Continuo* (PRI[3] continuo), que son los predominantes hasta la fecha, se debe usar una secuencia que varíe el tiempo entre pulsos para de esta forma poder resolver la ambigüedad y poder conocer con seguridad la posición del blanco. Alrededor de 1970 encontramos patentes [3] que indagan en este principio y se crean sistemas capaces de emplearlo satisfactoriamente.

Las dos técnicas mencionadas mejoran drásticamente la inmunidad a intercepción enemiga, al mismo tiempo que consiguen una detección más fiable y precisa de los elementos en el rango de actuación del sistema. Éstas no se ha dejado de mejorar, cada vez más radares en uso las emplean y se incorporan al equipamiento de más fuerzas militares.

Por ello, creemos que es necesario generar plataformas de contramedidas capaces de sostener el desarrollo de las capacidades anti-jamming[4] de los sistemas de detección.

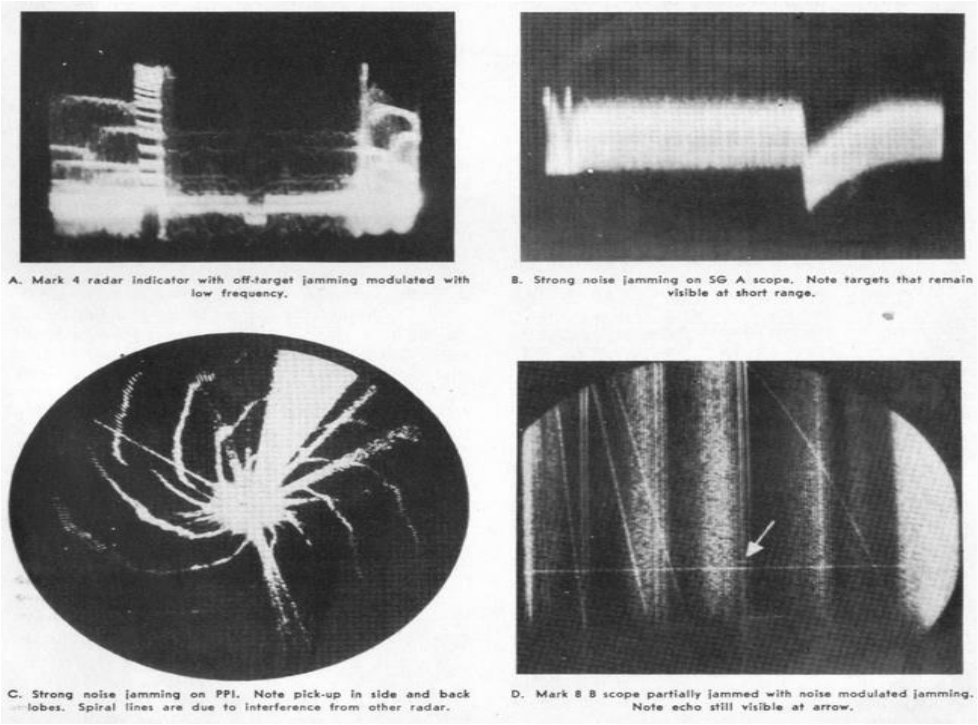


Ilustración 2. Figuras habituales en medidas de RADAR bajo el efecto de distintos tipos de jamming *[4]*.

[3] PRI – Pulse Repetition Interval, el intervalo entre cada pulso que emite el radar.

[4] Jamming – Interferencia generada artificialmente con el objetivo de corromper un sistema

Los rangos de actuación de los radares modernos son enormes, y el acceso a la tecnología es cada vez mayor, lo que supone que los sistemas instalados se multipliquen en número. Esto implica que para garantizar la seguridad de las operaciones emplear naves con el menor perfil de detección posible sea necesario pero no resulta suficiente. Dada la alta probabilidad de detección, debemos, además, conseguir ser capaces de manipular las medidas de sus aparatos para desviarlos de naves aliadas.

Se prevé que en el futuro nos enfrentaremos a sistemas que incluyan a la vez, varias o todas, de las siguientes características [5], las cuales tendremos que contramedir:

- Gran número de emisores. Esto genera un entorno en el que se lidia con multitud de señales concurrentes en el tiempo.
- Ráfagas de frecuencias. Requerimiento de receptores capaces de variar con velocidad la frecuencia de sintonización y/o ser capaces de detectar y procesar bandas anchas de espectro.
- Onda Continua (CW) puede ser empleada ocasionalmente.
- Compresión de Pulso. Lo que permite resolver ambigüedades y mejorar la capacidad de percepción del sistema, dando como resultado una medición a la vez más fiable y precisa de los blancos.
- Haz Ágil. La tecnología más moderna, basada en arrays de antenas controladas con sistemas digitales, permite el apuntamiento y configuración del haz RADAR de forma electrónica, pudiendo cambiar instantáneamente a la forma y dirección deseadas, lo que los hace más versátiles y rápidos.
- Radares LPI[5]. Suponen un reto para el frontend de detección, los sistemas ELINT[6] y EW[7].
- Múltiples PRF[8] y elevados. Permiten, además de resolver ambigüedades, ser más precisos y difíciles de contramedir.

Estas cualidades y modos de funcionamiento, definen nuestros objetivos en el campo de las Contramedidas. Habrá que evaluar pues, las condiciones que establece cada uno de ellos para un sistema final capaz de hacerles frente.

[5] LPI – Low Probability of Intercept. Son radares diseñados específicamente para no ser detectados por segundos sistemas pasivos. Usan grandes anchos de banda, saltos de frecuencia y modulaciones diversas con poca potencia para minimizar la probabilidad de intercepción.

[6] Electronic Intelligence – Sistemas destinados a recoger información sobre las operaciones del enemigo en el espacio electromagnético.

[7] Early Warning – Son los sistemas encargados de detectar la presencia de RADAR enemigo y alertar de ello al resto de sistemas para que procedan con sus acciones designadas.

[8] Pulse Repetition Frequency – Frecuencia de repetición de pulso (Inverso de PRI).

En cuanto a la tecnología empleada, durante la Segunda Guerra Mundial se desarrolló desde los tubos de vacío, pasando por semiconductores, hasta los microcontroladores. Los materiales, componentes, tecnologías de producción y elementos de procesamiento evolucionaron, y no han parado, notablemente.

Tan básico como una chispa fue el primer modo de transmisión empleado en experimentos que llevarían a la creación del RADAR tal como lo conocemos ahora. En los primeros años de uso la tecnología maduró empleando mayoritariamente tubos de vacío y magnetrones de gran potencia. Debido a lo básico de los elementos de sincronización, y la calidad de los receptores, las potencias empleadas eran enormes, con haces muy poco eficientes. No obstante durante ese periodo de Guerra, las tan sólo veintiuna torres de radar de las que disponía Gran Bretaña les dieron una ventaja táctica inimaginable a los aliados hasta el momento. Minimizaron los daños y salvaron vidas de muchos civiles gracias a los sistemas de Alerta Temprana (EW), que permitían dirigir la población a búnkeres, y monitorizar la flota alemana para coordinar la defensa aérea. [6]

Ilustración 3. Sistema de Alerta Temprana en 1962, Alaska.

Posteriormente se desarrolló el equipo de detección que, junto con nuevos instrumentos, invalidasen los sistemas radar enemigos que desde entonces todos los ejércitos comenzarían a incorporar. Con su mayor repunte de actividad durante la "Guerra Fría", el equipamiento empleado ha ido tomando los elementos más vanguardistas de la industria y dando lugar a nuevos avances.

En 1974 se emplea por primera vez un minicomputador para controlar el radar de instrumentación digital AN TPQ-39, pudiendo ser manejado por un solo operador, o puesto en modo "manos libres". [2] Es el principio de la automatización de las tareas de detección.

Se empiezan a utilizar a partir de entonces microcontroladores y microprocesadores para llevar a cabo el control del radar y cada vez más se van sustituyendo partes analógicas por sus equivalentes digitales.

Hoy en día, toda la mezcla, síntesis y demodulación de señales es digital. Incluso el apuntamiento se consigue con radares de matriz de elementos apuntados electrónicamente. Las condiciones de operación, con escenarios cada vez más complejos y con sistemas más exigentes puestos en juego por el enemigo y también por los aliados, obligan a realizar la mayor parte de la operación del radar de forma automática. Al mismo tiempo, los ordenadores o microcontroladores corriendo software, cada vez se ven más limitados al intentar abastecer de información con la temporización adecuada para atender a un gran número de amenazas.

Es por eso que este trabajo indica las pautas para implementar las tareas de interés en sistemas electrónicos digitales basados en FPGA[9]. En concreto se estudia el empleo eficiente de SoC[10] para repartir las tareas según su tipo, tiempo de ejecución y requerimientos de temporización, entre microprocesadores y lógica programable, de forma que se alcance un sistema versátil, escalable y rápido.

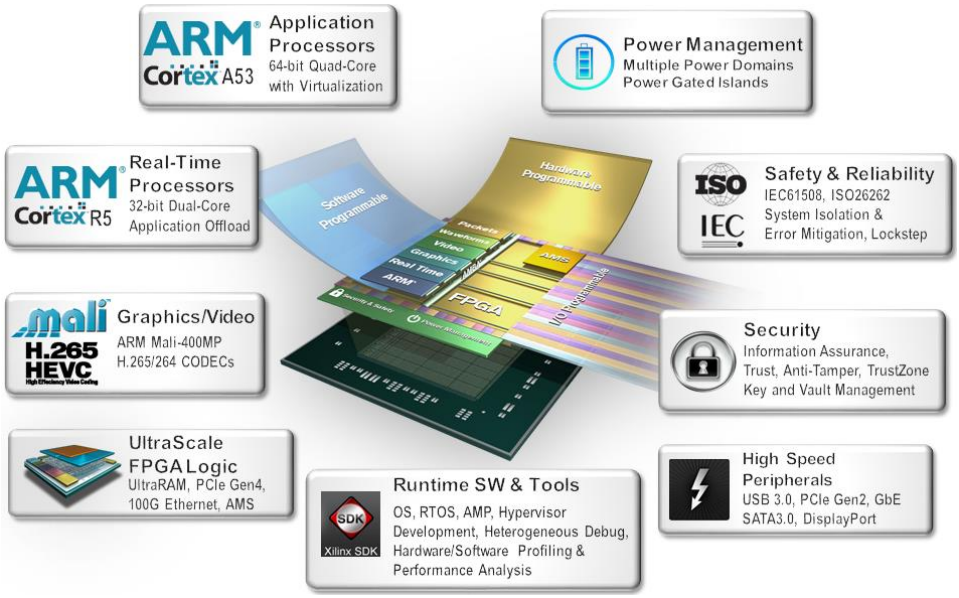


Ilustración 4. Diagrama de la composición de un SoC (Zynq UltraScale+). *[7]*

[9] FPGA - Field-Programmable Gate Array. Es un tipo de circuito integrado reconfigurable capaz de implementar lógica digital empleando bloques lógicos con conexiones configurables.

[10] System On Chip – Sistemas integrados en un solo chip, que comprometen FPGA y microprocesadores ahorrando espacio y al mismo tiempo aumentando la capacidad y velocidad de intercambio de datos.

La tecnología de procesado digital, tiene en la actualidad una demanda tal, que lleva a la electrónica de consumo hasta los elementos más vanguardistas. Este fenómeno no ocurría hace pocos años. Entonces, el equipamiento militar empleaba elementos al alcance de sólo unos pocos, disponer de elementos a años de desarrollo del resto de competidores garantizaba control y estabilidad. Sin embargo, en estos momentos el acceso a la tecnología más moderna se ha extendido hasta tal punto que es inevitable que terceros fuera del ámbito de confianza accedan a radares que estén equipados con mayor capacidad anti-intercepción, y será necesario mantener la capacidad de intervenirlos para mantener nuestra capacidad defensiva.

Dados los precedentes explicados, y teniendo en cuenta la situación futura que se prevé para el espacio electromagnético, se ve necesario el establecimiento de una arquitectura que sea capaz de:

- Operar en tiempo real en un ambiente poblado de blancos.
- Tratar sistemas que emplean técnicas anti-interferencia.
- Ser modular y escalable para evolucionar al igual que harán las amenazas.

1.2 OBJETIVOS DEL TRABAJO

Del anterior análisis de la situación actual y la tendencia en la industria se establece que el principal objetivo del trabajo es la definición de los elementos que compondrán una arquitectura de predicción de pulsos capaz de trabajar con patrones complejos de PRI.

Esta es una representación del escenario en el que debe trabajar el Predictor. Se muestra el ecosistema en el que trabaja, los sistemas con los que interactúa, y los bloques que necesita para su funcionamiento:

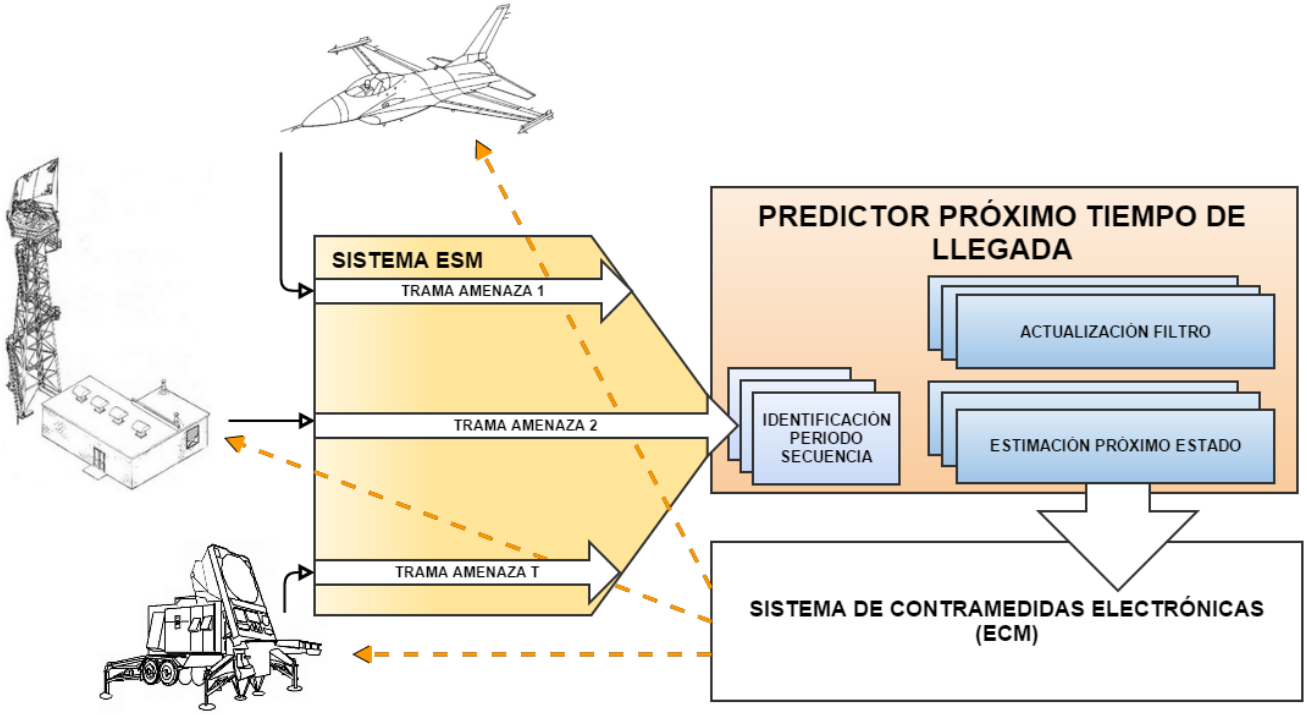


Ilustración 5. Escenario de operación del Predictor y estructura del mismo[11].

Se presentarán técnicas que se han encontrado útiles para predecir pulsos RADAR en entornos militares y civiles, y centraremos el estudio de su implementación en el estimador basado en filtrado Kalman. En concreto de aplicación a patrones Stagger[12], ya que son la forma de PRI complejo más extendida y que está reemplazando a los radares de anterior generación.

[11] ESM – Electronic Support Measures. Son sistemas de apoyo a las contramedidas, que emplean características del pulso de radar para diferenciar emisores y aportar a los sistemas de Contramedidas ECM información necesaria para realizar sus técnicas de engaño.

[12] Patrones Stagger – Son secuencias de pulsos radar en las que se va variando el PRI de pulso a pulso, para por un lado eliminar ambigüedades y por otro, impedir el funcionamiento de contramedidas electrónicas convencionales

La forma de pulsos de RADAR básica es la de ancho de pulso y PRI constante:

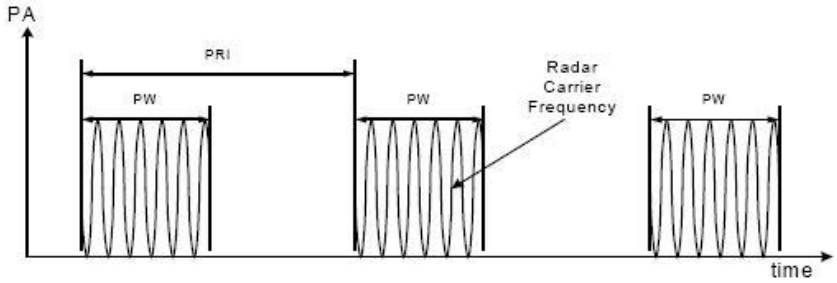


Ilustración 6. Ejemplo PRI constante.

PA es la amplitud del pulso y PW el ancho del pulso. El objeto de interés, el PRI en este caso no varía pulso a pulso, sin embargo esta es la forma que toma un patrón de pulsos Stagger en el tiempo:

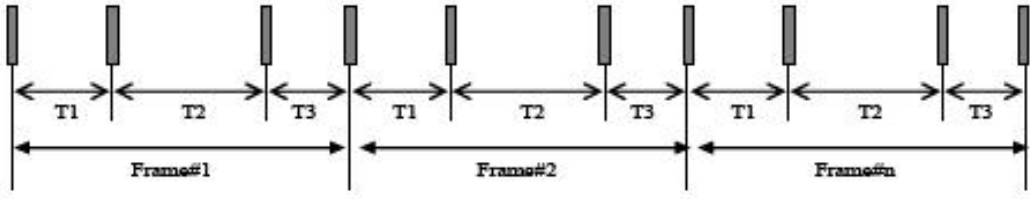


Ilustración 7. Ejemplo secuencia Stagger de PRI *[8]*.

Las secuencias Stagger están compuestas por un PRI básico, que es para cada pulso multiplicado un número arbitrario de veces. Este número por el cual se multiplica el PRI básico es un elemento. La secuencia de denomina por el número de elementos distintos que la componen, y el número total de posiciones que forman un periodo. Este es un ejemplo:

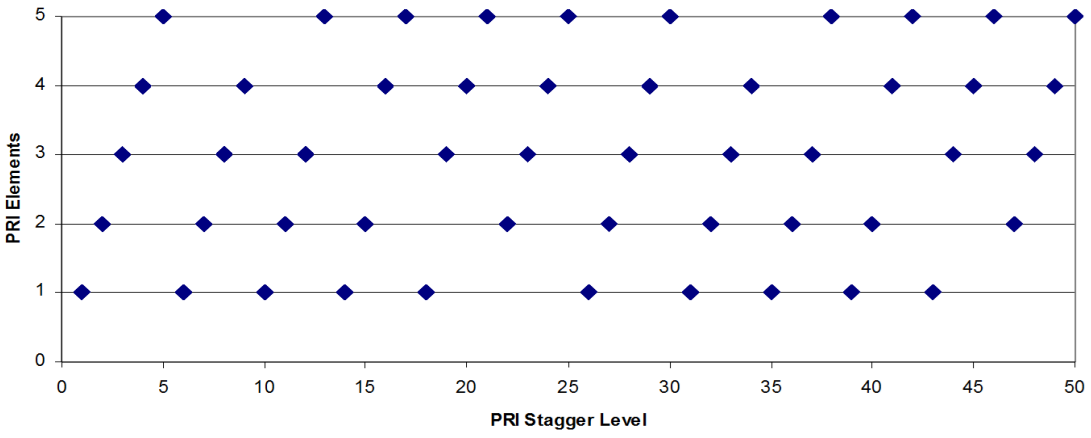


Ilustración 8. Dos periodos de una secuencia Stagger de 5 elementos, 25 posiciones *[9]*.

El objetivo es identificar cuanto antes el patrón y estimar el tiempo de llegada correspondiente al siguiente pulso en la secuencia. De esta manera quedan establecidos los siguientes objetivos que estructuran el trabajo:

1 Descripción de técnicas matemáticas que pueden ser empleadas en la predicción de pulsos. Se hace una recopilación de las técnicas que se han encontrado más interesantes con vistas a ser implementadas en un sistema de las características descritas. Se detalla más extensamente el predictor basado en filtro Kalman ya que es el que se ha encontrado más apropiado para los patrones tipo Stagger.

2 Estudio de las variables que influyen a la hora de dictar los requerimientos de un sistema de este tipo. Cuando se crece en complejidad, conviene concretar los elementos y temporización necesarios para que el sistema sea capaz de dar respuesta al número de amenazas deseado, con características determinadas y con un nivel de confianza establecido.

3 Propuesta de arquitectura que dé respuesta al problema, atendiendo a los requisitos establecidos anteriormente se persigue una arquitectura que sea capaz de evolucionar dada la necesidad futura de aumentar o cambiar el tipo de las amenazas a contramedir. Debe dotarse de robustez y es deseable que se realice de forma parametrizable. Es decir, que dependiendo la capacidad del sistema electrónico donde se va a implementar, se pueda adaptar para dar el máximo de prestaciones dentro de los límites de las piezas hardware a emplear para ello.

4 Simulaciones en Matlab y Simulink de las dos partes que conforman el predictor de pulsos elegido. Consta de un identificador de periodo de la secuencia Stagger, y el propio filtro Kalman que requiere del primero para su correcta configuración.

5 Indicaciones para configuración del Sistema Electrónico Digital basado en SoC. Se indicará los elementos que deben localizarse en cada parte del SoC de acuerdo a las necesidades de cómputo deducidas por las simulaciones, de cara a que los bloques generados se traduzcan en un futuro de manera casi inmediata a HDL que configure hardware real.

1.3 METODOLOGÍA

Para alcanzar los objetivos anteriormente marcados, se seguirán los siguientes Métodos.

Problema a abordar:

Un sistema ESM provee los tiempos de llegada de pulsos radar de cada uno de los emisores detectados, con ruidos generados en origen, y debidos a la precisión de la medida. Se debe inferir el periodo de la secuencia Stagger con la que trabaja cada emisor y proporcionar como salidas, para un siguiente sistema, el tiempo de llegada predicho para el próximo pulso de cada amenaza, así como un indicador de seguridad de la predicción.

Ya que el objetivo final es acabar dando una visión general del sistema deseado, se detallan la forma de implementarlo y las herramientas empleadas para ello:

1 Recopilación y estudio de publicaciones y libros de texto que tratan el problema de predicción de patrones desde distintos puntos de vista. Se escogen los más representativos y se resumen aquí las partes más útiles para este sistema.

2 Para establecer los requisitos técnicos se estudiarán los fenómenos físicos en los que se basa el RADAR y se verá qué determina cada uno de ellos en la forma de operar del sistema de Contramedidas. Posteriormente se analizarán las capacidades de un sistema de Contramedidas en función de sus capacidades de funcionamiento.

3 Mediante investigación, exploración de distintas opciones y discusión con especialistas en el tema, se concluye una arquitectura y organización adecuadas para tratar el problema de la predicción de pulso y los sistemas multiamenaza.

4 Para efectuar simulaciones de las dos partes del predictor se emplea: En el caso del identificador de periodo Stagger, un script Matlab, que paramétricamente genera un modelo Simulink del identificador. Se crea también el modelo Simulink del filtro Kalman y dado que ambos son flexibles, se muestran los resultados obtenidos con las distintas opciones.

5 Para poder establecer las pautas concretas de cara a implementar el sistema simulado en hardware, se tienen en cuenta los conocimientos necesarios en uso de FPGA para procesado digital. Se indica, de cara a una futura traducción de los bloques generados a HDL listo para FPGA, los elementos adicionales necesarios para hacer el sistema funcional, versátil y capaz de trabajar bajo los requerimientos deseados.

CAPÍTULO 2:
TÉCNICAS MATEMÁTICAS ÚTILES PARA LA PREDICCIÓN DE PULSOS

CAPÍTULO 2: TÉCNICAS MATEMÁTICAS ÚTILES PARA LA PREDICCIÓN DE PULSOS

2.1 IDENTIFICACIÓN DEL PERIODO DE LA SECUENCIA

Multitud de algoritmos necesitan como parámetro para realizar la predicción la información del número de elementos que conforman un periodo de la secuencia. De esta forma es posible ordenar los elementos, procesar la información y actualizar las estimaciones para el próximo estado. Es por ello que se investigan las cualidades de los distintos métodos de identificación de ciclos para confeccionar con elementos de estos la solución empleada.

2.1.1 MÉTODO DE TABLA DE LISTA ABREVIADA [10]

Se emplea este método para encontrar de forma computacionalmente eficiente un patrón que se encuentra repetido en una secuencia de símbolos almacenados en una memoria. Es una mejora respecto al Árbol de Máximo Sub-patrón[13].

- Resumen

  Se escanea la secuencia para encontrar la Tabla de Lista Abreviada (ALT[14]) que incluye la frecuencia de aparición de cada elemento, a continuación se escanea de nuevo para crear el árbol de máximo subpatrón, que identifica patrones frecuentes.

- Procedimiento y explicación

  Si un elemento aparece periódicamente, dado un periodo T sus posiciones en la secuencia formarán una serie aritmética que puede ser capturada con tres parámetros: Posición del periodo, periodo, y cuentas (frecuencia).

  Por ejemplo, en la secuencia S = `bdcadabacdca ', las ocurrencias de' a 'para el período = 2 son 3; 5; 7; 11. Podemos representarlo como (1; 2; 4), donde 1 es la posición periodo de la ocurrencia de `a 'con período = 2, y 4 es la frecuencia del patrón en la serie. Para un período determinado T, para cada elemento, tenemos que mantener T representaciones y las llamamos Lista Abreviada.

[13] Max-subpattern tree – Algoritmo precursor [25]

[14] Abbreviated List Table – Traducción Tabla de Lista Abreviada

El algoritmo, con el que creamos una matriz tridimensional para almacenar cada uno de los parámetros. Es el siguiente.

Algorithm ALT1(ALT, W, S)
1. **while**(S still has elements){
2. Read element e from S,
   //$SeqPos$ is e's position in S;
3. Get the index idx of e in D;
4. **for**($p := 1$; $p \leq W$; p++){
5. pos:=$SeqPos$ mod p;
6. ALT[idx][$p-1$][pos]++;}}
7. //truncate sequence for all periods

Algorithm ALT2 (ALT, W, min_sup, n)
1. **for** (idx:=0; $idx < |D|$; idx++){
2. get element e whose index equals to idx;
3. **for** (p:=1; $p \leq W$; p++){
4. threshold:= $\lfloor n/p \rfloor \times min\_sup$;
5. **for** (pos:=0;$pos < p$; pos++){
6. if (ALT[idx][$p-1$][pos] $\geq$ threshold)
7. output p, pos and element e;
8. }}}

Ilustración 9. Algoritmo ALT para cálculo de periodo.

Para encontrar los patrones frecuentes tenemos en cuenta que para cada período frecuente encontrado en el paso 1, el algoritmo construye el árbol max_subpattern (máximo subpatrón) en el paso 2 utilizando F1 y las listas invertidas. Las listas de los elementos frecuentes se combinan para reconstruir los fragmentos y ellos a su vez se insertan en el árbol max_subpattern. Por último, tenemos patrones frecuentes atravesando todo los árboles.

Este algoritmo se ha desarrollado y comprobado como una mejora sustancial del de árbol de sub-patrones.

- Comparación

Cuando se compara con el algoritmo en el que está basado, se nota una mejora en la velocidad de cómputo y la eficiencia con la que lo hace. En un ejemplo práctico se comparan ambos obteniendo las siguientes figuras.

period pos	0	1	2	3	4
period=1	6				
period=2	4	2			
period=3	2	2	2		
period=4	2	0	1	2	
period=5	1	0	2	2	1

time(s)	[1] +tree	ALT+tree
find period	71.34	5.59
find F_1	2.12	0
build trees	2.07	1.11
mine pat.	2.02	2.01

n	period	num of freq. pat.
100K	100	12
500K	100	42
1000K	100	76
2000K	100	133

Ilustración 10. Tabla ALT generada en la prueba, comparación de coste y patrones encontrados.

La eficiencia en función de la ventana de periodo máximo que se le da, tiene una evolución con menor pendiente que el algoritmo de sólo árbol, por lo que se comporta mejor independientemente del tamaño.

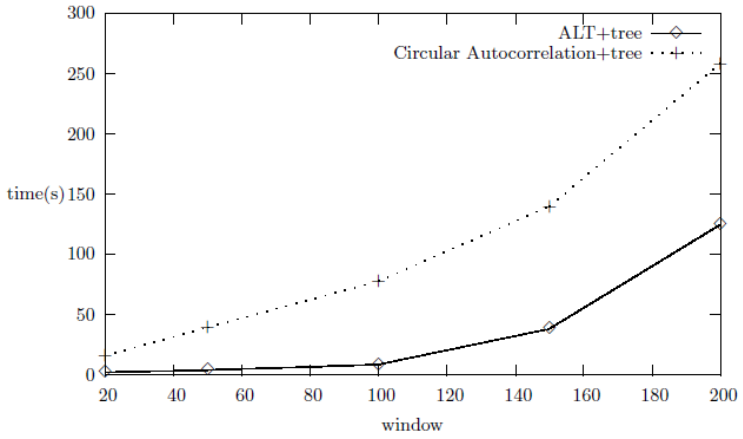


Ilustración 11. Comparación de tiempo de cómputo vs tamaño de la ventana.

Por último, ya que uno de los requisitos del sistema es la escalabilidad es importante conocer de qué forma escala este algoritmo para ver si podría usarse en un sistema de mayor envergadura, con más pulsos y/o mayor memoria.

Para ello es muy representativa esta figura en la que se aprecia que la pendiente con la que aumenta el tiempo de procesamiento aumentando el número de puntos, es muy inferior a la del algoritmo anterior.

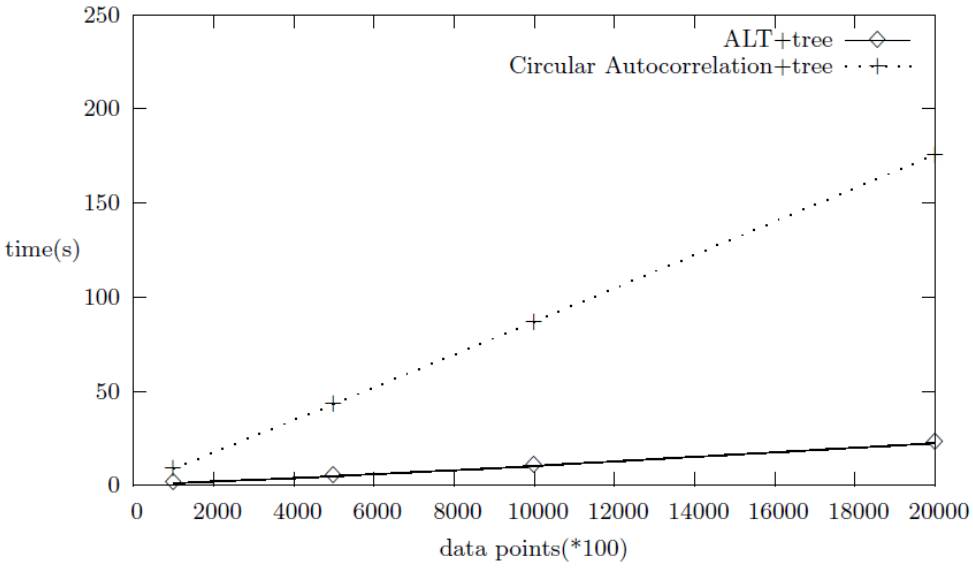


Ilustración 12. Escalabilidad del algoritmo.

- Aplicación

  Analizando el algoritmo y su desempleo, se ve que hay elementos del mismo que resultan útiles para el sistema objetivo. Si bien no es aceptable la necesidad iteración sobre una memoria con un histórico de puntos para llegar a obtener el periodo, se toman varios conceptos que sí serán de aplicación como se explica a continuación. Se toman:

  -Búsqueda de sub-patrones de distinto tamaño.

  -Ocurrencias de un subpatrón y su efecto en el patrón total para identificar el periodo completo.

2.1.2 ALGORITMO DE DETECCIÓN DE CICLO DE FLOYD [11]

A menudo se usa este algoritmo para encontrar bucles infinitos en máquinas de estado o secuencias en ramas de programación de software. Se trata de un algoritmo sencillo y con una relativamente alta eficiencia computacional.

- Resumen

  Se toman dos punteros, ambos comienzan en la primera posición. Cuando uno avanza una posición, el otro avanza dos y se comprueba si se han encontrado. Distintas variantes se emplean para obtener el ciclo más corto o más largo pero se trata básicamente del mismo principio.

- Procedimiento y explicación

  Situando los valores en una lista como una colección de punteros a valores que contienen una dirección, se puede visualizar la secuencia como una serie de μ elementos iniciales que llegan a un punto en el que se repiten con periodo λ. Entonces para encontrar cuándo se ha entrado al bucle basta con identificar la situación en la que $i = m\lambda \geq \mu$.

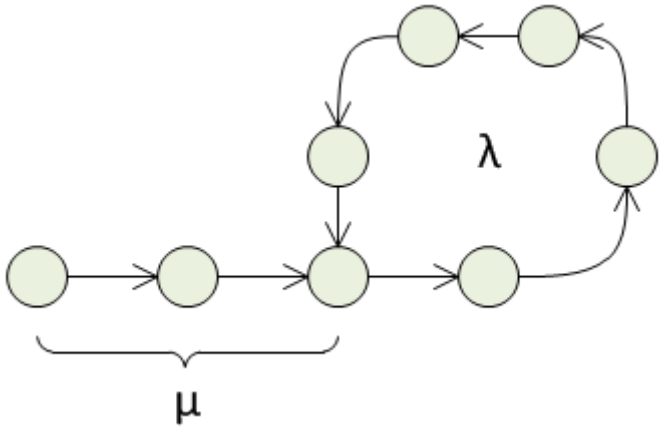


Ilustración 13. Representación de una serie de PRIs que llegaría a ser cíclica tras λ muestras *[12]*.

Basta entonces añadir al algoritmo únicamente la parte en la que una vez que ha encontrado que está en el ciclo, devuelva el λ correspondiente al periodo de la secuencia.

Este sencillo código pyton encuentra el ciclo y muestra la longitud del mismo iterando en los pasos que tarda en volver a llegar al punto del otro puntero.

```
def detect_cycle_and_start(head):
    # The function returns a tuple (Flag of cycl
    tortoise = head
    hare = head

    # Determine if there is a cycle.
    while hare:
        tortoise = tortoise.next
        hare = hare.next
        if hare:
            hare = hare.next
            if tortoise is hare:
                print(tortoise)
                break
    else:
        return (False, None)

    # Determine the length of the tail mu.
    hare = head
    mu = 0
    while hare is not tortoise:
        hare = hare.next
        tortoise = tortoise.next
        mu += 1
    return (True, mu)
```

Ilustración 14. Algoritmo que encuentra un ciclo y su periodo *[12]*.

Es particularmente eficiente ya que es O(n) tiempo y O(1) espacio.

- Aplicación

De esta forma de encontrar bucles se toma la idea de que ambos punteros se estén moviendo, lo que posibilita que corra en tiempo real con un buffer FIFO almacenando los PRIs. Sin embargo no se puede emplear ya que la identificación no puede garantizar inmunidad a pulsos faltantes o corruptos y se puede dar el caso que dentro de una secuencia Stagger mayor, se den varias veces distintas secuencias más pequeñas, y debemos hacer el tracking y predecir dentro del periodo completo para acertar incluso en el final o cambio de sub-secuencia.

2.1.3 APLICACIÓN DE LA AUTOCORRELACIÓN

La correlación es un proceso empleado ampliamente en telecomunicaciones para identificación emparejamientos y sincronización de emisores y transmisores. La autocorrelación es la correlación cruzada de una señal consigo misma.

- Resumen

Es una operación que da como resultado una medida de la similaridad entre dos series, sin embargo también la podemos utilizar para calcular el periodo comparando la serie de referencia consigo misma retrasada un número de tiempos de muestreo.

- Procedimiento y explicación

Disponer de copias retrasadas de una secuencia discreta es inmediato en un sistema electrónico digital. Para hacer la operación de autocorrelación, sólo hay que hacer la correlación cruzada entre la secuencia más reciente y las sucesivas copias retrasadas y registradas.

Representando un ejemplo de aplicación de la autocorrelación para la identificación de periodicidad [13] arroja un resultado como el siguiente.

Dada esta secuencia de valores en el tiempo, que representa el uso de electricidad en una casa monitorizada cada hora:

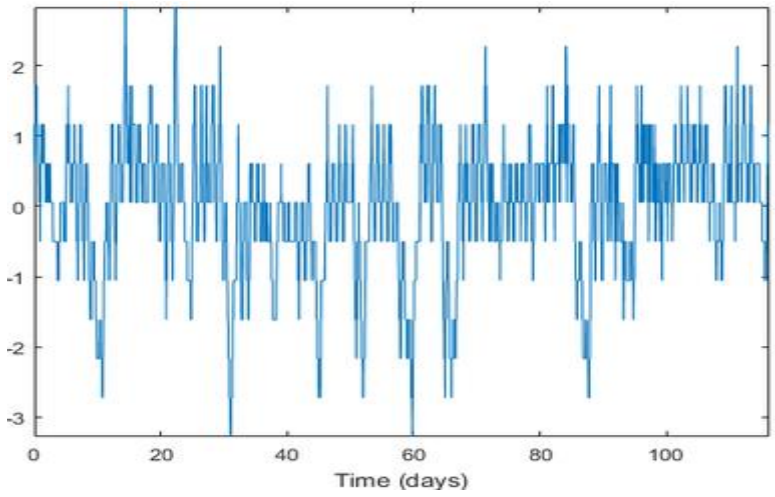


Ilustración 15. Secuencia a autocorrelar.

Si realizamos sobre ella una autocorrelación, obtendremos este resultado:

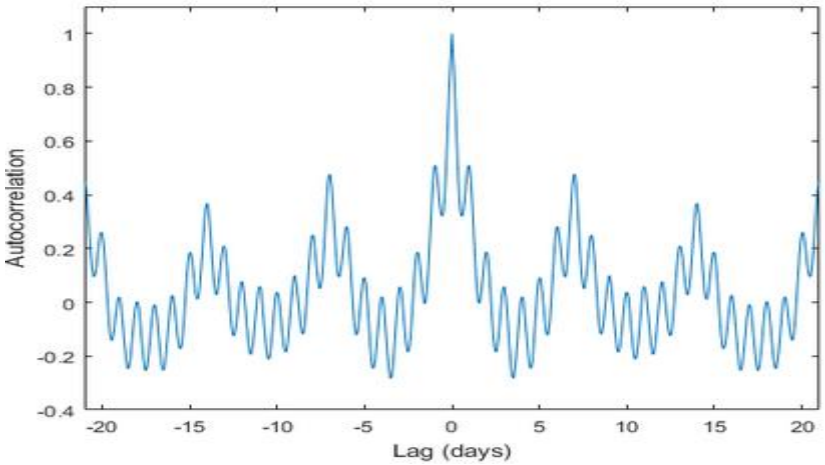


Ilustración 16. Resultado de la autocorrelación.

Primero vemos la analogía Lag (retraso), con una señal que se registra sucesivamente. Para obtener la información relevante sobre periodicidad, se tienen que examinar los máximos.

Si tenemos en cuenta solo los números naturales para "días", el máximo absoluto se encuentra en 7, que en la secuencia analizada se intuye visualmente que es el periodo fundamental.

Además, analizando los máximos locales encontramos periodicidad 1, debido a los cambios noche-día dentro de un mismo día.

Para tiempo discreto, tal como es el procesado digital, la operación de correlación entre las funciones f y g se define como:

$$(f \star g)[n] \stackrel{\text{def}}{=} \sum_{m=-\infty}^{\infty} f^*[m]\ g[m+n].$$

Ilustración 17. Función de correlación cruzada.

Se puede ver entonces que la implementación en una FPGA [14] será previsiblemente simple, con pocos recursos y operación con alto *throughput*[15].

[15] Throughput – Cantidad de información procesada por unidad (normalmente de tiempo).

- Proceso de implementación

Para obtener el HDL que describa la operación, se reduce a los elementos aritméticos más básicos. Estaremos muestreando la misma secuencia X pero la consideraremos distinta de su retrasada y se llama Y.

$\{X(i)\}$
$\{Y(i)\}$ Siendo i= 0, 1, 2,…, 4N-1

Cuando se paraleliza la secuencia por 1:4 se computa de la siguiente forma:

$\{X_a(n), X_b(n), X_c(n), X_d(n)\}$
$\{Y_a(n), Y_b(n), Y_c(n), Y_d(n)\}$

$$R_0(j) = \frac{1}{4N}\left\{\sum_{n=0}^{N-1} X_a(n-j)Y_a(n) + \sum_{n=0}^{N-1} X_b(n-j)Y_b(n) + \sum_{n=0}^{N-1} X_c(n-j)Y_c(n) + \sum_{n=0}^{N-1} X_d(n-j)Y_d(n)\right\}$$

$$R_1(j) = \frac{1}{4N}\left\{\sum_{n=0}^{N-1} X_d(n-1-j)Y_a(n) + \sum_{n=0}^{N-1} X_a(n-j)Y_b(n) + \sum_{n=0}^{N-1} X_b(n-j)Y_c(n) + \sum_{n=0}^{N-1} X_c(n-j)Y_d(n)\right\}$$

$$R_2(j) = \frac{1}{4N}\left\{\sum_{n=0}^{N-1} X_c(n-1-j)Y_a(n) + \sum_{n=0}^{N-1} X_d(n-1-j)Y_b(n) + \sum_{n=0}^{N-1} X_a(n-j)Y_c(n) + \sum_{n=0}^{N-1} X_b(n-j)Y_d(n)\right\}$$

$$R_3(j) = \frac{1}{4N}\left\{\sum_{n=0}^{N-1} X_b(n-1-j)Y_a(n) + \sum_{n=0}^{N-1} X_c(n-1-j)Y_b(n) + \sum_{n=0}^{N-1} X_d(n-1-j)Y_c(n) + \sum_{n=0}^{N-1} X_a(n-j)Y_d(n)\right\}$$

Se traducen las operaciones aritméticas básicas obtenidas, en HDL para FPGA. Hecho esto se puede generar el esquemático RTL del diseño realizado.

Se obtiene el siguiente diagrama para el correlador paralelo.

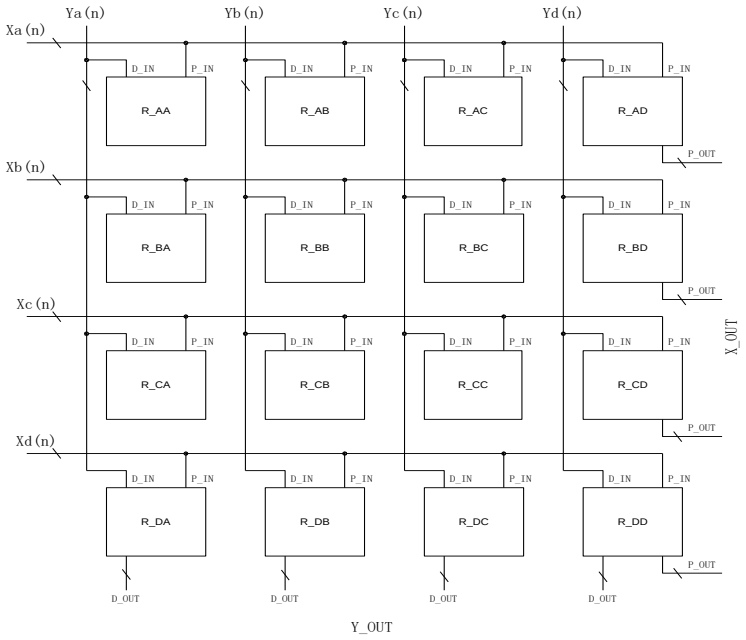


Ilustración 18. Correlador paralelo, esquema del HDL.

En los experimentos [14] en los que implementaron el correlador en hardware real, se obtuvo un tiempo mínimo de ciclo de 3.9ns en una Virtex-2. Es un resultado que valida esta opción, ya que permite procesar a ciclo de FPGA que es mucho más rápido que el de las secuencias RADAR.

- Aplicación

Este método se presenta como fuerte candidato para su uso en la identificación del periodo de la secuencia.

-Presenta características deseables como son: Baja latencia en el cómputo, inmunidad a imperfecciones en la secuencia y escalabilidad.

-Si bien por otra parte puede llegar a requerir muchos recursos útiles en otras partes del sistema como multiplicadores o slices DSP.

2.1.4 HISTOGRAMA DE ESCALERA DE COMPARADORES

Tras el estudio y valoración de las distintas formas de resolver la identificación del periodo, se opta por una solución que incluye conceptos tomados de los métodos referenciados y es adaptada para adecuarse a los objetivos.

Se busca además de los atributos mencionados hasta ahora, que sea una solución parametrizable, de forma que se pueda ajustar antes de síntesis o compilación a un sistema de las dimensiones deseadas. Esta característica hace que sea adaptable a escenarios de distinta envergadura y permite explorar las cualidades del sistema.

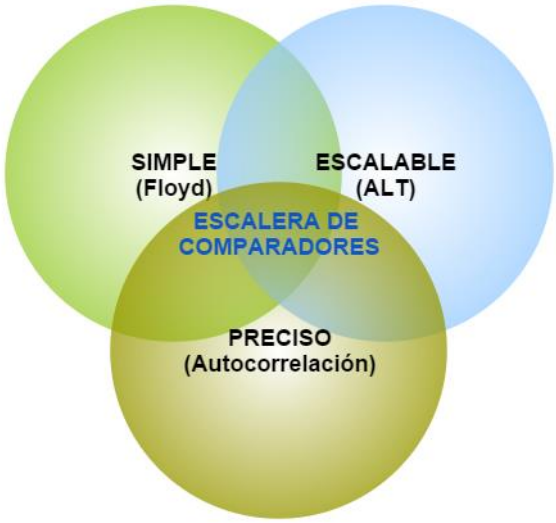


Ilustración 19. Confección del método elegido.

Partiendo de que la entrada a nuestro sistema es la información proporcionada por el sistema ESM, en el que se separan los pulsos por emisor empleando variables como el ángulo de recepción del pulso a la antena, procesamos de la siguiente manera la información. El sistema, en función de las capacidades del receptor RADAR, generará T tramas, una por cada amenaza.

- Una primera etapa, hace la conversión inmediata entre Tiempo de Llegada (TOA[16]), y PRI. El PRI se obtiene simplemente como la diferencia del último TOA con el anterior.

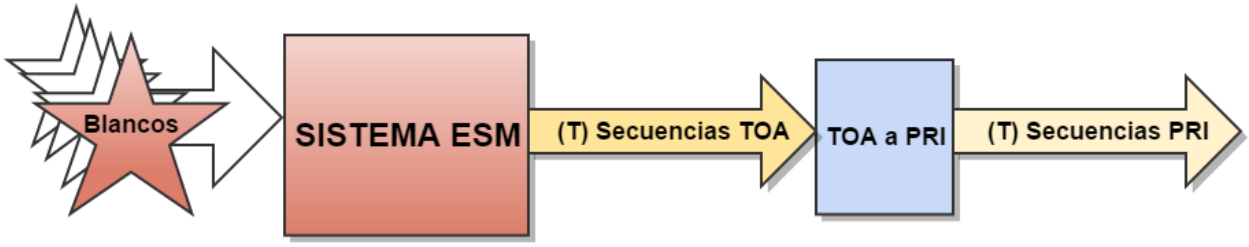


Ilustración 20. Diagrama secuencias de entrada.

- Se crearán T buffer(s) FIFO, de tamaño a parametrizar, que almacenará los pulsos en orden de llegada. Todos los elementos del buffer se acceden en paralelo, teniendo así acceso a las últimas S muestras ordenadas temporalmente.

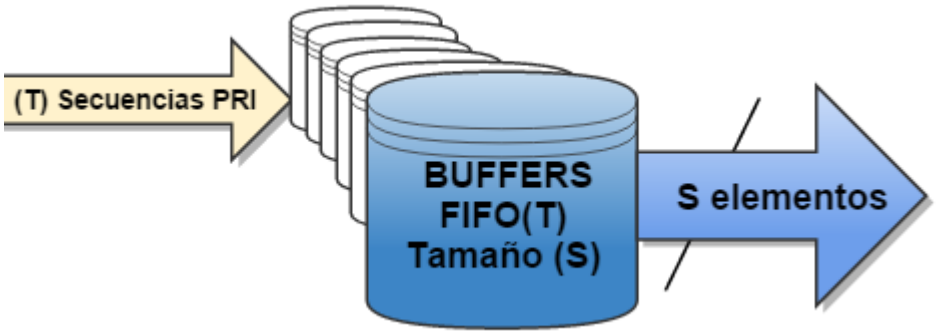


Ilustración 21. Buffer con salida paralelo S palabras.

- Cada elemento de los S que componen un buffer, se conectará a comparadores siguiendo el siguiente razonamiento. Si llamamos N al número de posiciones Stagger que corresponden a un periodo completo de la secuencia, debemos tener un buffer en el que al menos quepa un periodo completo, a mayor tamaño del buffer y los comparadores asociados, mayor robustez tendrá el cálculo del periodo.

[16] TOA – Time Of Arrival. Tiempo de llegada de un pulso.

Identificaremos el número de veces que se repiten los elementos para cada distancia entre ellos. Sabemos que en una secuencia periódica, cada N elementos, estos se repetirán. Sin embargo algoritmos simples como el de Floyd, podrían identificar una repetición en la que una sub-secuencia se repita dentro de un periodo mayor. Esto es, para la secuencia [2 4 6 4 6 8 2 4 6 4 6 8 2 4 6 4 6 8] vemos que la sub-secuencia [4 6] de periodo 2 se repite varias veces, siendo en realidad el periodo total de 6.

Una primera aproximación sería comparar todos los elementos entre sí y comprobar el número de repeticiones. La distancia N que más se repita, sería el periodo. Sin embargo multitud de casos en la secuencia, como por ejemplo una secuencia larga con una sub-secuencia corta que se repite muchas veces, darían lugar a resultados erróneos.

Como se ha deducido en varios otros algoritmos, se mejora la robustez ante este tipo de secuencias particulares, cuando las comparaciones se realizan en un rango variante de elementos. Esto es, iremos disminuyendo sucesivamente la distancia máxima entre comparaciones. Esto además, reduce considerablemente los recursos necesarios para implementación en hardware. Si llamamos BUF_SIZ al tamaño del buffer, donde se requerían BUF_SIZ^2 comparadores, ahora se requerirán sólo BUF_SIZ *∑i para i=1 hasta BUF_SIZ.

Se deberá reproducir entonces, para el tamaño de buffer escogido, la siguiente estructura de comparadores:

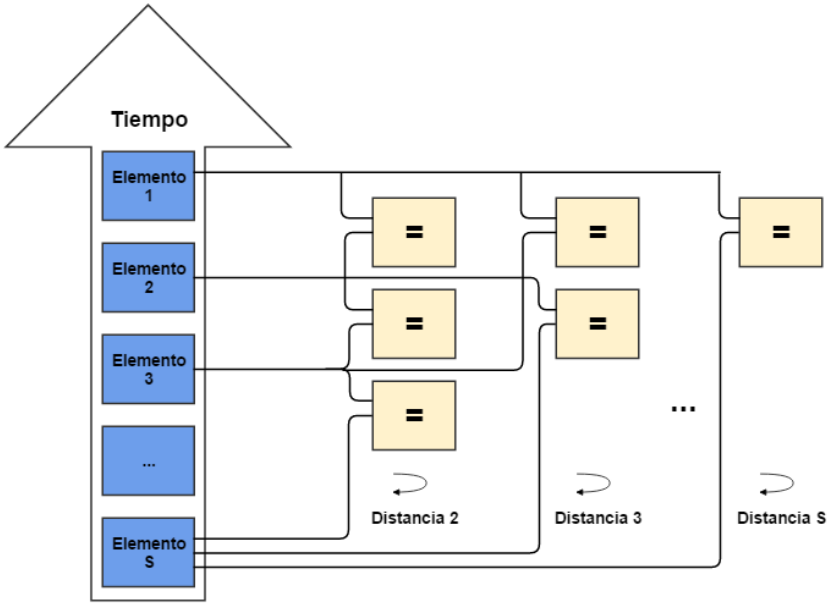


Ilustración 22. Estructura de comparadores.

Como se indica en el diagrama, los comparadores se sitúan a una distancia creciente en el buffer. De la misma forma, a medida que crece se reduce el número posible de comparaciones de ese tamaño, y se reduce el número de comparadores usados. En esta disposición se ve claramente entonces que si se suman las cuentas registradas por cada columna, se obtiene el número de ocurrencias en las que a las *n* posiciones, se repite el elemento. Además, dado que todas las comparaciones son concurrentes, y por otro lado también lo son las sumas, sólo se produce un retardo de un ciclo para obtener el resultado de las cuentas de las comparaciones.

Si el buffer tiene el tamaño suficiente, ya que todos los elementos a partir del número N se repetirán hasta el final del buffer, las cuentas para el tamaño real de la secuencia, N, superarán en número a cualquiera de las sub-tramas que pudieran estar incluidas o no en un periodo.

- Para realizar una adaptación a las medidas del mundo real, se han de tener en cuenta las tolerancias en la medida y el ruido introducido en la generación del pulso. Esto da lugar a valores que no son exactos y para ello se adapta la operación de comparación de la siguiente manera.

Las variables a tener en cuenta [15] siguen una distribución normal con media 0. Son el ruido de sistema, con desviación típica σ_w^2 y el ruido de medida con desviación σ_v^2. Tal como se detalla posteriormente, la varianza del PRI una vez ha sido medido y lo vamos a procesar es de $\sigma_w^2 + 2\ \sigma_v^2$. El valor que se elige por conveniencia es de 4 desviaciones estándar, que englobará la práctica totalidad de los pulsos absorbiendo las variaciones anteriores. Tomamos este valor de referencia de 4 desviaciones estándar porque será una constante en los siguientes cálculos.

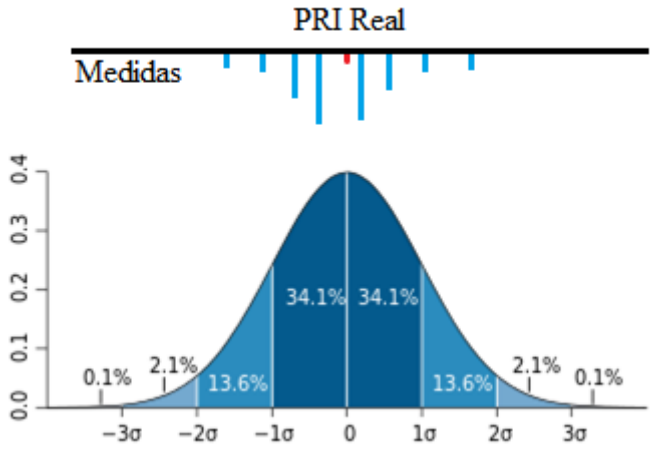


Ilustración 23. Con 4 sigmas del ruido combinado de sistema y de medida, se cubre la totalidad de los pulsos que siguen una distribución normal.

La estructura de comparadores será la misma. Únicamente se sustituirá el comparador simple, válido en el caso ideal, por uno de los dos siguientes mecanismos ideados para resolver el problema:

- Truncamiento.
  -Es una aproximación menos exacta pero reduce con mucho el número de elementos hardware necesario para su implementación.

  Se puede codificar el PRI mínimo y máximo esperable en una resolución preferida con un número arbitrario de bits, preferiblemente elevado. Se toma el valor de 4 sigmas del ruido presente en el elemento. A continuación se redondea el valor de sigma, para hacerlo coincidir con el peso correspondiente al que representa un número *t* de bits. Estos bits son truncados de todas las medidas y se emplean comparadores simples para el resto de bits.

  No se emplea esta técnica por ser menos precisa, pero se indica como alternativa para una situación en la que haya gran limitación de recursos.

- Umbrales.
  -Resultados de precisión a costa de hacer el proceso de comparación más complejo. Aun así, sigue empleando recursos simples y abundantes en cualquier FPGA.

  Para absorber la tolerancia aceptable en la medida se efectúa en primer lugar, y sólo es necesario hacerlo una vez, el valor correspondiente a 4 desviaciones estándar en la medida.

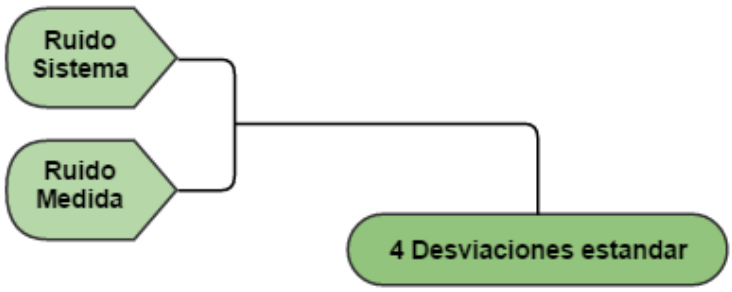


Ilustración 24. Cálculo tolerancia por ruido.

Disponiendo de este valor, todos los bloques de comparador simple serian sustituidos por el siguiente subsistema, que permite una diferencia entre los elementos siempre que sea menor a la desviación calculada:

Ilustración 25. Comparador con umbral.

- Una vez realizadas las comparaciones, obtenemos las cuentas para cada periodo encontrado dentro del buffer. Este es el histograma de periodos. Simplemente queda ahora seleccionar el que acumule más cuentas, y si se ha elegido el tamaño de buffer suficiente para hacerlo robusto a las condiciones de operación esperadas, este será el periodo de la secuencia Stagger, N.

Ilustración 26. Conjunto identificador de periodo.

2.2 ESTIMACIÓN DEL SIGUIENTE VALOR EN LA SECUENCIA: FILTRO RECURSIVO KALMAN

Se han desarrollado muchos tipos de filtros recursivos, uno de los más extendidos es el de Kalman, también conocido como estimador lineal cuadrático. Se comenzó a desarrollar en 1960 y a día de hoy es muy ampliamente usado en economía, control y procesado de señal entre otros. Posee características muy deseables, una estructura bastante óptima, es fácilmente adaptable a operación en tiempo real y consigue una mejora muy notable de la inmunidad al ruido. El aspecto más interesante y por el que se decide emplear este método, es que efectúa una predicción del estado posterior para iterativamente mejorar la media. Esta predicción precisamente, será la salida de nuestro sistema.

- Modelado del ruido

Como introducíamos en el identificador de periodo, es necesario caracterizar el tipo de ruido que vamos a encontrar en nuestro sistema. Se puede modelar [16] de dos formas, dependiendo las características que se espera que tengan el emisor y el receptor, del cual obtenemos la información.

-Si se espera que el predominante sea el ruido introducido por el receptor, ya que la fuente emite con gran estabilidad, se usaría el modelado para Tiempo de Llegada siguiente:

$$t_j = (j-1)T + t_0 + e_j, \qquad j = 1, \dots, N.$$

Siendo T el PRI básico, y e_j ruido Gaussiano con media cero.

El ruido Gaussiano modela la acción aleatoria de diversos factores físicos independientes entre sí, cuyos efectos se ven añadidos a la variable de interés. Sigue la siguiente función de densidad de probabilidad:

$$p_G(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(z-\mu)^2}{2\sigma^2}}$$

Ilustración 27. PDF Ruido Gaussiano.

La distribución, que sigue se conoce como Gaussiana o también Normal, se caracteriza con la media μ, desviación típica σ y el parámetro *z*. Variando estos tres valores se cambia la forma de la curva, como en este ejemplo:

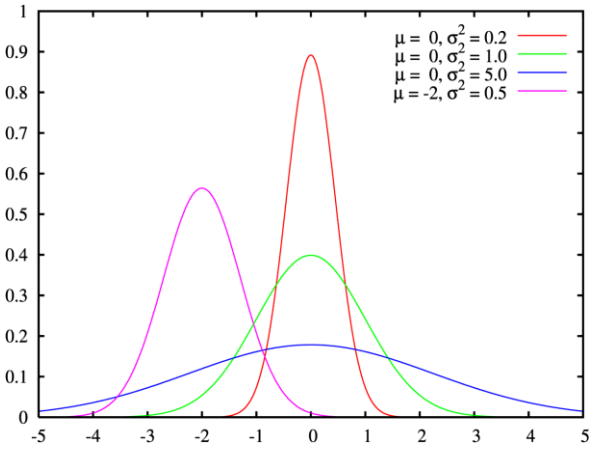


Ilustración 28. Curvas gaussianas con distintos parámetros *[17]*.

Es habitual encontrar ruidos debidos a variables aleatorias en sistemas de telecomunicación, debido a interferencias que se han de caracterizar de esta manera según su media y magnitud porque resulta imposible conocer su valor de otra manera.

-Sin embargo, es común que el emisor no emita con estabilidad que asume el modelado anterior, ya que muchas veces, incluso intencionadamente para intentar ser más difícilmente interceptados, los emisores añaden ruido blanco superpuesto en el PRI. Es por ello que es más conveniente el siguiente modelado para jitter[17] acumulativo.

$$t_j = t_{j-1} + T + v_j, \qquad j = 1, \dots, N.$$

Siendo v_j también ruido Gaussiano de media cero.

Se puede expresar también de forma más conveniente en diferencias de tiempo:

$$\tau_j = t_j - t_{j-1},$$

$$\tau_j = T + v_j, \qquad j = 2, \dots, N.$$

Para modelar el sistema en variables de estado, adaptándolo a la estructura del filtro Kalman, con la nomenclatura que empleamos, basta con identificar las ecuaciones que modelan la llegada de un nuevo pulso e incorporar el ruido de sistema w_j y el ruido de medida v_j. Es inmediato encontrar la relación:

$$TOA_j = TOA_{j-1} + PRI_j + w_j$$

$$\tau_j = TOA_j + v_j$$

[17] Jitter – Variabilidad en el tiempo.

Siendo τ_j el TOA medido, y w_j y v_j distribuciones Gaussianas de media cero y desviación estándar σ_w^2 y σ_v^2 respectivamente, como se adelantaba en el cálculo del umbral de los comparadores. Para terminar de justificar el valor tomado en los comparadores como desviación estándar del valor medido ($\sigma_w^2 + 2*\sigma_v^2$), sólo hay que sustituir las dos ecuaciones anteriores en esta, ya que nos interesan los PRIs, que no son más que la diferencia entre sucesivos TOA:

$$PRI\_medido_j = TOA\_medido_j - TOA\_medido_{j-1}$$

Y se obtiene:

$$PRI\_medido_j = PRI_j + w_j + v_j - v_{j-1}$$

Como el ruido de medida v_j es una variable aleatoria, el término j y el término j-1 no se pueden sumar, sino que sus deviaciones se suman, resultando que la desviación estándar total en la medida de PRI es:

$$Var(PRI\_medido_j) = \sigma_w^2 + 2*\sigma_v^2$$

- Modelado del sistema

Conocidas las ecuaciones que modelan el sistema, se obtiene la expresión de las partes que conforman el filtro Kalman [15]. Extendiendo las ecuaciones de TOA y TOA medido, desde el momento en que se recibe el primer pulso (TOA_0), hasta el tiempo correspondiente a la recepción del pulso n, quedan de la forma:

$$TOA_n = TOA_0 + \sum_{i=1}^{n} PRI_i + \sum_{i=1}^{n} w_i$$

$$\tau_n = \sum_{i=1}^{n} PRI_i + \sum_{i=1}^{n} w_i + v_n$$

Este es el funcionamiento esquematizado del filtrado que se empleará. Como se hace notar en la figura, nuestro mayor interés está en la estimación que hace a cada paso del siguiente estado, lo que para nosotros significa una predicción del Tiempo de Llegada (TOA) del próximo pulso.

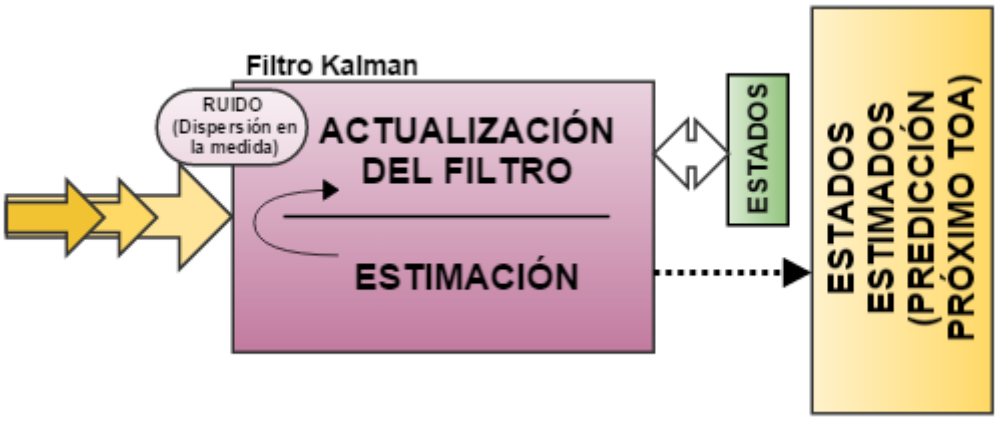


Ilustración 29. Uso del filtro Kalman como predictor.

En el caso general, el filtro Kalman consta de las siguientes dos partes:

-Estimación del estado del sistema:

$$X_k = F_{k-1}X_{k-1} + G_{k-1}u_{k-1} + w_{k-1}$$

Donde X es la matriz con los estados del sistema, F es la matriz de transición de estados, G la de control y u el vector de entrada.

-Observador (medida):

$$Y_k = H_kX_k + v_k$$

Donde H es la matriz de medida.

Se modela cada posición PRI como un *estado.* Si se ha encontrado que la secuencia tiene periodo N, entonces la matriz de estado X tendrá tamaño Nx1. Para realizar el filtrado del ruido y la estimación de estado siguiente en una secuencia que consta de muchos elementos, debemos entonces haber inferido el periodo con el sistema anterior.

El tamaño de la matriz de estados determina el orden del filtro Kalman y por tanto tamaño del resto de matrices del filtro. Se muestra a continuación la expresión que toma cada una [15] en función del número de posiciones de la secuencia, y del índice de periodo *k* en el que nos encontramos. Cada N posiciones recibidas, *k* aumenta 1.

- Matriz de medida, H_k.

Sabiendo que los TOA son el resultado de la suma sucesiva de los PRI cada periodo *k*, es inmediato obtener:

$$H_k = \begin{bmatrix} k & k-1 & k-1 & k-1 & \dots & k-1 \\ k & k & k-1 & k-1 & \dots & k-1 \\ k & k & k & k-1 & \dots & k-1 \\ \vdots & & & & & \\ k & k & k & k & \dots & k \end{bmatrix}_{NxN}$$

Ilustración 30 Expresión de *Hk*.

- Matriz de covarianza de medida, R_k.

Para afinar la capacidad del filtro para rechazar el ruido, incluye la matriz de covarianza de la medida, que almacena el ruido aditivo esperable en cada medida, tal como se había escogido al modelar el ruido que se esperaría en el sistema.

Estudiando un ejemplo, en el primer periodo *k*, de N=3 pulsos, se encuentran los siguientes ruidos añadidos a la medida con la ecuación de observación:

$$\underbrace{\begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix}}_{Y_1} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}}_{H_1} \underbrace{\begin{bmatrix} PRI_1 \\ PRI_2 \\ PRI_3 \end{bmatrix}}_{X_1} + \begin{bmatrix} w_1 + v_1 \\ w_1 + w_2 + v_2 \\ w_1 + w_2 + w_3 + v_3 \end{bmatrix}$$

La desviación en esa medida es entonces:

$$R_1 = \begin{bmatrix} \sigma_w^2 + \sigma_v^2 & \sigma_w^2 & \sigma_w^2 \\ \sigma_w^2 & 2\sigma_w^2 + \sigma_v^2 & 2\sigma_w^2 \\ \sigma_w^2 & 2\sigma_w^2 & 3\sigma_w^2 + \sigma_v^2 \end{bmatrix}$$

Tras recibir N elementos más, *k*=2 y tomará la forma:

$$\underbrace{\begin{bmatrix} \tau_4 \\ \tau_5 \\ \tau_6 \end{bmatrix}}_{Y_2} = \underbrace{\begin{bmatrix} 2 & 1 & 1 \\ 2 & 2 & 1 \\ 2 & 2 & 2 \end{bmatrix}}_{H_2} \underbrace{\begin{bmatrix} PRI_1 \\ PRI_2 \\ PRI_3 \end{bmatrix}}_{X_2} + \begin{bmatrix} w_1 + w_2 + w_3 + w_4 + v_4 \\ w_1 + w_2 + w_3 + w_4 + w_5 + v_5 \\ w_1 + w_2 + w_3 + w_4 + w_5 + w_6 + v_6 \end{bmatrix}$$

Y las desviaciones en la medida de los TOAs acumuladas serán:

$$R_2 = \begin{bmatrix} 4\sigma_w^2 + \sigma_v^2 & 4\sigma_w^2 & 4\sigma_w^2 \\ 4\sigma_w^2 & 5\sigma_w^2 + \sigma_v^2 & 5\sigma_w^2 \\ 4\sigma_w^2 & 5\sigma_w^2 & 6\sigma_w^2 + \sigma_v^2 \end{bmatrix}$$

La expresión normalizada observando la evolución de R_k y en función de N es:

$$R_k = \begin{bmatrix} (Nk-N+1)\sigma_w^2 + \sigma_v^2 & (Nk-N+1)\sigma_w^2 & \dots & (Nk-N+1)\sigma_w^2 \\ (Nk-N+1)\sigma_w^2 & (Nk-N+2)\sigma_w^2 + \sigma_v^2 & \dots & (Nk-N+2)\sigma_w^2 \\ \vdots & \vdots & & \vdots \\ (Nk-N+1)\sigma_w^2 & (Nk-N+2)\sigma_w^2 & \dots & (Nk)\sigma_w^2 + +\sigma_v^2 \end{bmatrix}_{NxN}$$

Ilustración 31. Expresión de *Rk*.

- Matriz de covarianza del error inicial, P_0.

De forma similar, se obtiene la covarianza del error, que es la diferencia entre el estado X_k y el estado estimado, denotado como $\hat{X}$. La expresión del estado estimado es sencilla, igual que se hacía para identificar la desviación estándar del error, en forma matricial será:

$$\hat{X}_0(1) = Y_1 - Y_0 = PRI_1 + w_1 + v_1 - v_0$$
$$\hat{X}_0(2) = Y_2 - Y_1 = PRI_2 + w_2 + v_2 - v_1$$
$$\hat{X}_0(3) = Y_3 - Y_2 = PRI_3 + w_3 + v_3 - v_2$$

$$\hat{X}_0 = \begin{bmatrix} PRI_1 + w_1 + v_1 - v_0 \\ PRI_2 + w_2 + v_2 - v_1 \\ PRI_3 + w_3 + v_3 - v_2 \end{bmatrix}$$

Y la diferencia con los PRI reales es:

$$X_0 - \hat{X}_0 = \begin{bmatrix} PRI_1 \\ PRI_2 \\ PRI_3 \end{bmatrix} - \begin{bmatrix} PRI_1 + w_1 + v_1 - v_0 \\ PRI_2 + w_2 + v_2 - v_1 \\ PRI_3 + w_3 + v_3 - v_2 \end{bmatrix} = \begin{bmatrix} -w_1 - v_1 + v_0 \\ -w_2 - v_2 + v_1 \\ -w_3 - v_3 + v_2 \end{bmatrix}$$

Por lo que P_0, el valor inicial de la covarianza del error de predicción es:

$$P_0 = \begin{bmatrix} \sigma_w^2 + 2\sigma_v^2 & -\sigma_v^2 & 0 \\ -\sigma_v^2 & \sigma_w^2 + 2\sigma_v^2 & -\sigma_v^2 \\ 0 & -\sigma_v^2 & \sigma_w^2 + 2\sigma_v^2 \end{bmatrix}$$

Ilustración 32. Expresión de *Po*.

- Los estados son constantes, puesto que cada elemento es un estado, y se intenta obtener la estimación y filtrar el ruido permaneciendo el valor real de cada posición del periodo.

  $X_k = X_{k-1}$, y la covarianza del ruido de proceso, $Q = 0$.

Matrices del filtro Kalman.

Definido y modelado el sistema en variables de estado, así como los valores de las matrices de condiciones iniciales, se obtienen las matrices con las que se realiza el proceso iterativo. Cada vez que se decida comenzar, parar y reiniciar el seguimiento del tren de pulsos, en un primer paso, quedan fijadas las condiciones iniciales. A partir de ahí, cada periodo Stagger, tal como indica el algoritmo de filtrado de Kalman, las siguientes matrices se operan para obtener la estimación de próximo TOA.

- Covarianza del residuo, S_k.

$$S_k = H_k \hat{P}_k^- H_k^T + R_k$$

- Ganancia Kalman, K_k.

$$K_k = \hat{P}_k^- H_k^T S_k^{-1}$$

- Residuo o innovación. Se obtiene como la diferencia entre los estados medidos, y los que se habían estimado:

$$\text{Inn} = Y_k - H_k \hat{X}_k^-$$

- Actualización de la predicción de estados:

$$\hat{X}_k = \hat{X}_k^- + K_k \, \text{Inn}$$

- Covarianza del error de predicción de estado actualizada, P_k.

$$\hat{P}_k = \hat{P}_k^- - K_k S_k K_k^T$$

Queda con esto definida la aritmética necesaria para identificar el periodo de la secuencia, N. Identificado este, se inicializan las matrices iniciales a partir de los pulsos del primer periodo, y se inicia el contador de periodos transcurridos, quedando así también definidas las operaciones de filtrado de la medida y predicción de próximo estado.

CAPÍTULO 3:

ESTUDIO PARA ESTABLECIMIENTO DE REQUERIMIENTOS TÉCNICOS DE UN SISTEMA ECM

CAPÍTULO 3: ESTUDIO PARA ESTABLECIMIENTO DE REQUERIMIENTOS TÉCNICOS DE UN SISTEMA ECM

Es conveniente estudiar cuantitativamente los efectos involucrados en un sistema antes de diseñarlo, para poder asignar recursos y esfuerzos de forma conveniente. Al diseñar un sub-sistema de ECM o un sistema que interactúe con él, se ha de identificar de entre los efectos expuestos a continuación, los que afectan a las áreas de actuación del sistema a diseñar. Hecho esto, con los objetivos de funcionamiento deseados, se pueden calcular los requisitos técnicos.

El tipo más básico de contramedida es el *jamming* por ruido. Sin embargo son deseables técnicas más elaboradas que emplean sistemas como memorias de radiofrecuencia y el predictor en cuestión para permitir introducir información falseada en el receptor objetivo. En este capítulo se introduce en primer lugar en detalle qué es un sistema ECM y cómo opera, ya que es al que el Predictor va a prestar apoyo.

En la siguiente parte, se exponen conceptos y fórmulas que asistan al establecimiento de requerimientos técnicos de un sistema de Contramedidas Electrónicas (ECM) o uno que interactúe con él.

3.1 SISTEMA ECM: DESCRIPCIÓN Y CONCEPTOS GENERALES

Se conoce como sistema ECM, o Sistema de Contramedidas Electrónicas, a aquel sistema de guerra electrónica capaz de actuar sobre la emisión de un radar y ocasionarle una perturbación o engaño en sus resultados.

Hasta pasada la segunda guerra mundial era poco lo que se podía hacer para evitar que un radar obtuviera datos fiables de los blancos a los que iluminaba. Típicamente, existían dos tipos de contramedidas:

- Chaff: Consiste en liberar una nube de pequeños elementos metálicos o metalizados para provocar una gran cantidad de señal reflejada hacia el radar y poder ocultarse en ella. También se utiliza para engañar a cabezas de misil guiadas por radar para que exploten en la nube metálica en vez de en el objetivo.

- Flare: Al igual que en el caso de Chaff, la idea consiste en liberar bengalas ardiendo o incandescentes para engañar a cabezas de misiles guiadas por infrarrojos. Se libera una bengala para que haga de señuelo y el misil siga a la misma en vez de al blanco.

Con el desarrollo de la tecnología radar vino aparejado un gran desarrollo del mundo de la guerra electrónica. Se desarrollaron sistemas capaces de detectar emisores radares en tiempo real (ESM), detectores de gran sensibilidad y capaces de analizar emisores con gran detalle (ELINT) y, como no, sistemas de contramedidas (ECM) capaces de generar emisiones radioeléctricas para intentar engañar y/o perturbar a los radares seleccionados. En cuestión de poco tiempo pasamos de protegernos liberando al aire metal o material incandescente a hacerlo generando señales en la banda de los radares enemigos.

Las primeras contramedidas electrónicas que se desarrollaron fueron las técnicas de perturbación mediante ruido en banda, también llamado ruido de barrera o BJAM. El concepto es sencillo: dado que el radar va a recibir relativamente poca energía debida a la reflexión de energía sobre el blanco, y la agilidad en frecuencia de los emisores radar en el pasado no era demasiado grande, el sistema de contramedidas sólo tiene que transmitir un ruido de alta potencia en la banda del radar para que este sea incapaz de reconocer los ecos de su señal transmitida. De esta manera, la señal reflejada en el blanco queda enmascarada por el ruido transmitido y el radar es incapaz de realizar una detección fiable. A día de hoy sigue siendo una contramedida de las más utilizadas pero, en caso de realizarse una ejecución a destiempo (cuando el radar te está iluminando desde una distancia grande y aún no ha sido capaz de detectar el blanco) , puede tener un efecto pernicioso pues estaríamos radiando energía y provocando una detección no deseada (nos convertimos en un radio-faro).

A medida que fue avanzando la tecnología, y los sistemas eran capaces de responder en tiempos mucho más cortos, se comenzó el desarrollo de las técnicas de engaño y los módulos DRFM o Memorias Digitales de Radiofrecuencia. A diferencia del caso de las técnicas de perturbación, el objetivo de estas últimas es la de generar blancos falsos al radar y ocultar nuestra posición real. Con el desarrollo de las memorias digitales de radiofrecuencia se abrió un enorme abanico de posibles contramedidas a realizar:

- Range Gate Pull In/Out o RGPI/O:

  Esta contramedida consiste en digitalizar la señal emitida por el radar y retransmitir una réplica en un instante distinto al que, por naturaleza, se reflejaría en el blanco. Si se retransmite la réplica pasado el instante en el que se produciría el reflejo estaríamos hablando de un RGPO y, por tanto, provocando que el radar viera el blanco más lejos, puesto que aumentamos el tiempo de vuelo de la señal. Y si la retransmitimos un poco antes de que llegue la señal al blanco, estaríamos acortando el tiempo de vuelo y provocando que el radar viera el blanco más cerca de lo que lo está en realidad. Parece evidente decir que el transmitir antes de que llegue el pulso es sencillo cuando hablamos de radares con PRI fijo, pero cuando hablamos de PRIs complejos la problemática aumenta cualitativamente y se requiere la asistencia del sistema Predictor.

Para que la contramedida sea efectiva, y se produzca realmente un engaño, es necesario seguir una serie de fases:

- Fase de solape y/o robo de puerta: Al comenzar la contramedida se debe suponer que se está siendo detectado por el radar. El algoritmo de tracking del radar esperará por tanto que la posición del blanco no cambie en exceso y, por supuesto, dentro de las capacidades que se le presuponen al blanco en cuanto a velocidad y aceleración. Es por esto que la contramedida comienza radiando pulsos justo encima de los ecos reales (solape). Como ahora la señal recibida por el radar va a tener una mayor potencia, este ajustará sus ganancias para "ver" correctamente al blanco. Disminuirá sus ganancias para evitar las saturaciones y dejará de ver el blanco real para ver sólo el pulso transmitido por el sistema de contramedidas. A este efecto se le denomina "robo de puerta", entendiendo por puerta el margen temporal en el que el radar espera recibir el eco del blanco. Una vez se "ha robado la puerta" podemos comenzar a desplazar el falso blanco.

- Fase de aceleración: Una vez robada la puerta, debemos empezar a desplazar el eco de forma creíble. Para ello simularemos que el blanco comienza a moverse con una cierta aceleración hacia fuera (RGPO) o hacia dentro (RGPI). En esta fase se irán calculando los instantes temporales correspondientes a la distancia relativa entre el radar y el falso blanco y se retransmitirán los pulsos en dichos instantes. En esta fase, si el robo de puerta ha sido satisfactorio, el radar habrá dejado de ver el blanco real y estará siguiendo al falso blanco que se separa del blanco real.

- Fase de velocidad constante: Una vez acelerado el falso blanco, cuando alcanza una velocidad determinada, se mantiene la velocidad hasta que logramos desplazar el falso blanco una determinada distancia.

- Fase de deceleración: Al igual que sucede en el caso de la fase de aceleración, cuando ya hemos alejado suficiente el falso blanco, entramos en una fase de deceleración para intentar mantener al falso blanco alejado de una manera creíble.

- Fase final: El falso blanco se mantiene fijo a una determinada distancia relativa al blanco real hasta que finalice la contramedida.

Lo explicado en este apartado respondería al comportamiento de una ley RGPI/O sencilla, pero existen multitud de variantes y combinaciones. Por ejemplo, se puede dejar un falso blanco permanentemente en la fase final mientras se comienza la ejecución la ley otra vez (gancho), o se pueden ejecutar múltiples leyes distintas simulando más de un blanco moviéndose a nuestro alrededor (RANRAP), se puede generar un RGPI con mucha aceleración y velocidad para simular lanzamientos de misiles (aunque esto es muy peligroso en situaciones reales), etc…

- Velocity Gate Pull Out/In:

Un efecto que se mide en radar para calcular la velocidad relativa del blanco detectado es el efecto Doppler. Esta contramedida se basa en modificar ligeramente la frecuencia en la que se radia la réplica del pulso recibido para que el radar estime de manera equivocada la velocidad del blanco. En este caso, en vez de modificar el tiempo, se modifica únicamente la frecuencia. El efecto que se busca en el radar es perturbar la estimación de velocidad y, con ello, el instante temporal en el que el radar esperaría encontrar el blanco en la siguiente iluminación. Al no cuadrar la velocidad medida con la distancia medida en el blanco siguiente, el radar no sería capaz de "enganchar" el blanco. Esta contramedida es especialmente útil en caso de ser iluminado por un radar en caso de lanzamiento de un misil. Si el radar no es capaz de enganchar el blanco, no podrá calcular trayectorias de lanzamiento fiables y, dado que los misiles son recursos limitados, no realizará el lanzamiento.

Al igual que en el caso de los RGPO/I, las ejecución de una ley VGPO/I se realiza por fases para mantener la credibilidad de la señal lo máximo posible. Las fases serían las mismas pero sustituyendo el tiempo por frecuencia en cuanto a variable a modificar. También se utiliza esta técnica en conjunción con los RGPO/I para simular los efectos Doppler de los pulsos retransmitidos en las fases de aceleración y velocidad constante y obtener una señal más realista.

- SJAM, NCP y otras técnicas de ruido coherente:

En el caso de las contramedidas de perturbación estábamos hablando de generar un ruido de una gran potencia y ancho de banda que no tenía por qué ser coherente con la señal del radar. El objetivo era bloquear e impedir la operativa radar en una cierta banda. Por ello, se generaba ruido durante un cierto porcentaje del tiempo y no tenía por qué coincidir con la señal del radar. Hay que entender que tener ruido el 100% del tiempo no sólo impide trabajar al radar enemigo, puede desensibilizar nuestros sistemas también e impedir nuestra operativa. Es por ello que, dado que se tiene capacidad de operar con la señal del radar (en tiempo y frecuencia), aparecen contramedidas cuyo objetivo es generar ruido de menor ancho de banda y coherente en frecuencia y tiempo con la señal del radar: SJAM o Spot Jamming genera ruido totalmente coherente con el pulso del radar y NCP (por ejemplo) genera ruido de banda estrecha comenzando un poco antes del comienzo del pulso real del radar y acabando un poco después, provocando que el pulso quede enmascarado en ruido pero sólo en ciertos instantes temporales, permitiendo así la operatividad del resto de sistemas.

Las técnicas descritas en este apartado son una pequeña muestra de todas las nuevas capacidades que tienen los sistemas de ECM actuales. Es importante hacer notar la importancia de determinar correctamente el tiempo y la frecuencia en la que vamos a recibir los pulsos del radar. En el caso de las técnicas de engaño en distancia con anticipación (RGPI) es necesario radiar el pulso correspondiente al falso blanco antes de haber recibido el transmitido por el radar y, si se quiere seguir una cierta ley, es importante no sólo transmitir el pulso, sino hacerlo en el instante correcto.

3.2 FENÓMENOS FÍSICOS PRESENTES EN EL USO DE RADAR

El principio básico de funcionamiento de un radar primario es simple. Se emite un pulso de radiofrecuencia, que es reflejado parcialmente si encuentra una superficie en su dirección de propagación. La medida del tiempo transcurrido en el trayecto de ida y vuelta, teniendo en cuenta la velocidad de propagación en el medio, determina la distancia a la que se encuentra el obstáculo.

- Función del tiempo de escucha, se obtiene fácilmente la primera ecuación trivial de radar. Determina la distancia en línea de visión al blanco, en función del tiempo de viaje del pulso (t), siendo c_o la velocidad de la luz:

$$Rango = \frac{c_0 * t}{2}$$

Esto impone una primera restricción a cualquier sistema RADAR, y por extensión a un ECM. El rango máximo frente a velocidad de refresco. En un radar sin adicionales mecanismos de mejora de la detección, el rango máximo equivale a la mitad de la distancia recorrida por el pulso en el tiempo entre pulsos, el PRI.

Aparece también un parámetro de gran importancia en el diseño de ECM. Como es habitual en sistemas montados en aeronaves, se comparte la antena, tendiendo un tiempo de transmisión (t_{tx}), y un tiempo de recuperación (t_{rec}) para poder escuchar el eco. El mínimo rango de medida. Este está relacionado con estos dos tiempos:

$$Rmin = \frac{c_0 * (t_{tx} + t_{rec})}{2}$$

Algunos tipos de contramedida requieren que se procese el pulso como máximo en este tiempo mínimo, para aplicar técnicas incluso a poca distancia del receptor.

En un radar de PRI constante se ha de sopesar la elección rango máximo contra frecuencia de refresco de la medida, ya que son parámetros inversos.

Una operación con secuencias complejas de PRI puede, entre otras ventajas, solventar este aspecto haciendo medidas con distintos tiempos inter-pulso, alcanzando mayores rangos y también rangos más cortos con mayor frecuencia de refresco. Este hecho motiva la transición de más radares a este modo de funcionamiento, lo que aumenta la necesidad de desarrollo de predictores para efectuar contramedidas en ellos.

- El ancho del pulso de *jamming* transmitido tiene una gran importancia. El procesado y filtrado de las mediciones que se hace es esencial para obtener el ancho de pulso preciso otorga dos ventajas significativas:
  - Se emplean los elementos de transmisión el tiempo necesario, permitiendo contramedir varios objetivos al mismo tiempo multiplexando el uso de la antena.
  - Se reduce la interferencia con los elementos de la propia nave.

- La potencia radiada influye de forma obvia en el rango de actuación. Sin embargo puede influir más notablemente un esfuerzo por mejorar la sensibilidad de recepción.

Veamos la siguiente ecuación en la que se puede observar la contribución de cada uno de los factores tenidos en cuenta. Es la de Potencia recibida. Analizaremos la expresión para un sistema en el que la antena receptora y emisora se encuentran en el mismo lugar. No es necesario que lo estén pero es el caso más normal para un sistema como el que se desarrolla.

$$P_r = \frac{P_t G_t A_r \sigma F^4}{(4\pi)^2 R^4}$$

Se puede adelantar que la potencia recibida (P_r) a su retorno será muy baja. Es directamente proporcional a:

- Potencia transmitida (P_t). La confección del pulso por distintas técnicas de modulación de pulso y conformación de trenes de pulso varían la potencia para obtener distinto tipo de información. Puede hacerse incluso en el mismo radar en tiempo de operación. Estos cambios deben por tanto seguirse para adaptar la contramedida a la técnica empleada por el receptor en el momento de la aplicación.

- Ganancia de la antena (G_t). Más correctamente se denomina directividad, ya que no se puede aumentar la potencia de la señal al hacerla pasar por la antena, sólo concentrarla y dirigir el haz en la forma que se considere más beneficioso para la operación del radar. Este es un parámetro que los radares modernos de Conformación Electrónica de Haz (Electronic Beamforming), pueden variar para realizar diversos engaños a receptores que perciben de forma distinta la señal según se varíen las fases emitidas por los elementos del array radiante.

- Apertura efectiva de la antena receptora (A_r), que depende de la longitud de onda y la directividad de la antena RX.

- σ es un coeficiente que modela la sección radar del objeto detectado.

Los factores para los cuales un cambio se nota elevado a la cuarta en la potencia recibida son:

- Factor de propagación (*F*), que será distinto de 1 cuando se deje de tomar la simplificación de que las ondas viajan en el vacío.

- Distancia al objetivo (*R*). Como es lógico se requiere más potencia a mayor distancia, es remarcable que la relación es también elevada a 4. Para doblar el rango empleando únicamente potencia adicional, se debe multiplicar por 16 la potencia transmitida. Hacer esto no es práctico y se recurre a técnicas de modulación de pulso o ensanchamiento de espectro para aumentar el alcance.

En el dimensionado de las etapas de TX y RX, resulta más rentable la inversión en sensibilidad del receptor.

- Métodos como Control Automático de Ganancia (AGC), transmisión en banda ancha y apuntamiento electrónico requieren menor potencia para identificar blancos.
- Una menor potencia también implica menor probabilidad de intercepción.

- Efecto Doppler. Cuando hay un movimiento del blanco relativo al transmisor, se produce una desviación en la frecuencia de la onda reflejada. Siendo *v* la velocidad del blanco y *c* la de la luz en el vacío, un emisor emitiendo en frecuencia f_{tx} tendrá esta magnitud:

$$f_d = 2v\frac{ftx}{c}$$

Este efecto tiene obvias implicaciones en el diseño de un sistema de detección, como es la posibilidad de identificar la velocidad de objetivos en movimiento[18], pero tenemos que analizarlo desde el punto de vista de ECM para confeccionar un pulso de las características apropiadas para inducir la falsa información de velocidad v_{falsa} deseada en el receptor. Se tendrá que aplicar entonces desde nuestro sistema un desplazamiento tal que, si el emisor transmite en ftx, se ha de aplicar un factor $f_{tx\ ECM}$ de:

$$f_{tx\ ECM} = 2v_{falsa}\frac{ftx}{c}$$

[18] MTI Radar – Moving Target Indication. Capaces de discriminar un objetivo en movimiento de entre el ruido del entorno ya que genera un desplazamiento en la frecuencia que se puede detectar con receptores coherentes, y haciendo una medida del mismo, puede obtener la velocidad con la que se mueve.

Adicionalmente, en caso de que el ESM caracterice el blanco como móvil, pudiendo ser otro avión, se deberá sensar la velocidad a la que se está moviendo nuestro conjunto radar, y añadir la velocidad objetivo. Resulta un desplazamiento del pulso falso que se va a transmitir de $f_{tx\ movimiento}$:

$$f_{tx\ movimiento} = 2(v_{emisor} + v_{objetivo})\frac{ftx}{c}$$

Las características de estos ejemplos de radares reales pueden ayudar a establecer requerimientos temporales concretos teniendo en cuenta las relaciones expuestas, y de sistema, para un/ ECM que debe actuar dentro de los parámetros de operación del receptor objetivo:

Unidad radar	Precisión angular	Precisión rango	Precisión altitud
BOR–A 550	< ±0.3°	< 20 m	
LANZA	< ±0.14°	< 50 m	340 m ≈ 1150 feet (at 100 NM)
GM 400	< ±0,3°	< 50 m	600 m ≈ 2000 feet (at 100 NM)
RRP–117	< ±0,18°	< 463 m	1000 m ≈ 3000 feet (at 100 NM)
MSSR-2000	< ±0.049°	< 44.4 m	
STAR-2000	< ±0.16°	< 60 m	
Variant	< ±0.25°	< 25 m	

Tabla 1. Características de radares reales *[18]*.

3.3 CARACTERÍSTICAS DEBIDAS AL FUNCIONAMIENTO DE SISTEMAS RELACIONADOS

Se estudian los modos de funcionamiento tanto de sistemas que cooperan con el nuestro, como de los emisores a contramedir, para obtener las características necesarias para ser eficaz.

- Tiempo muerto. Es el tiempo transcurrido desde que finaliza la escucha, que determina el rango máximo explicado anteriormente, y la siguiente transmisión. Radares modernos de apuntamiento electrónico necesitan este tiempo para reconfigurar los desplazadores de fase con los que se conforma el haz. Dada su proliferación es necesario que el sistema ECM lo considere.

  Los tiempos habituales rondan los 200 microsegundos. No tiene por qué ser constante, ya que cambia en función de la tarea a realizar entre transmisiones. El tracking y estimación de estos tiempos permiten:

  - Liberar recursos computacionales del sistema durante los tiempos muertos.
  - Sincronizar las transmisiones para crear blancos falsos coherentes con el funcionamiento que espera el receptor.

- Limitación de PRI mínimo (PRF máximo). Conviene estudiar en conjunto el sistema en que se engloba el predictor para conocer los requisitos y capacidades que tendrá. Es un buen ejemplo el PRI mínimo que será capaz de contramedir. Lo determinan los siguientes factores:

  - Retardo máximo entre las líneas del sistema. Con la escala de tiempos de que se dispone, no se puede contar con la idealización de transmisiones instantáneas.
    En sistemas reales toman tiempos del orden de nanosegundos al microsegundo.

  - Tiempos de conmutación de los elementos de RF. Compartir en el tiempo parte de la cadena de radiofrecuencia implica tiempos de espera.
    Se encuentran en el entorno de 1.5 microsegundos.

  - El Tiempo de Proceso es variable dependiendo de las operaciones que requiera el sistema ESM que se efectúen sobre la señal de entrada.

  - *Jitter* aleatorio incluido en la señal. Es de gran utilidad el modelado y filtrado que se incluía en el capítulo anterior, para ser capaces de trabajar en la franja de tiempo adecuada.

  - Obviamente, el tiempo de cálculo de la predicción. Ya que es aditivo al resto de elementos anteriores, es esencial minimizarlo para poder trabajar con varias amenazas al mismo tiempo.

- Mecanismos de validación. Es la capacidad del sistema para discriminar qué pulsos corresponden a la señal de interés y evitar la interferencia de señales no deseadas. Es esencial para ello mantener una buena relación señal ruido (S/N) y realizar las escuchas con los parámetros oportunos. Para ello se realimentan los sub-sistemas de sensado y proceso. Veremos qué parámetros que se usan para validar pulsos recibidos deben llevar un tracking asociado con un estimador que reduzca la incertidumbre en la discriminación de blancos.

    - Ancho de pulso. Es una característica básica que permite distinguir emisores.

    - Ángulo de Llegada (AOA[19]). Se aprovecha la dirección de apuntamiento de la antena, conociendo la posición del receptor, y las características y/o configuraciones que definen la directividad. Con alto grado de fiabilidad distingue las distintas fuentes, es información muy valiosa para el conjunto de identificación y predicción.

    - Otras características intrínsecas, como modulación intrapulso…, etc., que pueden contener información única con la que asociarla a un emisor.

    - TOA. Como hemos analizado, los sucesivos TOA registrados, si han sido medianamente bien clasificados con los demás métodos, la predicción hecha con la información hasta el momento puede deshacer la ambigüedad sobre la proveniencia de un pulso.

    - Frecuencia:

    Se trata de un parámetro de especial importancia ya que es clave en la intercepción de un radar. Dada la virtualmente infinita cantidad de bandas en las que puede operar un receptor, incluso limitando los anchos de banda realmente usados, y barriendo en frecuencia con contramedidas de gran ancho de banda; sigue siendo imposible abarcar todo el espectro necesario para combatir con seguridad cualquier objetivo.

    Adicionalmente, radares de última generación con tecnología como SDR[20], son capaces de varias prácticamente de forma instantánea la frecuencia RF de transmisión. De esta manera, sin un sistema que prediga la siguiente banda en la que se localizará el pulso, no se puede contramedir, ya que puede radiarse sincronizado en el tiempo, pero la medida será rechazada por el receptor por encontrarse fuera de su nueva banda de

[19] AOA – Angle Of Arrival. Siglas traducción Ángulo de Llegada.

[20] SDR – Software Defined Radio. Es un sistema de comunicaciones en el que elementos que habitualmente empleaban hardware y electrónica analógica, se sustituyen por procesado digital.

sintonización. Para ello se propone el siguiente esquema que realimenta el sistema ESM por medio de dos estimadores, y proporciona controles útiles a la etapa de RF para operar en la frecuencia esperada del receptor objetivo:

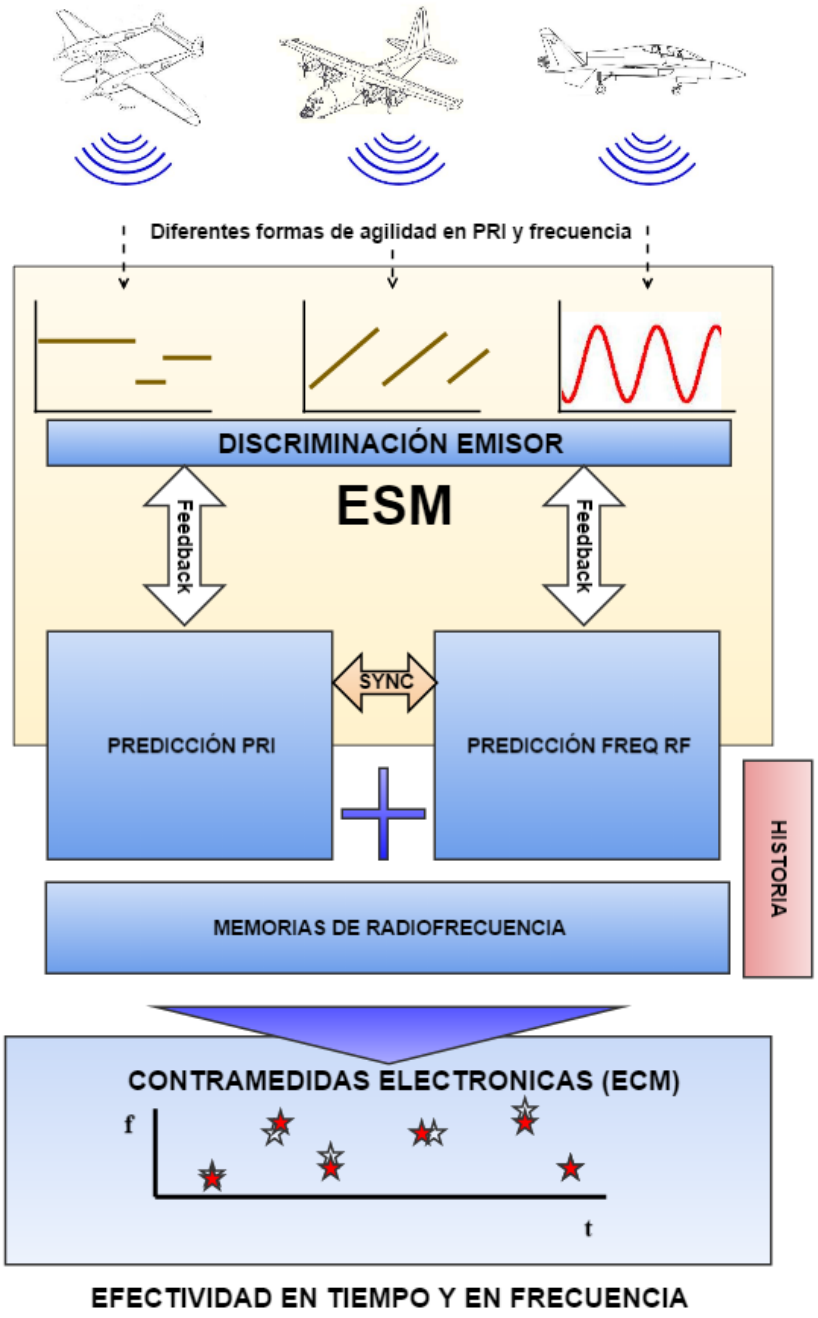


Ilustración 33. Propuesta de interacción de sistemas para identificación de parámetros de amenazas.

En esta disposición, todos los sistemas cuentan con información adicional que puede ayudarles a desempeñar su función con más exactitud. Haciendo el recorrido que realiza la señal, se puede apreciar de qué manera aporta cada uno al conjunto.

Se asume un escenario en el que coexisten varios emisores, que pueden tener distintas formas de agilidad, tanto en PRI que sería en el tiempo, como en RF, que es la banda de radiofrecuencia en la que emiten sus pulsos. Este es el camino que sigue un pulso en el sistema:

1- Estos pulsos en primer lugar atraviesan la etapa analógica de recepción, siempre y cuando esta se haya habilitado y preparado con la configuración necesaria para ello.

2- Una vez acondicionada la señal, es digitalizada y pasa al ESM, el sistema de Soporte a las Contramedidas. El ESM ha sido configurado con conocimiento previo del ELINT, el sistema de Inteligencia Electrónico, y obtiene información sensorial principalmente de la cadena de recepción.

   Con esto, discrimina la procedencia de cada pulso, y en su salida, lo asocia a un emisor determinado. Se pasa al resto de sistemas la información escogida, se decide comunicar a los demás sistemas información de tiempo y también de frecuencia de cada emisor.

3- Esta información pasa por los predictores. La estructura y el funcionamiento de ambos es el mismo. Un emisor puede tener un patrón de variación de PRI, otro de variación de RF, sólo uno de ellos… y cualquier combinación, por lo que ambos predictores están preparados para identificar el patrón y dar como salida el símbolo previsto.

   Esta información realimenta al resto de sistemas de la siguiente forma:

   - El paso 1 puede preparar una configuración de la etapa de recepción más precisa. Será capaz de comenzar la escucha en el tiempo exacto apropiado, y requerirá un menor ancho de banda ya que conocerá de antemano la banda en la que se recibirá el pulso.
   - En el paso 2, ante una duda, se cuenta con información adicional para identificar el emisor. Si se esperaba que un emisor mandase un punto en ese momento, probablemente sea suyo.
   - El propio paso 3 se realimenta iterativamente. Mejora las predicciones con el tiempo.

4- Toda la información que finalmente recibe el sistema de Contramedidas Electrónicas es mucho más exacta. Esto le permite:

   - Emplear más técnicas de engaño imposibles de realizar sin conocimiento del tiempo de llegada del próximo pulso.

   - Aplicar engaños más creíbles y precisos.

En resumen, una mayor interacción entre los sistemas mostrados permite:

a. Mejor aprovechamiento en el tiempo de los elementos de transmisión y recepción. Esto se traduce en un mayor número de objetivos que se pueden identificar y contramedir concurrentemente.
b. Correcta clasificación de un mayor número de pulsos. La función de probabilidad asociada a cada pulso recibe nuevos sumandos, que en conjunto pueden representar la diferencia entre una falsa atribución y un seguimiento correcto.
c. Mayor efectividad sobrepasando las capacidades Contra-Contramedidas de los receptores RADAR. Se consigue sintetizar pulsos falsos con propiedades más realistas.

La interacción representada en el diagrama es posible con la tecnología actual ya que nuevos enlaces de fibra o alta velocidad como el AXI bus proveen altas tasas de transmisión sincronizables, minimizando el retraso introducido por la transferencia de información y haciendo que la mejora conseguida sea rentable. Las relaciones entre los sistemas son las siguientes:

- ESM – Predictor PRI

Mediante los parámetros registrados por los sensores que se explicaban anteriormente, el ESM es capaz de separar los pulsos con un cierto grado de acierto. Estos son tomados en un primer momento por el Predictor. Tal como se explicaba en su análisis, el proceso iterativo mejora la exactitud de la predicción. Llegado a este punto, las propias predicciones pueden ser usadas por el ESM para filtrar pulsos mal identificados. Este filtrado redunda también en una mejora del Predictor que no recibe entradas discordantes.

- Predictor PRI – Predictor Frecuencia RF

El Predictor de PRI indicará los TOA de los futuros pulsos de cada emisor. De esta manera, el receptor radar de nuestro sistema no tiene que buscar en todo el espectro todo el tiempo, lo que sería imposible. Centrará sus fases de escaneo en los TOA predichos. Allí ha de encontrar la frecuencia del pulso con la que inicializará el Predictor de Frecuencia RF. A partir de aquí se produce de nuevo una realimentación que reduce los anchos de banda al escanear zonas predichas de ancho de banda, y sólo en los tiempos que se prevé, con mayor exactitud, la llegada de un pulso.

- ESM – Predictor Frecuencia RF

La información de banda donde encontrar el pulso es crucial para mantener un tracking sólido de un objetivo, de esta manera todos los siguientes subsistemas mantienen un flujo constante de información sin pulsos perdidos.

- Las Memorias de Radiofrecuencia almacenan pulsos característicos del emisor, para poder manipularlos y reenviarlos en el TOA y banda de RF apropiados para ejecutar la técnica de Contramedidas apropiada. Aquí se muestra un ejemplo de transmisión con patrones de

variación instantánea tanto de la banda de RF en la que se transmite, como del tiempo entre pulsos:

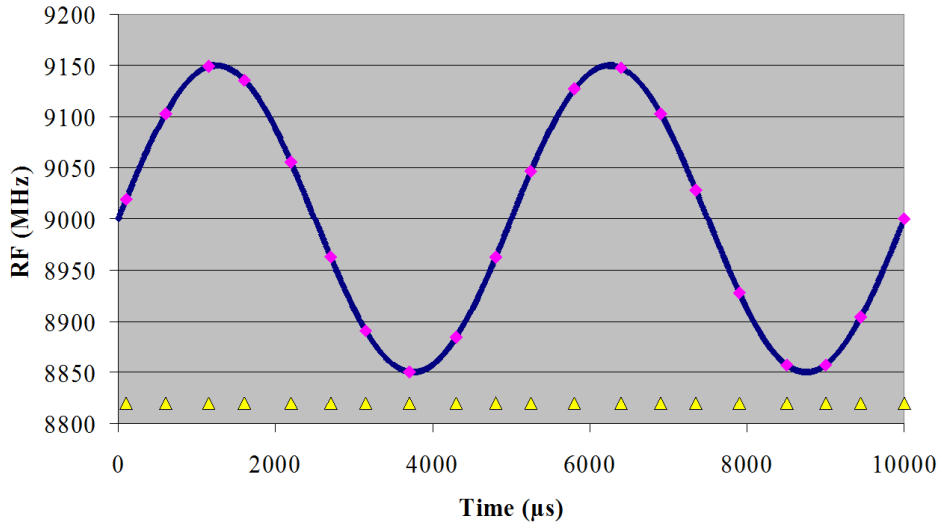


Ilustración 34. Ejemplo predicción PRI (Triángulos amarillos, tiempo) y Frecuencia RF (Puntos morados) *[9]*.

- Histórico:

La recopilación de todos los datos generados por los predictores y las medidas del ESM son de gran utilidad, ya que un problema común es que los radares terrestres habitualmente tienen patrones de escaneo del espacio en forma circular, cónica o sectorial. Los constantes cambios en el apuntamiento, ya sea por medios mecánicos o por conformación electrónica de haz, hacen que nuestro sistema reciba únicamente un determinado número de pulsos en cada iluminación.

Esto dificulta la identificación, caracterización, seguimiento y por tanto predicción de la secuencia de pulsos que emite.

Para lidiar con el problema se almacenan las características combinadas de todos los pulsos, estos son procesados por un sistema de Inteligencia Electrónica (ELINT), que organiza la información y reconstruye el patrón de apuntamiento.

Conocer las direcciones de apuntamiento del emisor permite emplear técnicas sincronizadas con las iluminaciones como:

- Transmisión sólo en lóbulos secundarios del patrón de radiación de nuestra antena (que es conocido). Mediante la modulación adecuada se radia principalmente en ellos, lo que hace percibir la señal como procedente de otro ángulo.
- Transmisión con ganancia inversa a la potencia recibida, para generar movimientos falsos lo más realistas posible.

3.4 INTEROPERABILIDAD

Son analizados los tiempos dedicados a los sistemas involucrados, así como aquí los mecanismos empleados para trabajar en armonía.

- Tiempo de transpondedor.

Se define como el tiempo mínimo en el que el sistema es capaz de retransmitir un pulso amenaza recibido. Idealmente se debe aproximar a 0. En un sistema real se consideran buenos tiempos menores a 150ns.

Este tiempo se hace especialmente importante para radares de Baja Probabilidad de Intercepción (LPI) que tienen anchos de pulso muy pequeños, de alrededor de 50 nanosegundos. Si el retardo mínimo entre el eco real y el pulso de la contramedida es mayor que la puerta temporal que abre el detector del radar amenaza, no será posible conseguir un enganche de puerta, y no se podrá engañar al receptor.

Influyen factores como:

- Procesamiento Software: Debe ser reducido o eliminado. Es uno de los motivos por los que se propone emplear un sistema electrónico basado en FPGA que emplea firmware mucho más específico y rápido.

- Procesamiento de la cadena de RF: Se ha de disminuir el número de componentes y la latencia entre ellos.

- Retardos por cableado RF: Se deben situar físicamente próximos el receptor, procesador y transmisor.

- Tiempos de conmutación entre RX y TX. Una buena predicción del momento en el que va a llegar el pulso permite tener los elementos previamente listos para retransmitirlo en el menor tiempo posible.

- Capacidades multiamenaza.

Una de las características más deseables en un sistema avanzado, una vez realizados los esfuerzos necesarios en la discriminación, puede verse aumentada con:

- Recursos para procesamiento de elementos de RF en paralelo. Un SoC se postula a este respecto también como la elección óptima, pudiendo procesar en paralelo incluso información situada en distintos dominios de reloj.

- Multiplexación de recursos en el tiempo. Cuanto más precisa sea la ventana de tiempo predicha, más tiempo deja disponible los recursos para otras amenazas. Esto es especialmente interesante en equipos compactos como normalmente son los aeroportados, que tienen un número muy limitado de elementos de transmisión-recepción.
- Reducir el tiempo de apuntamiento a los objetivos. Para ello se emplean arrays de antenas apuntadas electrónicamente, que como se ha comentado, varían el apuntamiento únicamente cambiando la fase constructiva o destructivamente para dirigir el haz.

- Necesidades de transmisión.

Todos los subsistemas detallados confluyen en la sincronización que equilibra las necesidades de escucha (detección, grabación de pulsos y predicción de secuencias), con las de transmisión.

Para lograrlo se emplean los siguientes mecanismos síncronos:

- LookThrough. Son los intervalos en los que se detiene la transmisión para permitir la escucha.

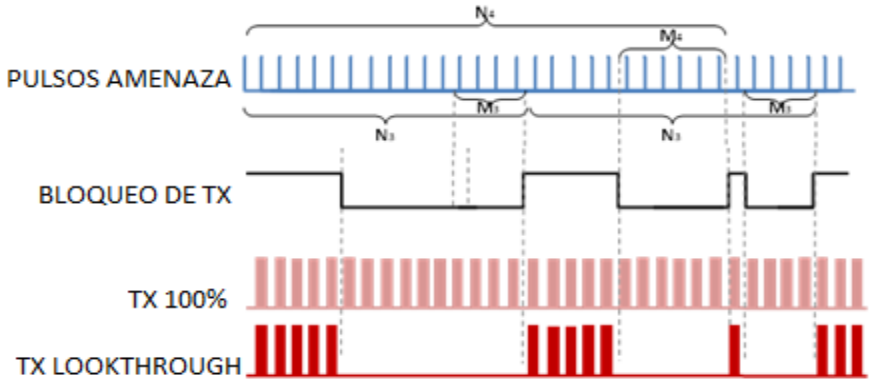


Ilustración 35. Ejemplo de uso de LookThrough.

La orden de transmisión, *TX*, está supeditada a habilitación con un nivel bajo de la señal de *BLOQUEO* de transmisión. Esta señal de bloqueo de TX permite los intervalos de LookThrough en los que escuchar los Pulsos Amenaza. El predictor es capaz de estrechar la duración de estos intervalos, y por tanto se pueden enviar más pulsos manipulados.

- LookOver. Empleando control de ganancia se suben los umbrales de detección para poder seguir escuchando incluso mientras se transmite, y se sensa el propio pulso transmitido sumado a los pulsos ajenos. Para que el LookOver sea más efectivo, se ha de mejorar el aislamiento electromagnético entre receptor y transmisor.

CAPÍTULO 4:
ELEMENTOS PARA ARQUITECTURA DE PREDICTOR ECM EFICAZ

CAPÍTULO 4: ELEMENTOS PARA ARQUITECTURA DE PREDICTOR ECM EFICAZ

A fin de mejorar la versatilidad del conjunto, se presentan a continuación una serie de mecanismos de utilidad en un sistema compuesto por Predictores para el soporte de Contramedidas Electrónicas. La arquitectura desarrollada de esta manera incluye los elementos necesarios para el dimensionado del Predictor de acuerdo a las necesidades esperables, el mecanismo para operación en modo multiamenaza, y componentes adicionales que aportan información útil para el control del sistema.

4.1 DIMENSIÓN PARAMETRIZABILE

Esta es una característica siempre deseable, ya que permite primero dimensionar el sub-sistema y adecuarlo al hardware donde se implementará, y también incluirlo como bloque que se adapte a un conjunto mayor. Para hacer el predictor parametrizable se establecen:

- Tamaños de buffer configurables.

Como se ha explicado anteriormente hay sub-sistemas que recogen un histórico de pulsos y operan con ellos para obtener distinta información relativa al emisor. Por el contra, debido a los requerimientos desarrollados en el capítulo 3, la operación en tiempo real se tiene forzosamente que limitar a un número de muestras para poder dar el resultado dentro de un tiempo máximo.

Los buffer del Predictor son empleados al completo en la fase de identificación de periodo y parcialmente, sólo las últimas N muestras, para efectuar el filtrado iterativo Kalman.

- Tamaño de la escalera de comparadores.

Se define con el tamaño del buffer de pulsos. Como se adelantaba en el capítulo 2, el número de bloques comparadores será:

$$Número\ bloques\ comparador = \sum_{i=1}^{BUFF\_SIZ} BUFF\_SIZ * i$$

Adicionalmente se requerirán BUF_SIZ sumadores de tamaño *i* hasta BUF_SIZ para realizar el histograma.

- Tamaño del filtro Kalman.

El orden del filtro es igual al número de estados. Tal como hemos modelado la secuencia de pulsos, el número de estados es igual al número de elementos que componen un periodo, es decir N.

El filtro emplea una gran cantidad de recursos en la FPGA, por lo que se implementa en hardware una sola vez. Esta instancia es multiplexada en el tiempo para procesar todas las amenazas que concurran.

Ya que el hardware es fijo, se preestablece un tamaño máximo de periodo de secuencia esperable, N_MAX. Determinará el tamaño de todas las matrices involucradas en el filtro, y por tanto, los recursos hardware empleados para la implementación de las operaciones aritméticas que las relacionan.

Durante la operación, para procesar tramas de distintos tamaños, se establece como entrada al filtro el vector de N posiciones de la trama almacenada, y se concatenan ceros hasta N_MAX. En cada cambio de trama a filtrar, con la N obtenida del bloque de identificación de periodo, se seleccionan los N elementos a procesar del buffer de pulsos. El resultado del cálculo matricial será el mismo que de haber elegido un tamaño de matriz N, puesto que las columnas no usadas se mantendrán a 0 y tomaremos como resultado de cada operación de filtrado los N primeros elementos de la matriz de estimación.

- Número de amenazas concurrentes.

En modo multiamenaza el sistema es capaz de abastecer de predicciones actualizadas para múltiples objetivos. Se parametriza el sistema con el número máximo de amenazas a contramedir, antes de la implementación en FPGA.

El uso de hardware no aumenta excesivamente al aumentar el número de amenazas, gracias al multiplexado en el tiempo que se explica a continuación.

La latencia máxima dependerá de la implementación en HDL realizada, en la que habrá una latencia del filtro y un número de ciclos que toma el cambio de amenaza. Se establece también la frecuencia con la que operará esa sección del diseño. Para obtener el número máximo de amenazas posibles, se emplea el capítulo anterior y dependiendo de las características del sistema global, tras restar los efectos explicados se obtendrá un tiempo disponible para predicción, al que se aplica la siguiente fórmula.

$$Amenazas_{max} = \frac{t_{predicción} * f_{clk}}{\left(ciclos_{cambio\ amenaza} + latencia_{predictor}\right)}$$

4.2 MULTIPLEXACIÓN EN EL TIEMPO

Para que el Predictor pueda crecer en capacidades a un ritmo mayor que el que crece la cantidad de recursos que emplea, se recurre a la multiplexación de los elementos comunes. De esta forma se puede hacer un tracking y predicción de varios objetivos concurrentemente.

El uso de esta técnica tiene un efecto que ha de ser tomado en cuenta en el dimensionado del sistema a partir de los requisitos. Si dos amenazas requieren actualización de la predicción en el exacto mismo momento, se procesarán una a continuación de la otra.

Dado que sumar el número de amenazas y aplicar la latencia resultante sería un peor caso extremadamente infrecuente, se debe caracterizar su efecto recurriendo a una función de probabilidad en la que se caracterice el tipo de tramas esperadas, y la probabilidad de ocurrencia de pulsos muy próximos que además necesiten actualización de la predicción en un tiempo cercano a ser tan pequeño como el de actualización de predicción, que se espera del orden de decenas de nanosegundos.

La siguiente figura muestra los recursos fijos asociados a cada amenaza, y los operadores, que emplean la información almacenada de cada amenaza. En cada momento los multiplexores sitúan los operandos apropiados en los operadores, en función de la amenaza que toque procesar. El sistema está dimensionado con el N máximo introducido como parámetro.

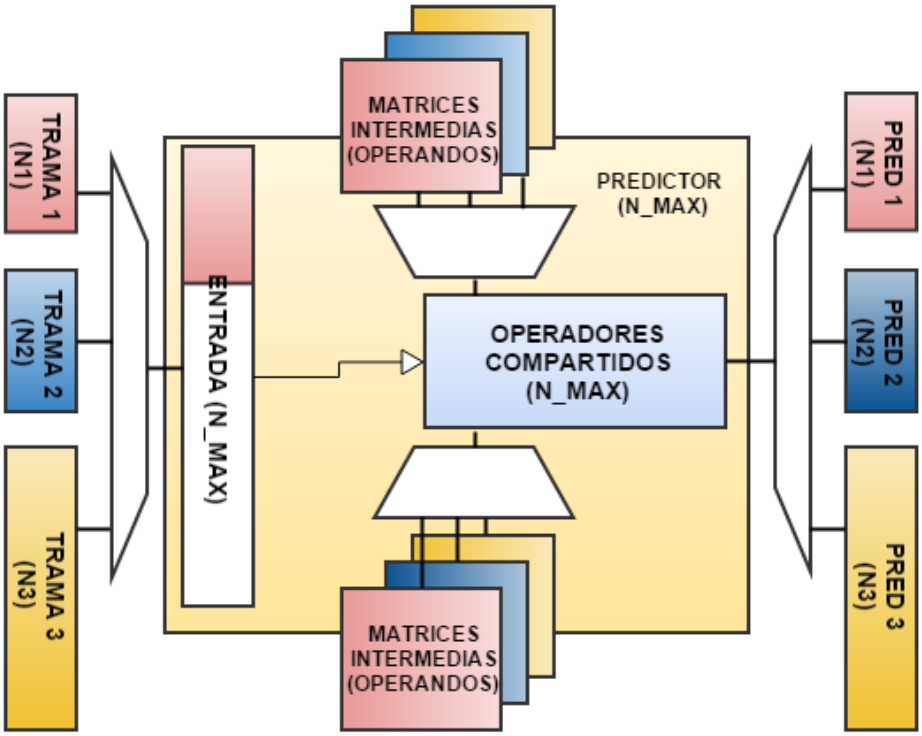


Ilustración 36. Esquema para multiplexación de recursos de cómputo.

4.3 MONITORIZACIÓN DE CAMBIOS EN EL PERIODO IDENTIFICADO

En determinados escenarios, una identificación errónea de los pulsos, y una contramedida mal ejecutada, puede ser más peligroso que la no-actuación. Para ello se propone el siguiente mecanismo.

Tal como se ha planteado la estructura del identificador de periodos, se pueden monitorizar constantemente los periodos más repetidos en la memoria del buffer. Rápidos cambios en estos elementos indican un patrón inestable que puede además contener pulsos mal clasificados.

Durante la inicialización y la recepción de los primeros pulsos de cada amenaza, este bloque puede reconocer cuándo se ha recibido un periodo completo, puesto que los índices de mayor número de hits permanecerán constantes. De esta forma se consigue una cualidad de enorme importancia en un sistema de ECM con Predictor, la capacidad de predecir a partir de haber capturado un solo periodo de la secuencia. Muchos sistemas en cambio requieren de una memoria con un gran número de muestras grabadas para obtener las características necesarias para comenzar la predicción.

La identificación del momento en que se cuenta con un periodo completo puede activar la inicialización del filtro y por tanto la puesta en marcha inmediata de la predicción.

Por sí solo, o en conjunción con medidas adicionales, se puede también emplear para habilitar la actuación del sistema de Contramedidas ECM a partir del momento en que se cuente con una uniformidad suficiente en la predicción.

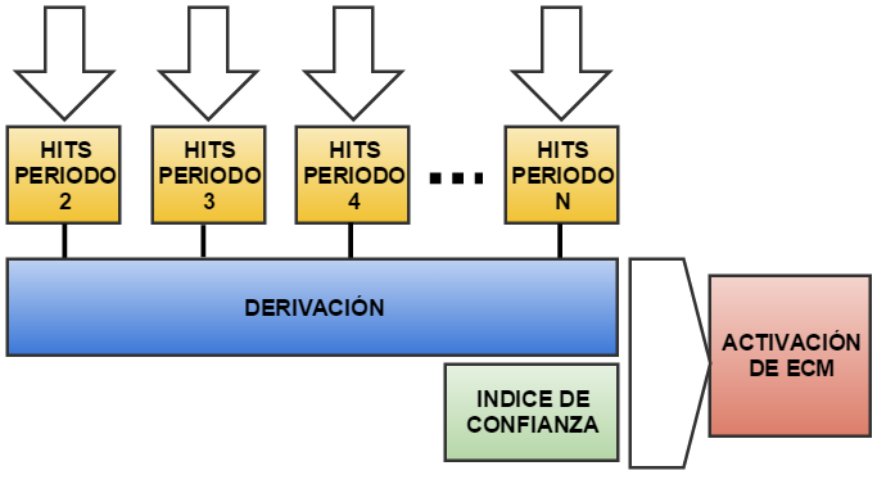


Ilustración 37. Esquema monitorización de cambios en identificación de periodo.

Como se muestra en la figura, cada acumulador de cuentas se asocia una entrada al monitor de cambios en el periodo, aparte de al resto de componentes a donde estaba conectado.

El funcionamiento de este monitor es el siguiente:

- Se realiza la derivada del valor acumulado en cada columna. Esto, en tiempo discreto es inmediato como incremento por unidad de tiempo.
- La pendiente con la que cambian los elementos no será igual para cada columna, ya que dependiendo de las características de la trama y el tamaño del identificador, se acumulan más o menos hits en los contadores que coinciden con sub-periodos. Según el comportamiento deseado, se puede cambiar el peso que tiene cada uno de los elementos.
- La suma ponderada de la pendiente de cambio por el peso asignado, dará como resultado un índice de cuánta variación se está produciendo en la identificación del periodo.
- Dado que el índice de confianza o seguridad, se puede cambiar en tiempo de ejecución, este mecanismo puede adaptarse a situaciones complejas o cambiantes.

- Elección del índice de confianza:

Se ha de establecer este coeficiente de seguridad de forma que se cuantifique la tolerancia a imperfecciones admisible en la salida. Esto dependerá de las consecuencias que pueda tener en cada situación de operación el error en la predicción.

Además, en situaciones como por ejemplo una aeronave entrando en territorio desconocido, hay un factor que aporta un pequeño margen de tiempo en el que tomar medidas hasta obtener una predicción estable. Mientras se entra en el rango de un radar enemigo, si se cuenta con un receptor de buena sensibilidad, se comenzará a percibir su señal cuando aún es lo suficientemente débil para no poder ser detectada tras el camino de vuelta. Este tiempo se podría emplear en dejar que el filtro afine sus predicciones antes de activar las contramedidas.

CAPÍTULO 5:
SIMULACIONES DEL PREDICTOR PARAMETRIZABLE

CAPÍTULO 5: SIMULACIONES DEL PREDICTOR PARAMETRIZABLE

Para demostrar la efectividad de los métodos propuestos se realizan simulaciones de los bloques bajo diferentes configuraciones y entradas. Se simulará el identificador de periodo con distintos tamaños de buffer, y el estimador Kalman con distintos tamaños de la secuencia de entrada.

Toda la arquitectura está orientada a una futura implementación hardware. Los sub-sistemas creados para simulación se realizan teniendo en mente que se trate de lógica y procesos implementables en un SoC compuesto por FPGA y microprocesador. De esta manera posteriormente se pueden emplear las herramientas de generación de HDL a partir de los modelos Simulink, e implementar el sistema parametrizado y simulado en un dispositivo hardware.

Es necesaria una simulación paramétrica para poder estudiar el comportamiento en las distintas configuraciones que permite el sistema. Simulando las entradas se puede valorar su desempeño con antelación y optimizarlo antes de su implementación definitiva en hardware. Se aportan las conclusiones extraídas e indicaciones para un correcto funcionamiento.

5.1 SIMULACIONES DEL IDENTIFICADOR DE PERIODO

Este bloque se ha creado de forma que se pueda variar el tamaño de buffer de muestras con el que opera. Este parámetro determina la cantidad de lógica empleada y la robustez del filtro. Para realizar las simulaciones en el entorno Matlab del sistema corriendo en tiempo real y discreto, se emplea la herramienta Simulink.

Se ha conseguido un subsistema de identificación de periodo veloz, que emplea elementos aritmeticológicos simples y presenta una buena inmunidad a errores del sistema ESM que ocasionen pulsos faltantes.

A fin de obtener el sistema dimensionado a partir del parámetro de tamaño de muestras, se elabora un script que conforma el modelo deseado en interconecta los bloques para el BUF_SIZ deseado. Se puede encontrar el script completo en el *ANEXO I*. El objetivo es poder crear automáticamente los modelos para identificadores de periodo de distintas características sin intervención manual, y de la misma forma simular su comportamiento para las secuencias de entrada deseadas.

5.1.1 PARAMETRIZACIÓN DEL TAMAÑO

Antes de incluir este módulo en un sistema, conviene estudiar el efecto que tiene variar su tamaño en función del tipo de trama esperable durante la operación del mismo. Al comienzo del script, se puede establecer de dos maneras.

Primero se introduce el máximo de elementos que se espera que pueda contener un periodo Stagger. Es obvio que no se podrá calcular un periodo mayor que los pulsos que ha almacenado.

Esta información es habitualmente obtenida a partir del análisis *off-line* de las observaciones realizadas por un ELINT. El tamaño total del buffer se establece entonces:

a. Indicando directamente el tamaño de BUF_SIZ.
b. Escogiendo el número de periodos completos de la secuencia mayor que se quiere almacenar, y un coeficiente de incremento de la robustez ROB_INC.

A continuación se puede introducir manualmente una secuencia Stagger, si se quiere simular el desempeño al tratar con un emisor de características conocidas de antemano. Si no, se usarán valores aleatorios.

El script creará ahora un nuevo modelo Simulink, y se procede a la incorporación de los bloques mediante el modo línea de comandos. En este modo, se pasan parámetros a Simulink que permiten la manipulación de bloques, conexiones, parámetros y configuraciones desde el exterior. Para emplearlo, se ha aplicado lo siguiente:

- Formato de los argumentos.

Todos los argumentos, incluso aunque sean números, han de pasarse como *strings*[21].

- Ruta de acceso a los bloques.

Cuando se usan bloques incluidos en la suite básica de Simulink, se invocan bajo la ruta simbólica built-*in*. Si están incluidos en alguna toolbox, como se ha empleado por ejemplo la de DSP, se ha de encontrar el nombre de ruta que se le asocia, en este caso *dspstat3*. Los bloques se acceden como *'nombreToolbox/nombreBloque'*.

[21] String – Cadena de caracteres (chars).

- Localización dentro del espacio de trabajo.

  Se ha de pasar la posición del bloque dentro del espacio de trabajo, para ello se crea una variable de referencia para cada grupo de bloques. Esta recorrerá el espacio variando los ejes por los incrementos seleccionados. Todas las posiciones son absolutas, respecto a la esquina superior izquierda.

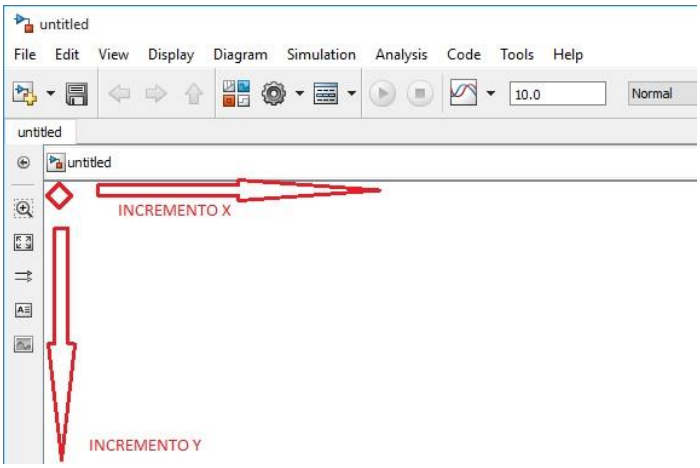


Ilustración 38. Sistema de coordenadas para Simulink command-line.

- Tamaño.

  El tamaño de los bloques se establece indicando los dos puntos de la diagonal que definen el rectángulo. De la misma manera que se define una posición, un vector con las dos coordenadas de dos puntos definen al mismo tiempo localización y tamaño de un bloque.

- Parámetros de cada bloque.

  Los mismos parámetros que se configuran manualmente abriendo los diálogos de bloque en Simulink, se han de pasar como argumentos en el momento de la creación del mismo. Las propiedades que se pueden configurar en cada caso pueden ser consultadas en la página de Parámetros Específicos de Bloque [19].

  Definidos todos los argumentos necesarios, se crea el bloque con la función *add_block()*. Un ejemplo de sintaxis queda de esta manera:

```
add_block('built-in/Mux',[sys muxName],'Position',pos, 'Inputs',num2str(sizePulseBuffer-1) );
```

Ilustración 39. Ejemplo add_block().

- Interconexión de los bloques.

  Es necesaria una organización en los nombres de los bloques, ya que para que este mecanismo sea útil, las conexiones han de hacerse programáticamente. Los nombres de bloque se confeccionan concatenando *strings* con los que se identifican en el diagrama. De la misma manera, se crea una cadena para el puerto correspondiente del bloque de origen y el de destino.

```
compName = ['/' 'comparator' num2str(j) 'N' num2str(i+1)];

sourcePort = ['demuxSamples' '/' num2str(j)];
destPort   = ['comparator' num2str(j) 'N' num2str(i+1) '/' '1'];
```

Ilustración 40 Ejemplo denominación de puertos para conexión.

Una vez se han definido los parámetros necesarios para la conexión de bloques, se pasan a la función *add_line()*, seleccionando 'autorouting' para que evite los obstáculos.

```
add_line(sys,sourcePort,destPort,'autorouting','on');
```

Ilustración 41. Ejemplo *add_line()* con *autorouting.*

Analizando el algoritmo y la estructura planificada para el identificador de periodo basado en escalera de comparadores, se ha de encontrar la relación existente entre los índices de cada puerto de cada elemento que compone el sub-sistema. Hecho esto, se automatiza en bucles la conexión sucesiva de cada elemento con el apropiado bloque.

Una vez se han definido todos los bloques, su localización y se han establecido las conexiones entre ellos, introducimos los parámetros iniciales de ejemplo y al correr el script obtenemos el siguiente modelo como resultado:

Parámetros ejemplo:

MAX_STAGGER_POSITIONS = 7;	Mayor secuencia esperable 7 posiciones
BUF_SIZ_TIMES_MAXN = 2;	Buffer para alojar 2 secuencias completas
ROB_INC = 2;	Posiciones adicionales a almacenar 2

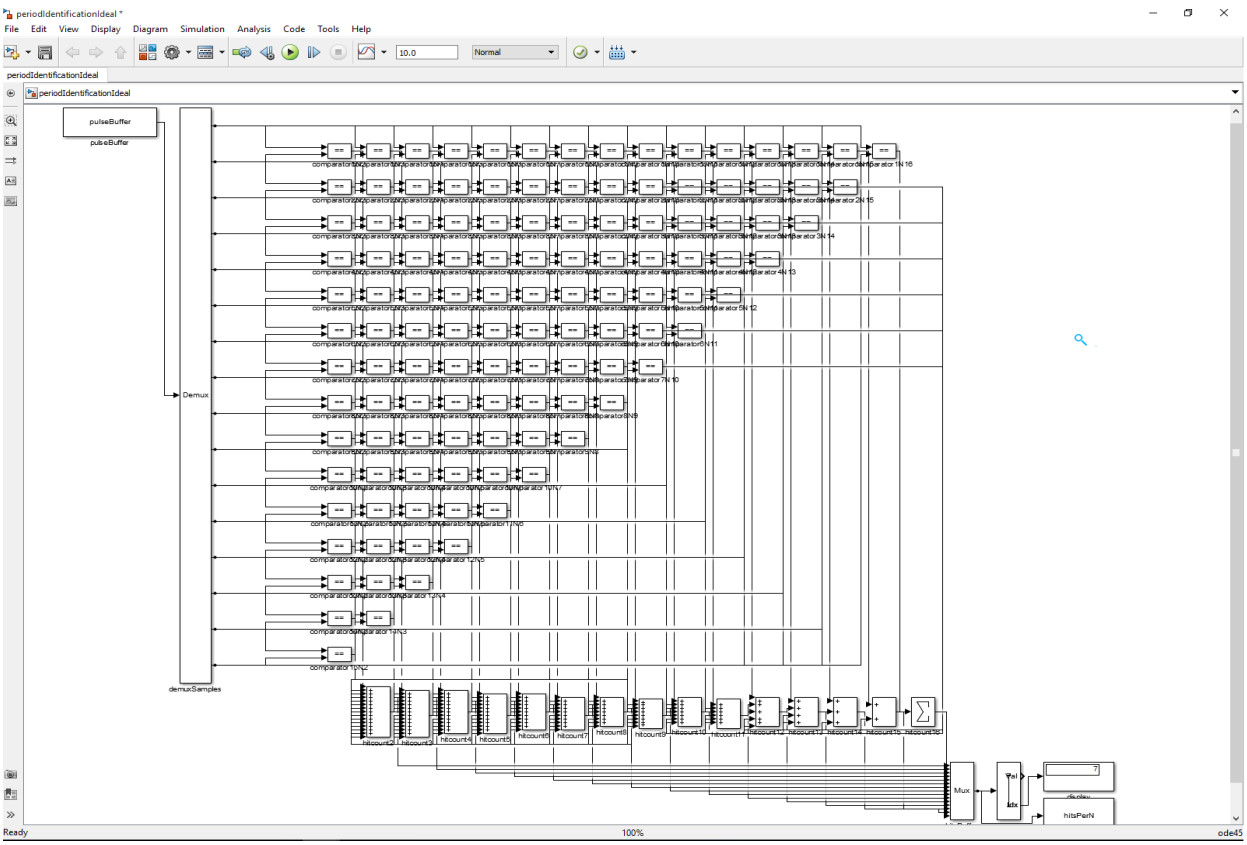
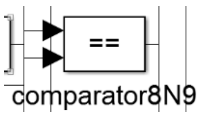


Ilustración 42. Ejemplo creación parametrizada modelo identificador de periodo.

Se ha denominado cada bloque comparador de forma que permite identificar la contribución que hace a la identificación. Por ejemplo, este será el que compare el elemento 8 para un N = 9.



Con esta disposición, se ha ordenado por columnas los comparadores para cada N. Por ello se sitúan también por columnas los acumuladores que recogen los hits (cuentas) detectadas para cada posible periodo.

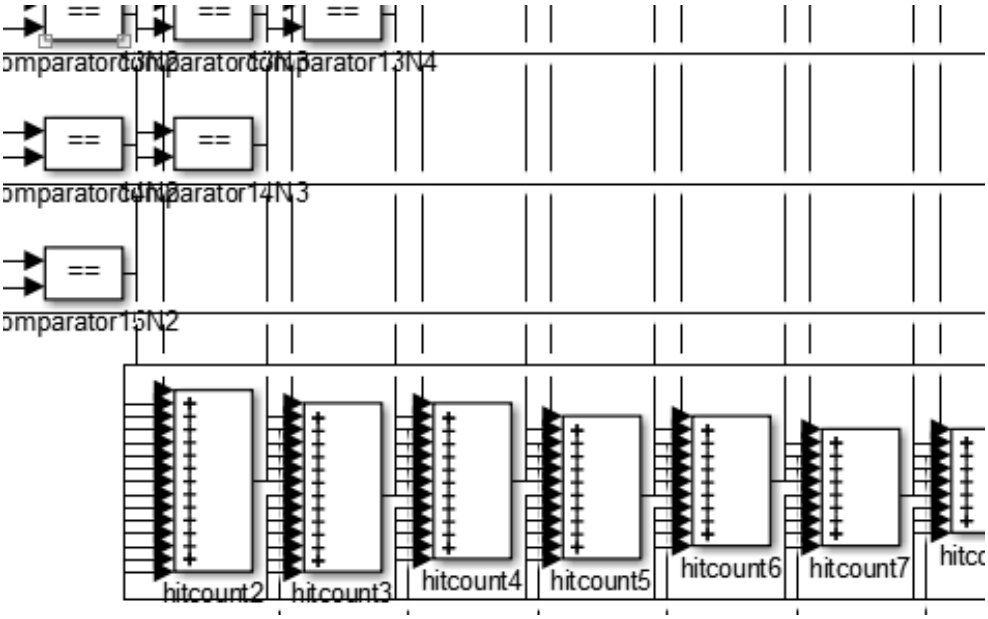


Ilustración 43. Acumuladores de cuentas.

El número de acumuladores es también determinado por los parámetros, por lo que se puede implementar a continuación un bloque, de tamaño constante, que selecciona cual contiene más hits y proporciona como salida el índice correspondiente.

Este es el periodo de la secuencia Stagger contenida en el buffer, N.

5.1.2 SIMULACIÓN

Tras la confección del modelo, se genera una trama simulada para comprobar el funcionamiento del sub-sistema. En el modelo se incluye la salida del contenido de todos los comparadores en cada ciclo de simulación.

La trama de elementos Stagger, se elabora con el tamaño empleado para parametrizar el tamaño del identificador. Se simula una trama de periodo igual al máximo esperado, para la cual se pueden introducir los elementos de una trama conocida en el vector "staggerSequence", o simular una trama con elementos escogidos aleatoriamente.

Este es el resultado para la simulación de la secuencia de ejemplo siguiente: [50 25 70 25 80 120 35]. Se simula sobre el identificador de ejemplo creado antes.

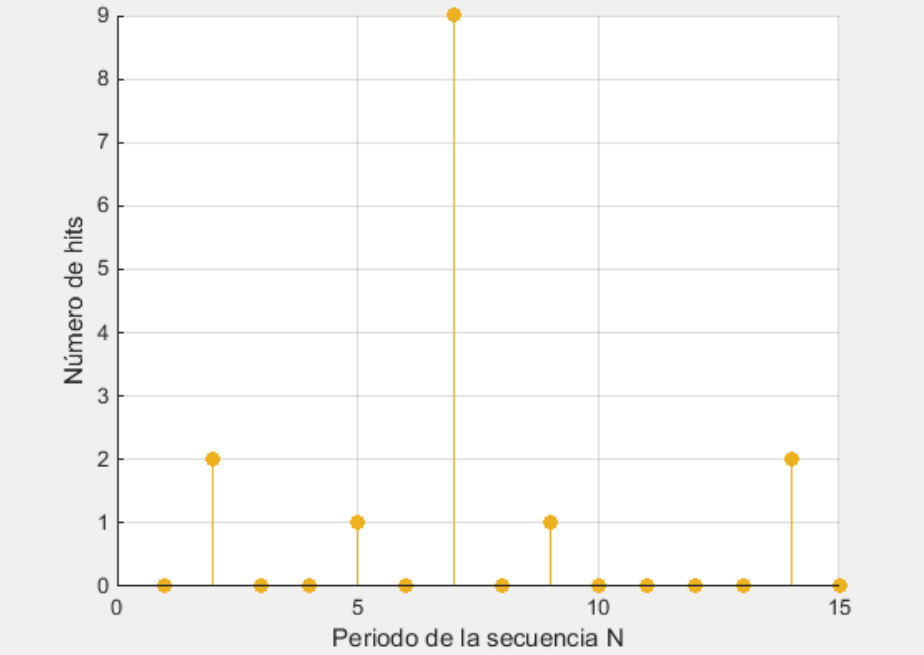


Ilustración 44. Resultados de la simulación de identificador de periodo (En este caso 7), con la trama ejemplo.

Los resultados encajan con lo esperado de acuerdo al planteamiento del identificador de periodo. Comprobamos el número de cuentas obtenidas para cada columna:

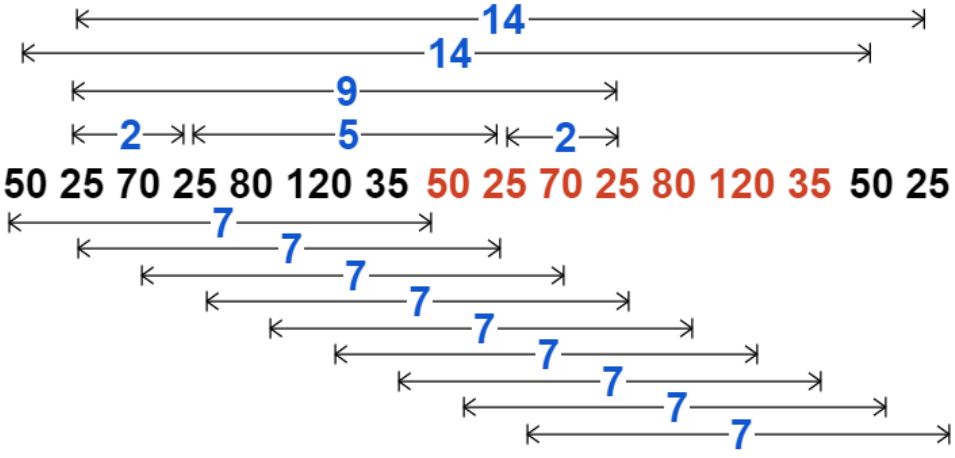


Ilustración 45. Comprobación hits con periodo identificado.

De acuerdo al tamaño del buffer estas son las posiciones que se encuentran almacenadas. Efectivamente el histograma representa correctamente el número de veces que está presente cada periodo de la señal. Puesto que es un buffer FIFO, en cada paso de simulación, se desplazan y entra el nuevo elemento. En la imagen obtenida de Matlab, se ha representado con persistencia activada y un color distinto los hits en cada ciclo. El hecho de que sólo haya un color, indica que durante todos los ciclos, el resultado se ha mantenido constante. Esto indica un correcto funcionamiento también, ya que es igual analizar estas tramas:

[50 25 70 25 80 120 35 50 25 70 25 80 120 35 50 25]

[25 70 25 80 120 35 50 25 70 25 80 120 35 50 25 70]

5.1.3 CONCLUSIONES DE LA SIMULACIÓN

Sería ideal por supuesto contar con un buffer infinito para obtener con la mayor seguridad el periodo. Sin embargo esto no es posible por dos motivos:

- La obvia limitación a una cantidad de recursos finitos.
- El requerimiento de proporcionar una estimación cuanto antes al haber capturado un periodo de la señal.

Para asistir de manera razonada la decisión de dónde situar el compromiso entre consumo de recursos, y desempeño del sub-sistema. Se realizan las siguientes simulaciones adicionales y se observan los efectos en la variación de los parámetros. Todas las simulaciones se realizan con tramas de periodo igual al que se indica como máximo esperado, ya que el desempeño del identificador será mejor cuanto mayor sea el tamaño relativo de este con el periodo a identificar. Un periodo más pequeño del máximo previsto siempre se identificará mejor, por lo que nos mantenemos en el peor caso para los parámetros que se escojan.

- Número de periodos almacenados.

Se incluye en el *ANEXO II* el script preparado para simular los modelos variando un parámetro de interés. Se puede introducir un número de simulaciones en las que variar el tamaño de la secuencia y también un número en el que variar el tamaño del identificador en sí. Sucesivamente va creando el modelo con los parámetros elegidos, simula y presenta los resultados para los distintos tamaños y pasa al siguiente. Esto nos permite identificar cómo afecta a su comportamiento.

Continuamos usando los parámetros de ejemplo anteriores, con la secuencia Stagger de N=7 de elementos que incluye repeticiones más pequeñas dentro del periodo completo para comprobar su funcionamiento al ampliar el tamaño de buffer.

Añadir un número de celdas múltiplo del periodo máximo esperable, aumentará de forma evidente tanto la capacidad para identificar el periodo, como la resistencia a imperfecciones en la trama recibida.

Para realizar una simulación de peor caso, se variará el tamaño del buffer siempre siendo números muy pequeños. Esto es para poner a prueba el sistema en el peor caso y ver su capacidad. En una implementación real, el tamaño de buffer será mucho mayor, ya que se ha diseñado empleando los recursos más simples y abundantes en una FPGA. En esta simulación el tamaño se seleccionará en varias iteraciones, creando cada vez un modelo para albergar 1 periodo completo, 3 (el de 2 se ha mostrado en el ejemplo anterior), 5 y 7 periodos. Se mantienen constantes los incrementos INC_ROB en 2.

- Resultados obtenidos:

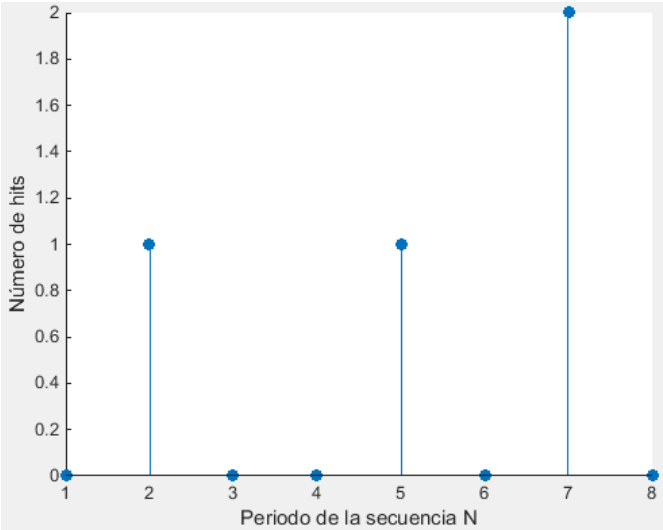
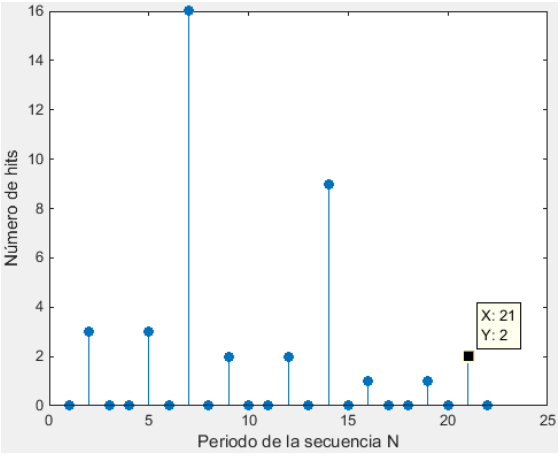
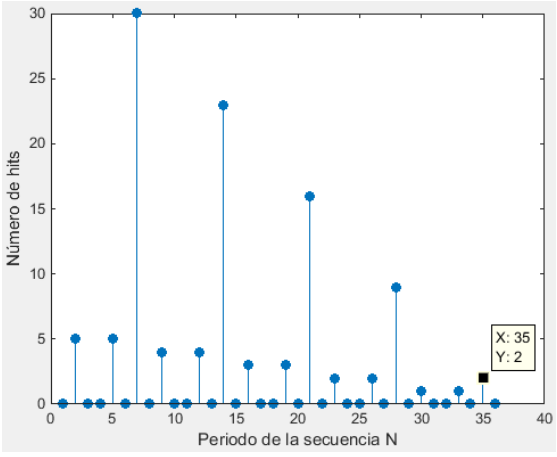


Ilustración 46. Identificación N=7, BUF_SIZ=1 periodo, INC_ROB=2.

El caso en el que el buffer FIFO sólo pudiese almacenar un periodo no es utilizable, ya que no podría haber ni siquiera una repetición con periodo N. Para ello se incluye el incremento de robustez ROB_INC. Pese a no ser utilizable, se muestra como punto de partida para comprobar lo que es esperable, ya que la secuencia Stagger introducida a propósito tiene las características que ponen de manifiesto los comportamientos del identificador. Como se aprecia, y se puede ver igual en la ilustración 45, aparecen las sub-secuencias contenidas en un periodo de la secuencia completa. Estas son las ocurrencias para periodo 2 y 5. Por otro lado se identifica correctamente el periodo 7, ya que recibe dos elementos extra gracias al ROB_INC introducido. Estos dos elementos, serían dos 7 representados en la parte inferior de la ilustración 45.





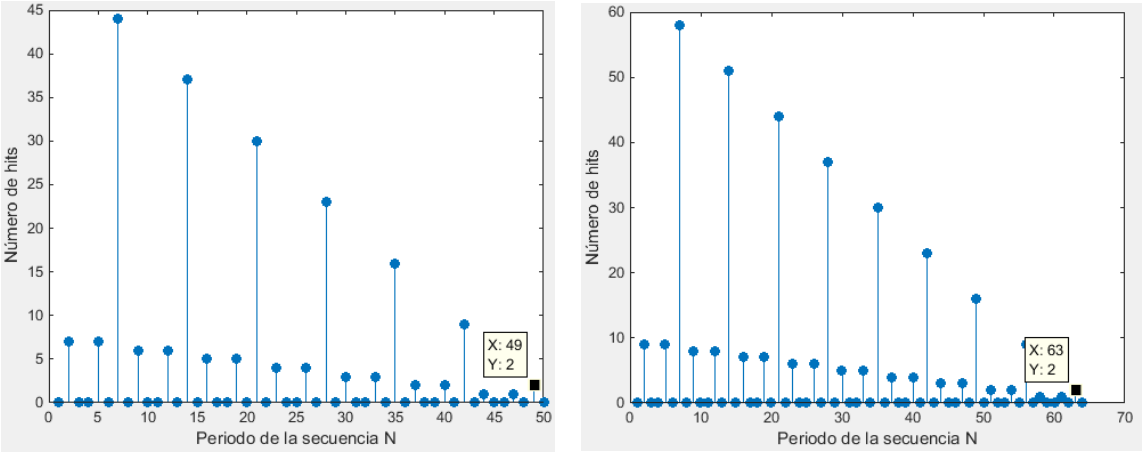


Ilustración 47. Identificación N=7, INC_ROB=2, BUF_SIZ=3, 5, 7, 9 periodos respectivamente.

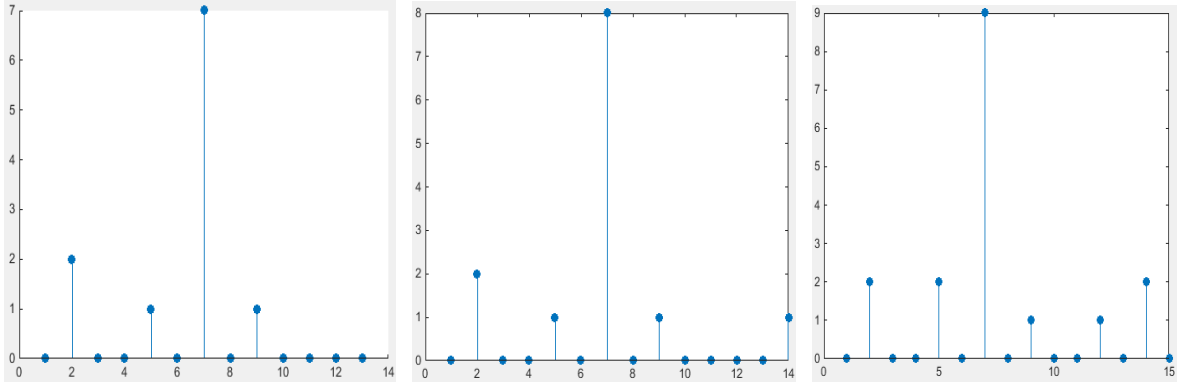
Se aprecia claramente la tendencia al aumentar el número de periodos que puede contener el buffer. Se ha marcado el último máximo local para subrayar cómo los picos se dan en el producto del número de periodos que aloja el buffer multiplicado por el número de posiciones de que consta un periodo.

De la misma manera, como es esperable, el número de hits es multiplicado por el número de periodos almacenados. Se ve también cómo decrecen las cuentas para las sub-secuencias. La pendiente que tienen los picos múltiplos de un valor de sub-secuencia disminuye con pendiente igual al número de hits en un periodo.

- Incremento de la robustez.

Se incluye la posibilidad de añadir una cantidad inferior a N_MAX de elementos al buffer. Ya que en abundancia de recursos aumentar el tamaño en más múltiplos de N sería siempre deseable, este parámetro está indicado para sistemas con escasez de recursos y grandes periodos máximos esperados.

Una simulación análoga a la anterior, variando el número de elementos añadidos con ROB_INC desde 0 hasta 5 muestra lo siguiente:



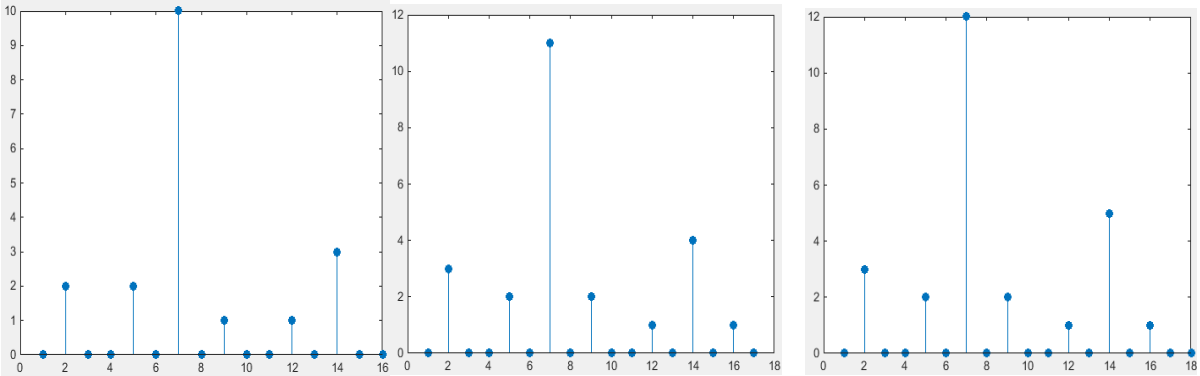


Ilustración 48. Identificación N=7, BUF_SIZ= 2 periodos, INC_ROB=0, 1, 2, 3, 4, 5 respectivamente.

Los hits para el periodo real, 7, aumentan en 1 por cada incremento adicional. Sin embargo, sólo en algunos casos, aumenta también el número de hits para un N menor, ya que no se terminan de almacenar todos los elementos de algunos de estos periodos pequeños, que se pueden encontrar repartidos a lo largo de la secuencia mayor.

Lo que permite asegurar una identificación correcta del periodo incluso con imperfecciones en los pulsos recibidos, es una mayor diferencia entre los hits del periodo real, y los de los sub-periodos más pequeños. Puesto que con este parámetro se consigue incrementar los hits del periodo real una cantidad INC_ROB, pero no todos los sub-periodos incrementan en el mismo número, se recomienda en los casos que se necesite, añadir un número de elementos *no* múltiplo del máximo periodo esperado, empleando el parámetro INC_ROB para ello.

- Inmunidad a pulsos perdidos.

El sistema ESM, que proporciona los pulsos al Predictor, no es infalible. Por ello es esperable que ocasionalmente se pierda algún pulso. El bloque de identificación de periodo se ha diseñado de forma que sea inmune a la pérdida de pulsos.

Esta inmunidad es proporcional al número de periodos completos que es capaz de almacenar el buffer que analiza.

Para simular los pulsos perdidos se prepara una modificación del script (*ANEXO III*), que simula el identificador ideal de forma que emule pulsos perdidos. Para que la simulación pueda emular cualquier escenario deseado, se introduce el parámetro "pulseMissingProb". Se trata de un vector. Cada elemento representa el número de pulsos que se avanzará la secuencia. El funcionamiento normal es que avance de uno en uno, sin embargo si el ESM falla y pierde un pulso, se habrá producido un salto de 2 posiciones. El valor del elemento es la probabilidad de que ocurra ese salto, por ello todos los elementos deben sumar probabilidad 1.

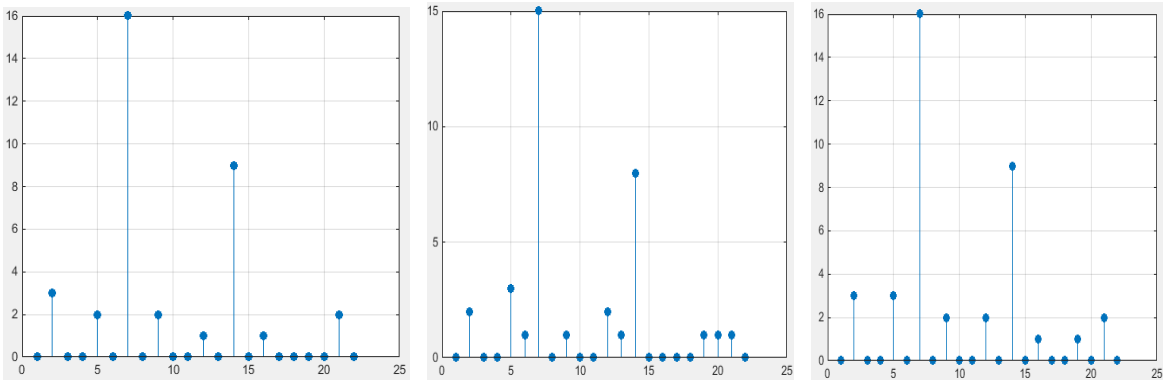
```
2       %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3       %%% CONFIGURATION PARAMETERS %%%
4       %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5
6 -     pulseMissProb = [0.9 0.07 0.02 0.01];
```

Ilustración 49. Ejemplo de introducción de probabilidad de pulso perdido: 90% pulsos correctos, 7% un pulso perdido, 2% dos pulsos perdidos, 1% tres pulsos perdidos.

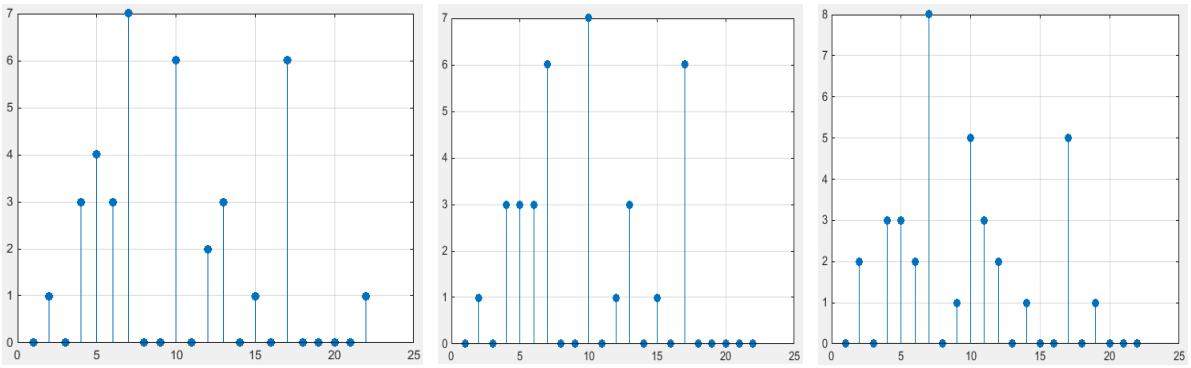
- Prueba de recursos reducidos

Se realiza una primera simulación, con el identificador de periodo dimensionado para albergar 3 periodos máximos. El script muestra a cada paso el histograma de hits para cada periodo. Se simulan 30 pulsos. Como siempre, con hits en el eje *y*, periodo N en el eje *x*; se muestra la evolución a través de las capturas más representativas. Además de simularse con tamaño de trama igual al máximo, este es un dimensionado muy pequeño, se verán sus efectos:

> *(Se han perdido 5 pulsos en total de los 30 simulados, es consistente con las probabilidades de fallo que se ha introducido. De nuevo, esta es una tasa de error demasiado alta, se pretende que el bloque falle, más que imitar un comportamiento de escenario real)*



El histograma sufre pequeñas oscilaciones mientras el número de fallos es cero o pequeño.



Durante el tiempo en el que se acumulan más errores, los índices para periodos que no son el real aumentan. Tal como se ve en esta sucesión, durante algunos breves instantes, un periodo erróneo puede llegar a sobrepasar al real, antes de que este vuelva a encabezar el histograma.

Se obtienen valiosas conclusiones de esta forma de simulación *worst-case*[22]. Aunque se ha simulado un supuesto demasiado malo para ser real, las relaciones son extrapolables para mejorar un sistema completo:

- Uno o varios pulsos perdidos dentro de un periodo recibido, provocan que desaparezcan todas las contribuciones que los elementos de ese periodo habrían hecho al real. Además, se verán como un posible sub-periodo, y sumará sus hits a los ya presentes, por lo que suben las cuentas de los sub-periodos y se reduce la diferencia con el real, el periodo completo N.
- En el momento que se unen todas las peores condiciones, el periodo identificado puede saltar momentáneamente a uno falso.
  En este punto, para sistemas con una gran limitación en recursos, sería de gran ayuda el monitor de velocidad de cambio de periodo explicado en el capítulo 4.3. El monitor puede actuar como filtro paso-bajo en el periodo identificado. Variando su parámetro de seguridad, se puede limitar la velocidad con la que cambie la identificación de periodo. De esta manera, picos repentinos como los observados en esta simulación, quedarían absorbidos y no afectarían al resto del sistema, aumentando la inmunidad a pulsos perdidos incluso disponiendo de pocos recursos para la identificación.

[22] *"worst-case"* – Peor caso.

Para obtener con total seguridad el periodo se debe cumplir:

- Sin pulsos perdidos el buffer tiene que tener mayor número de elementos restando un periodo, que las veces que se repite cualquier sub-periodo dentro de la secuencia (*n*).

$$(BUF\_SIZ - N) > n$$

- Cuando se espera la pérdida de pulsos, se ha de tener en cuenta que uno o varios pulsos perdidos dentro de un periodo, restan ~N hits a la identificación del periodo verdadero. Si se dispone de información de Inteligencia que indique la probabilidad de los distintos periodos y sub-periodos, así como estadísticas de los pulsos perdidos, se puede aproximar cuántas veces más se han de añadir N elementos al tamaño del buffer para mantener la inmunidad a pulsos perdidos.

- Prueba con recursos apropiados

La siguiente simulación emplea un tamaño de buffer más razonable, de 10 periodos. Se comprueba ahora el claro aumento de la inmunidad a pulsos perdidos.

Se introducen las siguientes probabilidades de fallo:

```
2      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3      %%% CONFIGURATION PARAMETERS %%%
4      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5
6 -    pulseMissProb = [0.8 0.15 0.03 0.02];
```

Ilustración 50. Probabilidades de pulso perdido: 80% pulsos correctos, 15% un pulso perdido, 3% dos pulsos perdidos, 2% tres pulsos perdidos.

Las probabilidades de fallo se han aumentado. Se han producido 9 pérdidas de los 30 pulsos entrantes simulados, es una altísima tasa de fallo, de nuevo se pone a prueba con dureza la resistencia del sub-sistema a pulsos perdidos. A continuación se muestra la evolución del histograma bajo el nuevo dimensionado del bloque, mostrando los momentos de más actividad.

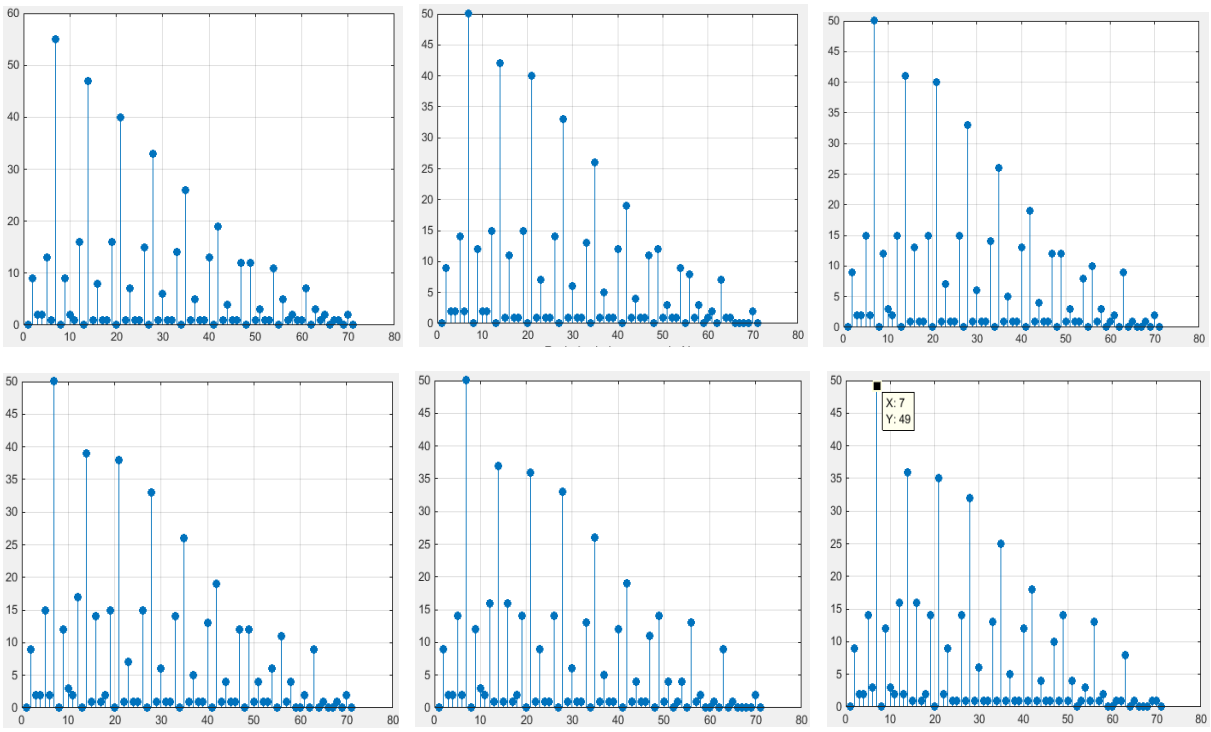


Ilustración 51. Desempeño del identificador en caso de pulsos faltantes con dimensionado apropiado.

Pese a que la tasa de fallos se ha incrementado, en ningún momento se produce una identificación errónea del periodo. Se puede ver cómo las variaciones producidas en el periodo de mayor número de fallos no pueden alcanzar el número de hits de los elementos correctamente recibidos. Tras el momento de mayor pérdida de pulsos, se vuelve a ir recuperando y haciendo mayor aún la diferencia entre el siguiente sub-periodo encontrado y el verdadero periodo N.

Se puede concluir que el identificador de periodo efectivamente resulta versátil para adaptarse a las necesidades del sistema que lo albergue y presenta una gran inmunidad a pulsos perdidos.

5.2 SIMULACIONES DEL FILTRO ESTIMADOR KALMAN

A partir de las relaciones aritméticas obtenidas del estudio de las ecuaciones de filtro Kalman, en el capítulo 2.2, se confecciona el modelo para operación en tiempo real. El objetivo es que el funcionamiento del modelo sea el mismo que se realizaría en el sistema electrónico hardware elegido para su implementación. De esta manera, no sólo se obtienen simulaciones realistas de los resultados esperables, sino que también deja un modelo que mediante herramientas de traducción de lenguaje de alto nivel, puede ser llevado de manera casi inmediata a una FPGA o SoC.

Para ello se han de inferir las reglas con las que construir cada matriz, en función de los parámetros de funcionamiento. Con esto se obtiene una forma flexible de simular el estimador para distintas situaciones.

El filtro Kalman se ha elegido por presentar una algoritmia ideal para la operación en tiempo real. Se basa en una operación recursiva, lo que significa que no necesita memorias que almacenen gran cantidad de información pasada, únicamente necesita información de las entradas actuales, y los estados anteriormente estimados con su matriz de incertidumbre. El algoritmo funciona en dos pasos. Primero estima los estados futuros, y después con la siguiente medida, actualiza las predicciones.

Esta es una visión general del resultado:

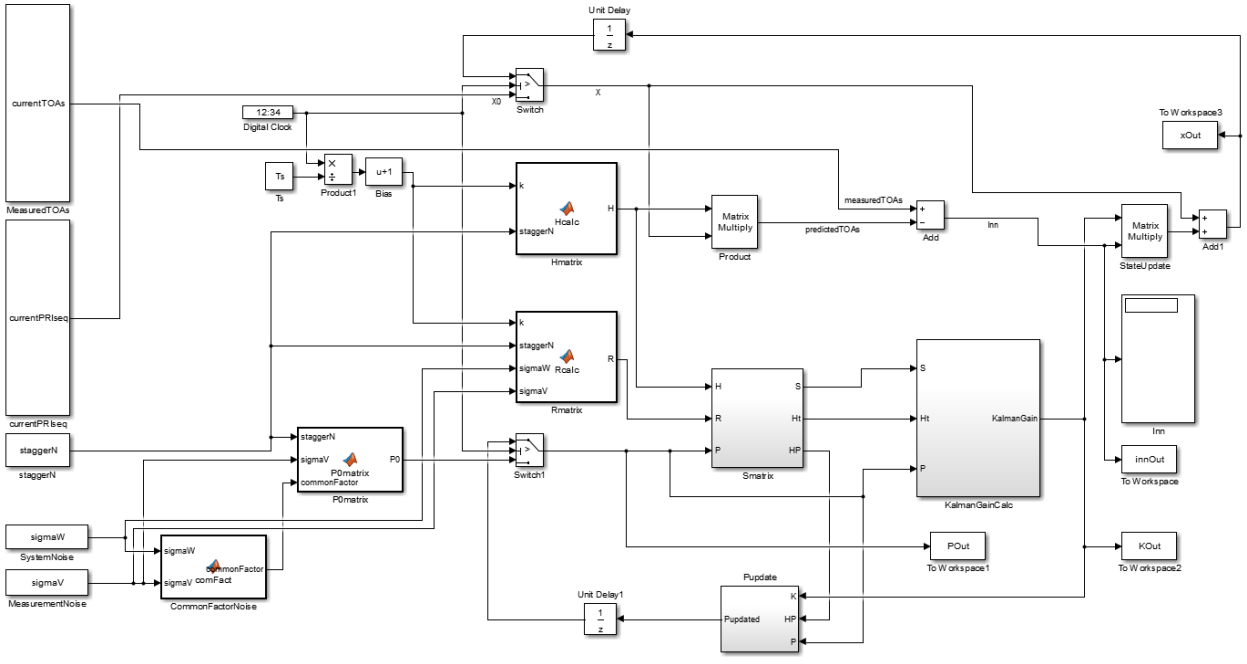


Ilustración 52. Visión del sub-sistema estimador.

Se pueden ver a la izquierda los datos que son obtenidos de otras partes del sistema, como los puertos de entrada de pulsos, y el tamaño que se ha identificado para la secuencia. Hacia la derecha se realiza el proceso de filtrado, pasando por la Ganancia Kalman. A partir de ahí se produce la realimentación, por la parte superior de la estimación de estados, y por la inferior de la desviación de los resultados.

5.2.1 COMPOSICIÓN DE MATRICES

Las matrices necesarias para la operación del filtro se introducen de forma que sigan la representación obtenida en la deducción teórica de las fórmulas. De esta forma son extensibles a cualquier condición de trabajo.

En un sistema multiamenaza, el hecho de que el filtro Kalman sea iterativo es sumamente beneficioso en el ahorro de recursos. Únicamente se han de conservar las matrices de estado, predicción y dispersión del error, para cada emisor. Por ello, si se decide seguir la arquitectura propuesta de multiplexación de los elementos de cálculo en el tiempo, estas matrices se replicarán una vez por cada emisor al que se quiera realizar seguimiento. Esta cualidad no sólo ahorra recursos, sino que hace el proceso de predicción mucho más rápido.

- Matrices de inicialización del filtro.

En el instante que el identificador de periodo notifica que se ha recibido un pulso, con el nivel de confianza elegido, se ha de inicializar el filtro Kalman. Requiere las dos matrices iniciales siguientes para realizar la primera estimación y arrancar el proceso iterativo:

- Matriz de estados iniciales X_0.

El tamaño de la matriz, al igual que el orden del filtro, está determinado por el tamaño N de la secuencia identificada. Por ello en el momento que se ha registrado un periodo completo, se toman los N elementos que serán los estados iniciales.

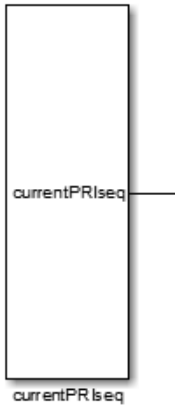


Ilustración 53. Puerto currentPRIseq.

Los N elementos de la secuencia que se está identificando se mapean al puerto “currentPRIseq” del filtro. De esta manera en cuanto se habilite la operación del filtro, los elementos registrados se emplearán como estado inicial.

- Covarianza del error inicial P_0.

A partir de la magnitud del ruido de sistema (desviación típica σ_w^2), y el de medida (desviación típica σ_v^2), con los que se establece que trabajará el sistema, se forma la matriz.

Se crean los bloques que toman los parámetros necesarios de otras partes del sistema:

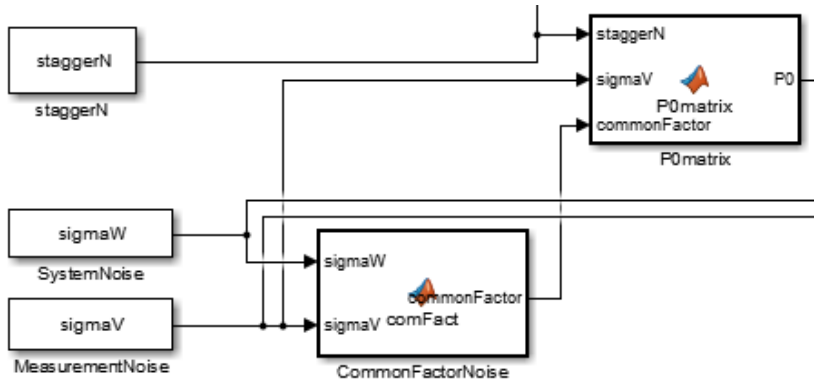


Ilustración 54. Bloques obtención *Po.*

A partir del valor de sigmaW y sigmaV, se calcula el valor de la varianza de los elementos de PRI, deducido en el capítulo 2.1.4. Este es a continuación situado en la diagonal principal de P0. Cada bloque incluye el código necesario para replicar la fórmula matemática que define la matriz:

$$P_0 = \begin{bmatrix} \sigma_w^2 + 2\sigma_v^2 & -\sigma_v^2 & 0 \\ -\sigma_v^2 & \sigma_w^2 + 2\sigma_v^2 & -\sigma_v^2 \\ 0 & -\sigma_v^2 & \sigma_w^2 + 2\sigma_v^2 \end{bmatrix}$$

Ilustración 55. Expresión *Po*

```
function P0 = P0matrix(staggerN, sigmaV, commonFactor)

  P0=zeros(staggerN,NstaggerN);

  P0(1,1)=commonFactor;
  P0(1,2)=-sigmaV^2;
  for i=2:(staggerN-1)
    P0(i,i-1)=-sigmaV^2;
    P0(i,i)=commonFactor;
    P0(i,i+1)=-sigmaV^2;
  end
  P0(staggerN,staggerN-1)=-sigmaV^2;
  P0(staggerN,staggerN)=commonFactor;
```

Ilustración 56. Código expresión *Po*

- Matrices de operación del filtro.

Las siguientes son las matrices con las que se opera en cada periodo de pulsos *k* recibido. Su valor es función del índice de periodo en que se encuentra la amenaza seleccionada.

- Matriz de medida, *Hk*.

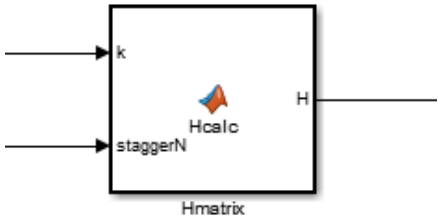


Ilustración 57. Bloque *Hk*.

Se extrae fácilmente el código para el valor de H según el índice de periodo:

$$H_k = \begin{bmatrix} k & k-1 & k-1 & k-1 & \dots & k-1 \\ k & k & k-1 & k-1 & \dots & k-1 \\ k & k & k & k-1 & \dots & k-1 \\ \vdots & & & & & \\ k & k & k & k & \dots & k \end{bmatrix}_{NxN}$$

```
function H = Hcalc(k,staggerN)

    H=zeros(N_MAX,N_MAX);
    for i=1:staggerN
        for j=1:staggerN
          if j <= i
            H(i,j)=k;
          else
            H(i,j)=k-1;
          end
        end
    end
```

Ilustración 58. Fórmula y código para configurar *Hk*.

- Matriz de covarianza de la medida, *Rk*.

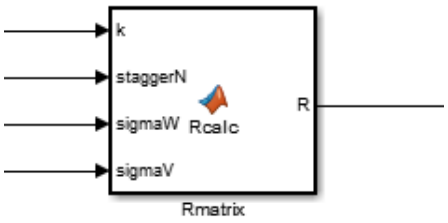


Ilustración 59. Bloque *Rk*.

Se toma la fórmula que describe el valor de R en el tiempo en función de los parámetros de entrada mostrados, y se traduce al siguiente código:

$$R_k = \begin{bmatrix} (Nk-N+1)\sigma_w^2+\sigma_v^2 & (Nk-N+1)\sigma_w^2 & \dots & (Nk-N+1)\sigma_w^2 \\ (Nk-N+1)\sigma_w^2 & (Nk-N+2)\sigma_w^2+\sigma_v^2 & \dots & (Nk-N+2)\sigma_w^2 \\ \vdots & \vdots & & \vdots \\ (Nk-N+1)\sigma_w^2 & (Nk-N+2)\sigma_w^2 & \dots & (Nk)\sigma_w^2++\sigma_v^2 \end{bmatrix}_{NxN}$$

```
function R = Rcalc(k,staggerN,sigmaW,sigmaV)

    R=zeros(N_MAX,N_MAX);

    comAux=staggerN*k-(staggerN-1);
    for i=1:staggerN
       R(i,i)=comAux*(sigmaW^2)+sigmaV^2;
       if i<staggerN
          R((i+1):staggerN,i)=comAux*(sigmaW^2);
          R(i,(i+1):staggerN)=comAux*(sigmaW^2);
       end
       comAux=comAux+1;
    end
```

Ilustración 60. Fórmula y código para configurar *Hk*

- Cambio de inicialización a cálculo iterativo.

Mediante un switch, se seleccionan los estados iniciales registrados durante el primer periodo *k*, tras la inicialización, este switch cierra el lazo de realimentación. Es necesario uno para conmutar cada matriz de inicialización, *Xo* y *Po*.

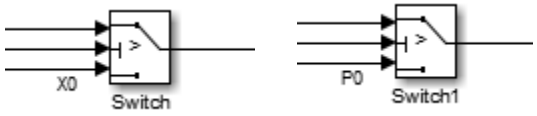


Ilustración 61. Switches inicialización-realimentación.

- División de las dos fases de filtrado.

Como se explicaba anteriormente, el filtro Kalman tiene una fase de predicción, y una de actualización. Para que esta sucesión se dé periodo a periodo, y no en un bucle infinito. Se incluyen retardos unitarios a la salida de cada cálculo iterativo. Se podrían registrar además otras partes intermedias del cálculo para procesarlo en *pipeline*.

- Secuencia de operaciones.

Se muestra a continuación el proceso efectuado sobre las señales para obtener el resultado deseado, que es la estimación de los próximos TOA. Partimos ahora de que se han realizado la primera fase con las condiciones iniciales.

- Cálculo de la covarianza del residuo, matriz *S*.

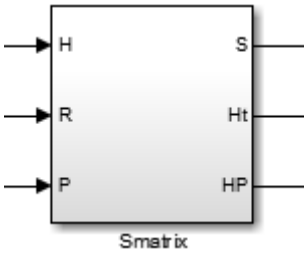


Ilustración 62. Bloque cálculo matriz *S.*

El bloque de cálculo para la obtención de S es un subsistema que realiza las operaciones expresadas en el desarrollo teórico:

$$S_k = H_k \hat{P}_k^- H_k^T + R_k$$

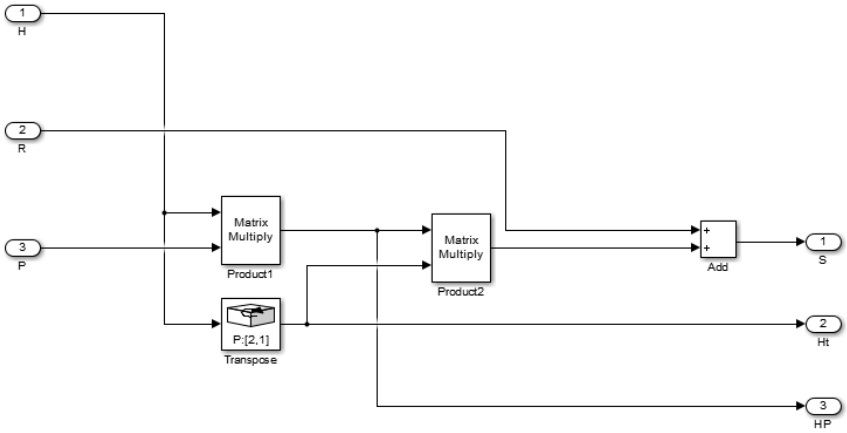


Ilustración 63. Fórmula y subsistema obtención de la matriz *S.*

Se han identificado las partes comunes con siguientes operaciones, y a parte de dar el resultado de S, se proporcionan los resultados de los cálculos intermedios. De esta manera el cálculo sólo se realiza una vez y se puede reducir la latencia del proceso.

- Cálculo de la ganancia Kalman, *K*.

Es el elemento crucial en el filtrado, adapta las estimaciones ponderando los elementos con mayor acierto.

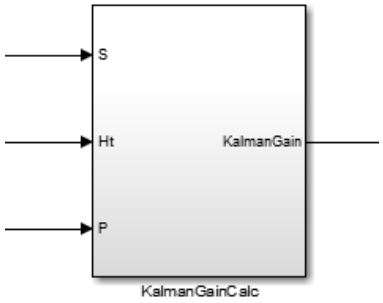


Ilustración 64. Bloque cálculo matriz *K*.

Su cálculo se realiza empleando operaciones parciales realizadas en el paso anterior, reduciendo con mucho las operaciones necesarias como se puede ver en el siguiente subsistema:

$$K_k = \hat{P}_k^- H_k^T S_k^{-1}$$

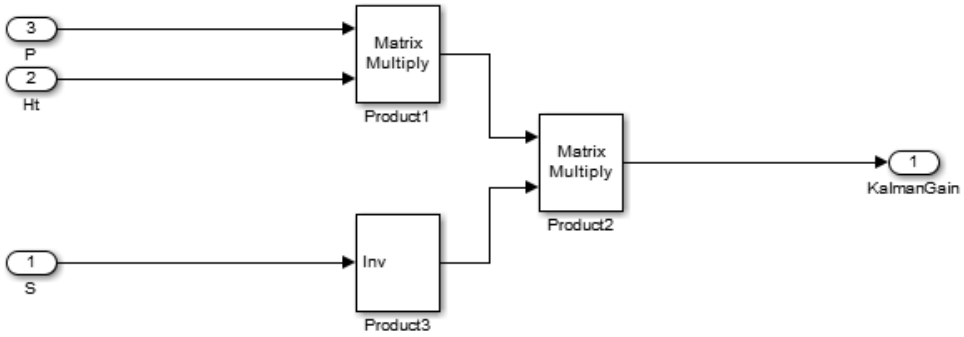


Ilustración 65. Fórmula y subsistema obtención de la matriz K.

- Residuo.

También conocido como innovación (*Inn*), se obtiene como la diferencia entre el estado predicho y el medido.

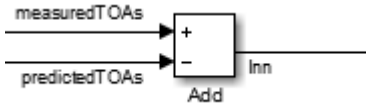


Ilustración 66. Cálculo del residuo.

o ACTUALIZACIÓN DE LA PREDICCIÓN DE ESTADOS

Este punto es el objetivo de todo el sistema Predictor. Gracias al ajuste de la ganancia Kalman y su ponderación de los elementos con mayor tasa de acierto, en este paso se pueden obtener estimaciones de los estados futuros, que son a su vez cada vez más exactas.

- La fórmula para la actualización de la predicción es: $\hat{X}_k = \hat{X}_k^- + K_k \, \text{Inn}$

- Esta se traduce en los siguientes bloques:

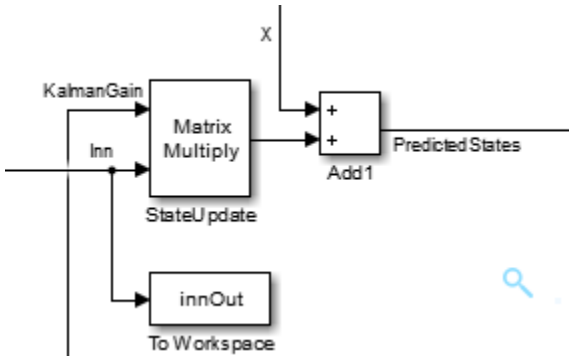


Ilustración 67. Bloques actualización predicción de estados.

- Actualización de la covarianza del error de estimación.

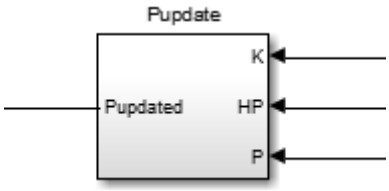


Ilustración 68. Bloque actualización *P*.

De nuevo, empleando partes precalculadas en otros pasos se reducen las operaciones a realizar para actualizar la matriz de covarianza del error de estimación siguiendo la siguiente expresión:

$$\hat{P}_k = \hat{P}_k^- - K_k S_k K_k^T$$

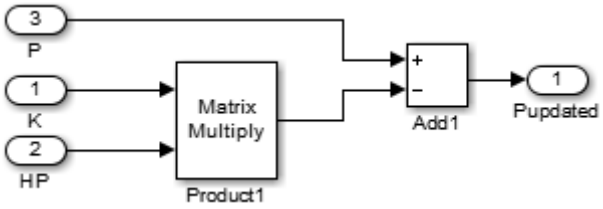


Ilustración 69. Ecuación y subsistema de actualización de la matriz P.

5.2.2 SIMULACIÓN

Elaborado el modelo empleando los elementos descritos, se introducen varias tramas y se simula para comprobar que se ha aplicado bien la teoría y se ha implementado correctamente para disfrutar de las ventajas del uso del filtrado recursivo.

Primero, ha de configurarse el modelo Simulink para el tamaño de secuencia que se desea comprobar. Se prepara un script (disponible en el *ANEXO IV*) que lanzará la simulación y mostrará los resultados.

Los parámetros que configuran la simulación son:

staggerSequence	Vector que contiene los elementos de un periodo Stagger tamaño N max. Ej.: [50 25 70 25 80 120 35]
pulsesSim	Número de pulsos recibidos a simular Ej.: 2000
sigmaW, sigmaV	Desviación típica de los ruidos de sistema (*w*) y de medida (*v*). Ej.: 0.5, 0.01

Tabla 2. Parámetros simulación Predictor.

- A partir de estos, se simulan tantos pulsos como se ha indicado, siguiendo el patrón de la secuencia Stagger introducida, y añadiendo los ruidos de sistema y medida a cada posición.

Tras la simulación, se muestra una ventana para cada estado sobre la que se representa con persistencia la evolución de la estimación. De esta forma se pueden ver superpuestos los resultados para cada estado (elemento Stagger).

- Simulación 1.

Se realiza una primera simulación con el mismo patrón de entrada que habíamos expuesto como ejemplo durante el testeo del identificador de periodo. De esta forma continuamos el que sería el camino que recorrería la trama al entrar en el sistema Predictor.

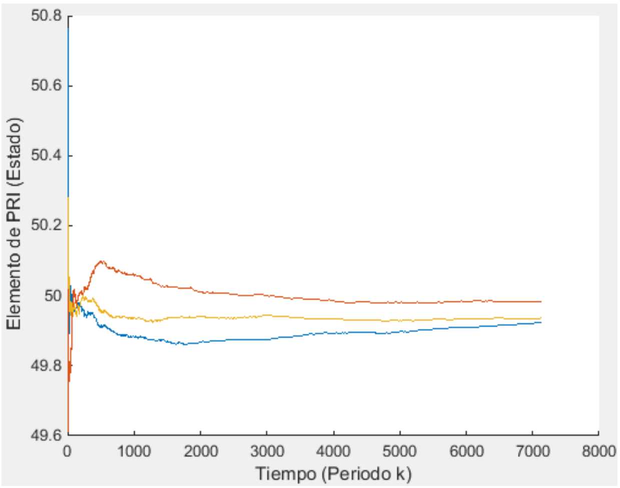
La configuración del script es:

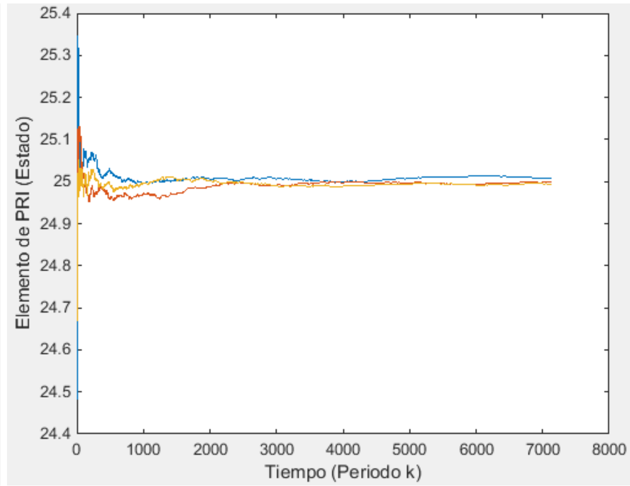
staggerSequence	[50 25 70 25 80 120 35]
pulsesSim	50000
sigmaW, sigmaV	0.5, 0.015

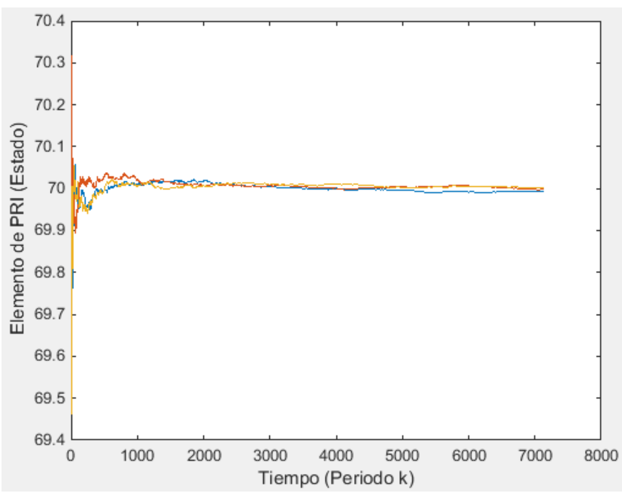
Tabla 3. Configuración simulación 1 Predictor.

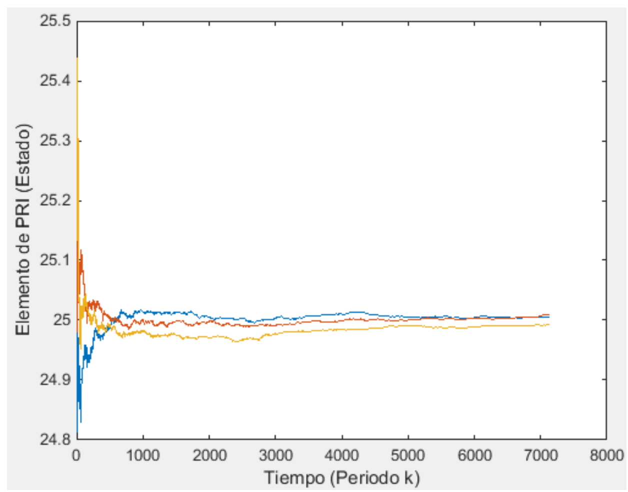
Se efectúan varias simulaciones en las que los valores aleatorios de ruido darán distintos resultados, y se comprueba la consistencia de los resultados.

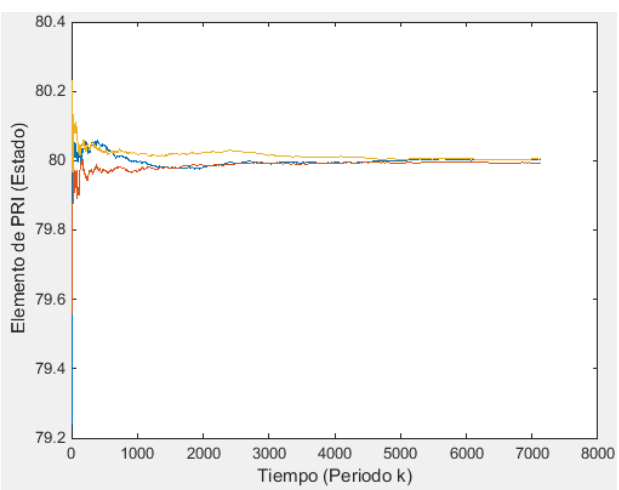
Dado que estamos procesando una secuencia de 7 posiciones, tendremos 7 estados. Esta es la evolución que sigue la estimación en cada uno de ellos:

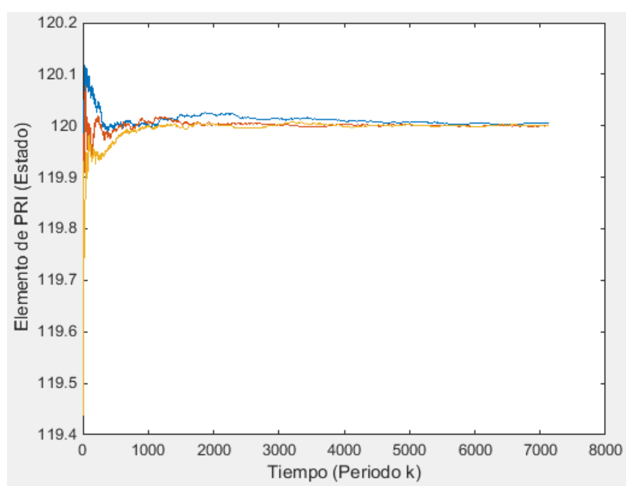












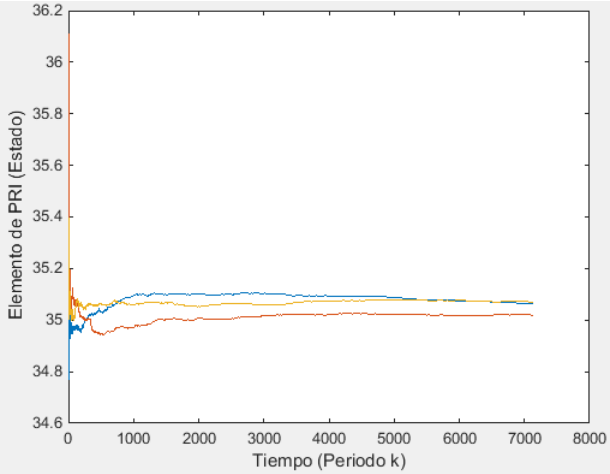


Ilustración 70. Evolución de la estimación de cada uno de los 7 estados.

Los resultados de la simulación son satisfactorios. Responde completamente al comportamiento esperado en un filtro de este tipo:

- Durante los primeros instantes, las grandes fluctuaciones en los datos de entrada se van reduciendo muy significativamente.
- Se estabiliza la medida dentro de una franja de error. Este es un comportamiento inevitable por la naturaleza de la señal, de la cual es imposible filtrar la absoluta totalidad del ruido.

- Simulación 2.

Para ver con más detalle la cantidad de ruido que hay al arrancar el filtro hacemos una simulación sola para los primeros instantes:

staggerSequence	[50 25 70 25 80 120 35]
pulsesSim	500
sigmaW, sigmaV	0.5, 0.015

Tabla 4. Configuración simulación 2 Predictor.

Se muestra la evolución de dos estados, que es representativa del resto.

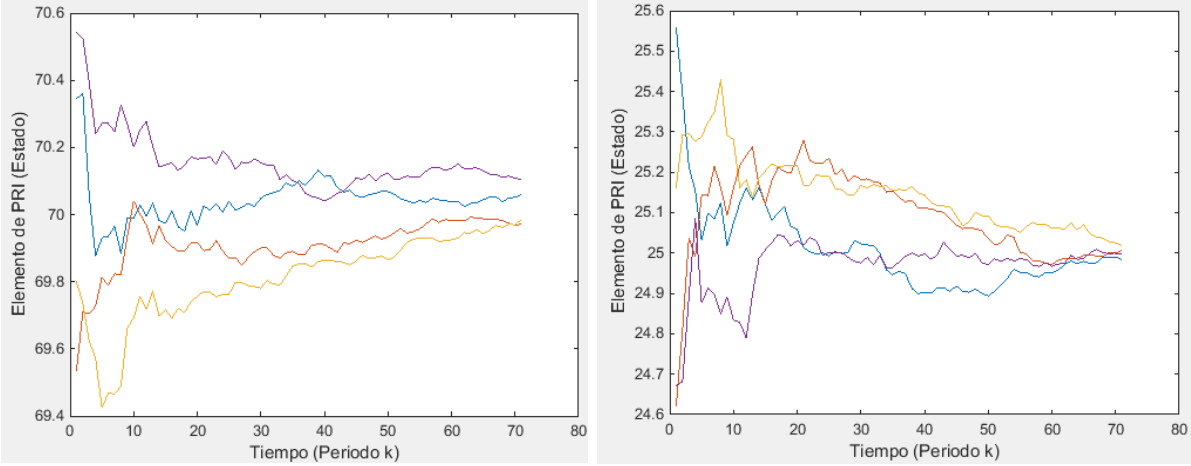


Ilustración 71. Evolución de la estimación de estados en los primeros ciclos de predicción.

Se puede ver con claridad cómo el error presente al inicio del proceso de predicción es muy superior a la banda de error en la que quedan confinadas posteriormente los estados.

Para comprobar por otro lado, cómo se comporta a largo plazo, se realiza una simulación con 2.000.000 de pulsos. Este es el resultado:

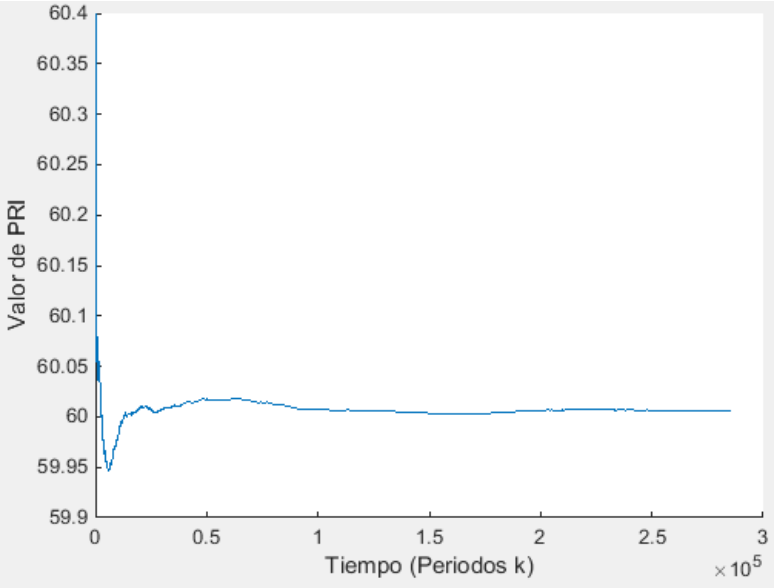


Ilustración 72. Simulación 2.000.000 de pulsos recibidos.

En régimen permanente de funcionamiento la estimación corresponde prácticamente con total exactitud al valor real del elemento Stagger, en este caso 60.

- Simulaciones con distintos tamaños de secuencia.

Para terminar de validar el correcto funcionamiento del Predictor, se realizan más simulaciones con distintos tamaños de secuencia N. Se resumen a continuación los resultados, mostrando la evolución de la estimación de uno de los estados, y todos los estados estimados en el último momento de la simulación:

- staggerSequence=[50 25 70 25 80 120 35 85 30];

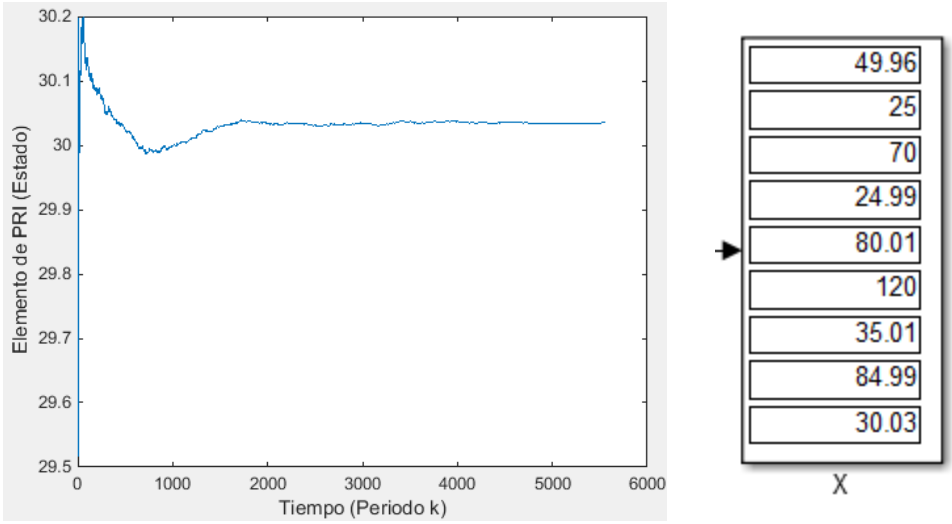


Ilustración 73. Simulación Predictor *N=9.*

- staggerSequence=[50 25 70 25 80 120 35 85 30 40 80];

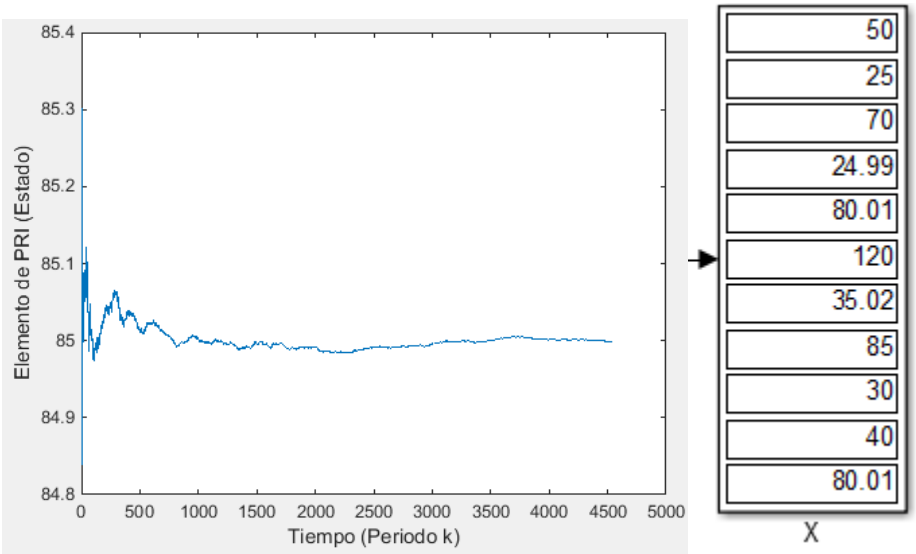


Ilustración 74. Simulación Predictor *N=11.*

5.2.3 CONCLUSIONES DE LA SIMULACIÓN

Sin duda queda de manifiesto con las simulaciones efectuadas, que el estimador Kalman ha sido implementado de forma satisfactoria, y aporta una enorme calidad a la estimación de futuros pulsos.

El predictor ha sido puesto bajo distintos tamaños de secuencia, con distintas tramas y simulando diversas combinaciones de ruido de sistema y de medida. En todos los casos el resultado es muy positivo, reduciendo drásticamente el error de predicción.

Dado que filtra con gran eficacia tanto el ruido propio de la generación y sensado de la señal, como el introducido como medida anti-sabotaje, se concluye que es un método eficaz para la predicción de Tiempos de Llegada para asistir en las Contramedidas hacia objetivos con agilidad en PRI.

CAPÍTULO 6:
INDICACIONES PARA LA ELECCIÓN DEL HARDWARE Y TRABAJO FUTURO

CAPÍTULO 6: INDICACIONES PARA LA ELECCIÓN DEL HARDWARE Y TRABAJO FUTURO

Todo el trabajo se ha orientado a sentar las bases de una implementación hardware de un sistema de apoyo a las Contramedidas basado en predictores.

La arquitectura es compatible con procesamiento en tiempo real, los sub-sistemas emplean formas de operación acordes, y se han elaborado de forma que empleen los recursos presentes en la mayoría de piezas comerciales.

Nuevos tipos de dispositivos ofrecen hoy en día tanto capacidades de procesamiento multiplicadas, como nuevas funcionalidades. Esto permite reinventar sistemas capaces de emplear técnicas que hasta ahora se encontraban limitadas por la tecnología disponible.

Se hace una revisión de los dispositivos que se consideran más apropiados para el desempeño de la tarea de predicción, los System On a Chip. Se hará también una selección de la familia más apropiada para la tarea que se ha desarrollado en este trabajo, y se indica qué partes del procesado se deberían realizar en cada parte del *SoC*.

6.1 INDICACIONES PARA LA ELECCIÓN DEL HARDWARE

Desde que la aparición de la electrónica, esta ha ido evolucionando a pasos agigantados. Uno de los parámetros que mejor ha medido esta evolución ha sido la miniaturización de los dispositivos electrónicos. Cuanto más pequeños son los dispositivos mayor es el número de ellos que cabe en un mismo espacio y, por lo tanto, la potencia y eficiencia de estos sistemas electrónicos mejora notablemente.

Los primeros dispositivos electrónicos, ampliamente usados en control de RADAR, dependían de tubos de vacío, los cuales ocupaban mucho espacio. A finales de 1940, aparecieron los primeros transistores, que ocupando menos espacio y consumían una pequeña fracción de energía que usaban los tubos de vacío. En la década de 1960, aparecieron los circuitos integrados. En el mismo espacio donde antes había un transistor, se podía poner un circuito integrado que contenía veinte transistores. La evolución de estos integrados fue predicha en su día por Gordon Moore, fundador de Intel Corporation, en su primera ley, la cual establece que el tamaño de un transistor se reduce un 50% cada año y medio.

Debido a la gran evolución en las tecnologías de fabricación, hoy en día es posible integrar todos los componentes de un sistema en un único circuito integrado. Por lo tanto, un SOC (del acrónimo System on a Chip) es un circuito integrado que contiene varios módulos que antes se encontraban en diferentes circuitos integrados.

Los módulos básicos que contiene un SOC suelen ser:

- Un microcontrolador o microprocesador, que hace las veces de cerebro del sistema.
- Módulos de memoria, ya sea RAM, ROM, EEPROM, o memoria flash.
- Controladores de memoria, los cuales se encargan de gestionar la memoria para su uso a través del micro o cualquier otro módulo que se incluya en el sistema.
- Un sistema de reloj que se deriva a todo el sistema, según las necesidades de cada módulo puede ajustarse la fase y la frecuencia.
- Interfaces externas para comunicarse con otros periféricos o sistemas externos, ya sea a través de buses USB, Ethernet, FireWire, SPI, I2C, etc.
- Interfaces analógicas, para recibir o crear señales analógicas.
- Y muy importante, buses de comunicación interna, para comunicar los diferentes módulos dentro del propio chip. Estos buses son mucho más veloces al estar integrados en el mismo chip, que si se realizara un rutado con pistas convencionales entre dos circuitos aislados.

Dependiendo las necesidades del producto, cada SOC puede tener una serie distinta de módulos internamente. En nuestro caso en concreto, nos interesan los SOC que contienen FPGAs. Este tipo de dispositivos pueden ofrecer mejoras clave sobre ciertos tipos de soluciones, tales como acortar tiempos para comercializar el producto, disminuir los riesgos del diseño, disminución del consumo energético y del coste, y miniaturización del diseño.

Una de las ventajas más importantes, es la flexibilidad, ya que este tipo de dispositivos son completamente programables, tanto en software, como en hardware. Haciendo que cualquier cambio de última hora se pueda llevar a cabo en diseños cerrados, algo imposible con hardware convencional, y también hace que sus actualizaciones y mejoras sean más sencillas.

Si comparamos un SOC frente a un procesador y una FPGA en circuitos integrados por separado, podemos encontrar que la reducción de espacio y consumo alcanza hasta un 50%, elimina costes logísticos y de empaquetado del circuito integrado. Si además, cada dispositivo poseían memoria por separado, esta se puede compartir para ambos, incluso integrar en el mismo SOC. Añadir las líneas de comunicación entre la FPGA y el procesador dentro del integrado, reduce el consumo sustancialmente, pudiendo aumentar el ancho de banda de comunicación, y reduciendo la latencia entre ambos dispositivos, además de evitar tediosos problemas de errores en la comunicación.

En la comparación de este tipo de SOC que incluye una FPGA, con otro SOC que incluya un sistema ASIC, podemos observar que la posibilidad de reconfiguración que posee la FPGA, puede hacerlo más flexible y actualizable, lo que hace que se adapte mejor a los cambios de los requerimientos del mercado y a los estándares emergentes. El diseño de un ASIC puede conllevar un elevado recargo en el diseño de las máscaras, o puede requerir la producción de un mínimo de unidades muy alto para obtener rentabilidad.

Hoy en día los SOC con FPGA mas importantes han sido diseñados por las principales empresas del sector: Xilinx, Altera y Microsemi.

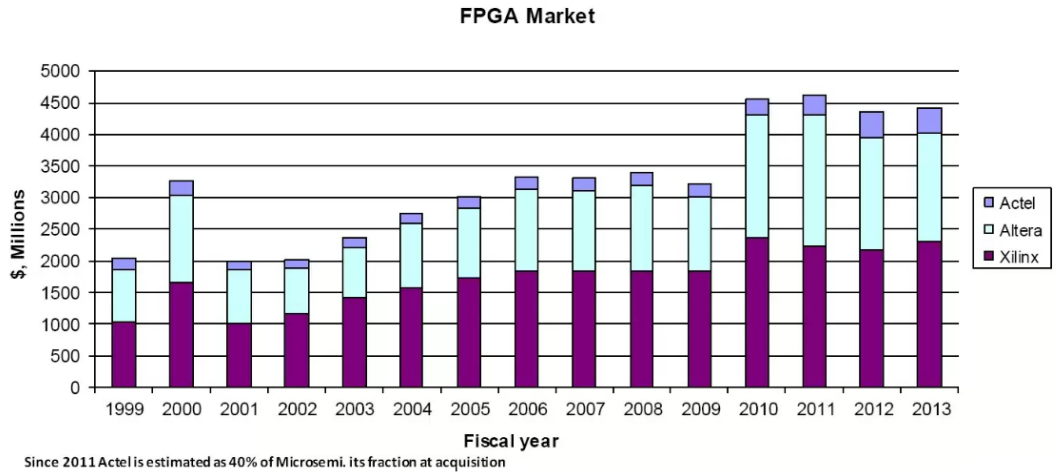


Ilustración 75. Beneficio de Altera, Actel (Adquirida por Microsemi) y Xilinx en los últimos años *[20]*

Estas tres empresas emplean procesadores ARM, la arquitectura más demandada actualmente, con todos sus periféricos, actuando igual que el procesador que se puede adquirir independientemente. Se resumen las características que se han encontrado interesantes del modelo de cada fabricante:

- *Zynq-7000* es el SOC creado por Xilinx, incluye un ARM Cortex-A9.

Es Dual Core a 1.0Ghz, con memoria Caché de nivel 1 de 32Kb, y memoria Caché de nivel 2 de 512Kb. La memoria RAM dedicada al procesador es de 256Kb, además posee un controlador de memoria externa, soportando LPDDR2, DDR2, DDR3, DDR3L, con detección y corrección de errores de 16 bits, cuya frecuencia máxima del bus es de 533Mhz.

Es útil a la hora de integrarlo en un sistema más complejo, el hecho de que el procesador posee ciertos periféricos para comunicaciones, 1 quad SPI, 1 controlador de memoria estática. 2 controladores para 10/100/1G Ethernet. 2 controladores USB 2.0 OTG (On-The-Go). 2 controladores SD/SDIO. 2 UART. 2 controladores IIC. 2 controladores CAN. 2 Controladores SPI (maestro o esclavo). 2 contadores/temporizadores de triple modo a 16bits. 1 temporizador watchdog de 24bits. La FPGA que contiene puede ser Artix-7 o Kintex-7 con una densidad de celdas lógicas de entre 28k y 444k.

Contiene también, muy útiles para comunicar con otras partes del sistema ECM, transceivers de alta velocidad.

Además como contiene dos conversores analógico-digital de 12bit a 1MSPS puede emplearse para monitorizar algún sensor directamente.

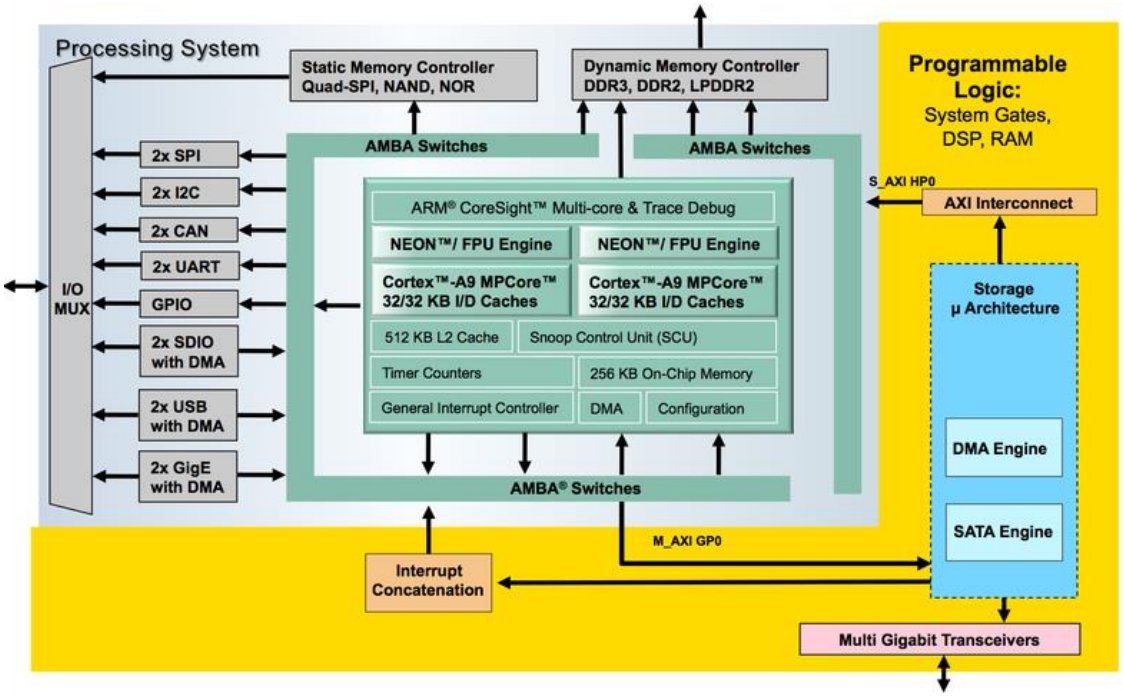


Ilustración 76. Diagrama de bloques de Zynq-7000 de Xilinx. *[21]*

- El SOC de Altera se compone de un ARM Cortex-A9, orientado a aplicaciones.

El procesador puede tener 1 o 2 núcleos a 1.05Ghz, con memoria Caché de nivel 1 de 32Kb, y memoria Caché de nivel 2 de 512Kb con detección y corrección de errores. La memoria RAM dedicada al procesador es de 64Kb, con control de acceso directo a memoria de 8 canales (ARM DMA330) y 32 peticiones de periféricos, además posee un controlador de memoria externa que soporta LPDDR2, DDR2, DDR3, DDR3L, también con detección y corrección de errores automáticas.

En cuanto a periféricos de comunicación, contiene 1 quad SPI, o un controlador de dual quad SPI con selector de 4 integrados. 2 controladores para 10/100/1G Ethernet. 2 controladores USB OTG (On-The-Go). 1 controlador SD/MMC/SDIO. 2 UART. 4 controladores IIC. 2 controladores CAN. 2 controladores SPI maestro y 2 controladores SPI esclavo. 2 temporizadores de propósito general a 32bits. 2 temporizadores watchdog de 32bits. La FPGA que contiene puede ser Cyclone V o Arria V con una densidad de celdas lógicas de entre 25k y 452k.

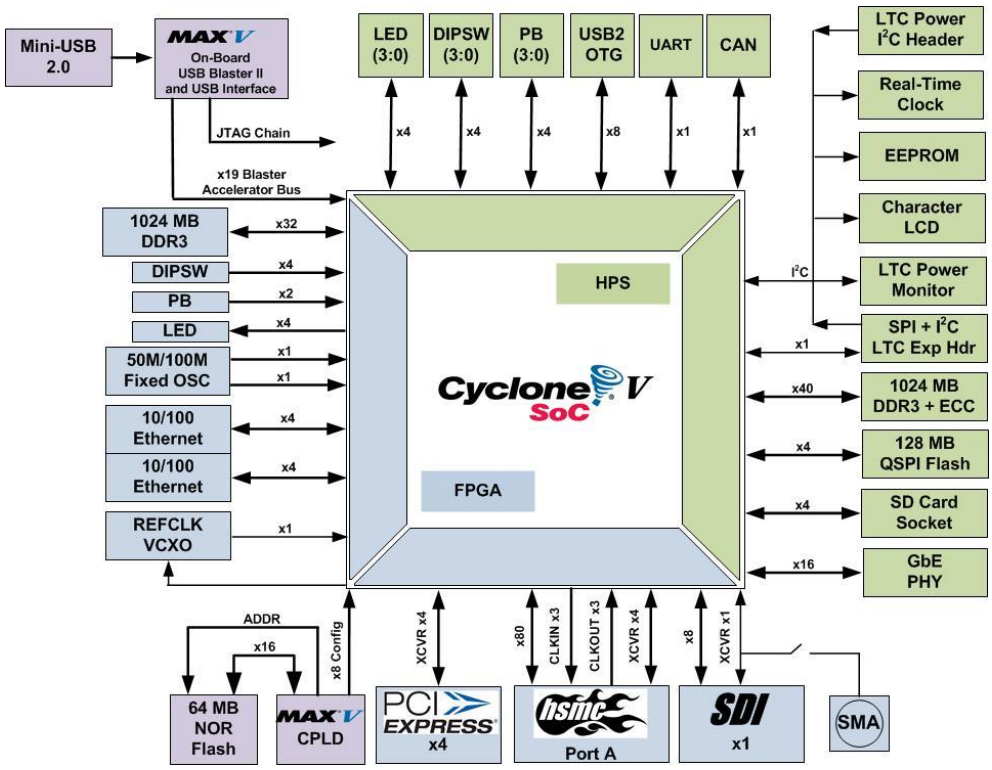


Ilustración 77. Diagrama de bloques de un SOC de Altera con Cyclone V *[22]*.

- Por último en la comparativa, *Smartfusion 2* de Microsemi.

Está compuesto por un ARM Cortex-M3, este tipo de procesador está orientado a realizar funciones de microcontrolador. Posee un solo núcleo a 166 Mhz, con memoria Caché de nivel 1 de 8Kb. Posee una unidad de gestión de memoria (MMU), y un controlador genérico de interrupciones anidado y vectorizado. La memoria RAM dedicada al procesador es de 64Kb, con control de acceso directo a memoria de 1 canal (HPDMA) y 4 peticiones de periféricos, además posee un controlador de memoria externa, soportando LPDDR, DDR2, DDR3, con detección y corrección de errores de 8, 16 y 32bits, cuya frecuencia máxima del bus es de 333Mhz.

Tiene los habituales puertos de comunicación serie y un reloj de tiempo real (RTC). La FPGA que contiene es una Fusion2 con una densidad de celdas lógicas de entre 6k y 146k. También posee transceivers de alta velocidad.

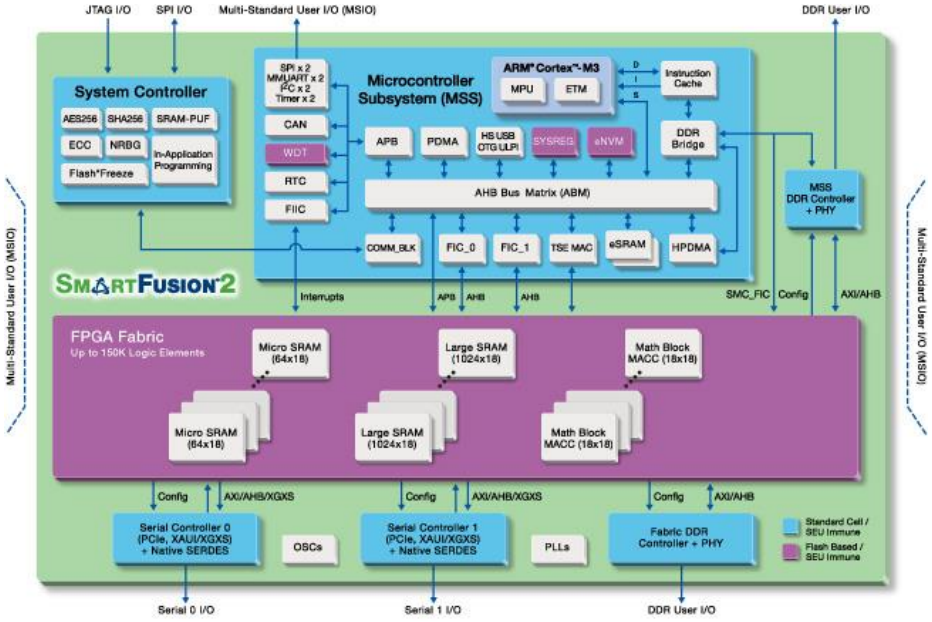


Ilustración 78. Diagrama de bloques de SmartFusion2 de Microsemi. *[23]*

En un primer análisis sobre las características de estos tres dispositivos, podemos ver que la SmartFusion2 de Microsemi posee un ARM Cortex-M3, el cual puede ser demasiado limitado en algunas de sus funciones como microcontrolador. Esta característica es el mayor punto débil del sistema, ya que podría limitar el diseño de la parte de procesado software. Por lo tanto se selecciona un SOC de Altera o Xilinx.

Ambos tienen características similares, pero se ha encontrado que diferencias en sus herramientas software son decisivas al decantarnos por Xilinx. Por ejemplo Altera usa el entorno de desarrollo de ARM, el cual es de pago (aunque se puede pedir versiones de evaluación), y la configuración del ARM a través de Quartus II es bastante complejo. Xilinx, en cambio integra todo en su herramienta Vivado.

Adicionalmente, y de especial utilidad tal como se ha planteado este trabajo, es que Xilinx provee herramientas que interactúan con Vivado para traducir los diseños realizados para simulación en Matlab y Simulink, y permiten generar VHDL directamente implementable en sus SoC.

Estas herramientas [24] son *HDL Coder*, que permite generar código VHDL a partir de los diseños en Matlab; y *System Generator*, que funciona con bloques Simulink de toolboxes nativas de Matla, suyas propias y otras como las de DSP[23].

[23] Digital Signal Processin – Procesado digital de señal

6.2 TRABAJO FUTURO

El siguiente paso hacia la realización completa del sistema Predictor de apoyo a ECM, sería la implementación hardware que se ha mantenido en todo momento como condición de diseño. Al mismo tiempo, se pueden implementar los mecanismos expuestos y desarrollar otros nuevos para mejorar el funcionamiento del sistema, aumentando su exactitud o su inmunidad a imperfecciones de otros sistemas de los que depende.

6.2.1 IMPLEMENTACIÓN EN SoC

El gran aliciente que presentan los SoC, es que permiten al diseñador no tener que prescindir de ninguna ventaja, ya que engloban microprocesadores y FPGAs, se puede aprovechar lo mejor de cada uno de los dos modos de procesado.

Los módulos creados en este trabajo, se pueden convertir en VHDL o bloques *black-box*[24] empleando las herramientas de Vivado mencionadas anteriormente. Para ello se han de adecuar los bloques a la representación en coma fija o flotante según las opciones disponibles, y posteriormente incluir los bloques específicos de Xilinx para traducción de lenguajes de alto nivel. Se puede incluso en ese momento, realizar simulaciones aceleradas con el hardware real conectado.

- División del procesado.

Tal como se ha propuesto, sería beneficiosa la multiplexación en el tiempo de los elementos de cálculo. Estos se implementarán en software en el microprocesador o en hardware en la parte de FPGA, según las características de cada uno.

- El proceso de identificación de periodo está especialmente creado para implementación en FPGA, que empleará los mismos componentes que se han usado en la simulación.

- Manteniendo aisladas las matrices con información de cada amenaza, la programación multi-hilo puede aprovecharse plenamente para realizar procesado software en virtual concurrencia. Las operaciones con matrices tales como una inversión, son más indicadas para ser realizadas por software en la parte de microprocesador.

[24] Black-box - Caja negra. Representación de un subsistema en el que se tienen entradas y salidas que funcionan de una forma determinada pero no se tiene acceso a los procesos internos que se desarrollan para ello.

6.2.2 EXPLORACIÓN DE OPCIONES PARA MEJORA DE LA PREDICCIÓN

La cantidad de métodos estadísticos utilizados para predicción en los más diversos campos es enorme. Desde economía hasta biología se encuentra utilidad a desarrollos matemáticos que pueden estimar o predecir distintos efectos. Con ciertas modificaciones se pueden adaptar al tipo de procesado de este sistema otras técnicas alternativas.

Además, varias técnicas de predicción podrían convivir dentro del mismo sistema incorporando un árbitro de la siguiente forma:

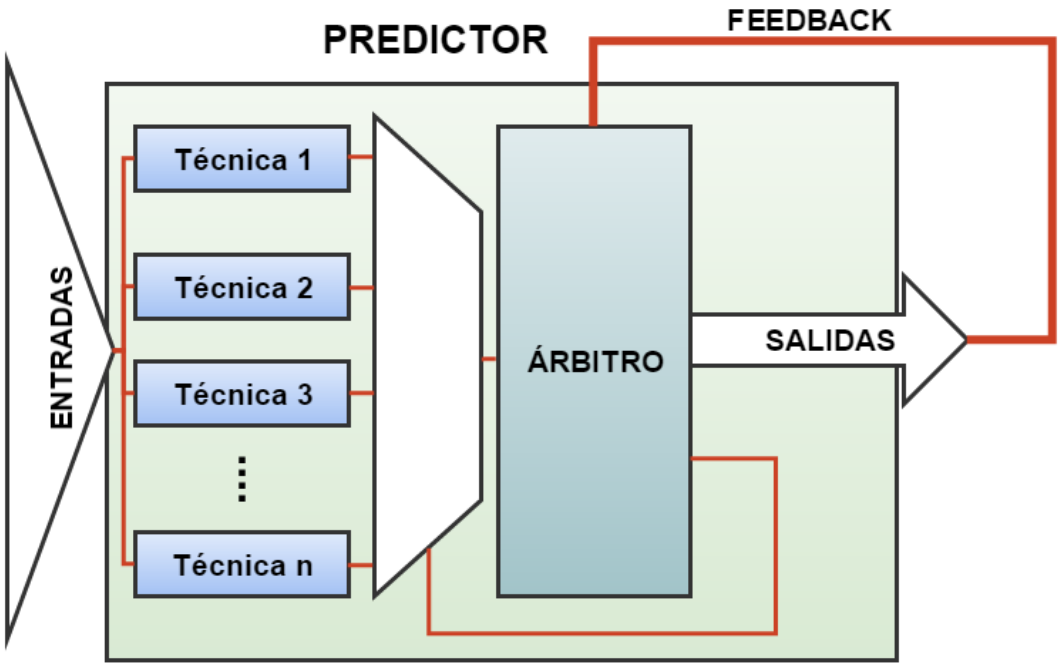


Ilustración 79. Propuesta árbitro para incluir más técnicas de predicción.

Todas las técnicas de predicción recibirían a la vez las entradas. Estas son procesadas y cada técnica proporciona su estimación. El árbitro, realimentándose con cada estimación y cada entrada siguiente, seleccionará de entre las técnicas la que más tasa de aciertos esté mostrando en el tiempo reciente.

Se abre así la posibilidad de inclusión de varias nuevas formas de predicción, junto con mecanismos para la selección en tiempo de ejecución de los estimadores más exactos.

PRESUPUESTO

El siguiente presupuesto engloba los costes en material y gastos de mano de obra, los gastos generales, el beneficio industrial, el coste de ejecución, la redacción del proyecto y el presupuesto total.

· EQUIPOS UTILIZADOS

EQUIPO	PRECIO	DURACIÓN	USO	TOTAL
Ordenador Portátil Intel i7	1100€	2 años	1 año	550€
Cables para el Ordenador	20€	2 años	1 año	10 €
SUBTOTAL EQUIPOS				660€

Tabla 5. Presupuesto equipos utilizados.

· PRESUPUESTO DE MATERIAL

DESCRIPCIÓN	UNIDADES	€/UNIDAD		TOTAL
Matlab R2014b	1	2.000€		2000€
DSP System Toolbox	1	1.250€		1250€
Simulink	1	3.000€		3000€
Libros de texto	3	15€		45€
SUBTOTAL MATERIALES				6.295€

· COSTE DE MANO DE OBRA

TRABAJADOR	Nº DE HORAS	€/HORA		TOTAL
Graduado	300	35€		10.500€
SUBTOTAL MANO DE OBRA				10.500€

· TOTAL:

Total coste de materiales:		
	Coste de equipos	660€
	Coste de material	6.295€
	Coste de mano de obra	10.500€
TOTAL	**COSTE DE MATERIALES**	17.455€

· GASTOS GENERALES

Aquí se incluyen los gastos de uso de instalaciones, y el coste de ejecución material es estos gastos sumado al coste de los materiales.

Los gastos generales son los producidos por la utilización de las instalaciones, por lo que el coste de ejecución material es el producido por los gastos generales, más el coste de materiales.

Coste de materiales	17.455€
Gastos generales	500€
COSTE DE EJECUCIÓN MATERIAL	17.955€

· PRESUPUESTO DE EJECUCIÓN POR CONTRATA

Resultado de la suma del coste de ejecución material, más el beneficio industrial, y añadiendo los impuestos correspondientes.

Coste de ejecución material	17.955€
Beneficio Industrial (7%)	1.256,85€
I.V.A. (21%)	4.034,48€
PRESUPUESTO DE EJECUCIÓN POR CONTRATA	23.246,28€

Honorarios por redacción del proyecto (7%)	1344.77€
TOTAL PRESUPUESTO	24.591.05€

El presupuesto total del proyecto es de VEITICUATRO MIL QUINIENTOS NOVENTA Y UNO EUROS CON CINCO CÉNTIMOS DE EURO.

BIBLIOGRAFÍA

[1] C. Wolff, "Radar Historical Overview," [Online]. Available: http://www.radartutorial.eu/04.history/hi04.en.html.

[2] Microwaves&RF, "Tracking the evolution of Radar," [Online]. Available: http://mwrf.com/military/tracking-evolution-radar.

[3] S. Willem, "Staggered pulse repetition frequency radar providing discrimination between first and second returns". Patent US 3491360 A, 1970.

[4] J. C. M. COOKE, "The capabilities and limitations of shipborne radar," UNITED STATES FLEET. [Online].

[5] N. A. W. C. (California), Electronic Warfare And Radar Systems Engineering Handbook, 2012.

[6] S. N. America, "History of Radar," [Online]. Available: http://www.decaturelectronics.com/information-center/history-of-radar/.

[7] Xilinx, "Zynq UltraScale+ MPSoC," [Online]. Available: http://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html.

[8] OP. [Online]. Available: http://defence pk/threads/pafs-acquisition-of-integrated-air-defense-network.81085/page-7.

[9] MC-CM, Enabling Technology, 2005.

[10] D. W. C. a. N. M. Huiping Cao, Discovering Partial Periodic Patterns in Discrete.

[11] R. W. Floyd, Floyd's cycle-finding algorithm, 1967.

[12] I. Yurchenko, "Finding a cycle in a linked list," [Online]. Available: https://ivanyu.me/blog/2013/11/24/finding-a-cycle-in-a-linked-list/.

[13] Mathworks, "Find Periodicity Using Autocorrelation," [Online]. Available: http://es.mathworks.com/help/signal/ug/find-periodicity-using-autocorrelation.html.

[14] Z. Y. Y. Q. L. Zhenhui, Digital Correlator Design Using Virtex-2 FPGAs, 2003.

[15] S. Avcu, Radar Pulse Repetition Interval Tracking, 2006.

[16] D. Gray, Parameter estimation for periodic discrete event processes, 1994.

[17] WikiMedia. [Online]. Available: https://es.wikipedia.org/wiki/Funci%C3%B3n_gaussiana#/media/File:Normal_distribution_pdf.png.

[18] C. Wolf, "Radar accuracy," [Online]. Available: http://www.radartutorial.eu/01.basics/Radars%20Accuracy.en.html.

[19] Matlab, "Block-Specific Parameters," [Online]. Available: http://es.mathworks.com/help/simulink/slref/block-specific-parameters.html.

[20] eetimes, "FPGAs as ASIC alternatives," [Online]. Available: http://www.eetimes.com/author.asp?doc_id=1322021.

[21] Xilinx, "Zynq-7000 All Programmable SoC," [Online]. Available: http://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html.

[22] Altera, "Cyclone V SoCs," [Online]. Available: https://www.altera.com/products/soc/portfolio/cyclone-v-soc/overview.html.

[23] Microsemi, "SmartFusion2 SoC FPGAs," [Online]. Available: http://www.microsemi.com/products/fpga-soc/soc-fpga/smartfusion2.

[24] M. Xilinx, "Xilinx FPGAs and Zynq SoCs," [Online]. Available: http://es.mathworks.com/solutions/fpga-design/simulink-with-xilinx-system-generator-for-dsp.html.

[25] G. D. a. Y. Y. J. Han, Effcient mining of partial periodic patterns in time, 1999.

ANEXOS

ANEXO I:

Script simulación identificador de periodo ideal

```
clear; clc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

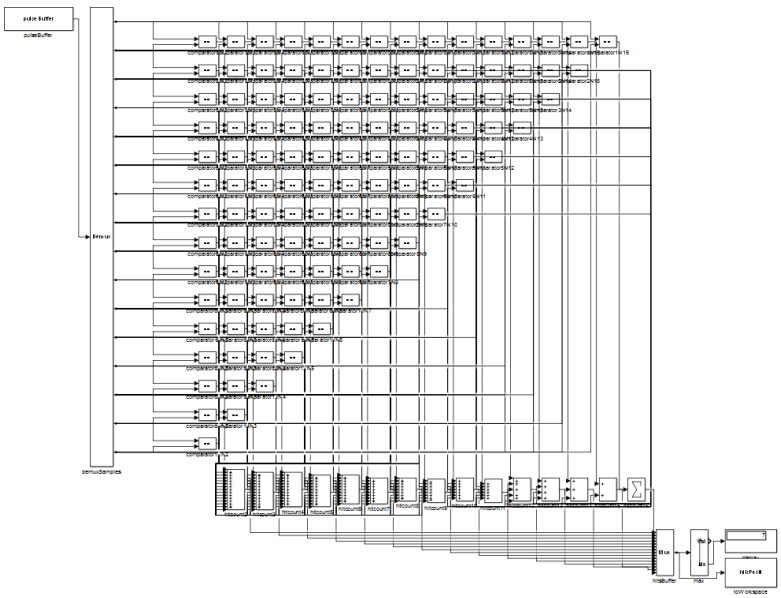
CONFIGURATION PARAMETERS

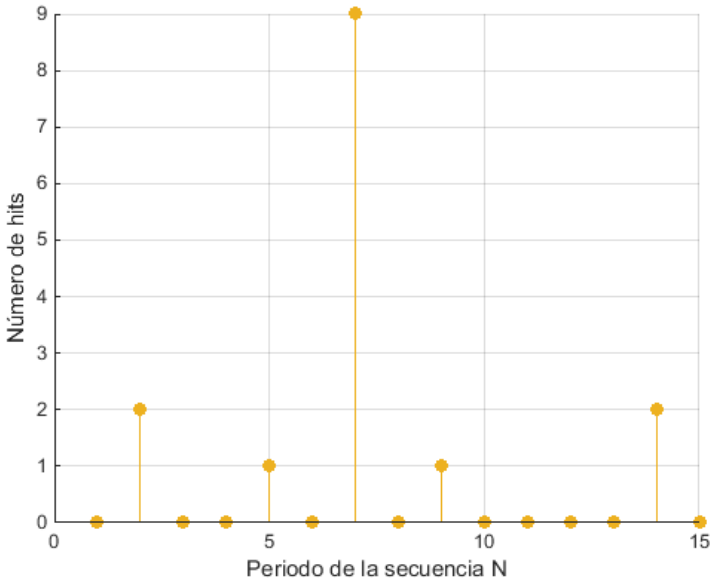
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
MAX_STAGGER_POSITIONS = 7;

BUF_SIZ = 0;
BUF_SIZ_TIMES_MAXN = 2;
ROB_INC = 2;

staggerSequence = [50 25 70 25 80 120 35];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
pulsesSim = 3;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if BUF_SIZ ~= 0
    BUF_SIZ_TIMES_MAXN = fix(BUF_SIZ/MAX_STAGGER_POSITIONS);
    ROB_INC = rem(BUF_SIZ,MAX_STAGGER_POSITIONS);
end

if staggerSequence == 0
    staggerSequence = randi([50 200],1,MAX_STAGGER_POSITIONS);
end
pulseBuffer = [];

for i=1: BUF_SIZ_TIMES_MAXN
    pulseBuffer = [pulseBuffer staggerSequence];
end
for i=1: ROB_INC
    pulseBuffer = [pulseBuffer staggerSequence(i)];
    pulseIndex = i;
end

sizePulseBuffer = size(pulseBuffer);
sizePulseBuffer = sizePulseBuffer(2);

sys = 'periodIdentificationIdeal';
new_system(sys)
open_system(sys)
```

```
x = 50;
y = 30;
w = 120;
h = 40;

pos = [x y x+w y+h];
add_block('built-in/Constant',[sys '/pulseBuffer'],'Position',pos,
'Value','pulseBuffer' );

x = x+w+30;
xs = 50;
ys = 50;
w = 40;
pos = [x y x+w y+sizePulseBuffer*ys];
add_block('built-in/Demux',[sys '/demuxSamples'],'Position',pos,
'Outputs',num2str(sizePulseBuffer) );

add_line(sys,'pulseBuffer/1','demuxSamples/1','autorouting','on');

x = x+w+100;
w = 30;
h = 20;
pos = [x y+h x+w y+h];

ya = y+h+sizePulseBuffer*ys;
xa = x+xs;
ha = 40;

ym = ya+ha+ys;
pos = [xa+sizePulseBuffer*xs ym xa+w+sizePulseBuffer*xs ym+sizePulseBuffer*5];
muxName = ['/' 'hitsBuffer'];
add_block('built-in/Mux',[sys muxName],'Position',pos,
'Inputs',num2str(sizePulseBuffer-1) );

pos(1) = pos(1)+w+30;
pos(3) = pos(3)+w+30;
maxName = ['/' 'max'];
add_block('dspstat3/Maximum',[sys maxName],'Position',pos );
sourcePort = ['hitsBuffer' '/' '1'];
destPort   = ['max' '/' '1'];
add_line(sys,sourcePort,destPort,'autorouting','on');

pos = [pos(1)+w+30 ym pos(3)+w+90 ym+sizePulseBuffer*5/2];
displayName = ['/' 'display'];
add_block('built-in/Display',[sys displayName],'Position',pos );
sourcePort = ['max' '/' '2'];
destPort   = ['display' '/' '1'];
add_line(sys,sourcePort,destPort,'autorouting','on');
```

```
pos(2) = pos(2)+sizePulseBuffer*5/2+10;
pos(4) = pos(4)+sizePulseBuffer*5/2+20;
toWorkspaceName = ['/' 'toWorkspace'];
add_block('built-in/ToWorkspace',[sys toWorkspaceName],'Position',pos,
'VariableName','hitsPerN', 'SaveFormat','Array', 'MaxDataPoints',num2str(1) );
sourcePort = ['hitsBuffer' '/' '1'];
destPort   = ['toWorkspace' '/' '1'];
add_line(sys,sourcePort,destPort,'autorouting','on');

for i = (sizePulseBuffer-1):-1:1
    pos = [xa+i*xs ya xa+w+i*xs ya+ha];
    adderName = ['/' 'hitcount' num2str(i+1)];
    add_block('built-in/Sum',[sys adderName],'Position',pos,
'Inputs',num2str(sizePulseBuffer-i), 'OutDataTypeStr', 'single' );

    sourcePort = ['hitcount' num2str(i+1) '/' '1'];
    destPort   = ['hitsBuffer' '/' num2str(i)];
    add_line(sys,sourcePort,destPort,'autorouting','on');
end



compName = 'Comparator name';
sourcePort = 'Pulse index';
destPort   = 'Comparator';

for j = 1:(sizePulseBuffer-1)
    for i= 1:(sizePulseBuffer-j)
        pos = [x+i*xs y+j*ys x+w+i*xs y+h+j*ys];
        compName = ['/' 'comparator' num2str(j) 'N' num2str(i+1)];
        add_block('built-in/RelationalOperator',[sys compName],'Position',pos,
'Operator','==' );

        sourcePort = ['demuxSamples' '/' num2str(j)];
        destPort   = ['comparator' num2str(j) 'N' num2str(i+1) '/' '1'];
        add_line(sys,sourcePort,destPort,'autorouting','on');
        sourcePort = ['demuxSamples' '/' num2str(i+j)];
        destPort   = ['comparator' num2str(j) 'N' num2str(i+1) '/' '2'];
        add_line(sys,sourcePort,destPort,'autorouting','on');
        sourcePort = ['comparator' num2str(j) 'N' num2str(i+1) '/' '1'];
        destPort   = ['hitcount' num2str(i+1) '/' num2str(j)];
        add_line(sys,sourcePort,destPort,'autorouting','on');
    end
end

hold on
pulseIndex = MAX_STAGGER_POSITIONS ;
for i = 1:pulsesSim
    pulseBuffer = circshift(pulseBuffer,1,2);
    pulseBuffer(1) = staggerSequence(pulseIndex)
```

```
    pulseIndex = pulseIndex-1;
    if pulseIndex<1
        pulseIndex = MAX_STAGGER_POSITIONS;
    end
    sim('periodIdentificationIdeal',0.01)

    stem(hitsPerN,'filled')
    xlabel('Periodo de la secuencia N');
    ylabel('Número de hits');
    grid on;
end
```





ANEXO II:

Script simulación identificador de periodo variando sucesivamente tamaño de periodo y tamaño del identificador

```
clear; clc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

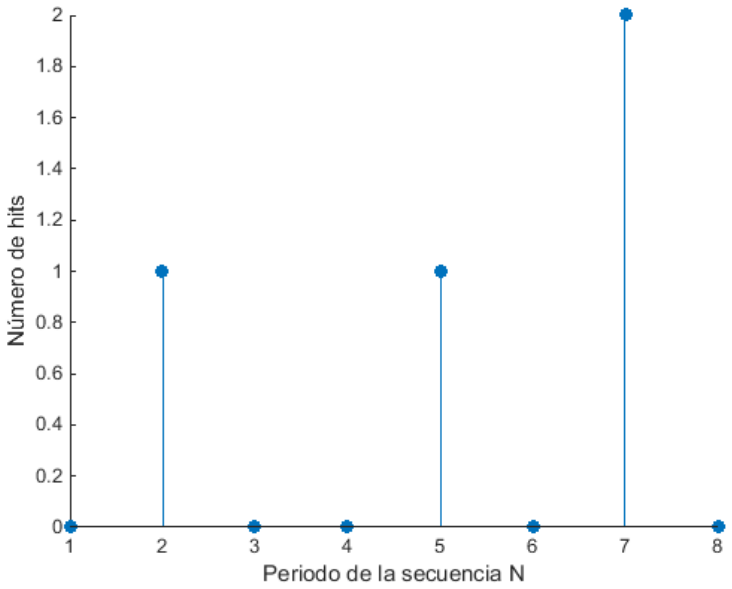
SIMULATION PARAMETERS

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
MAX_STAGGER_POSITIONS = 7;
staggerSequence = [50 25 70 25 80 120 35];
runsN = 1;
runsBufSiz = 1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

hold on
for varyN = MAX_STAGGER_POSITIONS:MAX_STAGGER_POSITIONS:MAX_STAGGER_POSITIONS*runsN
    for varyBuffSiz = 1:2:2*runsBufSiz

MAX_STAGGER_POSITIONS = varyN
BUF_SIZ_TIMES_N = varyBuffSiz

ROB_INC = 2;


if staggerSequence == 0
    staggerSequence = randi([50 200],1,MAX_STAGGER_POSITIONS);
end
pulseBuffer = [];

for i=1: BUF_SIZ_TIMES_N
    pulseBuffer = [pulseBuffer staggerSequence];
end
for i=1: ROB_INC
    pulseBuffer = [pulseBuffer staggerSequence(i)];
end

sizePulseBuffer = size(pulseBuffer);
```

```
sizePulseBuffer = sizePulseBuffer(2);

sys = 'periodIdentificationIdeal';
new_system(sys) % Create the model
open_system(sys)


x = 50;
y = 30;
w = 120;
h = 40;

pos = [x y x+w y+h];
add_block('built-in/Constant',[sys '/pulseBuffer'],'Position',pos,
'Value','pulseBuffer' );

x = x+w+30;
xs = 50;
ys = 50;
w = 40;
pos = [x y x+w y+sizePulseBuffer*ys];
add_block('built-in/Demux',[sys '/demuxSamples'],'Position',pos,
'Outputs',num2str(sizePulseBuffer) );

add_line(sys,'pulseBuffer/1','demuxSamples/1','autorouting','on');

x = x+w+100;
w = 30;
h = 20;
pos = [x y+h x+w y+h];

ya = y+h+sizePulseBuffer*ys;
xa = x+xs;
ha = 40;

ym = ya+ha+ys;
pos = [xa+sizePulseBuffer*xs ym xa+w+sizePulseBuffer*xs ym+sizePulseBuffer*5];
muxName = ['/' 'hitsBuffer'];
add_block('built-in/Mux',[sys muxName],'Position',pos,
'Inputs',num2str(sizePulseBuffer-1) );

pos(1) = pos(1)+w+30;
pos(3) = pos(3)+w+30;
maxName = ['/' 'max'];
add_block('dspstat3/Maximum',[sys maxName],'Position',pos );
sourcePort = ['hitsBuffer' '/' '1'];
destPort   = ['max' '/' '1'];
add_line(sys,sourcePort,destPort,'autorouting','on');

pos = [pos(1)+w+30 ym pos(3)+w+90 ym+sizePulseBuffer*5/2];
```

```
displayName = ['/' 'display'];
add_block('built-in/Display',[sys displayName],'Position',pos );
sourcePort = ['max' '/' '2'];
destPort   = ['display' '/' '1'];
add_line(sys,sourcePort,destPort,'autorouting','on');

pos(2) = pos(2)+sizePulseBuffer*5/2+10;
pos(4) = pos(4)+sizePulseBuffer*5/2+20;
toWorkspaceName = ['/' 'toWorkspace'];
add_block('built-in/ToWorkspace',[sys toWorkspaceName],'Position',pos,
'VariableName','hitsPerN', 'SaveFormat','Array', 'MaxDataPoints',num2str(1) );
sourcePort = ['hitsBuffer' '/' '1'];
destPort   = ['toWorkspace' '/' '1'];
add_line(sys,sourcePort,destPort,'autorouting','on');

for i = (sizePulseBuffer-1):-1:1
    pos = [xa+i*xs ya xa+w+i*xs ya+ha];
    adderName = ['/' 'hitcount' num2str(i+1)];
    add_block('built-in/Sum',[sys adderName],'Position',pos,
'Inputs',num2str(sizePulseBuffer-i), 'OutDataTypeStr', 'single' );

    sourcePort = ['hitcount' num2str(i+1) '/' '1'];
    destPort   = ['hitsBuffer' '/' num2str(i)];
    add_line(sys,sourcePort,destPort,'autorouting','on');
end



compName = 'Comparator name';
sourcePort = 'Pulse index';
destPort   = 'Comparator';

for j = 1:(sizePulseBuffer-1)
    for i= 1:(sizePulseBuffer-j)
        pos = [x+i*xs y+j*ys x+w+i*xs y+h+j*ys];
        compName = ['/' 'comparator' num2str(j) 'N' num2str(i+1)];
        add_block('built-in/RelationalOperator',[sys compName],'Position',pos,
'Operator','==' );

        sourcePort = ['demuxSamples' '/' num2str(j)];
        destPort   = ['comparator' num2str(j) 'N' num2str(i+1) '/' '1'];
        add_line(sys,sourcePort,destPort,'autorouting','on');
        sourcePort = ['demuxSamples' '/' num2str(i+j)];
        destPort   = ['comparator' num2str(j) 'N' num2str(i+1) '/' '2'];
        add_line(sys,sourcePort,destPort,'autorouting','on');
        sourcePort = ['comparator' num2str(j) 'N' num2str(i+1) '/' '1'];
        destPort   = ['hitcount' num2str(i+1) '/' num2str(j)];
        add_line(sys,sourcePort,destPort,'autorouting','on');
    end
end
```

```
        sim('periodIdentificationIdeal',0.01)
        stem(hitsPerN,'filled')
        xlabel('Periodo de la secuencia N');
        ylabel('Número de hits');
        pause
        figure

        close_system('periodIdentificationIdeal', 0)
    end
end
```



ANEXO III:

Script simulación identificador de periodo con pulsos perdidos

```
clear; clc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

CONFIGURATION PARAMETERS

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

pulseMissProb = [0.8 0.15 0.03 0.02];

MAX_STAGGER_POSITIONS = 7;

BUF_SIZ = 0;
BUF_SIZ_TIMES_MAXN = 3;
ROB_INC = 2;

staggerSequence = [50 25 70 25 80 120 35];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
pulsesSim = 5;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if BUF_SIZ ~= 0
    BUF_SIZ_TIMES_MAXN = fix(BUF_SIZ/MAX_STAGGER_POSITIONS);
    ROB_INC = rem(BUF_SIZ,MAX_STAGGER_POSITIONS);
end

if staggerSequence == 0
    staggerSequence = randi([50 200],1,MAX_STAGGER_POSITIONS);
end
pulseBuffer = [];
pulsesMissed = 0;

for i=1: BUF_SIZ_TIMES_MAXN
    pulseBuffer = [pulseBuffer staggerSequence];
end
for i=1: ROB_INC
    pulseBuffer = [pulseBuffer staggerSequence(i)];
    pulseIndex = i;
end
```

```
sizePulseBuffer = size(pulseBuffer);
sizePulseBuffer = sizePulseBuffer(2);

sys = 'periodIdentificationIdeal';
new_system(sys)
open_system(sys)


x = 50;
y = 30;
w = 120;
h = 40;

pos = [x y x+w y+h];
add_block('built-in/Constant',[sys '/pulseBuffer'],'Position',pos,
'Value','pulseBuffer' );

x = x+w+30;
xs = 50;
ys = 50;
w = 40;
pos = [x y x+w y+sizePulseBuffer*ys];
add_block('built-in/Demux',[sys '/demuxSamples'],'Position',pos,
'Outputs',num2str(sizePulseBuffer) );

add_line(sys,'pulseBuffer/1','demuxSamples/1','autorouting','on');

x = x+w+100;
w = 30;
h = 20;
pos = [x y+h x+w y+h];

ya = y+h+sizePulseBuffer*ys;
xa = x+xs;
ha = 40;

ym = ya+ha+ys;
pos = [xa+sizePulseBuffer*xs ym xa+w+sizePulseBuffer*xs ym+sizePulseBuffer*5];
muxName = ['/' 'hitsBuffer'];
add_block('built-in/Mux',[sys muxName],'Position',pos,
'Inputs',num2str(sizePulseBuffer-1) );

pos(1) = pos(1)+w+30;
pos(3) = pos(3)+w+30;
maxName = ['/' 'max'];
add_block('dspstat3/Maximum',[sys maxName],'Position',pos );
sourcePort = ['hitsBuffer' '/' '1'];
destPort   = ['max' '/' '1'];
add_line(sys,sourcePort,destPort,'autorouting','on');
```

```
pos = [pos(1)+w+30 ym pos(3)+w+90 ym+sizePulseBuffer*5/2];
displayName = ['/' 'display'];
add_block('built-in/Display',[sys displayName],'Position',pos );
sourcePort = ['max' '/' '2'];
destPort   = ['display' '/' '1'];
add_line(sys,sourcePort,destPort,'autorouting','on');

pos(2) = pos(2)+sizePulseBuffer*5/2+10;
pos(4) = pos(4)+sizePulseBuffer*5/2+20;
toWorkspaceName = ['/' 'toWorkspace'];
add_block('built-in/ToWorkspace',[sys toWorkspaceName],'Position',pos,
'VariableName','hitsPerN', 'SaveFormat','Array', 'MaxDataPoints',num2str(1) );
sourcePort = ['hitsBuffer' '/' '1'];
destPort   = ['toWorkspace' '/' '1'];
add_line(sys,sourcePort,destPort,'autorouting','on');

for i = (sizePulseBuffer-1):-1:1
    pos = [xa+i*xs ya xa+w+i*xs ya+ha];
    adderName = ['/' 'hitcount' num2str(i+1)];
    add_block('built-in/Sum',[sys adderName],'Position',pos,
'Inputs',num2str(sizePulseBuffer-i), 'OutDataTypeStr', 'single' );

    sourcePort = ['hitcount' num2str(i+1) '/' '1'];
    destPort   = ['hitsBuffer' '/' num2str(i)];
    add_line(sys,sourcePort,destPort,'autorouting','on');
end



compName = 'Comparator name';
sourcePort = 'Pulse index';
destPort   = 'Comparator';

for j = 1:(sizePulseBuffer-1)
    for i= 1:(sizePulseBuffer-j)
        pos = [x+i*xs y+j*ys x+w+i*xs y+h+j*ys];
        compName = ['/' 'comparator' num2str(j) 'N' num2str(i+1)];
        add_block('built-in/RelationalOperator',[sys compName],'Position',pos,
'Operator','==' );

        sourcePort = ['demuxSamples' '/' num2str(j)];
        destPort   = ['comparator' num2str(j) 'N' num2str(i+1) '/' '1'];
        add_line(sys,sourcePort,destPort,'autorouting','on');
        sourcePort = ['demuxSamples' '/' num2str(i+j)];
        destPort   = ['comparator' num2str(j) 'N' num2str(i+1) '/' '2'];
        add_line(sys,sourcePort,destPort,'autorouting','on');
        sourcePort = ['comparator' num2str(j) 'N' num2str(i+1) '/' '1'];
        destPort   = ['hitcount' num2str(i+1) '/' num2str(j)];
        add_line(sys,sourcePort,destPort,'autorouting','on');
    end
```

```
end

hold on
pulseIndex = MAX_STAGGER_POSITIONS ;
for i = 1:pulsesSim
    pulseBuffer = circshift(pulseBuffer,1,2);
    pulseBuffer(1) = staggerSequence(pulseIndex);

    obtRand = rand;
    missinProb = sum(obtRand >= cumsum([0, pulseMissProb]));

    pulseIndex = pulseIndex - missinProb;

    if missinProb ~= 1
        pulsesMissed = pulsesMissed + (missinProb-1)
    end

    if pulseIndex<1
        pulseIndex = MAX_STAGGER_POSITIONS;
    end
    sim('periodIdentificationIdeal',0.01)

    figure
    stem(hitsPerN,'filled')
    xlabel('Periodo de la secuencia N');
    ylabel('Número de hits');
    grid on;
end
```

ANEXO IV:

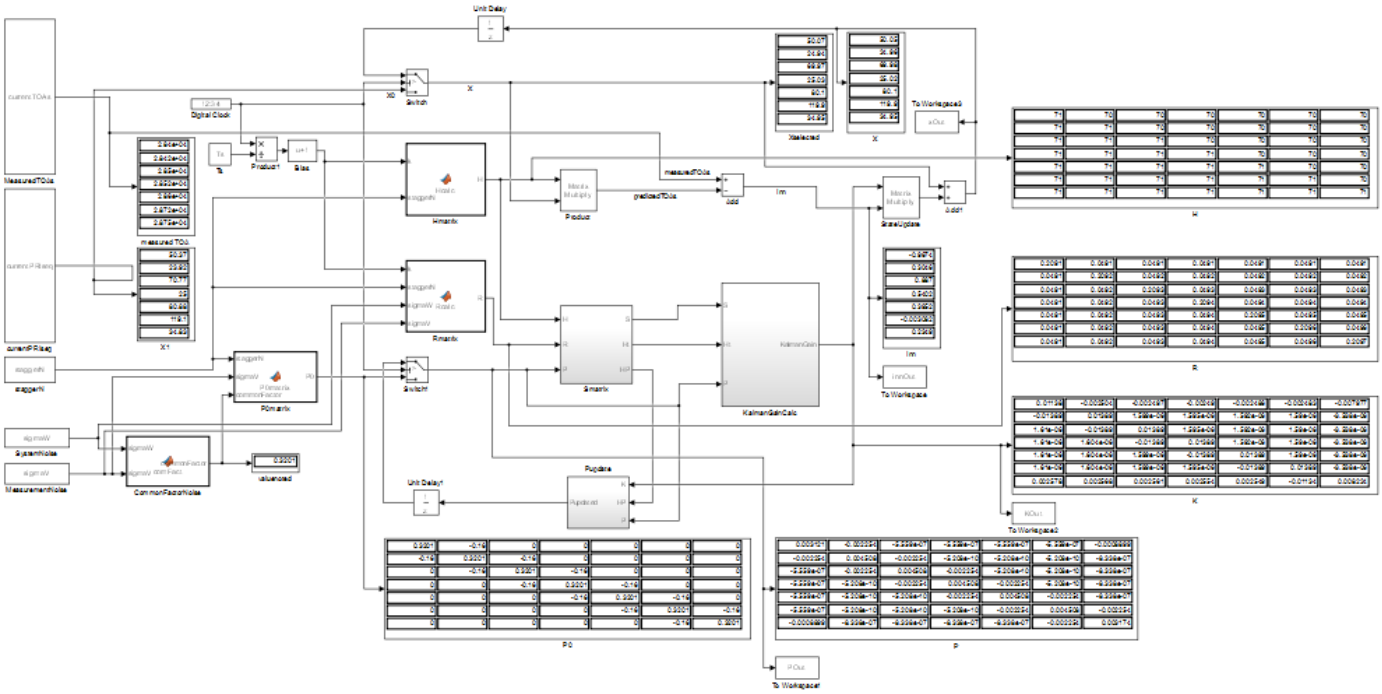
Script simulación predicción Kalman

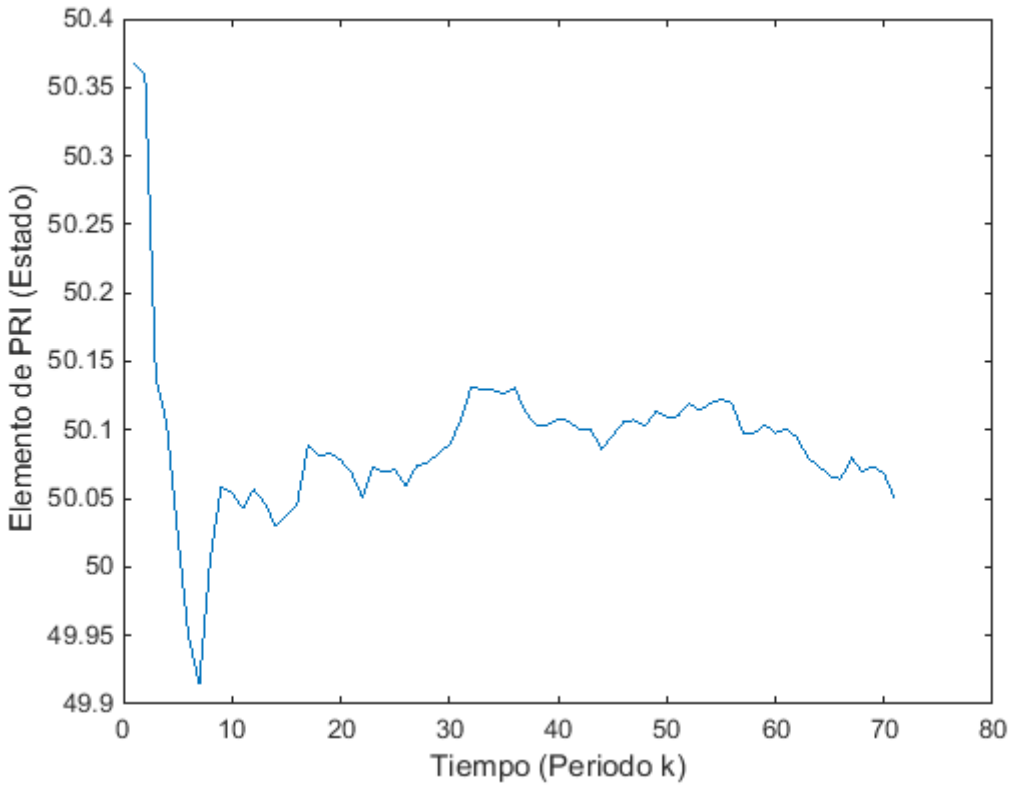
```
clear; clc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

CONFIGURATION PARAMETERS

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
open('KalmanFilteringModel');
Ts = 1e-3;
staggerSequence=[50 25 70 25 80 120 35];

pulsesSim = 500;

sigmaW=0.01;
sigmaV=0.4;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
staggerN=length(staggerSequence);

TOAarray=[];

incomingTOA=[];
incomingTOA=staggerSequence(1)+sigmaW*randn;
k=2;
for i=2:pulsesSim
    if k==(staggerN+1)
        k=1;
        incomingTOA(i)=incomingTOA(i-1)+staggerSequence(1)+sigmaW*randn;
    else
        incomingTOA(i)=incomingTOA(i-1)+staggerSequence(k)+sigmaW*randn;
    end
    k=k+1;
end

for i=1:pulsesSim
  TOAarray(i)=incomingTOA(i)+sigmaV*randn;
end

sizTOAs = size(TOAarray);
sizTOAs = sizTOAs(2);
cycles = 0;
```

```
for i=1:sizTOAs/staggerN
        currentTOAs(i,1)   = (i-1)*Ts;
        currentTOAs(i,2:staggerN+1) = TOAarray(staggerN*(i-1)+1:staggerN*i);
        cycles=i;
end
currentPRIseq(1,1)=currentTOAs(1,2);
 for i=2:staggerN
   currentPRIseq(i,1)=currentTOAs(1,1+i)-currentTOAs(1,i);
 end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



sim('KalmanFilteringModel', Ts*(cycles-1));
for i=1:staggerN
    hold on
    figure (i)
    plot(xOut(:,i))
    ylabel('Elemento de PRI (Estado)');
    xlabel('Tiempo (Periodo k)');
end
```




Elemento de PRI (Estado)
Tiempo (Periodo k)
50.4
50.35
50.3
50.25
50.2
50.15
50.1
50.05
50
49.95
49.9
0
10
20
30
40
50
60
70
80