Universitat Oberta de Catalunya

uoc.edu

Diseño y simulación firmware de una transmisión de squitters en modo S encriptados mediante RSA.

Jorge Mayo Vilches.
Master en ingeniería de telecomunicación.
Área de electrónica.

Nombre Consultor: Juan Antonio Ortega Redondo.
Nombre Profesor/a responsable de la asignatura: Carlos Monzo Sánchez.

Junio 2021.

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Diseño y simulación firmware de una transmisión de squitters en modo S encriptados mediante RSA*
Nombre del autor:	*Jorge Mayo Vilches*
Nombre del consultor/a:	*Juan Antonio Ortega Redondo.*
Nombre del PRA:	*Carlos Monzo Sánchez*
Fecha de entrega:	06/2021
Titulación:	*Master ingeniería de Telecomunicación.*
Área del Trabajo Final:	*Electrónica*
Idioma del trabajo:	*Castellano.*
Palabras clave:	*FPGA, Firmware, IFF.*

Resumen del Trabajo: *Con la finalidad, contexto de aplicación, metodología, resultados y conclusiones del trabajo.*

Este proyecto presenta el diseño y desarrollo de un sistema de comunicación basado en protocolo IFF (Identification Friend or Foe). IFF es un sistema de identificación criptográfica utilizado tanto en el campo civil como militar. En el campo civil se utiliza para controlar el tráfico aéreo, mientras que en el militar su función es la de distinguir los vehículos enemigos de los que no lo son.

Este proyecto diseña y desarrolla en concreto la transmisión y recepción de squitters encriptados, la encriptación se lleva a cabo mediante un sistema criptográfico de clave asimétrica basado en el algoritmo RSA.

Mediante el uso de una FPGA desarrollado en lenguaje de descripción HW VHDL se simula la comunicación entre dos equipos basados en protocolo IFF, en particular la transmisión de squitters en modo S, formado por un preámbulo y un conjunto de datos encriptados.

La parte del interfaz con el usuario se realiza mediante dos microcontroladores conectados a la FPGA por dos puertos series. Un microcontrolador simula la aeronave, ofreciendo un control de todos los parámetros del vehículo al usuario mediante una pantalla táctil, el otro microcontrolador simula la estación terrestre, el cual está conectado vía ethernet, para ofrecer una visualización de todas las aeronaves del espacio aéreo en una página web dinámica.

Como líneas futuras del proyecto, se realiza el estudio y diseño en una PCB de las etapas de transmisión y recepción del equipo, con el objetivo de conseguir una transmisión vía RF.

Abstract (in English, 250 words or less):

This project aims to present the design and development that takes place in a communication system based on the IFF (Identification Friend or Foe) protocol. IFF is a cryptographic identification system used in both civil and military fields. In the civilian field it is used to control air traffic, while in the military field its function is to distinguish enemy vehicles from non-enemy vehicles.

This project specifically designs and develops the transmission and reception of encrypted squitters, the encryption is carried out using an asymmetric key cryptographic system based on the RSA algorithm.

Using a FPGA and developing in Very High-Speed Integrated Circuit Hardware Description Language (VHDL) description language, the communication between two devices based on IFF protocol is simulated, in particular the transmission of S-mode squitters, consisting of a preamble and a set of encrypted data.

The user interface part is composed by two microcontrollers connected to a FPGA mentioned before via two serial ports. One microcontroller simulates the aircraft, simulating a user control via a touch screen, the other microcontroller simulates the ground station, which is connected via ethernet, to provide a visualization of all airspace aircraft on a dynamic web page.

Aiming for the future, the analysis and design of the equipment's transmission and reception stages will take place in a PCB , with the objective of achieving a transmission via RF.

CITA CELEBRE.

"La ingeniería es lo más cercano
a la magia que existe en el mundo"

Elon Musk.

AGRADECIMIENTOS.

Dedicado a mi familia. Por apoyarme y ayudarme en todos mis proyectos, fueran cuales fueran, por creer en mí, pero sobre todo por cuidarme y enseñarme cada día a ser mejor persona. Por un amor y apoyo incondicional durante todo este tiempo, recordándome que era capaz de conseguir cada una de las cosas que me propusiera en la vida.

Este logro es tanto mío como vuestro.

Glosario. Acrónimos.

1090 ES: 1090 MHz Extended Squitter

ADS: Automatic Dependent Surveillance

ADS-B: Automatic Dependent Surveillance-Broadcast

EUROCAE: European Organization for Civil Aviation Equipment

FAA: Federal Aviation Administration

FL290: 29.000 pies de altitud.

FL285: 28.500 pies de altitud

FW: Firmware

IFF: Identification Friend or foe

IFR: Instrumental flight rules

KTAS: Knots true airspeed (Velocidad real)

RF: Radiofrecuencia

PCB: Printed Circuit Board

ISM: Industrial, Scientific & Medical

SIF: Selective Identification Feature

TXP: Transponder.

FPGA: field programmable gate array

VHSIC: (Very High Speed Integrated Circuit)

HDL: Hardware Description Language.

VHDL: VHSIC (Very High Speed Integrated Circuit) + HDL (Hardware Description Language).

HW: Hardware

SW: Software

RSA: Rivest, Shamir y Adleman

TX: Equipo transmisor

RX: Equipo receptor

FSM: Finite state machine (Maquina de estados finita)

Índice

Lista de figuras

1. Introducción.

Este proyecto simula la transmisión de un squitter en modulación ASK y en la banda ISM europea enviado desde una aeronave a otra aeronave o equipo terrestre.

> *Un squitter es una transmisión espontanea generada de forma periódica en modo broadcast, que proporciona los datos de posición, altitud, velocidad y datos adiciones sin la necesidad de ser interrogado, reduciendo así el tráfico de información.*

Para este proyecto el squitter transmitido constara de un preámbulo propio del equipo IFF y de unos datos que estarán encriptados mediante un sistema criptográfico de clave pública basado en el algoritmo RSA.

1.1 Contexto y justificación del Trabajo.

El enorme crecimiento del tráfico aéreo y el hecho de que haya extensas zonas que no están guiadas por ATC (Air Traffic Control), hacen necesario el conocimiento del tráfico aéreo en tiempo real, no sólo para los controladores de tráfico aéreo, sino también para los propios pilotos. Los radares actuales son muy lentos, los controladores aéreos reciben información de un avión en particular cada seis o siete segundos, tasa de actualización adecuada, pero insuficiente en la actualidad.

La tecnología ADS-B (Automatic Dependent Surveillance – Broadcast) está basada en la transmisión espontanea en modo broadcast de squitter, esta tecnología busca ofrecer una solución para mantener un control del tráfico aéreo con una tasa de actualización más elevada que la actual.

La transmisión automática de squitters proporciona una mejora ya que envía información de manera periódica en intervalos de tiempo mucho más pequeños, intervalos del orden menor de un segundo y con una tasa de tráfico de 15 milisegundos, el dispositivo manda automáticamente información de posición (longitud y latitud,), altitud, velocidad, estado de emergencia, identificación, altura.

A continuación, se muestra una imagen obtenida de internet con el fin de ilustrar el tráfico aéreo mundial en el año 2019 (evitando el impacto de la COVID-19), asociado a la necesidad de tener controlado todo este tráfico y el mercado que puede generar nuestro equipo para la empresa que lo comercializa.

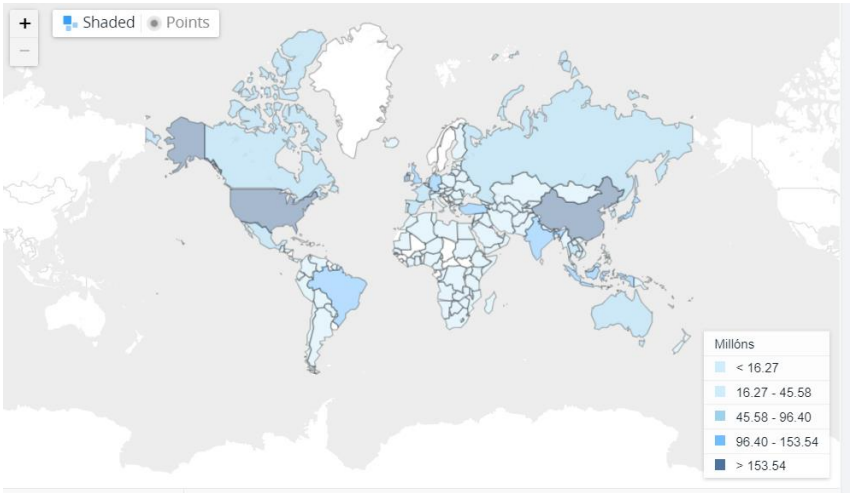


Ilustración 1. Mapa mundial de tráfico aéreo

Un detalle importante es el control de la frecuencia del squitter el cual está en un valor comprendido entre 0,8 y 1,2 segundos. Utilizando una cuantificación de tiempo no superior a 15 milisegundos. Esta frecuencia de trabajo nos ofrece una información muy periódica de las condiciones en las que se encuentra el equipo.

El proyecto transmitirá y recibirá squitters de modo S. En modo S las respuestas tienen el mismo formato que los squitters, en la siguiente imagen obtenida de la propia documentacion de INDRA se puede apreciar una captura de un squitter.

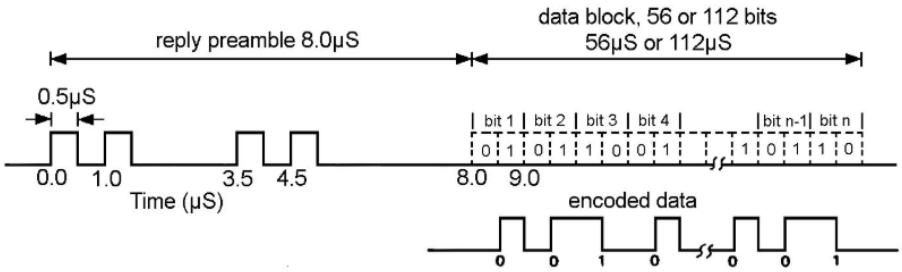


Ilustración 2. Formato de transmisión Squitter Modo S.

Se observa como los squitters de modo S constan de un preámbulo de 4 pulsos seguidos de 56 o 112 bits (en función del formato) modulados en posición (PPM). El formato depende de si se trata de un short squitter (DF11) o extended squitter (DF17).

1.2 Objetivos del Trabajo.

El objetivo del proyecto es diseñar y desarrollar un sistema de comunicación IFF con información encriptada controlado mediante una FPGA.

Con el fin de abaratar costes, se simula mediante un única FPGA la transmisión y la recepción por lo que la función de este dispositivo podría estar dividido en dos partes claramente diferenciadas.

- Diseño y desarrollo de la transmisión. La FPGA genera los pulsos digitales que forman el squitter que se quiere transmitir, el tren de pulsos estará formado por un preámbulo y un conjunto de datos encriptados mediante un sistema criptográfico de clave pública basado en el algoritmo RSA.
- Diseño y desarrollo de la recepción. La FPGA recibe los pulsos digitales, la primera parte confirma el preámbulo, la segunda recibe y desencriptan los datos recibidos.

Con la finalidad de poder comprobar los resultados, se implementa un interfaz externo a la propia FPGA, este interfaz se lleva a cabo mediante los dispositivos del propio kit de desarrollo y algunos añadidos en una protoboard. Este interfaz tiene la función de por un lado en la etapa de transmisión permitir al usuario elegir el tipo de mensaje y los parámetros transmitidos, mientras que por otro lado en la etapa de recepción su correcta visualización.

Desarrollado correctamente la transmisión en banda base. Se realiza un estudio y diseño de una PCB de las etapas de transmisión y recepción mediante una portadora centrada en la banda ISM de 868 MHz modulada con el tren de pulsos digitales obtenido en la FPGA con el objetivo de conseguir una transmisión ASK.

1.3 Enfoque y método seguido.

- ***En primer lugar, se estudia el enfoque genérico del proyecto.***

El proyecto se implementa mediante el kit de desarrollo Intel DK-DEV-10M50A MAX 10. Este kit de desarrollo destaca la FPGA de Intel de la familia MAX10, este dispositivo será configurado mediante el entorno de desarrollo Quartus (Quartus prime 18.1).

El sistema criptográfico utilizado está basado en la factorización de números enteros, el funcionamiento se basa en el producto, conocido, de dos números primos grandes elegidos al azar y mantenidos en secreto. Mediante teoremas matemáticos se calculan las claves públicas y privadas del equipo receptor.

La FPGA es la encargada por un lado de realizar la encriptación y transmisión de los mensajes recibidos desde el microcontrolador y por otro lado de la recepción y desencriptación de los mensajes recibidos desde el equipo transmisor.

Mediante dos microcontroladores conectados externamente a la FPGA se podrá simular la transmisión y recepción de squitters, encriptados mediante las claves calculadas previamente en función de los números primos elegidos. En el microcontrolador que simula la aeronave transmisora mediante un panel táctil permite al usuario elegir los tipos de mensajes que se quieren transmitir, por otro lado, con un joystick permite elegir el rumbo de la misma. En el microcontrolador que simula el equipo receptor, mediante una conexión ethernet, permite en una página web realizar una visualización del espacio aéreo de todas las aeronaves que se encuentran en el radio de alcance.

Concluido el desarrollo firmware, se avanza con el desarrollo fhardware de la circuitería externa en PCB para la transmisión RF, al igual que en la parte de la configuración de la FPGA, esta parte también está formada por dos partes claramente diferenciadas.

En la etapa de transmisión mediante una portadora centrada en la banda ISM de 868 MHz se modula una transmisión ASK. La señal modulada, será filtrada y amplificada para transmitirse vía RF hasta el receptor.

En la etapa de recepción se recibirá la señal RF, esta señal será filtrada (con un filtro paso banda centrado en 868 MHz) y a continuación, mezclada con el objetivo de reducir la frecuencia de la portadora de la señal. Mediante un extractor de video, obtendremos la señal digitalizada con el fin de poder extraer los datos transmitidos y que puedan ser procesados en la FPGA receptora.

- ***En segundo lugar, se resume el método seguido.***

Para la realización de este proyecto se ha decidido seguir una estrategia de creación de un nuevo producto partiendo de los diferentes kits y circuitos integrados que existen en el mercado. El prototipo resultante incorporara las funcionalidades establecidos en los objetivos del proyecto (apartado anterior).

Para el resultado que se pretende conseguir. Se considera como método de desarrollo óptimo de gestión de proyectos la metodología basada en objetivos. Cada escenario de trabajo requiere de una explicación detallada de cada tarea. El análisis de cada tarea se llevará a cabo mediante un planteamiento previo, un análisis teórico, un desarrollo conceptual y una simulación hardware o por computadora.

1.4 Planificación del Trabajo.

En una metodología orientada a objetivos, la planificación consiste en descomponer el trabajo en partes más pequeñas y manejables, que permita lograr cada uno de los objetivos en tiempo y coste esperados, trabajando en un escenario en el que se mantendrán estudiados previamente los posibles riesgos eventuales.

En conclusión, se recoge el siguiente plan de gestión global del proyecto, este plan de gestión se encuentra dividido en tres grandes planes claramente diferenciados.

- **Plan de hitos.** Este plan identifica el conjunto de acciones formadas por un conjunto de tareas. Los hitos del proyecto se resumen en la siguiente tabla.

Hito.	***Fecha finalización.***	***Objetivo.***
PEC1. Definición.	*28/02/2021*	*Definir el objetivo y alcance del TFM que se desarrollara en el semestre.*
PEC2. Estado del arte.	*14/03/2021*	*Redacción del estado del arte del proyecto. Ubicación del proyecto.*
PEC3. Diseño e implementación.	*07/05/2021*	*Diseño: Realizar un estudio y diseño del futuro desarrollo, tener en cuenta el alcance marcado en la primera parte (PEC1). Implementación: Desarrollar el proyecto con las especificaciones de diseño definidas y marcadas previamente por el alumno y tutor.*
PEC4. Memoria	*31/05/2021*	*Redacción de la memoria técnica y descriptiva del proyecto.*
PEC5. Presentación	*10/06/2021*	*Preparación de la defensa del proyecto realizado mediante una presentación.*

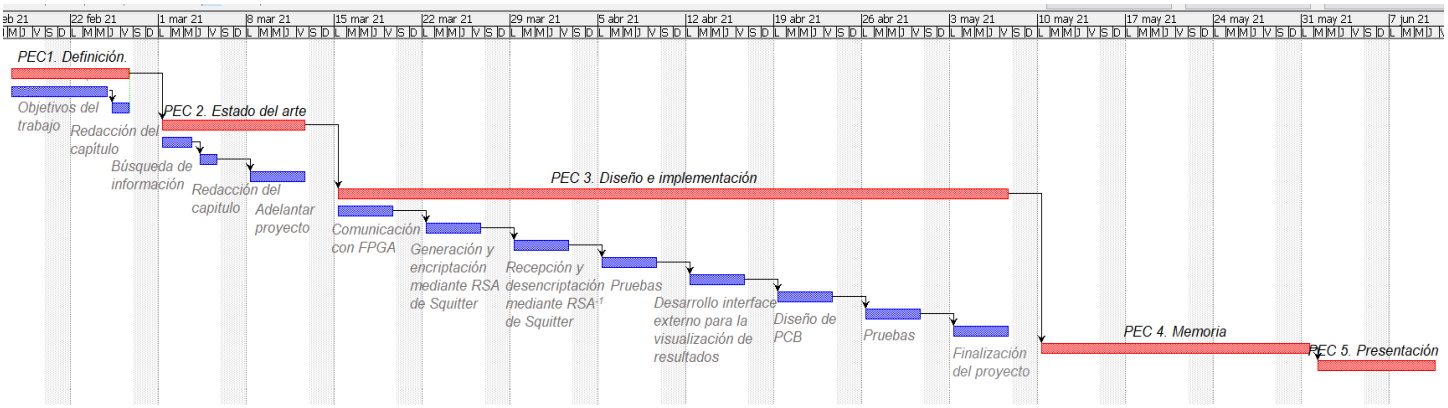
Tabla 1. Tabla de hitos del proyecto.

- **Plan de tiempos.** Este plan permite realizar una estimación de la duración asociada a cada tarea. Mediante un cronograma basado en un diagrama de Gantt se planifica cada tarea del conjunto de hitos descrito previamente. Este tipo de diagramas destaca por ofrecer en el eje de ordenadas el conjunto de tareas de cada proyecto mientras que en el de abscisas el tiempo de inicio y duración de cada una de ellas. Se implementa una breve planificación orientativa mostrada en la siguiente tabla.

Nombre.	***Duración.***	***Fecha inicio.***	***Fecha finalización.***
PEC1. Definición.	*8 días*	*17/02/2021 8:00*	*26/02/2021 17:00*
PEC 1.1. Objetivos del trabajo.	*6 días*	*17/02/2021 8:00*	*24/02/2021 17:00*
PEC 1.2. Redacción del capítulo.	*2 días*	*25/02/2021 8:00*	*26/02/2021 17:00*
PEC 2. Estado del arte	*10 días*	*01/03/2021 8:00*	*12/03/2021 17:00*
PEC 2.1. Búsqueda de información	*3 días*	*01/03/2021 8:00*	*03/03/2021 17:00*
PEC 2.2. Redacción del capitulo	*2 días*	*04/03/2021 8:00*	*05/03/2021 17:00*
PEC 2.3. Adelantar proyecto	*5 días*	*08/03/2021 8:00*	*12/03/2021 17:00*
PEC 3. Diseño e implementación	*40 días*	*15/03/2021 8:00*	*07/05/2021 17:00*
PEC 3.1. Comunicación con FPGA	*5 días*	*15/03/2021 8:00*	*19/03/2021 17:00*
PEC 3.2 Generación y encriptación mediante RSA de Squitter	*5 días*	*22/03/2021 8:00*	*26/04/2021 17:00*
PEC 3.3. Recepción y desencriptación mediante RSA^{-1} de Squitter	*5 días*	*29/03/2021 8:00*	*02/04/2021 17:00*
PEC 3.4. Pruebas de PEC 3.1 y 3.2	*5 días*	*05/04/2021 8:00*	*09/04/2021 17:00*
PEC 3.5. Desarrollo interface externo para la visualización de resultados.	*5 días*	*12/04/2021 8:00*	*16/04/2021 17:00*
PEC 3.6. Diseño de PCB	*5 días*	*19/04/2021 8:00*	*23/04/2021 17:00*
PEC 3.7. Pruebas de PEC 3.6	*5 días*	*26/04/2021 8:00*	*30/04/2021 17:00*
PEC 3.8. Finalización del proyecto	*5 días*	*03/05/2021 8:00*	*07/05/2021 17:00*
PEC 4. Memoria	*16 días*	*10/05/2021 8:00*	*31/05/2021 17:00*
PEC 5. Presentación	*8 días*	*01/06/2021 8:00*	*10/06/2021 17:00*

Tabla 2. Tabla de planificación temporal.

NOTA. El diagrama de Gantt completo se encuentra en el anexo II.



- **Plan de riesgos.** Este plan permite identificar, evaluar y así poder gestionar los posibles riesgos correspondientes al proyecto. Los riesgos inherentes del proyecto se resumen en la siguiente tabla.

Riesgo identificado.	***Probabilidad y Evaluación.***		***Gestión asociada.***
	Probabilidad.	***Evaluación.***	
Definición insuficiente del plan de trabajo.	*Baja*	*Media*	*Revisión y aclaración de requisitos y tiempos de entrega con el tutor.*
Incidencias en compras de materiales.	*Media*	*Media*	*. Buscar otro distribuidor de componentes electrónicos*
Problemas en la comunicación con la FPGA.	*Alta*	*Alta*	*Búsqueda de información adicional del fabricante sobre el kit de desarrollo.*
Falta de instrumentación Hardware.	*Alta*	*Muy alta*	*Dividir el sistema en etapas más pequeñas. Utilizar circuitos externos para poder filtrar los posibles fallos.*
Problemas de diseño y montaje de la PCB.	*Media*	*Alta*	*. Buscar otro distribuidor de fabricación.*
Exceso de la definición del plan inicial en la complejidad técnica.	*Media*	*Media*	*Replanteamiento de los objetivos parciales y del producto final ofrecido en el proyecto.*

Tabla 3. Tabla de riesgos del proyecto.

1.5 Breve sumario de productos obtenidos.

El producto resultante de este trabajo (TFM) es un prototipo de comunicación IFF de transmisión de un squitter con datos encriptados en modulación ASK en la banda ISM europea. Los productos desarrollados en este trabajo son los siguientes:

- ***Parte técnica y de desarrollo del TFM.***
  - Desarrollo de código VHDL para la transmisión y recepción del squitter, incluida la encriptación y desencriptación del mensaje.
  - Sistema de interface externo para la visualización del Squitter transmitido, tipo de mensaje y mensaje enviado.
  - Estudio y diseño del desarrollo de una PCB de la etapa de transmisión y recepción para la comunicación RF en modulación ASK en banda ISM 868 MHz.

- ***Redacción, memoria del TFM***
  - Memoria técnica y descriptiva del proyecto implementado.

- ***Presentación del TFM***
  - Exposición de los puntos más importantes. Defensa del proyecto realizado ante un tribunal.

1.6 Breve descripción de los otros capítulos de la memoria.

A continuación, se detalla y resume todos los capítulos que conforman la memoria del proyecto.

- ***Capítulo 1. Introducción.***
  Definir el objetivo y alcance del TFM que se desarrollara durante el semestre por el alumno. Puesta en marcha y visión global.

- ***Capítulo 2. Estado del arte.***
  Estudio del punto de partida. Breve definición de los temas más importantes del proyecto. Presentación de los dispositivos utilizados y elegidos.

- ***Capítulo 3. Diseño del proyecto. Análisis genera****l.*
  Análisis de alto nivel del proyecto con el objetivo de obtener una visión genérica, destaca el diagrama de bloques de las etapas y procesos más críticos del trabajo.

- ***Capítulo 4. Diseño y desarrollo de la etapa de transmisión y recepción encriptada del Squitter.***
  Análisis especifico del diseño y desarrollo VHDL de la generación y recepción del Squitter junto con su respectiva encriptación y desencriptación.

- ***Capítulo 5. Diseño y desarrollo de la interface externa de visualización.***
  Análisis especifico del diseño y desarrollo de la interface externa para la correcta visualización de las tramas generadas.

- **Capítulo 6. *Estudio y diseño de la etapa RF implementada en PCB*.**
  Análisis especifico del estudio y diseño firmware de la etapa de radiofrecuencia en una PCB.

- ***Capítulo 7. Valoración económica del trabajo.***
  Gastos asociados al desarrollo y mantenimiento del trabajo.

- ***Capítulo 8. Conclusiones.***
  Enumeración y descripción de las conclusiones del trabajo. Destacar una reflexión y análisis crítico de los logros en términos de los objetivos planteados inicialmente. Complementar con líneas de trabajo futuro que no se han podido explorar.

- ***Capítulo 9. Glosario.***
  Definición de los términos y acrónimos más relevantes utilizados dentro de la Memoria.

- ***Capítulo 10. Bibliografía*.**
  Lista numerada de las referencias bibliográficas utilizadas dentro de la memoria.

- ***Capítulo 11. Anexos*.**
  Listado de apartados que son demasiado extensos para incluir dentro de la memoria y tienen un carácter autocontenido.

2. Estado del arte.

El estado del arte pretende definir la base teórica sobre la que se sostiene el proyecto que se realiza. Este capítulo repasa las técnicas y productos utilizados y desarrollados, permitiendo así poder explicar los aportes a los campos desarrollados. Es decir, en otras palabras, su contenido consiste en hacer un repaso de los diferentes proyectos y técnicas relacionadas con el ámbito del TFM.

2.1 Introducción. Tecnología ADS-B

El sistema de Vigilancia Dependiente Automática - Difusión *(ADS-B por las siglas en inglés de Automatic Dependent Surveillance Broadcast)* se trata de una tecnología de vigilancia que permite a una aeronave emitir de manera periódica su posición a través de la navegación por satélite (en concreto mediante GNSS), permitiendo realizar un seguimiento de la misma.
La tecnología ADS-B opera mediante el estándar *"1090ES"*. Permite difundir información automáticamente, sin tener que esperar una interrogación selectiva de otra aeronave. *"1090ES"* significa la transmisión de extended Squitter en 1090 MHz.

Como se ha comentado en la introducción del proyecto, dentro de los sistemas IFF se encuentra la tecnología ADS-B, esta tecnología de carácter civil y militar está basada en la transmisión periódica y a difusión de Squitters.

"Un squitter es una transmisión espontanea generada de forma periódica en modo broadcast, que proporciona los datos de posición, altitud, velocidad y datos adiciones sin la necesidad de ser interrogado, reduciendo así el tráfico de información."

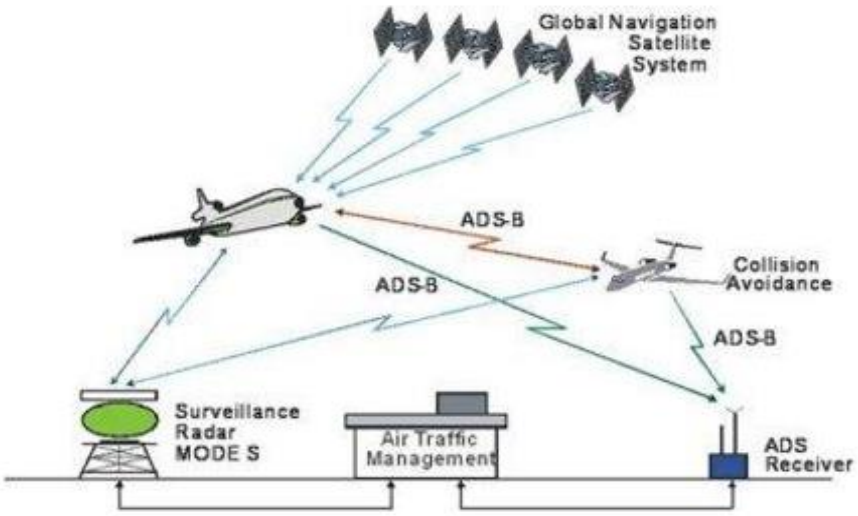


Ilustración 3. Ejemplo de tráfico aéreo con ADS-B

Las Agencias de aeronavegabilidad mundiales han emitido de manera global reglas y requerimientos correspondientes al equipo ADS-B, resumidos a continuación:

- **Europa:** La Unión Europea requerirá salida 1090ES ADS-B, para las aeronaves IFR con un peso máximo de despegue superior a 12,566 libras (5.700 kg) o una velocidad aérea de crucero máxima superior a 250 KTAS (463 km/h).

El uso de ADS-B es obligatorio en otros muchos países del mundo, a continuación, se resumen alguno de ellos:

- **Estados unidos:** En Estados Unidos el ADS-B es un componente integral de la nueva generación en estrategias del espacio aéreo nacional (NextGen) para mejorar o actualizar la infraestructura de la aviación y sus operaciones. Se requiere tecnología ADS-B cuando se opere en los 48 estados continuos, en todos los equipos que se encuentren por encima de los 10.000 pies y dependiendo de la ubicación si se encuentran por debajo de los 10.000 pies.
- **Australia:** En Australia es obligatorio el uso de ADS-B para todos los vuelos IFR, por encima y por debajo de FL290, en todo el continente.
- **México:** En México se requerirá ADS-B, en el espacio aéreo de Clase A, B y C, y en Clase E a más de 10,000 pies.
- **Hong Kong, Indonesia, Singapur, Sri Lanka, Vietnam, Taiwán, China:** En estos países, se requiere tecnología ADS-B para todos los vuelos por encima de FL 290 (29.000 pies).
- **Colombia:** se requiere de esta tecnología en todos los vuelos.
- **India:** es obligatorio que aeronaves estén equipadas con ADS-B para poder operar a FL285 (28.500 pies) o por encima, en el espacio aéreo continental.

En la siguiente imagen obtenida de internet se muestra con el símbolo de un avión en negro, los países en los que es obligatorio el uso de la tecnología ADS-B.

Ilustración 4. Países que implementan tecnología ADS-B

El proyecto implementa tecnología ADS-B, como se ha comentado previamente, esta tecnología está basada en la transmisión espontanea en modo broadcast de squitter, ofreciendo información de posición (longitud y latitud,), altitud, velocidad, estado de emergencia, identificación, altura y ofrecer así una solución para mantener un control automático del tráfico aéreo con una tasa de actualización elevada.

En la siguiente parte del capítulo se realiza un análisis de los aspectos más relevantes del diseño y desarrollo del proyecto permitiendo ofrecer una visión amplificada y más clara como punto de partida de los productos y tecnologías desarrolladas y utilizadas.

2.2 ADS-B como parte de la tecnologia IFF.

Los sistemas IFF comenzaron a desarrollarse en Alemania durante la segunda guerra mundial. El identificador amigo-enemigo, o IFF (Identification Friend or Foe), es un sistema de identificación criptográfica. Dentro del campo militar, sirve para distinguir a aeronaves o a vehículos enemigos de los que no lo son (específicamente sirve para identificar amigos, los no identificados se presumen enemigos), por el contrario, en el campo civil sirve para realizar un control del tráfico aéreo.

Actualmente se trabajan con los modos SIF y S para la parte civil y con los modos 4 y 5 para la parte militar.

La tecnología IFF se puede dividir en dos grandes modos de trabajo.

- **Interrogación/Respuesta**: Un equipo genera una interrogación de un modo de trabajo, y otro equipo le responde a esa interrogación generada. Si el equipo responde, se entiende que es “amigo” si por el contrario no obtenemos respuesta, deducimos que el equipo no es “amigo” y por lo tanto entendemos que es “enemigo”. Los sistemas IFF realizan el proceso de identificación en tres partes.
    1. Interrogación. Consiste en transmitir interrogaciones en uno o varios de los modos de interrogación.
    2. Respuesta. Las plataformas que detectan las interrogaciones, envían una respuesta que consiste en una serie de pulsos codificados, conforme al modo de interrogación utilizado.
    3. Reconocimiento. Las respuestas son recibidas y decodificadas por el equipo Interrogador, para obtener los códigos pudiendo así identificar la aeronave

- **Squitters**: Un equipo genera de forma automática una respuesta espontánea con carácter informativo genérico la cual le llega a todos los posibles receptores que se encuentren en el radio de alcance.

La siguiente imagen extraida de internet, muestra el ejemplo de un escenario miitar en el que un conjunto de vehiculos y aeronaves interrogan a una de ellas en concreto para discriminar si esta es amiga o enemiga.

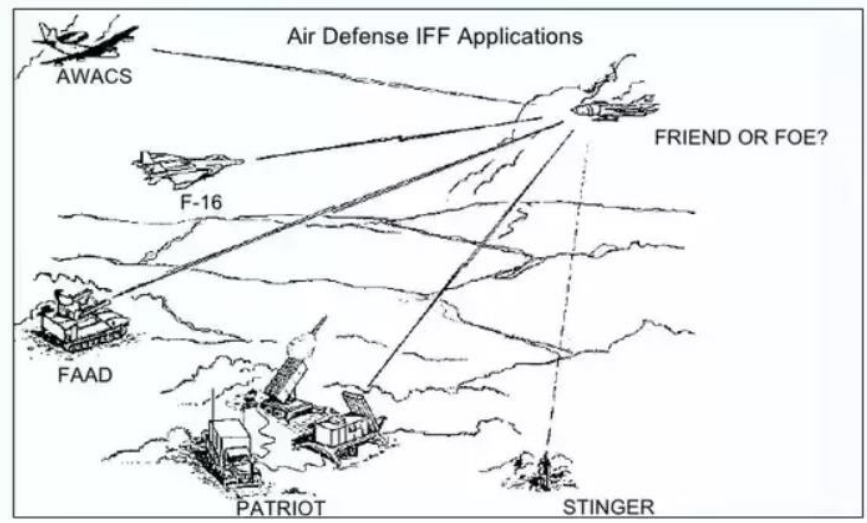


Ilustración 5. Ejemplo de sistema IFF en espacio militar.

El proyecto desarrolla y simula la transmisión y recepción de squitters encriptados por un número indefinido de aeronaves ubicadas en el espacio aéreo.

Dentro de la tecnología IFF existen diferentes modos de funcionamiento. Estos modos se podrían dividir en dos tipos claramente diferenciados, en función de su fecha de uso.

- **Sistemas IFF antiguos.**

Modos SIF: Sistemas de uso conjunto civil y militar constituido por un total de cuatro modos diferentes de operación:

  - **Modo 1:** Es un modo militar con un código de dos cifras, formado por un total de 32 códigos para identificación general. Se utiliza en control de tráfico aéreo militar, para determinar el tipo de aeronave y el tipo de misión
  - **Modo 2:** Es un modo de carácter militar con código de cuatro cifras, formado por un total de 4096 códigos para identificación individual. Permite identificar cada aeronave en particular.
  - **Modo 3/A:** Es un modo civil/militar de control de tráfico aéreo estándar. Se usa internacionalmente, formado por un total de 4096 códigos para conseguir identificación civil/militar individual. Este código es asignado por el aeropuerto de partida de la aeronave. Se reservan varios códigos para indicar situaciones concretas.
  - **Modo C:** Es un modo civil, se usa para transmitir la altura barométrica de la aeronave.

Modo 4: Sistema criptográfico, con uso exclusivo de carácter militar (OTAN). Genera únicamente identificación de amigo/enemigo (modo militar cifrado). La trama transmitida consta de una primera parte de pulsos de sincronización, seguido de un código cifrado.

- **Sistemas IFF nuevos y futuros.**

Modo S: Interrogación civil selectiva. Modo civil avanzado que permite establecer enlaces de datos selectivos con cada aeronave mediante una dirección y un número de vuelo formado por un total de 16.777.216 códigos de dirección.

Modo 5: Interrogación militar selectiva. Modo militar avanzado para las fuerzas militares de OTAN. Este modo utiliza técnicas criptográficas, codificación y modulación avanzada que superan el rendimiento y las limitaciones en aspectos de seguridad que presenta el modo 4. Su modo de funcionamiento es confidencial.

Entre los diferentes tipos de sistemas de localización, encontramos los siguientes tipos de radares junto con la tecnología ADS-B.

- **Radares primarios (PSR).** Utilizan las reflexiones de las ondas electromagnéticas, para determinar la distancia de las aeronaves con respecto a la estación radar en función del tiempo que transcurre desde que se emite la señal, hasta que ésta retorna.
- **Radares secundarios (SSR).** Este sistema está compuesto por dos subsistemas. El interrogador cuya función es la generación de interrogaciones, las cuales consisten en señales codificadas en forma de pulsos modulados en amplitud y en fase. Y el transpondedor, que se encuentra ubicado dentro de la aeronave, que funciona de receptor y el transmisor, este sistema recibe las interrogaciones, las procesa y responde al interrogador en función del tipo de interrogación indicando datos como su altitud e identificador.
- **Tecnología ADS-B.** Consiste en una tecnología de vigilancia cooperativa en la que un avión determina su posición a través del sistema de navegación por satélite GPS y la difunde periódicamente, lo que le permite realizar un seguimiento automático. Este sistema podría reemplazar a ambos sistemas radar como el principal método de vigilancia para el control del tráfico aéreo en todo el mundo.

En un futuro cercano, se pretenden eliminar los modos SIF y 4, debido a que en los modos S y 5, las interrogaciones se hacen en múltiples formatos y de forma selectiva, evitando de esta manera recibir respuestas de todos los equipos transpondedores que se encuentren dentro del alcance de la antena, y que éstas puedan solaparse y/o perderse (fenómeno conocido como "garbling" en terminología radar) y, por tanto, degradando la calidad de la vigilancia aérea.

- **Adaptación de la comunicación ADS-B como tecnología IFF en el proyecto desarrollado como TFM.**

El proyecto desarrollado como TFM, utilizara la misma modulación que los sistemas IFF (modulación PPM) también se respetan los tiempos asociados a los niveles altos y bajos del preámbulo, además del régimen binario y la longitud de la trama enviada en la parte de la transmisión de los datos. La transmisión se centra en paquetes de 56 bits, transmitiendo un squitter en formato short squitter DF11, manteniendo su valor propio de régimen binario de un bit por microsegundo.

Como mejora del proyecto, aparece la parte referenciada a la encriptación, en concreto criptografía asimétrica mediante tecnología RSA. Se decide encriptar los mensajes con el objetivo de que la transmisión de las diferentes aeronaves del espacio aéreo con la torre de control, puede ser recibida únicamente por esta última. Podría darse el caso de que apareciera una tercera persona con el objetivo de querer sabotear la comunicación, en caso de que no estén los mensajes encriptados, podría extraer el identificador de una aeronave transmisora, copiarlo y transmitir nuevos mensajes con valores falsos, creando así un completo caos por parte de la torre de control, ocasionando múltiples accidentes aéreos. El tipo de criptografía elegida (criptografía asimétrica) se fundamenta en que la torre de control, podría publicar sus claves publicas sin ningún problema y cualquier aeronave podría enviar información cifrada que únicamente podría descifrar la torre de control con su clave privada. En este marco teórico, se puede apreciar una gran facilidad de cara a transmitir información desde la aeronave hasta la torre de control.

El squitter transmitido mantiene la particularidad que ofrece el modo S frente al resto de modos civiles SIF. Esta particularidad consiste en transmitir en el propio squitter el identificador del origen transmisor, este concepto se denominada transmisión selectiva. Para ello los primeros 18 bits de información encriptada transmitida representaran el identificador de la aeronave. La estructura del squitter formado por un total de 56 bits es la siguiente:

- *2 bits a nivel alto constantes al principio de la trama*
- *18 bits a continuación: identificador de la aeronave encriptado.*
- *18 bits a continuación: tipo de mensaje encriptado.*
- *18 bits a continuación: mensaje enviado encriptado.*

Mediante una interrupción periódica. Se transmite constantemente y sin necesidad de ningún usuario, cuatro squitters seguidos con un contenido de los valores de longitud, latitud, velocidad y altitud de la aeronave. A parte de estos mensajes que se transmiten de forma periódica, el usuario podrá enviar cuando lo desee mensajes denominados de carácter especial, estos mensajes son los siguientes:

- Mensaje SOS. *Fallo de carácter general en la aeronave*.
- Mensaje del nivel de Gasolina. *Envía el nivel de combustible actual, pudiendo indicar problemas asociados a los niveles de carburante.*
- Mensaje de piloto automático activado. *Transmite que el piloto automático de la aeronave se encuentra activado.*
- Mensaje de solicitud de aterrizaje. *Transmite una solicitud por parte de la aeronave de aterrizaje inmediato.*

Tal y como se ha comentado en repetidas ocasiones a lo largo del documento, el proyecto se centra en el diseño, desarrollo y simulación de transmisión de squitters en modo S encriptados en RSA mediante tecnología ADS-B propia de equipos IFF.

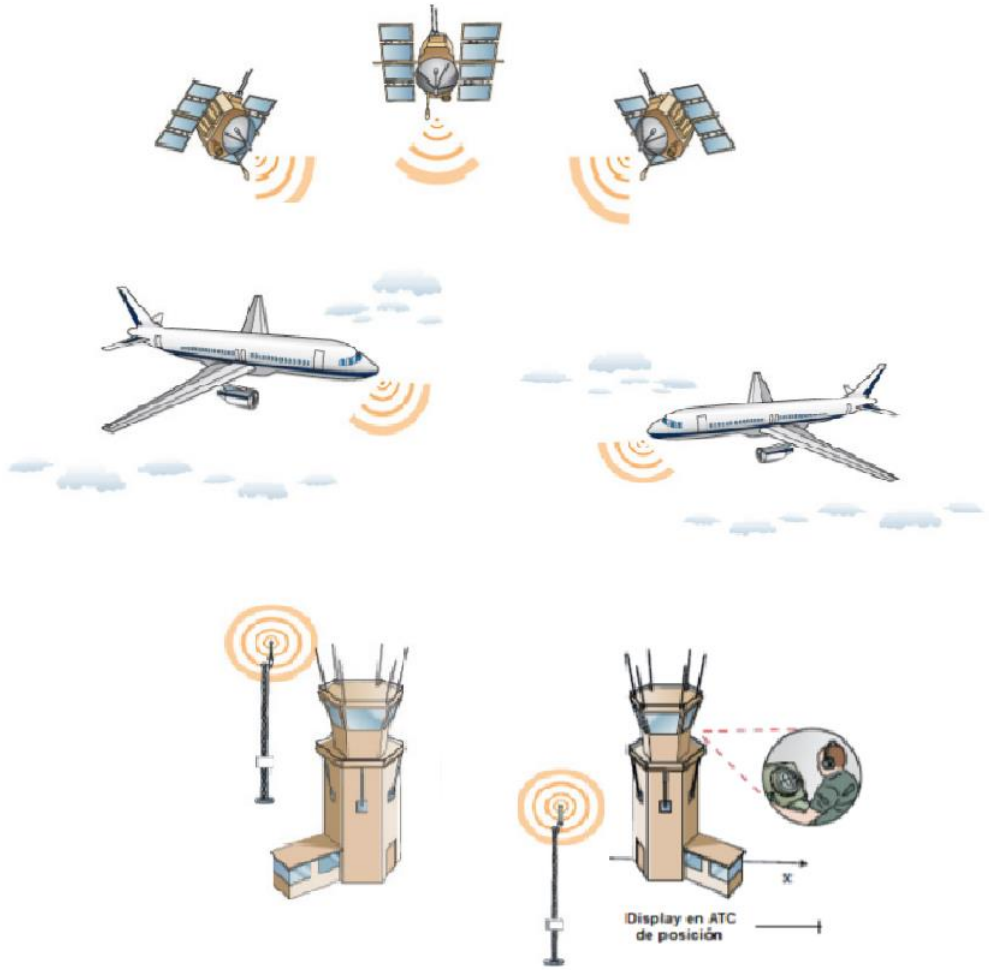


Ilustración 6. Ejemplo de tráfico aéreo con tecnología ADS-B

Concluida la introducción y explicación del marco de trabajo, entre lo que destaca la tecnología IFF, en concreto con el modo de trabajo implementado en tecnología ADS-B, el capítulo avanza con el estudio de los dispositivos, componentes y técnicas de mayor peso utilizados en el proyecto.

En la siguiente lista, aparecen los dispositivos, componentes y técnicas de mayor peso utilizados en el proyecto.

- *Dispositivo FPGA. Field programmable gate array.*
- *Dispositivo microcontrolador. µC.*
- *Sistema criptográfico. Algoritmo RSA.*
- *Espacio electromagnético. Sistemas RF.*
- *Placa base PCB. Printed Circuit Board.*

2.3 Dispositivo FPGA. Field programmable gate array.

Una FPGA (Field programmable gate array) es un dispositivo programable que contiene bloques de lógica cuya interconexión y funcionalidad puede ser diseñado y configurada con el objetivo de describir un circuito digital, mediante un lenguaje de descripción, especializado en un entorno de desarrollo.

Es de vital importancia comprender la diferencia entre una FPGA y un procesador. Los procesadores son dispositivos complejos con un conjunto fijo de instrucciones, cuando un usuario programa un microprocesador, desarrolla una serie de instrucciones para su ejecución de manera secuencial, por otro lado, cuando un usuario describe lógica digital en una FPGA, el circuito resultante contiene múltiples señales, que variarán al mismo tiempo, ejecutándose todas ellas de forma paralela.

Los procesadores están limitados en tiempo, un programa más complejo se traduce en más tareas, las cuales corresponden a mas instrucciones y estas a mas ciclos de reloj, es decir, más tiempo de ejecución. Sin embargo, una FPGA puede ejecutar múltiples tareas simultáneamente, eliminando el problema anterior. Las FPGAs son dispositivos diseñados y no programados en los cuales tras cargar el proyecto en el integrado desde el entorno de desarrollo se crea físicamente el circuito digital en el propio chip. Entre los fabricantes más importantes de FPGAs destacan Xilinx y Altera.

La siguiente disyuntiva que aparece, también asociada a la elección del dispositivo, es la elección entre implementar el diseño en una FPGA o en una CPLD.
Entre el análisis comparativo de los dos dispositivos, se puede destacar como punto de inflexión y por lo tanto de elección que, por norma general, las FPGAs contienen un gran número de bloques lógicos programables, los cuales se componen principalmente de tablas de consulta integrados en un océano de interconexiones programables. Por el contrario, las CPLDs ofrecen macrocelulas con arquitecturas internas preconfiguradas, ofreciendo menos libertad al usuario. Los bloques internos de las FPGAs ofrecen al usuario desarrollar funciones lógicas mucho más complejas, se podría llegar a decir que una FPGA puede contener incluso millones de bloques lógicos configurables en un solo dispositivo. En conclusión, parece obvio la elección de la FPGA frente a los dispositivos CPLD.

Concluido el razonamiento de cara a la elección de la FPGA, se considera interesante introducir un esquema a alto nivel de la arquitectura interna de la misma. La siguiente imagen ofrece una arquitectura básica en la que se puede observar cómo se diseña lógica en cada bloque, para a continuación mapear las conexiones con el fin de poder unir cada uno de ellos.

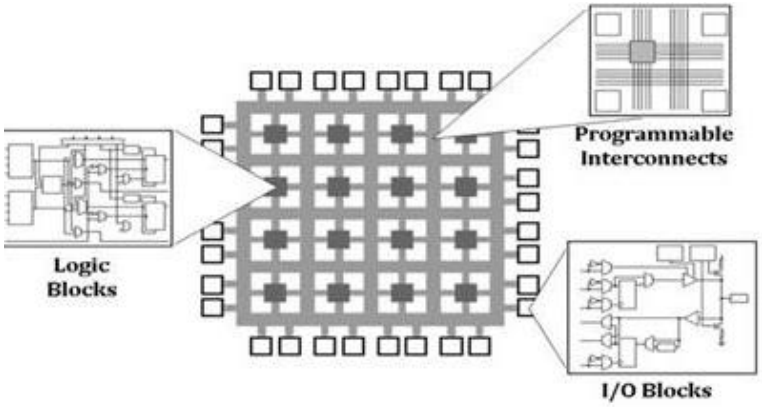


Ilustración 7. Arquitectura interna de una FPGA.

Mediante los lenguajes de descripción hardware (HDL) haciendo uso de estándares en modo texto formado por expresiones, declaraciones y estructuras de control se puede implementar estructuras de circuitos electrónicos digitales de manera más sencilla.

Entre los lenguajes de descripción hardware más importantes encontramos Verilog y VHDL, este último, utilizado en el proyecto consiste en un lenguaje de especificación diseñado por IEEE formado por la combinación de dos acrónimos: VHSIC y HDL.

A la hora de configurar una FPGA, a diferencia de la programación software, el primer paso es el diseño del circuito. Para este paso es recomendable fundamentar el diseño en electrónica digital, con el objetivo de desarrollar una clara arquitectura del proyecto. En el siguiente diagrama se aprecia una estructura recomendada de los pasos a seguir para realizar de manera correcta un diseño firmware en una FPGA.

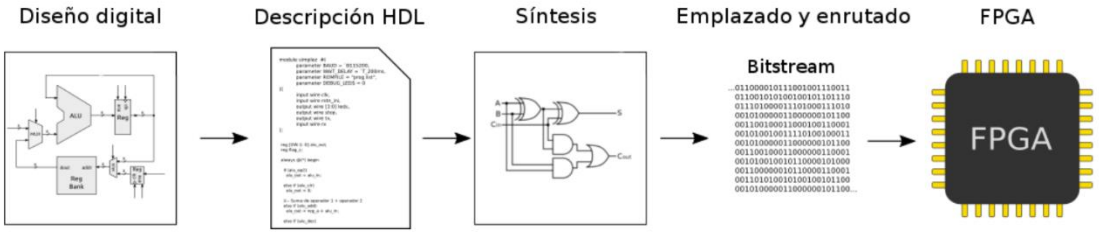


Ilustración 8. Etapas del diseño para una FPGA

El proyecto se implementa mediante el kit de desarrollo Intel DK-DEV-10M50A MAX 10. En este kit de desarrollo destaca la FPGA de Intel de la familia MAX10, este dispositivo será configurado mediante el entorno de desarrollo Quartus (Quartus prime 18.1) y un lenguaje de descripción VHDL. La siguiente imagen ofrece una visualización del kit de desarrollo.

Ilustración 9. Kit de desarrollo Intel DK-DEV-10M50A MAX 10.

En el kit de desarrollo se pueden destacar los siguientes dispositivos:

- *Fuente de reloj de oscilador externo de 25 MHz.*
- *Dos entradas SMA de convertidor analógico a digital (ADC).*
- *Un dispositivo convertidor de digital a analógico (DAC) de 16 bits con salida SMA.*
- *Dos conectores compatibles con Digilent Pmod de 12 pines.*
- *USB-Blaster II integrado (JTAG opcional directo a través del cabezal de 10 pines)*

La elección del kit, la familia de FPGA y el entorno de desarrollo, está fundamentada en una búsqueda de un marco de trabajo en el que el desarrollador se siente lo más cómodo posible, intentando que se conociera algún parámetro de los anteriores de manera previa al comienzo del TFM. El kit de desarrollo cumple holgadamente con todos los requisitos mínimos necesarios para el desarrollo del proyecto, junto con una guía de usuario para el desarrollo ampliamente detallada.

2.4 Dispositivo microcontrolador. µC.

Un microcontrolador es un es un circuito integrado programable, capaz de ejecutar las instrucciones grabadas en su memoria. Un microcontrolador está constituido por una unidad central de procesamiento CPU, una memoria de solo lectura tipo ROM y un conjunto de periféricos de entrada/salida. Los microcontroladores están diseñados por norma general para reducir el costo económico y el consumo de energía de un sistema en particular, para ello destaca el tamaño de la unidad central de procesamiento, la cantidad de memoria y los periféricos incluidos.

Mediante entornos de desarrollo software se programan los dispositivos, estos entornos de desarrollo permiten realizar una programación del dispositivo a un alto nivel como puede ser por ejemplo un lenguaje de programación C. La potencia de estos entornos de desarrollo reside en el puente implementado en la compilación del código que existe entre la programación de alto nivel y la programación de bajo nivel, ya que como es de esperar, el lenguaje de grabación en las memorias habilitadas de los microcontroladores es un lenguaje ensamblador.

Es interesante destacar las diferencias entre un microcontrolador y un microprocesador. El microprocesador, tiene la característica de que sus unidades están físicamente separadas, esto desencadena en que el dispositivo interactúa con una memoria RAM, una memoria ROM y con dispositivos de entrada y salida por medio de buses de comunicación, por el contrario, en el microcontrolador, el microcontrolador contiene su propio CPU, memorias RAM y ROM y dispositivos de entrada y salida ubicados en un único dispositivo. En conclusión y de cara a la elección del dispositivo, se puede garantizar la superioridad del microcontrolador debido a su reducido tamaño y capacidad de ser implementado en circuitos electrónicos.

El motivo de utilizar un microcontrolador en el proyecto es que se tratan de ordenadores con nivel computacional limitado, pero más que suficiente para desempeñar las funciones de este proyecto, destacar también que se tratan de dispositivos económicos y son relativamente fáciles de integrar en diseños de circuitos electrónicos más grandes como es nuestro caso, su capacidad para almacenar y ejecutar programas los hace ser extremadamente versátiles. Otro detalle de especial interés es la facilidad extrema que ofrecen para poder comunicarse con el exterior, facilitando así, de forma indirecta la conversión analógica digital en los pines dedicados del dispositivo. Por ultimo destacar, la predisposición del dispositivo para trabajar las interrupciones y poder dar saltos entre el código principal y las funciones asociadas a las interrupciones.

De cara al tipo de microcontrolador elegido, en primer lugar, diferenciarlos en lugar de los posibles bits de trabajo, además de que existen diversos tipos de microcontroladores los cuales varían dependiendo de la finalidad con la cual serán utilizados. Existen tres tipos de gamas de microcontroladores, analizadas a continuación de forma breve.

- *Gama baja. Dedicados fundamentalmente a tareas de control.*
- *Gama media. Utilizados para tareas de control en las que aparece cierto grado de procesamiento.*
- *Gama alta. Dedicados Fundamentalmente a procesamiento.*

Entre los principales fabricantes de microcontroladores, se podrían destacar: Atmel, Freescale (antes Motorola), Holtek, Intel, Microchip, NXP Semiconductors (antes Philips), Texas Instruments y Zilog.

El microcontrolador elegido es un LPC1768 Mini-DK2, se trata de un dispositivo basado en los procesadores de la serie NXP (NXP Semiconductors) LPC17XX (núcleo Cortex-M3). El dispositivo será programado en lenguaje C mediante el entorno de desarrollo Keil-MicroVision, siendo este un entorno ofrecido por la propia empresa ARM.

En la siguiente imagen se muestra una imagen del dispositivo elegido.

Ilustración 10. LPC1768 Mini-DK2, procesador de la serie NXP LPC17XX (núcleo Cortex-M3).

Con el fin de ofrecer la estructura interna del microcontrolador elegido para el proyecto, junto con todos los posibles grupos esclavos asociados a los posibles usos de los pines dedicados se adjunta la siguiente imagen de la hoja de características del fabricante.

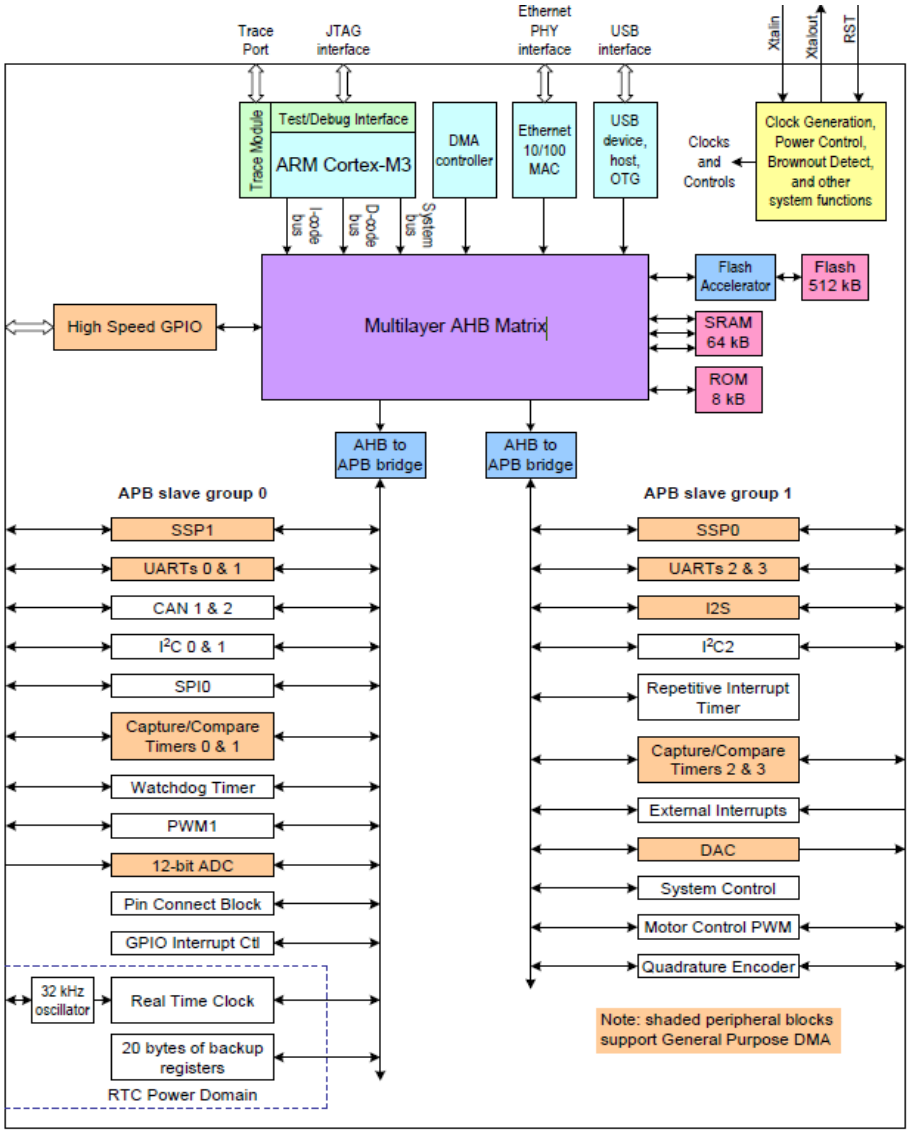


Ilustración 11. Diagrama de bloques internos del microcontrolador.

2.5 Sistema criptográfico. Algoritmo RSA.

La criptografía o cifrado describe el procedimiento mediante el cual se transforma un texto sin formato, denominado texto plano, en una secuencia a priori incomprensible de caracteres mediante una clave. El objetivo de la criptografía, es conseguir que el contenido del texto secreto resultante o criptograma denominado texto cifrado, solo sea accesible para aquellos que disponen de la clave para descifrarlo.

La criptografía o cifrado comenzó con métodos criptográficos sencillos, entre los que se pueden destacar dos tipos, por un lado, el cifrado por transposición o permutación basado en la permutando de caracteres aislados, palabras o frases enteras del texto plano del mensaje. Por otro lado, el cifrado por sustitución basado en el remplazo de caracteres por combinaciones alternativas. Entre los primeros y más sencillos métodos de cifrado se encuentra el cifrado César, datado en fechas cercanas al nacimiento de cristo (época del militar romano Julio Cesar). El objetivo era proteger su correspondencia de los enemigos, el sistema se basaba en la sustitución alfabética simple y consistía en sustituir cada letra por la que se encuentra algunas posiciones más adelantadas en el alfabeto, en este caso se utilizaba un desplazamiento de tres letras.

La codificación moderna se basa en funciones matemáticas de gran complejidad (denominados algoritmos). Esta codificación se basa en la combinación, la sustitución y la transposición, siendo técnicas parametrizadas conjuntamente mediante claves en la forma de contraseñas y códigos binarios. La criptografía moderna se divide de forma fundamental entre los métodos simétricos y los asimétricos partiendo del uso y la gestión de las claves en cada uno de ellos. En el cifrado simétrico se utiliza la misma clave tanto para el cifrado como para el descifrado de los datos. Por el contrario, en los sistemas asimétricos cada una de las partes genera su propio par de claves, compuesto de una clave pública y otra privada. Cuando se combinan la criptografía simétrica y la asimétrica se habla de criptografía híbrida.

En la segunda parte del epígrafe, se realiza el análisis de los sistemas de codificación moderna simétrica, asimétrica e hibrida más importantes utilizados en la actualidad. Destacando el sistema de criptografía RSA debido a que es el utilizado en el proyecto.

- **Sistema de criptográfica simétrica Data Encryption Standard (DES).**

Se trata de un método de cifrado desarrollado por IBM en la década de 1970, por medio de este sistema se sienta las bases de la criptográfica moderna. Es importante destacar que hoy en día se considera ya obsoleto, debido a la reducida longitud de 64 bits (56 bits útiles) de las claves. La potencia actual de computación de la tecnología, ha hecho a una clave DES tan vulnerable, que un ataque de fuerza bruta la descifra en unos pocos segundos.

Este algoritmo simétrico trabaja en paquetes de bits mediante cifrado de bloques, la implementación se basa en que el texto plano se subdivide en bloques de 64 bits, cada uno de los cuales se cifra con una clave. Se traduce así cada texto en un texto cifrado. El algoritmo de cifrado DES se estructura en una red Feistel que combina sustituciones y permutaciones durante 16 rondas.

El sistema DES se podría resumir en la siguiente secuencia, particularizando en cada ronda un bloque y una subclave diferente. El sistema comienza con una permutación del texto plano y de la clave, la cual se divide en dos subclaves. Realizadas ambas permutaciones, se continua el algoritmo con la ronda de cifrado, comenzando con una expansión y una unión con las subclaves mediante una operación XOR, el resultado es procesado mediante cajas de sustitución. Se concluye realizando una permutación de los datos en la salida inversa a la realizada en la entrada. El texto resultante de este último paso se puede considerar un mensaje cifrado.

El principal problema que presenta este estándar de encriptación es la escasa longitud de clave formado por un total de 56 bits, derivando en un espacio de claves también reducido que hoy en día no está en situación de afrontar ataques de fuerza bruta. Debida a esta vulnerabilidad el algoritmo de encriptación DES ha sido sustituido de forma definitiva por el algoritmo, igualmente simétrico, del estándar avanzado o AES, mucho más rápido y eficaz.

- **Sistema de criptográfica simétrica Advanced Encryption Standard (AES).**

Este sistema de criptografía simétrica, se posiciona como el sustituto del sistema DES a finales del año 2000. El sistema AES, subdivide el texto plano en bloques de 128, 192 y 256 bits y, en lugar de utilizar bloques de 64 bits, el estándar avanzado utiliza bloques más grandes de 128 bits. Estos paquetes de bits se cifran en varias rondas de manera consecutiva mediante una red de substitución-permutación (SPN). Al igual que en el sistema DES, AES usa en cada ronda una nueva subclave, que se deriva de forma recursiva de la clave inicial y se une al bloque a cifrar con XOR. Actualmente no se conoce ningún ataque relevante. Sin embargo, la información encriptada con AES solo está segura de los ataques externos mientras la clave permanezca en secreto, presentando serios problemas en relación con el intercambio de la clave, debido a que se usa una sola clave tanto para el cifrado como para el descifrado

El sistema AES se podría resumir en la siguiente secuencia. AES utiliza una nueva subclave en cada ronda, derivada por recursión de la primera. En la ronda previa se transpone en una tabla bidimensional y se une con la primera subclave mediante XOR. A continuación, aparecen las rondas de cifrado. Cada ronda recorre una serie de cuatro pasos, el número de rondas depende de la longitud de clave utilizada. Las instrucciones utilizadas en las rondas son SubBytes, ShiftRows y MixColumns. Para concluir, se produce una nueva unión con la subclave, denominado KeyAddition. En la etapa final mediante operaciones SubBytes, ShiftRows y KeyAddition se obtiene el texto cifrado.

El descifrado de datos encriptados con AES se fundamenta en la inversión del propio algoritmo, tanto en las instrucciones como en la orientación. AES ha demostrado poseer una elevada seguridad debido a su propio algoritmo, la longitud de clave de 128 bits anula la eficacia de los ataques de fuerza bruta y las operaciones ShiftRows y MixColumns realizan la mezcla óptima de los bits.

Entre las aplicaciones que usan este algoritmo, podrían destacar WPA2, SSH e IPSec y la compresión de archivos con herramientas 7-Zip o RAR. Como alternativa a AES también se pueden utilizar los algoritmos simétricos MARS, RC6, Serpent y Twofish,

- **Sistema de criptográfica asimétrica Rivest, Shamir, Adleman (RSA).**

El sistema RSA es un sistema criptográfico asimétrico considerado uno de los métodos más seguros, su nombre se debe a las siglas de sus tres fundadores (Rivest, Shamir y Adleman). La clave del sistema criptográfico RSA son las funciones matemáticas unidireccionales cuyo cálculo directo es sencillo, pero muy complejo a la inversa, ya que requiere muchas operaciones computacionales. Estas operaciones computacionales están basadas fundamentalmente en la aritmética modular.

El sistema RSA, es considerado el primer algoritmo publicado científicamente que permite la transferencia de datos cifrados sin intercambio de claves privadas.

El cifrado RSA utiliza un algoritmo basado en el producto de dos números primos relativamente grandes. La fortaleza del algoritmo se basa en que no existe hasta hoy ningún algoritmo suficientemente potente que sea capaz de descomponer el producto de los dos números en sus factores primos. En la actualidad, RSA ofrece la posibilidad de ajustar el algoritmo al desarrollo tecnológico, involucrando primos aún más grandes para la obtención de las claves.

Para implementar el algoritmo se generan las siguientes dos claves:

Clave pública: (e, n)	*Clave privada: (d, n)*

En primer lugar, se eligen dos números primos denominados *"p"* y *"q"* al azar, con el objetivo de garantizar la seguridad del algoritmo es recomendable que sean de una longitud similar y como mínimo de una dimensión de cien cifras.

El algoritmo avanza con el cálculo de *"n"*, denominado módulo RSA, este número es contenido en ambas claves y representa el producto de los dos números primos escogidos al azar.

$$n = p \cdot q$$

Por otro lado, se obtiene el valor de la función φ de Euler del módulo RSA, ***φ(n)***. Para el cálculo es recomendable aprovechar las propiedades de la función de Euler respecto a los números primos.

- *Propiedades.*

$$si\ "p"\ \ es\ primo\ \rightarrow\ \varphi(p) = (p-1)$$
$$si\ \ "m"\ y\ "n"\ son\ primos\ entre\ si\ \rightarrow\ \varphi(mn) = \varphi(m)\ \varphi(n)$$

- *Conclusión.*

$$f = \varphi(n) = (p-1) \cdot (q-1)$$

Para concluir los cálculos relacionados con la clave pública, se elige un número *"e"* al azar inferior a *"f"*. La condición para la elección de este número es que no tenga factores en común con *"f" (*parámetro que define la función φ de Euler del módulo RSA) en otras palabras, es necesario que "e" y "f" sean coprimos. Para el desarrollo de este paso, será recomendable el uso del algoritmo de Euclides. Este número *"e"* es conocido como el exponente de la clave pública, un exponente muy pequeño podría suponer un riesgo para la seguridad del algoritmo.

$$1 < e < f \rightarrow condicionado\ a\ que\ "e"\ sea\ coprimo\ a\ "f"$$

Tras la generación de la clave pública [*Clave pública:(e, N)*] los dos números primos involucrados en los cálculos deberán mantenerse en secreto para poder garantizar la seguridad del cifrado de RSA. Se destaca que el producto de ambos números es visible para todo el mundo en la clave pública. Condicionado a que en la actualidad no existe ningún algoritmo suficientemente eficiente, que pueda descomponer en sus factores al producto de dos números primos elevados en un tiempo relevante, resulta obvia que la garantía de seguridad del algoritmo se encuentra ligada a la potencia computacional.

Para la generación de la clave privada [*Clave privada: (d, N)*] únicamente es necesario un nuevo número *"d"* calculado mediante aritmética modular. Para el cálculo del nuevo número, es condición necesaria que el número *"d"* sea el multiplicador modular inverso de *"e mod (f)"*. Esto valor es recomendable calcularlo mediante el algoritmo de Euclides extendido. Este número *"d"* es conocido como el exponente de la clave privada,

$$af\ +\ de\ =\ mcd(e; f)\ =\ 1$$

En otras palabras, que el producto *"d"* por *"e"* menos uno, sea dividido exactamente por *"f"*. Esta condición se traduce en la siguiente identidad.

$$de\ \equiv\ 1\ (mod\ f\ )$$

En resumen. La clave pública es [*Clave pública: (e, N)]* traducido en exponente de cifrado y módulo. Mientras que la clave privada es [*Clave privada: (d, N)*] traducido en exponente de descifrado y módulo.

Usando las propiedades de la función de Euler, el Teorema de Euler y el Teorema del resto chino se puede garantizar:

$$mensaje \equiv mensaje^{ed} \ (mod\ n)$$

Concluido la parte de la generación de las claves públicas y privadas, el epígrafe avanza con la explicación del cifrado y descifrado del mensaje transmitido.

Cifrado del mensaje:

Un transmisor quiere enviar un mensaje a un receptor. El mensaje en texto plano *"M"* se cifra numéricamente a un número entero *"m"* menor que *"n"*. Con el mensaje en formato numérico se procede a aplicar la encriptación RSA y enviar el texto cifrado *"c"*.

$$c \equiv m^e \ (mod\ n)$$

Es recomendable el uso de la exponencial binaria para el cifrado del mensaje.

Descifrado del mensaje.

En el receptor se recibe el mensaje encriptado *"c"*. El receptor es capaz de desencriptar el mensaje con su clave privada, en particular mediante el multiplicador modular inverso *"d"*.

$$m \equiv c^d \ (mod\ n)$$

El procedimiento anterior funciona gracias a que se cumple la siguiente identidad. Para obtener esta conclusión se hace uso del teorema chino del resto y del teorema de Euler.

$$c^d = (m^e)^d \equiv m^{ed} \ (mod\ n) \qquad \rightarrow \qquad m^{ed} = m^{1+k\varphi(n)} = m(m^{\varphi(n)})^k \equiv m \ (mod\ n)$$

Concluida la explicación del algoritmo RSA, se concluye el epígrafe destacando algunos datos de interés sobre este sistema de encriptación. En primer lugar, en la actualidad los números primos con los que trabaja este algoritmo son del orden de 10^{300}, siendo un tamaño que podrá ir aumentando según avance la capacidad computacional de cálculo. La seguridad del RSA está basado en el problema matemático de factorizar números grandes y el propio problema RSA. El descifrado completo de un texto cifrado con RSA es computacionalmente intratable, no se ha encontrado un algoritmo eficiente todavía para resolver ambos problemas. Se prevé que en un futuro con la computación cuántica el algoritmo se pueda romper, pero hasta que llegue esa tecnología, es un método de cifrado seguro. El uso más extendido de este algoritmo es la firma digital.

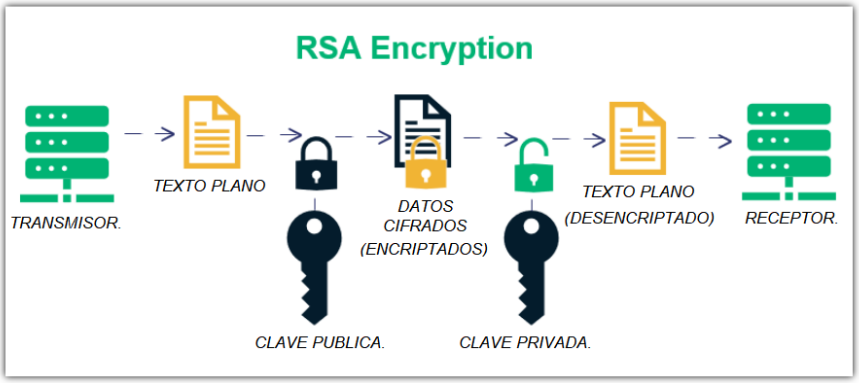


Ilustración 12. Sistema Criptográfico RSA

- **Sistema de criptográfica asimétrica Cifrado por clave pública basada en IP.**

Este sistema criptográfico nace en 1984, diseñado por uno de los desarrolladores del RSA, Adi Shamir, intentando solucionar la debilidad más importante de la criptografía asimétrica, es decir, la autentificación de los interlocutores. Este sistema propone un método basado en un ID. En el procedimiento de codificación basado en la identidad, la clave pública de un interlocutor se calcula a partir de un identificador inequívoco. Este concepto hace innecesaria la autentificación de las claves públicas, cuyo papel verificador adquiere ahora otra entidad, el denominado generador de clave privada (PKG) mediante un algoritmo central el destinatario de un mensaje cifrado puede obtener una clave de descifrado relativa a su identidad.
El método ID más conocido es el desarrollarlo por Dan Boneh y Matthew K. Franklin en 2001.

- **Sistema de criptográfica hibrido.**

Los métodos de cifrado híbrido consisten en la utilización conjunta de los dos tipos de algoritmos, tanto simétricos como asimétricos. El objetivo de esta combinación es compensar las debilidades de un sistema con las ventajas de otro. por un lado, el algoritmo simétrico AES al disponer de una clave privada común, presenta el problema del intercambio de claves, únicamente soluble con algoritmos asimétricos, el algoritmo asimétrico RSA separa ambas claves, la privada y la pública, permitiendo solucionar el problema del intercambio de la clave secreta con longitud de clave suficientemente grande, en concreto de 1976 bits como mínimo.

Una combinación muy habitual en la criptografía híbrida consiste en el cifrado simétrico de los datos útiles mediante AES y el consiguiente descifrado asimétrico de la clave de sesión con RSA. En otras palabras, la criptografía híbrida utiliza los algoritmos asimétricos únicamente para asegurar la transferencia de una clave de sesión simétrica en un canal no protegido. Esta clave permite, a su vez, descifrar eficazmente un texto codificado con ayuda de algoritmos simétricos.

Este esquema permite el cifrado y descifrado eficiente de datos útiles, incluso de gran volumen, a una gran velocidad. Al encriptarse asimétricamente solo una clave de sesión, se evitan los largos tiempos de procesamiento propios de los logaritmos asimétricos, así como el problema del intercambio de claves se reduce al problema de la autenticación de los usuarios.

Conclusión y elección del sistema criptográfico utilizado para el proyecto.

La criptografía simétrica utiliza la misma clave en ambas partes de la comunicación, por el contrario, en una comunicación cifrada de forma asimétrica ambos interlocutores generan un par de claves en cada lado, de modo que cada uno dispone de dos claves, una pública y una privada. Es en la clave privada donde reside la fortaleza de este tipo de sistema criptográfico. En la criptografía asimétrica las claves públicas son utilizadas para el proceso de cifrado, mientras que, las claves privadas que solo conoce el receptor se utilizan para descifrar el mensaje.

Tras realizar un estudio de los diferentes tipos de criptografía, se decide utilizar en el proyecto la criptográfica asimétrica RSA. El motivo principal, es el escenario aéreo de trabajo en el cual se considera inviable poder conseguir en algún momento un canal seguro para el intercambio de claves simetricas. Se ilustra con un ejemplo ese tipo de criptografía con el fin de entenderse más a fondo.

Supongamos el siguiente escenario. El usuario A quiere enviar un mensaje encriptado al usuario B, en un espacio aéreo donde también se encuentra un usuario C.

Para ello, el usuario A necesita la clave pública de B. La clave pública de B permite a A cifrar un mensaje que solo pueda ser descifrado con la clave privada que solo posee el usuario B, este detalle garantiza que solo B podrá leer el mensaje, ni siquiera el propio usuario A podrá leerlo después de cifrarlo. Como la comunicación es por espacio aéreo, solo se intercambiará la clave pública y se puede prescindir de la necesidad de disponer de otro canal más seguro, canal necesario en la comunicación simétrica.

El único detalle es la necesidad de poder garantizar la autenticidad del usuario que envía la información, pudiendo utilizarse cualquier código particular de vuelo asociado a la aeronave, de manera indirecta se estaría proponiendo un método similar al desarrollado por Adi Shamir ofreciendo al interlocutor un identificador inequívoco. El motivo es obvio, B no podrá estar seguro de que el mensaje proviene realmente de A, puesto que teóricamente C, podría utilizar la clave pública de B para cifrar y enviar mensajes

Anexo. Resumen expandido de los métodos de cifrado simétricos.

Data Encryption Standard (DES).

1. Permutación de entrada. El bloque de texto plano de 64 bits se somete a una permutación de entrada que modifica la secuencia de los bits.

2. Permutación de la clave. Los 56 bits útiles de la clave para el cifrado también se someten a una permutación, para después ser divididos en dos bloques de 28 bits.

3. Rondas de cifrado. Cada ronda recorre una serie de cuatro pasos. En primer lugar, se realiza una expansión del mensaje, el siguiente paso realiza una unión mediante una operación XOR del bloque de datos y la subclave, tras la unión de los datos y la subclave se realiza un procesado de cajas de sustitución, para concluir se realiza la operación lógica XOR de ambos bloques.

4. Permutación de salida. Una vez se han efectuado las 16 rondas de cifrado, los bloques se agrupan en un bloque de 64 bits y se someten a una permutación de salida inversa a la de entrada. El texto resultante de esta permutación se puede considerar un mensaje cifrado.

Advanced Encryption Standard (AES).

1. Expansión de la clave. AES utiliza una nueva subclave en cada ronda, la cual, derivada por recursión de la primera clave, se expande una longitud que permita generar el número necesario de subclaves de 128 bits.

2. Ronda previa. En la ronda previa el bloque de 128 bits se transpone en una tabla bidimensional y se une con la primera subclave mediante XOR.

3. Rondas de cifrado. Cada ronda recorre una serie de cuatro pasos, el número de rondas depende de la longitud de clave utilizada. En cada ronda tienen lugar las siguientes operaciones. En primer lugar, mediante la instrucción SubBytes se realiza una substitución alfabética simple, tras la sustitución los bytes de las casillas de la matriz se desplazan cíclicamente hacia la izquierda haciendo uso de la instrucción ShiftRows. Concluido el desplazamiento, se mezclan los datos dentro de las columnas de la tabla, este paso denominado MixColumns se basa en un recalculo de cada casilla en el que las columnas se someten a una multiplicación matricial, los resultados de este producto se unen mediante la puerta XOR. Para concluir, al final de cada ronda se produce una nueva unión con la subclave, denominado KeyAddition

4. Etapa final. En esta última etapa mediante operaciones SubBytes, ShiftRows y KeyAddition se obtiene el texto cifrado.

2.5 Espacio electromagnético. Sistemas RF.

Los sistemas electrónicos de comunicaciones transfieren información mediante energía electromagnética divida en las diferentes bandas de frecuencia. Mediante el espectro electromagnético se distribuye la energía de las ondas emitidas quedando marcado por las diferentes radiaciones de las mismas. En otras palabras, podemos resumir que las ondas suponen la propagación de la radiación.

El espectro electromagnético se divide en diferentes bandas en función de la frecuencia de emisión, cada banda tiene un intervalo frecuencial acotado en valores máximos y mínimos. Estas designaciones permiten la clasificación en función de las propiedades y múltiples aplicaciones asociadas a cada una de ellas. Las diferentes bandas de trabajo con su respectivo rango completo se resumen en la siguiente tabla.

Número de banda.	***Intervalo frecuencial.***	***Descripción y designación de la banda.***
Banda Nº 2	*30 Hz-300 Hz*	*ELF (Frecuencias extremadamente bajas). Señales de telemetría*
Banda Nº 3	*0.3 kHz-3 kHz*	*VF (Frecuencia de voz). Canales telefónicos y com. mineras*
Banda Nº 4	*3 kHz-30 kHz*	*VLF (Frecuencias muy bajas). Comunicaciones militares, especialmente en submarinos. Radio de gran alcance.*
Banda Nº 5	*30 kHz-300 kHz*	*LF (Frecuencias bajas). Navegación marítima y aeronáutica,*
Banda Nº 6	*0.3 MHz-3 MHz*	*MF (Frecuencias intermedias). Radio modulada en amplitud AM*
Banda Nº 7	*3 MHz-30 MHz*	*HF (Frecuencias altas). Radio de onda corta, banda de uso civil.*
Banda Nº 8	*30 MHz-300 MHz*	*VHF (Frecuencias muy altas). Usado en radios móviles. Emisión de TV y Radio modulada en frecuencia FM*
Banda Nº 9	*0.3 GHz-3 GHz*	*UHF (Frecuencias ultra altas). Telefonía móvil (GSM), WLAN, Radioaficionados uso de RFID. Tecnología radar.*
Banda Nº 10	*3 GHz-30 GHz*	*SHF (Frecuencias súper altas). Sistemas de radiocomunicación basado en microondas y satélites.*
Banda Nº 11	*30 GHz-300 GHz*	*EHF (Frecuencias extremadamente altas). Radares de alta resolución*
Bandas Nª 12 13 y 14	*0.3 THz- 300 THz*	*Luz infrarroja. Este intervalo no es visible por el ojo humano. Estas bandas no son consideradas ondas de radio.*
Banda Nº 15	*0.3 PHz-3 PHz*	*Luz visible. Utilizada en fibra óptica.*
Banda Nº 16	*3 PHz-30 PHz*	*Luz ultravioleta. Bandas demasiado elevadas, apenas utilizadas.*
Banda Nº 17	*30 PHz-300 PHz*	*Rayos X. Bandas demasiado elevadas, apenas utilizadas.*
Banda Nº 18	*0.3 EHz-3 EHz*	*Rayos gamma. Bandas demasiado elevadas, apenas utilizadas.*
Banda Nº 19	*3 EHz-30 EHz*	*Rayos cósmicos. Bandas demasiado elevadas, apenas utilizadas.*

Tabla 4. Tabla de bandas del espectro electromagnético.

La siguiente imagen ofrece el diagrama del espectro electromagnético con el objetivo de ofrecer una mejor visualización de los parámetros asociados a este concepto.

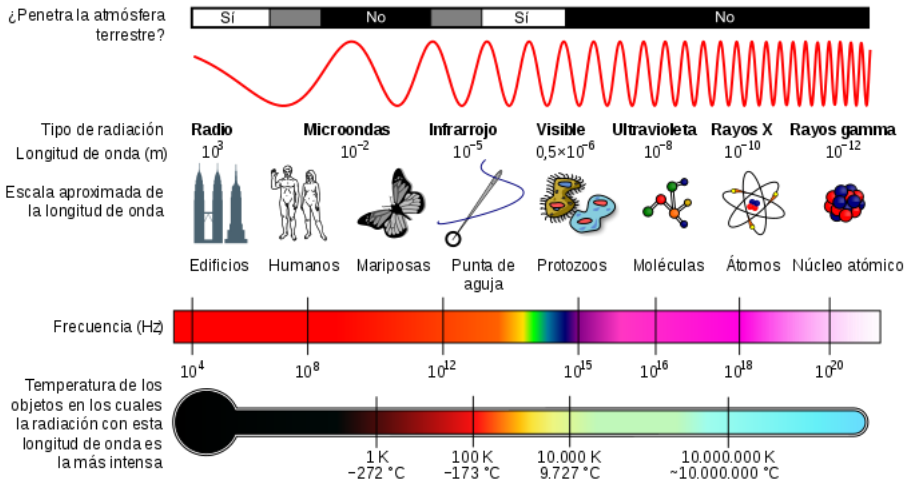


Ilustración 13. Diagrama del espectro electromagnético.

Las bandas frecuenciales del espectro electromagnético con menos energía asociada recibe el nombre de Radiofrecuencia (RF), en concreto abarcan hasta la banda número once, traducido en intervalo frecuencial desde los 30 Hz hasta los 300 GHz. Estas ondas por norma general son producidas mediante la variación de un campo electromagnético, es decir, una combinación de campos eléctricos y magnéticos oscilantes, los cuales son propagagados, a través del espacio transportando energía a la velocidad de la luz. La generación y propagación de estas ondas son compatibles con el modelo de ecuaciones matemáticas definido en las ecuaciones de Maxwell.

Estas ondas electromagnéticas pueden ser representadas como campos eléctricos y magnéticos autopropagados en forma de onda transversal. El diagrama muestra la propagación de una onda polarizada, esta onda está constituida por un campo eléctrico (color azul) definido sobre el plano vertical y un campo magnético (color rojo) sobre el plano horizontal. Los campos eléctrico y magnético en este tipo de ondas siempre forman una diferencia en fase de 90° una respecto a la otra.

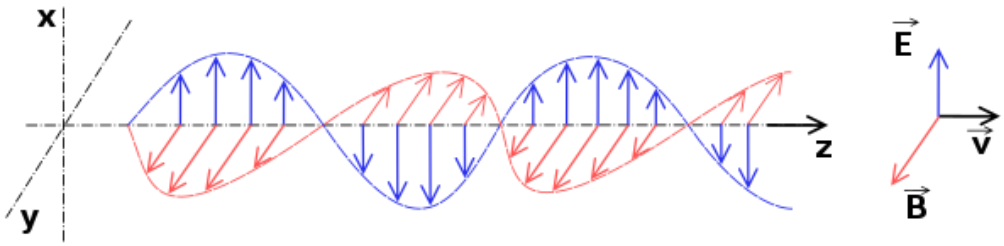


Ilustración 14. Transmisión de una onda electromagnética

Con el objetivo de realizar una transmisión inalámbrica entre las dos aeronaves el sistema hará uso de la tecnología RF. Para completar el diseño, es necesario elegir dos parámetros de diseño fundamentales. En primer lugar, la banda de trabajo y por lo tanto el intervalo frecuencial utilizado, en segundo lugar, el tipo de modulación utilizado.

En los sistemas IFF reales, la transmisión de squitter se implementa a una frecuencia de 1090 MHz, utilizando una modulación por posición de pulso (modulación digital PPM).

Respetando las regulaciones limitadoras internacionales, de cara a las bandas libres de trabajo se elige trabajar en la banda ISM europea centrada en 868 MHz. Por otro lado, debido a que la transmisión se basa únicamente en niveles binarios, se decide utilizar la señalización PPM, manteniendo el tipo de modulación digital que utiliza la tecnología IFF de cara a la codificación binaria de unos y ceros, en conjunto con la modulación por desplazamiento en amplitud ASK utilizando la portadora en la banda ISM decidida previamente con el objetivo de poder transmitir información vía RF. Cabe destacar que, como mejora si en un futuro el proyecto necesitara transmitir constelaciones de bits, con el fin de mejorar el régimen binario, sería necesario utilizar una modulación en fase.

La segunda parte de este epígrafe del estado del arte, analiza por separado los dos parámetros de diseño elegidos, la banda de trabajo y las modulaciones.

- **Banda ISM (Industrial, Scientific and Medical).**

Las bandas de radio industriales, científicas y médicas (Bandas ISM) son bandas del espectro de radio electromagnético, reservadas internacionalmente para el uso libre, se encuentran ubicadas en las bandas de radio frecuencia, actualmente la más conocida es la banda ubicada en 2.4 GHz. Se traducen en márgenes frecuenciales del espectro, idóneos para comunicaciones inalámbricas sin licencia de corto alcance y baja potencia, como el de nuestro proyecto.

Las bandas ISM están definidas por el Reglamento de Radiocomunicaciones de la UIT (artículo 5), el uso individual de los países de las bandas designadas en estas secciones puede diferir debido a las variaciones en los reglamentos de radio nacionales. El propio reglamento de Radiocomunicaciones define las bandas ISM como: *"Operación de equipos o dispositivos diseñados para generar y utilizar energía de radiofrecuencia local para fines industriales, científicos, médicos, domésticos o similares, excluidas las aplicaciones en el campo de las telecomunicaciones.".*

Las especificaciones originales e iniciales de ISM, preveían que las bandas se utilizarían principalmente para fines de no comunicación, como el calentamiento.

En la actualidad el dispositivo ISM que se encuentra con más frecuencia en el mercado, es el horno de microondas doméstico que funciona a 2.45 GHz y utiliza microondas para cocinar los alimentos. La congestión de estas bandas iniciales da lugar a la aparición de nuevas bandas de operación denominadas bandas no ISM, ya que no incluyen en áreas de aplicación "industrial, científica o médica" previstas originalmente en el estándar.
Entre las aplicaciones más desarrolladas encontramos la red inalámbrica (WiFi), en una segunda posición se encuentran todas las computadoras portátiles, tabletas, impresoras y teléfonos celulares. Otra tecnología que trabaja en esta banda es el bluetooth.

En Europa, el uso de la banda ISM está cubierto por las regulaciones de dispositivos de corto alcance emitidas por la Comisión Europea, basadas en las recomendaciones técnicas de CEPT y las normas de ETSI. Europa, África, Asia y Oceanía utiliza 80 o 40 canales con una separación de 50 KHz en un rango frecuencial de 866 a 869.9 MHz.
El proyecto trabaja en la banda no ISM europea más desarrollada de bajo consumo (LPWAN), esta banda se encuentra ubicada en los 868 MHz dividida en 6 sub-bandas diferentes.

- **Modulación PPM (Pulse Position Modulation).**

La modulación PPM consiste en un tipo de modulación donde la amplitud, la anchura y el periodo de los pulsos es constante, la parte variable de la modulación reside en la posición de inicio del pulso. Una de las principales dificultades en la implementación de esta técnica es que el receptor debe estar debidamente sincronizado para poder alinear el reloj local con el inicio de cada símbolo. Por otro lado, una de las principales ventajas de este tipo de modulación es que puede ser implementada de forma no coherente, de manera tal que el receptor no necesita utilizar un lazo de seguimiento de fase para obtener la información transmitida.

De cara a la codificación, trabajando en regímenes de transmisiones binarias, es decir, transmitiendo un único bit, en cada periodo de la señal, como es el caso del proyecto, el sistema transmisor ubica el pulso en la primera mitad del periodo de transmisión si se quiere transmitir un nivel alto, mientras que, por el contrario, el sistema transmisor ubicara el pulso en la segunda mitad del periodo, si se quiere transmitir un nivel bajo.

A continuación, se muestra un ejemplo de una transmisión con este tipo de modulación.

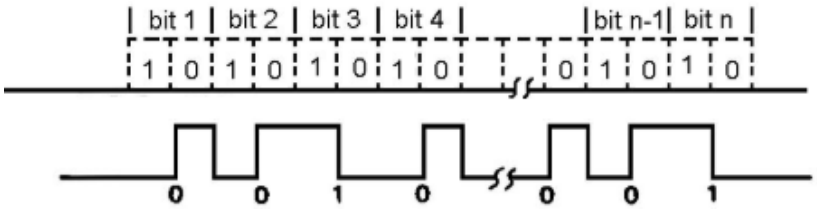


Ilustración 15. Ejemplo de modulación PPM.

- **Modulación ASK (Amplitude-shift keying).**

La modulación ASK, consiste en un tipo de modulación basada en desplazamiento de amplitud, modelando variaciones en la amplitud de la onda portadora y en consecuencia en la señal recibida. Los valores binarios de la señal digital moduladora, se representan por medio de dos amplitudes diferentes, por regla general, una de las dos amplitudes es cero. En conclusión, uno de los dígitos binarios se representa mediante la presencia de la portadora con amplitud constante, y el otro dígito se representa mediante la ausencia de la señal portadora. Se podría decir que la modulación ASK describe la modulación AM adaptada para sistemas digitales.

La siguiente ilustración, se visualiza la idea comentada en el párrafo anterior.

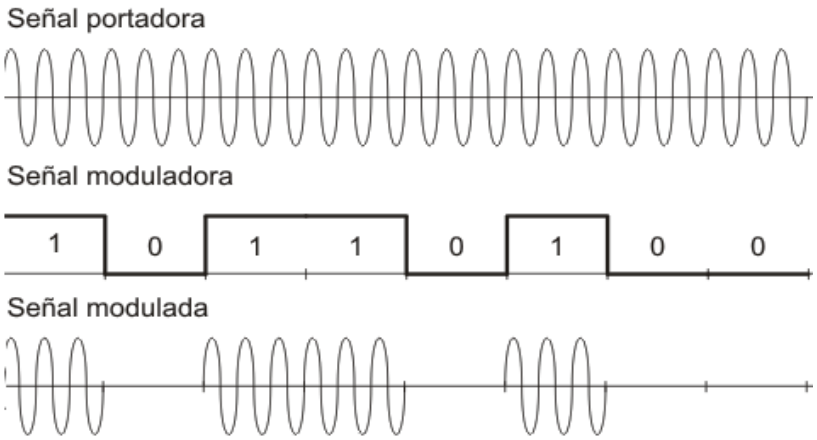


Ilustración 16. Ejemplo de modulación ASK.

Por lo tanto, podemos identificar la modulación ASK como, la conmutación de la señal sinusoidal portadora a través de su encendido y apagado mediante una señal binaria unipolar que, representa el mensaje que queremos transmitir. Considerando que la señal que se quiere transmitir tiene carácter binario, podemos deducir que únicamente puede tener dos niveles de tensión diferentes, el valor 0 representando un nivel bajo y, el valor A representando con un nivel alto. Su función se representa mediante la siguiente ecuación.

$$f_{ASK}(t) = f(t)\cos(\omega_c t)$$

- $Si\ f(t) = 0V \rightarrow f_{ASK}(t) = 0$
- $Si\ f(t) = AV \rightarrow f_{ASK}(t) = A\cos(\omega_c t)$

Concluimos identificando de los dos posibles símbolos de la modulación en un plano *"In phase and quadrature"*

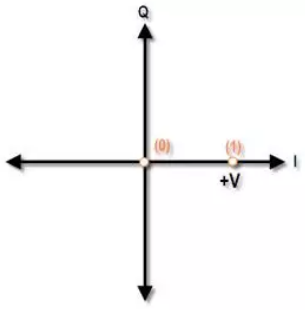


Ilustración 17. Representación In phase and quadrature de una modulación ASK

Las principales ventajas de este tipo de modulación, es la fácil adaptación a cualquier sistema de radio frecuencia, ya que basta con decidir la frecuencia de la portadora, para continuar adaptando la transmisión y recepción a la frecuencia de la señal. Aun así, ASK no es una de las modulaciones más utilizadas en la actualidad, debido principalmente al bajo régimen de transmisión, ya que únicamente puede transmitir un bit al mismo tiempo en una determinada frecuencia, otro motivo es, el laborioso trabajo que llevaría adaptar cada sistema a la frecuencia de la portadora, impidiendo hacerlo altamente escalable.

Para la parte de la transmisión y por lo tanto modular. La implementación más típica para realizar la modulación ASK, consiste en el uso de un interruptor modelando el on/off mediante la señal moduladora, cuando ésta sea un nivel alto el interruptor conduce la portadora a la salida, mientras que, si por el contrario es un nivel bajo, el interruptor se corta, garantizando un nivel bajo en la salida.

La siguiente imagen, modela la idea de la generación de la señal ASK.

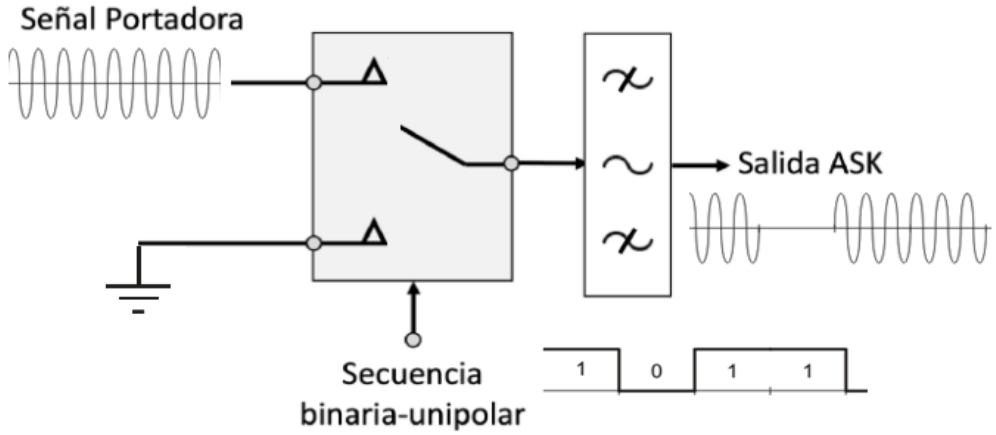


Ilustración 18. Método de generación de señal ASK.

Para la parte de la recepción y por lo tanto demodular. La implementación más típica para realizar la demodulación ASK, consiste en utilizar un detector de envolvente, el cual está constituido por un diodo rectificador y una red RC. El funcionamiento es simple, el diodo conduce en los semiciclos positivos de la señal y se corta en los negativos, en los semiciclos positivos, el condensador de la red RC mantiene la tensión constante en sus bornas con un pequeño transitorio. Como posible mejora, sería interesante introducir la señal de salida del demodulador en un comparador digital, con el objetivo de filtrar y blindar la señal envolvente obtenida en el detector.

La siguiente imagen, modela la idea de la recepción de la señal ASK.

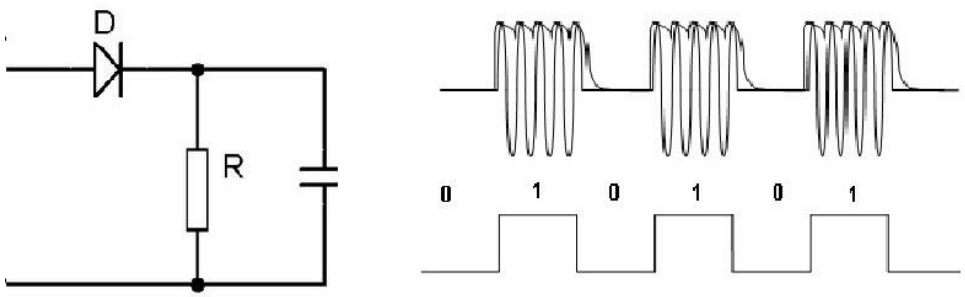


Ilustración 19. Método de recepción de señal ASK

2.6 Placa base PCB. Printed Circuit Board.

Una tarjeta de circuito impreso (PCB), es la base física que soporta el montaje de componentes y, la conexión eléctrica que permite el conexionado de los mismo mediante pistas o taladros. La tecnología de montaje superficial (SMT), es el sistema o conjunto de procesos usados para conectar física y, eléctricamente componentes de montaje superficial (SMD) a una placa de circuito impreso (PCB).

Entre las principales ventajas de la tecnología SMT frente a la convencional se puede destacar: la miniaturización en tamaños de 0.4x0.2mm, una alta fiabilidad con un alto rendimiento asociado, un control del proceso mucho más fácil y simple (permitiendo el control de procesos estadísticos), una reducción de costes, gran facilidad para la automatización del proceso y mejores características en altas frecuencias. La tecnología SMT se clasifica en función del lugar donde van montados los componentes y del tipo de componentes.
Los componentes SMD son dispositivos electrónicos miniaturizados. Se tratan de unos componentes que surgen para la tecnología de Circuitos Híbridos donde la fijación de los componentes al substrato cerámico precisaba de componentes provistos de terminales especiales.

Las tarjetas de circuito impreso pueden ser simple cara, en caso de que los componentes se monten en una sola cara de la PCB, o doble cara si por el contrario los componentes son montados en las dos caras de la PCB, o como última opción multicapa en caso de que lleven más de una capa de fabricación. La siguiente imagen muestra una ilustración de los tipos de montajes de PCB comentados previamente.

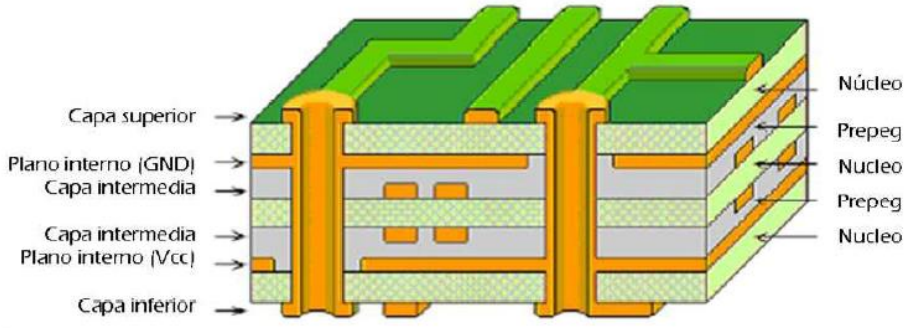


Ilustración 20. Tipos de tarjetas de circuito impreso.

Se puede garantizar que para implementar el proyecto de radio frecuencia (modulación y demodulación de la señal ASK en la banda ISM) implementar el diseño en una PCB mediante tecnología SMT y componentes SMD es la opción más factible. Este tipo de tecnología de fabricación impresa, destaca por minimizar los efectos inductivos y capacitivos traducidos en efectos parásitos en técnicas de RF.

El diseño se llevará a cabo mediante el entorno de desarrollo *"EasyEDA"*. Este entorno tiene un enlace de fabricación directo con la compañía *"JLCPCB"*. Las especificaciones de diseño más importantes para la implementación de prototipos de circuitos de RF en tarjetas PCB se muestran en la siguiente tabla.

Característica.	***Especificaciones técnicas de diseño PCB.***
Numero de capas	*Entre 2 y 20 capas*
Espesor del dieléctrico	*Entre 0.1 y 0.3 mm*
Aspectos tecnológicos y tipos de materiales	*Impedancias controladas y adaptadas y materiales de bajas perdidas.*

Tabla 5. Tabla resumen de especificaciones para circuitos RF en PCBs.

3. Diseño del proyecto. Análisis general.

Este proyecto presenta el diseño y desarrollo de un sistema de comunicación basado en protocolo IFF (Identification Friend or Foe). IFF es un sistema de identificación criptográfica utilizado tanto en el campo civil como militar. En el campo civil se utiliza para controlar el tráfico aéreo, mientras que en el militar su función es la de distinguir los vehículos enemigos de los que no lo son. Este proyecto diseña y desarrolla en concreto la transmisión y recepción de squitters encriptados, la encriptación se lleva a cabo mediante un sistema criptográfico de clave asimétrica basado en el algoritmo RSA. Mediante el uso de una FPGA desarrollado en lenguaje de descripción HW VHDL se simula la comunicación entre dos equipos basados en protocolo IFF, en particular la transmisión de squitters en modo S, formado por un preámbulo y un conjunto de datos encriptados.

El proyecto desarrollado como TFM, utilizara la misma modulación que los sistemas IFF (modulación PPM) también se respetan los tiempos asociados a los niveles altos y bajos del preámbulo, además del régimen binario y la longitud de la trama enviada en la parte de la transmisión de los datos. La transmisión se centra en paquetes de 56 bits, transmitiendo un squitter en formato short squitter DF11, manteniendo su valor propio de régimen binario de un bit por microsegundo.

Como mejora del proyecto, aparece la parte referenciada a la encriptación, en concreto criptografía asimétrica mediante tecnología RSA. Se decide encriptar los mensajes con el objetivo de que la transmisión de las diferentes aeronaves del espacio aéreo con la torre de control, puede ser recibida únicamente por esta última. Podría darse el caso de que apareciera una tercera persona con el objetivo de querer sabotear la comunicación, en caso de que no estén los mensajes encriptados, podría extraer el identificador de una aeronave transmisora, copiarlo y transmitir nuevos mensajes con valores falsos, creando así un completo caos por parte de la torre de control, ocasionando múltiples accidentes aéreos.

El tipo de criptografía elegida (criptografía asimétrica) se fundamenta en que la torre de control, podría publicar sus claves publicas sin ningún problema y cualquier aeronave podría enviar información cifrada que únicamente podría descifrar la torre de control con su clave privada. En este marco teórico, se puede apreciar una gran facilidad de cara a transmitir información desde la aeronave hasta la torre de control.

El squitter transmitido mantiene la particularidad que ofrece el modo S frente al resto de modos civiles SIF. Esta particularidad consiste en transmitir en el propio squitter el identificador del origen transmisor, este concepto se denominada transmisión selectiva. Para ello los primeros 18 bits de información encriptada transmitida representaran el identificador de la aeronave. La estructura del squitter formado por un total de 56 bits es la siguiente:

- *2 bits a nivel alto constantes al principio de la trama*
- *18 bits a continuación: identificador de la aeronave encriptado.*
- *18 bits a continuación: tipo de mensaje encriptado.*
- *18 bits a continuación: mensaje enviado encriptado.*

Se transmite de manera constantemente y de forma automática sin necesidad de ningún usuario, cuatro squitters seguidos con un contenido de los valores de longitud, latitud, velocidad y altitud de la aeronave. A parte de estos mensajes que se transmiten de forma periódica, el usuario podrá enviar cuando lo desee mensajes denominados de carácter especial, estos mensajes son los siguientes:

- Mensaje SOS. *Fallo de carácter general en la aeronave*.
- Mensaje del nivel de Gasolina. *Envía el nivel de combustible actual, pudiendo indicar problemas asociados a los niveles de carburante.*
- Mensaje de piloto automático activado. *Transmite que el piloto automático de la aeronave se encuentra activado.*
- Mensaje de solicitud de aterrizaje. *Transmite una solicitud por parte de la aeronave de aterrizaje inmediato.*

Tal y como se ha comentado en repetidas ocasiones a lo largo del documento, el proyecto se centra en el diseño, desarrollo y simulación de transmisión de squitters en modo S encriptados en RSA mediante tecnología ADS-B propia de equipos IFF.

De cara al desarrollo técnico, se puede ilustrar el desarrollo referente a los dispositivos utilizados en la transmisión y recepción en el siguiente diagrama de bloques.

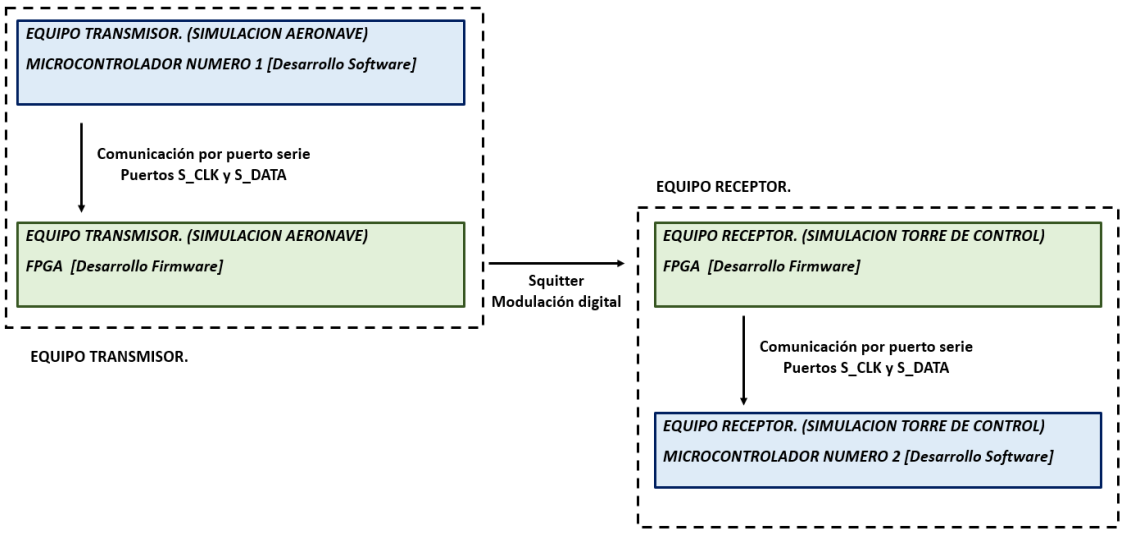


Ilustración 21. Diagrama de bloques genérico del proyecto

El proyecto está constituido por una FPGA y dos microcontroladores.

En primer lugar, respecto a la FPGA, con el objetivo de abaratar los costes del proyecto, se utiliza un único dispositivo. Es en el código firmware donde se estructura simulando dos dispositivos diferentes.

Por la parte de los microcontroladores. El primero de ellos, ubicado en la etapa de transmisión pretende realizar la simulación de la aeronave, mientras que el segundo ubicado en la etapa de recepción pretende realizar la simulación de la torre de control.

Para la comunicación entre los dos microcontroladores y la FPGA el proyecto utiliza una comunicación por puerto serie, se utilizan dos pines destinados al propósito general de ambos dispositivos para transmitir la señal de reloj y la señal de datos.

La comunicación entre las dos estructuras de código firmware se lleva a cabo con el mapeado de una señal de salida para la transmisión del squitter y una señal de entrada para la recepción del squitter, es importante destacar que la comunicación se realiza en banda base (modulación digital).

El proyecto de forma genérica, desde un alto nivel, realiza un control del tráfico aéreo mediante transmisiones periódicas de squitters. Para llevar a cabo este proyecto, se utilizan dos microcontroladores y una FPGA. La función de cada dispositivo se explica a continuación.

Un primer microcontrolador simula el avión. Su función es realizar el interfaz con el usuario. En primer lugar, mediante una pantalla táctil se eligen los dos números primos en los que está basado el algoritmo RSA. Estos números primos son enviados a la FPGA, para su futuro procesamiento. A continuación, mediante un nuevo menú, en la misma pantalla táctil el usuario puede elegir el identificador de la aeronave en la que se encontraría. Definidos estos dos parámetros de forma periódica se genera una interrupción que se transmite a la FPGA para realizar el envío de la posición a niveles de longitud, latitud, velocidad y altitud junto con algún mensaje de carácter especial solicitado por el usuario en el nuevo menú ofrecido por la pantalla táctil. Entre los mensajes de carácter especial que se pueden enviar en cualquier momento se encuentran: la solicitud de SOS, la reserva de combustible de la aeronave, la activación del piloto automático y por último la solicitud de pista de aterrizaje. El microcontrolador en los pines dedicados para la conversión analógica digital, tiene conectado un sensor de luminosidad, para configurar automáticamente el brillo de la pantalla, un sensor de temperatura, para conceder la temperatura exterior y un joystick que se utiliza para marcar el rumbo de la aeronave. Debido a que únicamente se disponen de dos microcontroladores, existe la posibilidad de apagar la aeronave, redefinir el identificador y realizar la simulación con otra aeronave, guardando los datos de la anterior por si se quiere retomar el envío de información desde esa aeronave. Toda la información se envía periódicamente por un puerto serie a la FPGA.

Un segundo microcontrolador simula la torre de control. Su función es la visualización en una página web de la información recibida en la FPGA. El microcontrolador se encuentra conectado a internet mediante una conexión ethernet, accediendo desde el ordenador a una URL previamente definida se visualizarán todas las aeronaves que hayan emitido su información. En la página web se ubicarán las aeronaves en función del identificador en las respectivas posiciones de longitud y latitud, junto con la velocidad y altitud asociadas a cada uno. Mediante una traza continua, se visualizan las posiciones de longitud y latitud por donde ha transitado la aeronave con el objetivo de conocer la ruta que lleva. Fuera del mapa se dispone de un histórico de los mensajes de carácter especial recibidos de cada identificador. En la cabecera de la página web también se visualizarán los números primos y las claves públicas y privadas del equipo receptor. La información histórica, se almacena en una cookie general, de la que cuelgan tantas cookies como aeronaves se hayan detectado. Como mejora, el equipo receptor dispone de la opción de predecir el movimiento de las aeronaves que lleven un tiempo sin recibir información, trazando su rumbo mediante un trazo discontinuo. Toda la información se recibe desde la FPGA por un puerto serie.

Por último, el dispositivo clave y más importante sin duda, la FPGA. La primera función de este dispositivo es generar las claves públicas y privadas del algoritmo RSA con los números primos transmitidos al comienzo desde el microcontrolador. Generadas ambas claves, la FPGA se divide de manera interna y por lo tanto ficticia en dos FPGAS, una para la transmisión y otra para la recepción, la conexión entre las dos particiones se realiza únicamente mediante un cable que conecta el squitter transmitido y el squitter recibido en modulación digital por posición de pulsos (PPM).

A continuación, se analiza cada una de las dos funciones de la FPGA.

La FPGA asociada a la transmisión, detecta la trama recibida desde el primer microcontrolador y de manera automática y en paralelo comienza generando el preámbulo del squitter mientras encripta los datos con la clave pública del receptor recibidos desde el microcontrolador. Una vez, finalizado el preámbulo, transmite los datos encriptados. El squitter generado en la FPGA se encuentra mapeado internamente a un pin de la FPGA y este a un puerto PMOD del kit de desarrollo.

La FPGA asociada a la recepción mapea el puerto de entrada del kit hasta el pin asociado a la recepción del squitter en la señal definida en el entorno de desarrollo hardware. La FPGA identifica el preámbulo y almacena los datos encriptados. Recibido el squitter por completo, desencripta los datos mediante la clave privada del receptor para concluir el proceso enviando dicha información al microcontrolador.

En la siguiente imagen se muestran todos los dispositivos utilizados y las conexiones entre los mismos.

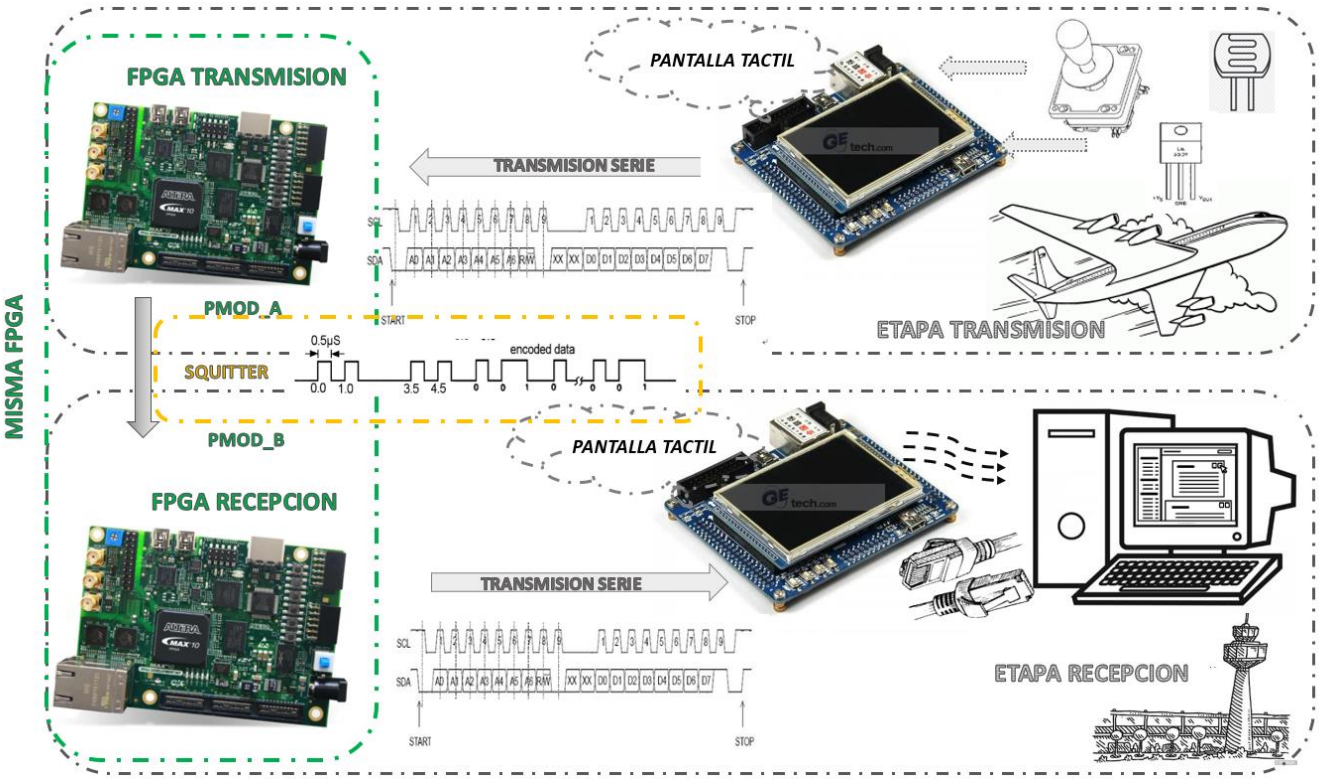


Ilustración 22. Diagrama de bloques genérico y técnico del proyecto.

Por otro lado, en paralelo, se encuentra el diseño hardware de la PCB para la etapa de transmisión y recepción con el objetivo de que esta se realice por radiofrecuencia con una modulación ASK y con portadora centrada en la banda ISM, en lugar de por medio guiado y en modulación banda base.

El punto actual del proyecto, así mismo, marcando el epígrafe relacionado con las líneas futuras y posibles mejoras es la fabricación en PCB de las etapas de transmisión y recepción.

Con el fin de enmarcar mejor el proyecto realizado, se realiza un análisis más a fondo por cada una de los dispositivos desarrollados. En este capítulo no se profundizará, esa parte se deja para los siguientes capítulos, el objetivo es ofrecer al lector una idea de carácter técnico a alto nivel del alcance del proyecto. El orden de análisis refleja el orden cronológico desarrollado, definido al comienzo del trabajo en el plan de tiempos.

3.1 Elección y comunicación con los dispositivos utilizados en el proyecto

En primer lugar, se realizó una búsqueda entre diferentes distribuidores de una FPGA con una tarjeta de desarrollo, se decidió utilizar un kit del distribuidor mousser electronics con una FPGA de la serie MAX 10, la tarjeta de desarrollo en concreto es Intel DK-DEV-10M50A MAX 10. La FPGA otorga una velocidad suficientemente alta para generar y encriptar los datos, tiene pintes mapeados de carácter analógico y un total de doce pines definidos como PMOD, para poder utilizar por el usuario. La comunicación se realiza mediante un Usb-Blaster especifico de grabación, los drivers son ofrecidos por el propio entorno de desarrollo Quartus.

En segundo lugar, se decide el microcontrolador utilizado. Se utiliza el microcontrolador LPC1768 de la familia NXP ARM Cortex-M3. La placa de desarrollo es una mini-DK2. La comunicación se realiza mediante un cable de grabación U-Link. Se decide utilizar esta tarjeta de desarrollo debido a que se ha trabajado previamente en el grado con ella y es familiar para el desarrollador del proyecto.

El proyecto desarrollado en Quartus para la FPGA, se crea desde cero. Para ello es de especial interés conocer la familia de la FPGA para todos los ajustes del proyecto y las hojas de características del fabricante con el fin de ubicar correctamente los pines dedicados y los pines de propósito general (PMOD).

A continuación, se ofrece una captura de simulación del entorno de desarrollo con los ajustes definidos y los pines asignados.

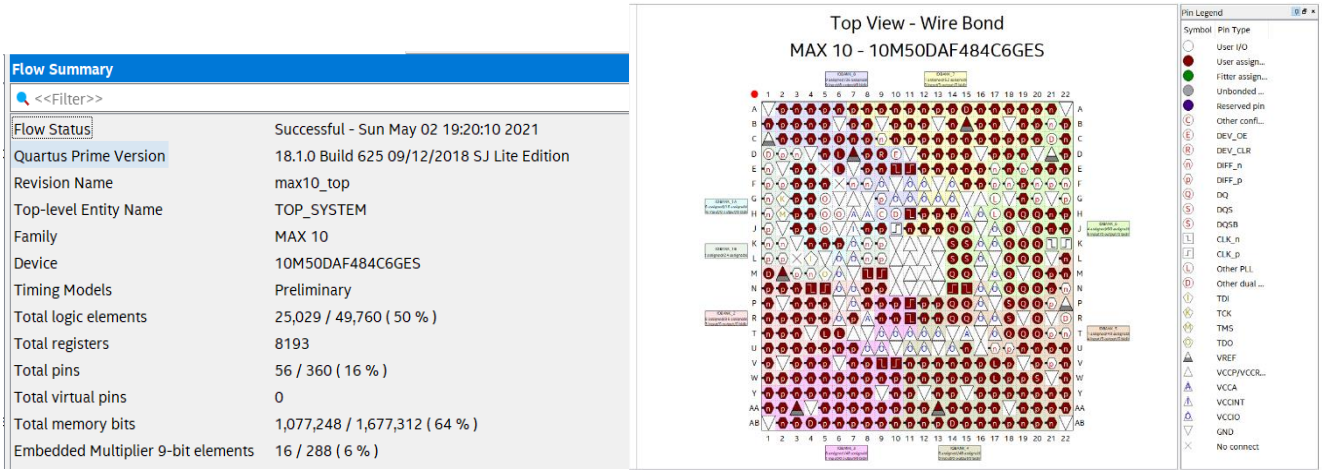


Ilustración 23. Captura del entorno de desarrollo.

Destacar que, en todo el código implementado con el objetivo de reproducir con la mayor fidelidad un código de descripción hardware (basado en electrónica digital) no se ha utilizado ninguna variable, evitando también los bucles de lazo de sentencias iterativas (for y while) y utilizando únicamente procesos y condiciones aumentando notablemente la complejidad del proyecto.

Por otro lado, para los proyectos desarrollados en el microcontrolador, se ha decidido reciclar como punto de partida un proyecto del grado, aprovechando únicamente la estructura del mismo, por lo que no es necesario crearlo desde cero.

3.2 Generación de las claves públicas y privadas del algoritmo RSA en la FPGA.

Las claves públicas y privadas del algoritmo RSA son generadas en la FPGA. Ambas claves son las propias del equipo receptor y son calculadas en función de los dos números primos elegidos al comienzo de la simulación por el usuario (denominados *"p"* y *"q"*). La clave pública será la utilizada por el transmisor para cifrar el mensaje transmitido, mientras que la clave privada, será la utilizada por el receptor para descifrar el mensaje recibido. Para el cálculo de las claves se utilizan operadores multiplicación y división, implementados en el entorno de desarrollo mediante LPMS de los catálogos de los módulos IP. A continuación, se analiza el cálculo de cada una de las claves.

En primer lugar, común a ambas claves, mediante dos LPMs se calcula por un lado el módulo RSA y por otro lado la función φ de Euler del módulo RSA

Para implementar el algoritmo se generan las siguientes dos claves:

Clave pública: (e, n)

Para el cálculo del exponente de cifrado se elige un número *"e"* al azar inferior a *"f"*. El número *"e"* inicial elegido es *"f/3"*, siendo el divisor (numero 3) un numero al azar elegido por el desarrollador debido a que no ser ni un número muy pequeño ni muy grande relativo al número *"f"*. Mediante el algoritmo de Euclides se discrimina si el número *"e"* inicial elegido es coprimo al número *"f"*. En caso afirmativo, se escoge este número *"e"* inicial como el exponente de cifrado de la clave pública, en caso negativo, se incrementa en una unidad el número *"e"* inicial y se vuelve a comprobar si este número es coprimo al número *"f"*, repitiendo este incremento hasta que finalmente lo sean. En el algoritmo, la discriminación que determina si el número *"e"* es coprimo al número *"f"* viene marcada por el valor del ultimo resto, ya que de forma indirecta está relacionado con el m.c.d. de ambos números. En caso de que el resto sea cero, los números no son coprimos, mientras que si, por el contrario, el resto es un uno, los números si lo serán.
Elegido el exponente de cifrado, se almacena en un vector todos los cocientes obtenidos en cada iteración en el algoritmo de Euclides ya que serán utilizados en el cálculo de la clave privada.

Clave privada: (d, n)

Para el cálculo del exponente de descifrado se calcula el número *"d"* resolviendo de manera inversa el algoritmo de Euclides, mediante los cocientes almacenados en un vector previamente en el cálculo de la clave pública. La condición de este número secreto es que el producto *"d"* por *"e"* menos uno, sea dividido exactamente por *"f"*.
La implementación más sencilla y más lenta, seria realizar un bucle de *"f"* iteraciones hasta obtener un resto de valor la unidad. El proyecto utiliza el algoritmo extendido de Euclides, apoyándose en la identidad de Bezout particularizando un máximo común divisor de valor unidad. Mediante estas propiedades se acelera descomunalmente el cálculo de la clave privada a muy pocas iteraciones en un proceso, evitando así realizar un bucle de *"f"* iteraciones. Este algoritmo, parte del resto con valor unidad obtenido en la última división de la clave pública para recorrer el vector de forma inversa aplicando las operaciones inversas y acabar llegando al exponente de descifrado *"d"*.

Finalizados los cálculos de las claves públicas y privadas se almacenan en la FPGA y se activa a nivel alto una señal mediante la cual se habilitará en un futuro la encriptación y desencriptación de los mensajes transmitidos y recibidos.

3.3 Generación y encriptación mediante RSA de squitters en la FPGA.

La FPGA genera la transmisión de un squitter cuando recibe un nuevo dato a transmitir desde el microcontrolador. Los paquetes de datos transmitidos por el puerto serie desde el microcontrolador se encuentran constituidos por tres campos, el primero de ellos el identificador de la aeronave, el segundo el tipo de mensaje y el tercero los datos del mensaje. Recibida la trama entera en la FPGA, de forma automática comienza a generar el preámbulo y en paralelo encripta los tres tipos de mensajes recibidos mediante la clave pública calculada al inicio. Es importante destacar que el preámbulo dura un total de ocho microsegundos, tiempo más que suficiente para poder encriptar los datos.

El squitter generado, replica por completo el tipo de modulación y los tiempos definidos del squitter original generado en IFF. El preámbulo está formado por un bloque de 4 pulsos con una duración a nivel alto constante de valor 0.5 µseg. estando separados 1, 3.5 y 4.5 µseg respectivamente unos de otros. El bloque de datos comienza 8 µseg. después del flanco de subida del primer pulso del preámbulo y consta de un paquete de 56 bits denominada *"short Squitter"*. Los bits de la trama son enviados en intervalos de 1 µseg. de duración, marcando un régimen de transmisión de 1Mbit/seg mediante una modulación binaria de pulsos en posición. En este tipo de modulación dependiendo del valor del bit a transmitir (nivel alto o bajo) se transmite un pulso de 0.5 µseg. de duración, en la primera o segunda mitad del intervalo.

El bloque de datos transmitido, se encuentra constituido por la trama recibida desde el microcontrolador, es decir, el identificador, el tipo de mensaje y los datos del mensaje. Al igual que en los paquetes transmitidos en IFF, los primeros bits marcan el identificador de la aeronave, con el objetivo de solucionar los problemas de los modos de trabajo más primarios. Un ejemplo del squitter transmitido es el siguiente:

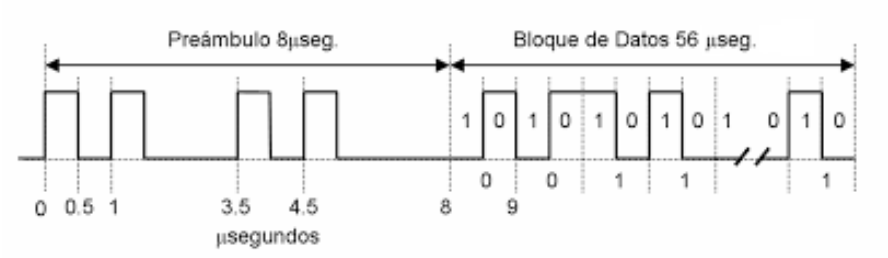


Ilustración 24. Squitter transmitido.

El preámbulo es generado mediante una máquina de estados con un contador asociado, el bloque de datos se transmite en un proceso mediante el cual se replica la modulación PPM comentada previamente al comienzo del capitulo.

RSA utiliza la siguiente ecuación para realizar la encriptación de un mensaje plano.

$$mensaje\_cifrado \equiv mensaje\_plano^{e} \ (mod\ n\ )$$

Para realizar la encriptación se recurre a la aritmética modular, la solución ofrecida de elevar al exponente de cifrado el mensaje plano son números muy elevados (cientos de miles de cifras) siendo imposible trabajar con números de tales dimensiones. Para solucionar este problema se utiliza la exponencial binaria, Este algoritmo es equivalente a descomponer el exponente (convirtiéndolo a base binaria) y calcular la base elevada a números más bajos. Resolviendo así el problema calculando únicamente la base en una secuencia de cuadrados y multiplicaciones, obteniendo como solución el producto de todos estos elementos, excepto los correspondientes a un cero en la representación binaria del exponente inicial.

3.4 recepción y desencriptación mediante RSA de squitters en la FPGA.

El kit de desarrollo enruta el pin de salida de la FPGA hasta un puerto libre de propósito general PMOD, mediante un cable, se une ese puerto de salida a otro puerto de entrada del mismo dispositivo, el cual ha sido enrutado hasta un pin de entrada de la propia FPGA. En resumen, se ha mapeado en la FPGA la salida y la entrada del squitter desde y hasta un puerto de propósito general, asociando posteriormente en el entorno de desarrollo estos pines con señales de entrada y salida.

Mediante una máquina de estados se confirma la detección del preámbulo, al igual que en el campo IFF se deja un tiempo de guarda relativo a un 10% del valor nominal para la anchura de los pulsos, es decir, pulsos de 0.5 μseg. ± 0.05 μseg.

Identificado el preámbulo, la FPGA almacena los datos recibidos secuencialmente en tres vectores diferentes obteniendo los valores encriptados del identificador, el tipo de mensaje y el propio mensaje. La recepción de datos, es coherente al tipo de modulación utilizada (PPM), por lo que únicamente es necesario saber si el nivel alto se encuentra en la primera mitad o en la segunda mitad del intervalo.

Para la parte de la desencriptación el receptor utiliza la clave privada para descifrar el mensaje recibido, es decir, mediante el exponente de descifrado el receptor puede obtener el mensaje en texto plano.

$$mensaje\_cifrado^{d}\ (mod\ n\ )\ \equiv\ mensaje\_plano$$

La lógica seguida es exactamente la misma que en la parte de encriptación, pero modificando el exponente. Se trabaja con números muy elevados, por lo que se utiliza la exponencial binaria para solucionar este problema.
Es importante recordar la identidad que resuelve la encriptación y desencriptación.

$$mensaje\ \equiv\ mensaje^{ed}\ (mod\ n\ )$$

Finalizada la desencriptación del squitter recibido, se transmite por el puerto serie los tres vectores concatenados al microcontrolador para su futura visualización.

PROTOCOLO DE COMUNICACIÓN ENTRE EL MICROCONTROLADOR Y LA FPGA.

La comunicación entre los microcontroladores y la FPGA se realiza via serie, para ello se utilizan dos pines, el primero marca el reloj y el segundo los datos. La transmisión y recepción en el microcontrolador se implementa en las handlers de los temporizadores asociados, mientras que en la FPGA se implementa con un simple proceso. Con el objetivo de blindar mejor el sistema, los datos son escritos en los flancos de subida de reloj y leídos en los flancos de bajada. La velocidad de transmisión es marcada por el dispositivo más lento, siendo en este caso el microcontrolador. La siguiente imagen ilustra cómo se realiza la comunicación entre ambas partes.

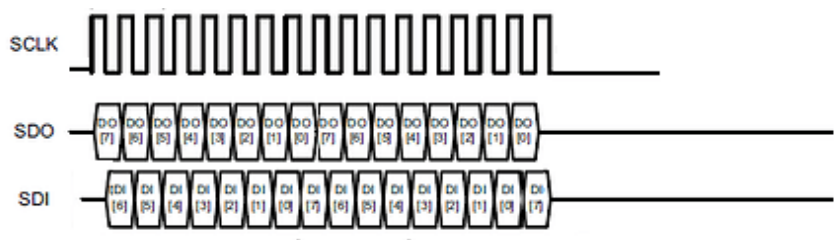


Ilustración 25. Protocolo de comunicación entre el microcontrolador y la FPGA.

3.5 Simulación del avión transmisor en el microcontrolador.

La principal función del microcontrolador es la de mediante una pantalla táctil realizar el interfaz con el usuario que controla la aeronave. La primera ventana permite al usuario elegir los números primos de entre un array previamente inicializado con los que la FPGA calculara las claves públicas y privadas y por consecuente en un futuro encriptara y desencriptará los mensajes. La segunda ventana inicializa la aeronave introduciendo el identificador.

Arrancada la aeronave, de forma automática y con un periodo predeterminado en los registros de los temporizadores, el microcontrolador transmite por el puerto serie a la FPGA los valores actualizados (con su respectivo identificador del tipo de mensaje) de longitud (7679), latitud (7665), velocidad (8669) y altitud (6576). El usuario puede pilotar la aeronave en una ventana específica, permitiendo variar la velocidad, la altitud y la dirección de la misma. El identificador del tipo de mensaje asociado, coincide con el valor ASCII de las dos primeras letras del tipo de mensaje transmitido.

Como mejora implementada, en los pines dedicados a la conversión analógica digital (ADC) del propio microcontrolador se encuentran conectados tres dispositivos, el primero es un sensor de temperatura (para conceder la temperatura exterior), el segundo es un sensor de luminosidad (para configurar automáticamente el brillo de la pantalla) el tercero y por ultimo un joystick (para marcar el rumbo y la velocidad). Es importante destacar que la configuración del ADC del microcontrolador se encuentra en modo *"burst"* permitiendo realizar las lecturas de los pines de entrada sin interrumpir al microcontrolador.

Para el cálculo de la longitud y la latitud en función de la velocidad y la dirección, se aplica trigonometría básica, identificando la velocidad como la hipotenusa del triángulo rectángulo y calculando los dos catetos asociados al incremento de longitud y latitud en función del seno y del coseno del ángulo (definido previamente como la dirección).

El código del microcontrolador implementa una máquina de estados que permite al usuario moverse entre las diferentes pantallas permitiendo enviar mensajes de carácter especial (con su respectivo identificador del tipo de mensaje) resumidos a continuación.

- Mensaje SOS (8379).
- Mensaje del nivel de Gasolina (7183).
- Mensaje de piloto automático activado (8073).
- Mensaje de solicitud de aterrizaje (6584).

El microcontrolador permite la opción de reiniciarse con el objetivo de cambiar el identificador y poder así simular otra aeronave. Tras reiniciarse la aeronave, cambiando el identificador, se almacena la posición donde se encontraba justo antes de reiniciarse junto con todos los mensajes de carácter especial que se han transmitido, por si el usuario quiere retomar la aeronave, conseguir que esta mantenga su estado anterior.

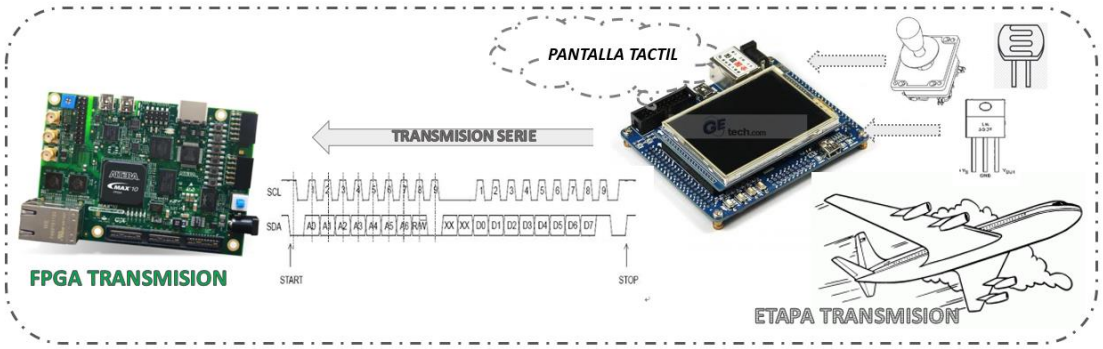


Ilustración 26. Microcontrolador transmisor (Avión)

3.6 Simulación e interfaz de visualización de la torre de control receptor.

Para la parte de la recepción, al igual que en la transmisión se utiliza un microcontrolador de la misma serie que la transmisión. En este caso, es el microcontrolador el que recibe los datos desencriptados desde la FPGA, el protocolo de comunicación al igual que en la transmisión es por puerto serie. Con el objetivo de una visualización de las aeronaves más realista, se conecta a internet el microcontrolador por puerto ethernet.

La velocidad de comunicación entre la FPGA y el microcontrolador es marcada por la velocidad de los periodos de interrupción marcados por los propios temporizadores del microcontrolador. Para poder sincronizar la comunicación se introducen ciclos de espera en la FPGA debido a la alta velocidad de este dispositivo. La página web se desarrolla en HTML, y JavaScript. De forma periódica, la página web solicita al servidor ubicado en el propio microcontrolador los datos actualizados recibidos desde la FPGA.

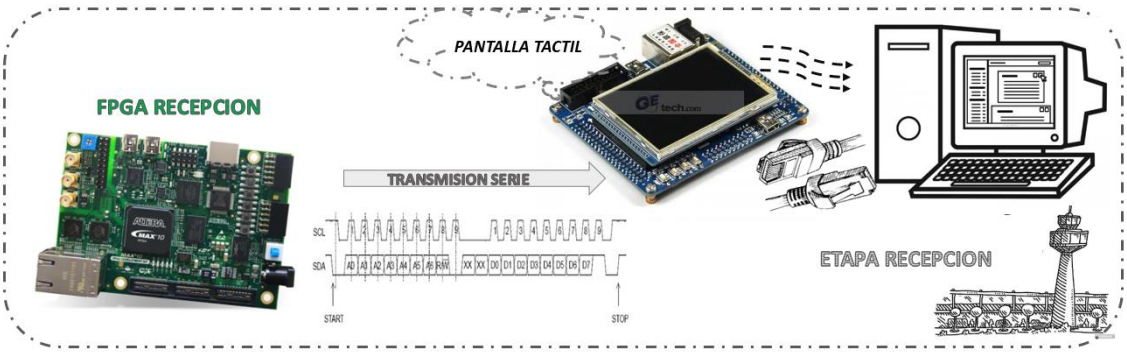


Ilustración 27. Microcontrolador recepción (Torre de control)

DESARROLLO DE LA PAGINA WEB.

La parte estática de esta Web, desarrollada en HTML, únicamente crea las zonas (marcos de la página web) donde serán presentadas cada una de las informaciones enviadas referentes al receptor. El diseño de la página la divide en tres zonas.

- La parte superior para el título de la página.
- La parte derecha en una columna visualizando en primer lugar los valores de los números primos y las claves referentes al algoritmo RSA y a continuación los datos actualizados y ordenados por el identificador de la aeronave más reciente recibido.
- La parte izquierda y central utilizada para ubicar las aeronaves de forma gráfica junto con el rumbo histórico que llevan en función de los valores de longitud y latitud en un mapa de España.

Cargada la parte estática, a través de JavaScript se realiza el proceso de relleno de las zonas variables. Para poder guardar los datos del histórico de los anteriores objetivos que no son el objetivo actual en el momento de la petición de datos se utilizan cookies. Mediante una cookie principal, se conoce la información de qué cookies existen, asociando cada una de estas a una aeronave y guardando en cada una de ellas el histórico de datos de la aeronave con ese identificador.

Mediante la pantalla táctil se puede modificar el formato de visualización de la aeronave en el mapa y habilitar o deshabilitar que la página web pueda realizar una predicción del movimiento de todas las aeronaves recibidas que no hayan actualizado su posición en la última recepción de datos. La predicción se realiza en trazo discontinuo para diferenciar el rumbo real, el cual es trazado con un trazo en línea continua.

3.7 Diseño de PCB de la etapa de transmisión.

Con el fin de poder conseguir una comunicación entre la aeronave y el equipo receptor por radiofrecuencia es necesario realizar el diseño de un circuito impreso en PCB. Para esta parte, es importante destacar que no se ha montado ni verificado, el estudio es únicamente de carácter teórico.

El objetivo principal de la etapa de transmisión es el de conseguir una señal filtrada y amplificada con una modulación por desplazamiento de amplitud (ASK) a partir de una señal moduladora binaria.

A continuación, se muestra una imagen con un diagrama de bloques de la etapa de transmisión con los dispositivos utilizados.

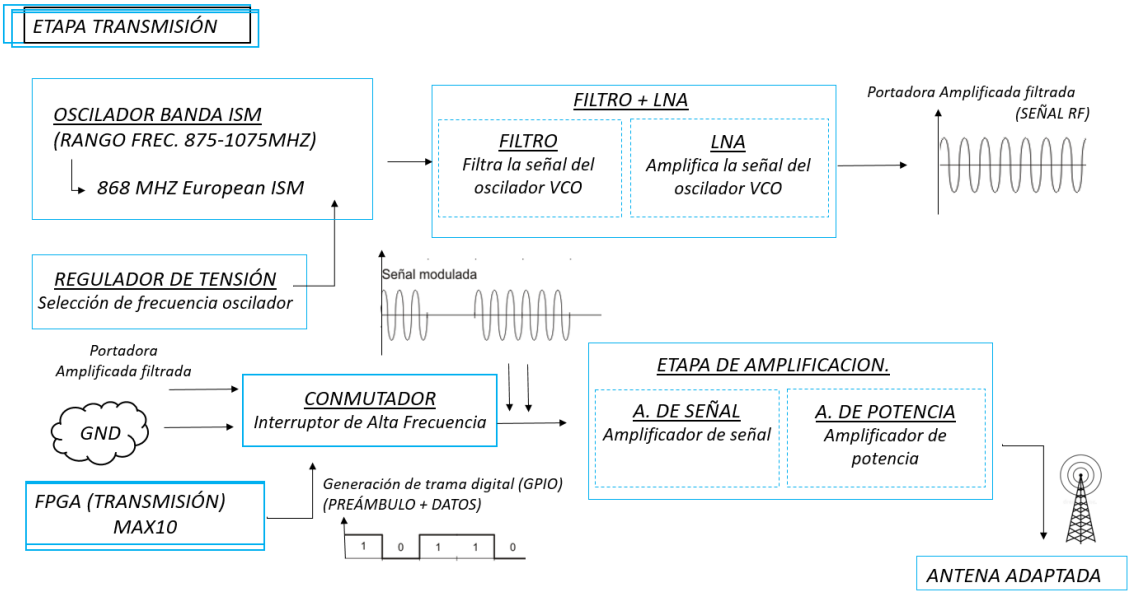


Ilustración 28. Esquemático de la etapa de transmisión.

La etapa de transmisión, utiliza un oscilador VCO para generar una portadora centrada en la banda europea ISM. Este oscilador genera una señal sinusoidal con frecuencia definida en función de la tensión de entrada del pin V_{TUNE}. De la hoja de características ofrecida por el fabricante, se identifica una frecuencia de 868 MHz para una tensión de entrada en el pin V_{TUNE} de valor 1.2v. La señal de control, es generada mediante un regulador de tensión regulado mediante un potenciómetro.

La señal portadora extraída del pin de salida del oscilador es filtrada mediante un filtro paso banda centrado en la banda ISM y amplificada con un amplificador LNA.

Mediante un interruptor de alta frecuencia controlado con la señal moduladora de salida de la FPGA, se habilita el paso de la señal portadora si la señal de control es un nivel alto o por el contrario se fija un nivel bajo constante en la salida del interruptor.

La salida del interruptor se introduce en la etapa de amplificación, el primer módulo amplifica la señal, este amplificador está constituido por un Par Dalington, el segundo módulo amplifica la potencia de la señal mediante un HPA.

Por último, la salida del bloque amplificador es conectada a una antena adaptada mediante la cual se emitirá la señal por el espacio aéreo vía RF.

3.8 Diseño de PCB de la etapa de recepción.

Con la misma finalidad que la etapa de trasmisión se realiza el diseño del circuito impreso de la etapa de recepción. Se destaca también el mismo marco que en la parte de la transmisión.

El objetivo principal de la etapa de recepción es el de conseguir una señal binaria digital que puede ser interpretada por la FPGA a partir de la recepción de una señal ASK.

A continuación, se muestra una imagen con un diagrama de bloques de la etapa de recepción con los dispositivos utilizados.

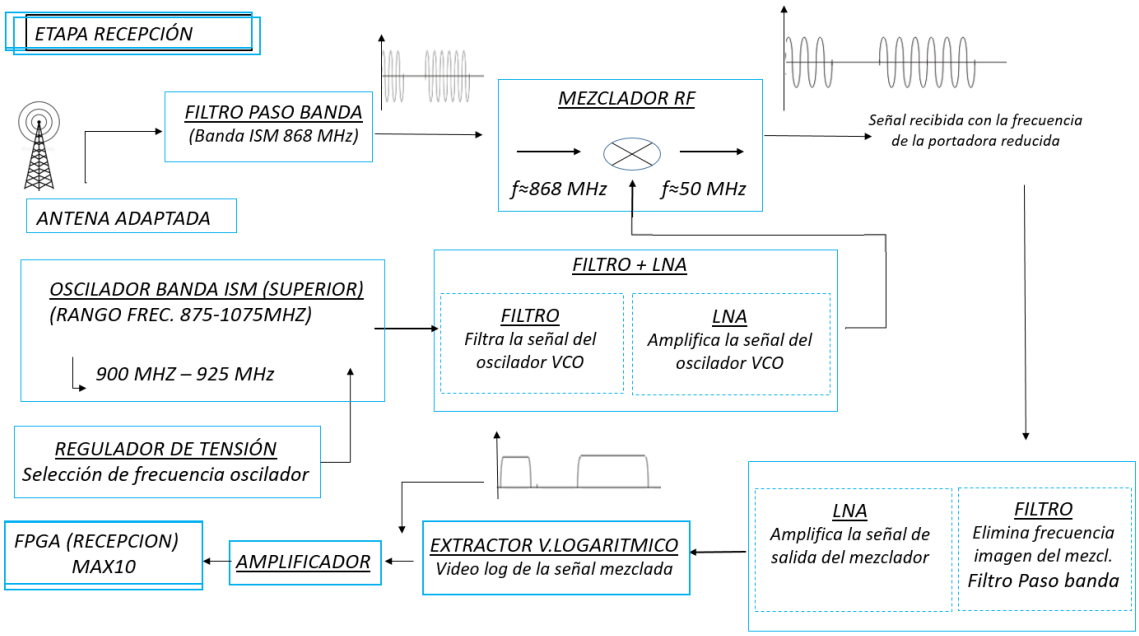


Ilustración 29. Esquemático de la etapa de recepción

La etapa de recepción comienza con un filtro adaptado a la banda ISM europea, el objetivo es obtener únicamente las señales del equipo transmisor. La salida del filtro es la primera señal que se introduce en un mezclador para reducir la frecuencia de la señal portadora a un rango de 50 MHz aproximados. La segunda señal mezclada, proviene de un nuevo oscilador, filtrado y amplificado de la misma serie que el utilizado en la transmisión, pero distinto modelo, debido a que necesitamos mezclar frecuencias ligeramente superiores a la transmitida.

La salida del mezclador se filtra y amplifica nuevamente, el motivo del filtro es eliminar la frecuencia imagen que aparece en altas frecuencias por la mezcla de señales y la amplificación es debida a las perdidas introducidas por el propio dispositivo.

La salida de esta etapa se introduce en un extractor de video logarítmico, consiguiendo una señal de salida proporcional a la potencia de la señal de entrada, de esta forma se consigue extraer la señal binaria de la modulada introducida.

La salida del extractor de video se conecta a un amplificador de amplitud de señal con el objetivo de aumentar a niveles de voltios la salida del extractor, para concluir, la salida del amplificador se conecta al pin de propósito general de la FPGA.

4. Diseño y desarrollo de la etapa de transmisión y recepción encriptada del Squitter desarrollado en la FPGA

Esta etapa constituye toda la parte referente al desarrollo firmware desarrollado en la FPGA. Como se ha comentado previamente, en la introducción del capítulo número tres con el objetivo de abaratar los costes del proyecto desarrollado, se ha utilizado un único dispositivo, implementando la división de las partes de la transmisión y de la recepción en el código firmware.

La siguiente imagen muestra la división desde muy alto nivel de la estructura firmware comentada previamente.

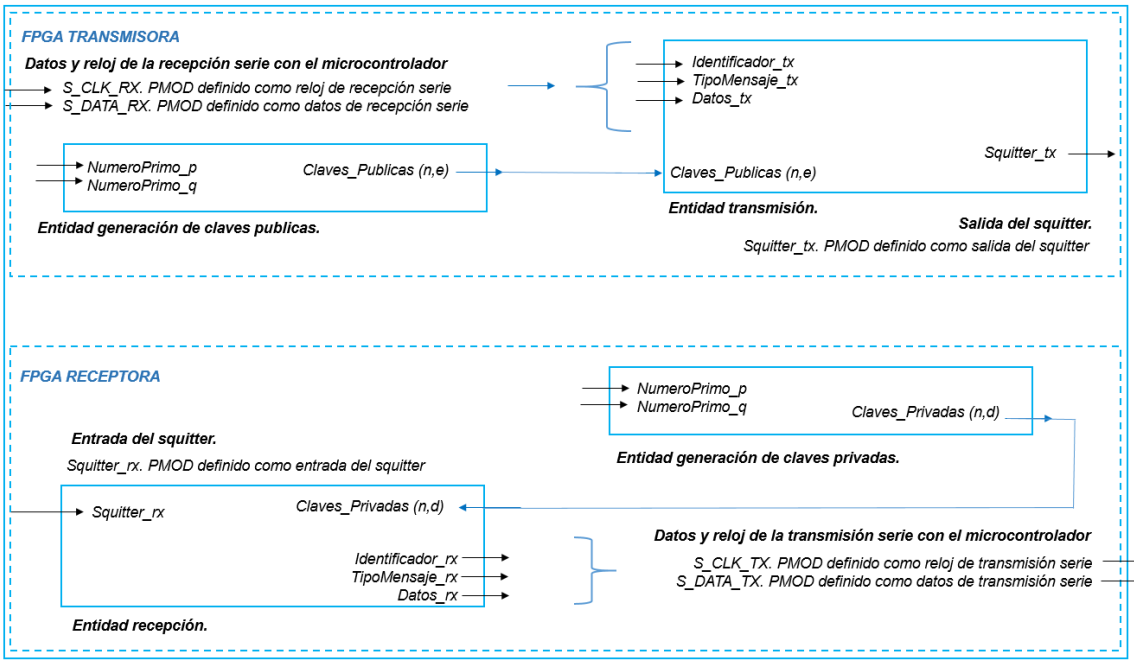


Ilustración 30. Diagrama de bloques del desarrollo firmware de la FPGA.

Tal y como se ha comentado, existe un desarrollo firmware que divide la FPGA de forma virtual en dos FPGAs diferentes, una de ellas para realizar la transmisión y otra para realizar la recepción. Se realiza un breve resumen de ambas entidades.

La FPGA asociada a la transmisión, dispone de un proceso de arbitración, para poder recibir datos desde el microcontrolador por el puerto serie, a su vez, se encuentra constituida por una entidad que genera las claves publicas necesarias para el cifrado a partir de los números primos elegidos por el usuario y otra entidad que genera el squitter con los datos transmitidos desde el microcontrolador cifrándolos con la clave pública calculada en la anterior entidad. El squitter generado en la FPGA se encuentra mapeado internamente a un pin de la FPGA y este a un puerto PMOD del kit de desarrollo.

La FPGA asociada a la recepción, dispone de una entidad que genera la clave privada de descifrado a partir de los números primos indicados previamente, por otro lado, se ha mapea el puerto de entrada del kit hasta el pin asociado a la recepción del squitter en la señal definida en el entorno de desarrollo. Mediante la segunda entidad, asociada a la recepción se recibe el squitter y se desencripta mediante la clave privada calculada en la anterior entidad. Por último, mediante el proceso de arbitración, se transmite el squitter desencriptado al microcontrolador mediante el proceso de arbitración de salida.

El diseño de esta etapa de desarrollo firmware se puede dividir en cuatro grandes entidades nombradas y resumidas a continuación.

- **TOP_SYSTEM.**

Se trata de la entidad superior. Tiene dos funciones asociadas. La primera, consiste en realizar el arbitraje de la comunicación con el microcontrolador tanto para los puertos de entrada como de salida. La segunda función (propia de la entidad superior) consiste en unir mediante señales intermedias las demás entidades principales.

- **GENERACION_CLAVES. (Ubicado en ambas FPGAs)**

La función de esta entidad como su propio nombre indica es generar las claves públicas y privadas del equipo receptor, en función de los números primos enviados desde el microcontrolador.

- **TRANSMISION_SQUITTER. (Ubicado en la FPGA transmisora)**

Esta entidad recibe en tres señales diferentes, los tres mensajes en texto plano que conforman el squitter a transmitir. Según recibe los datos, los introduce en una entidad inferior que se encarga de realizar su encriptación, en paralelo otra entidad genera el preámbulo del squitter. Finalizado el preámbulo, mediante una nueva señal se indica a otra nueva entidad que comience la transmisión de los datos encriptados.

- **RECEPCION_SQUITTER. (Ubicado en la FPGA receptora)**

Por último, esta entidad procesa del squitter recibido los tres mensajes encriptados. Su función es desencriptarlos y ofrecerlos en la salida como texto plano para su futura transmisión al microcontrolador del receptor.

La siguiente imagen ofrece un diagrama de bloques de la entidad superior del proyecto completo, definida como *"TOP_SYSTEM"*.

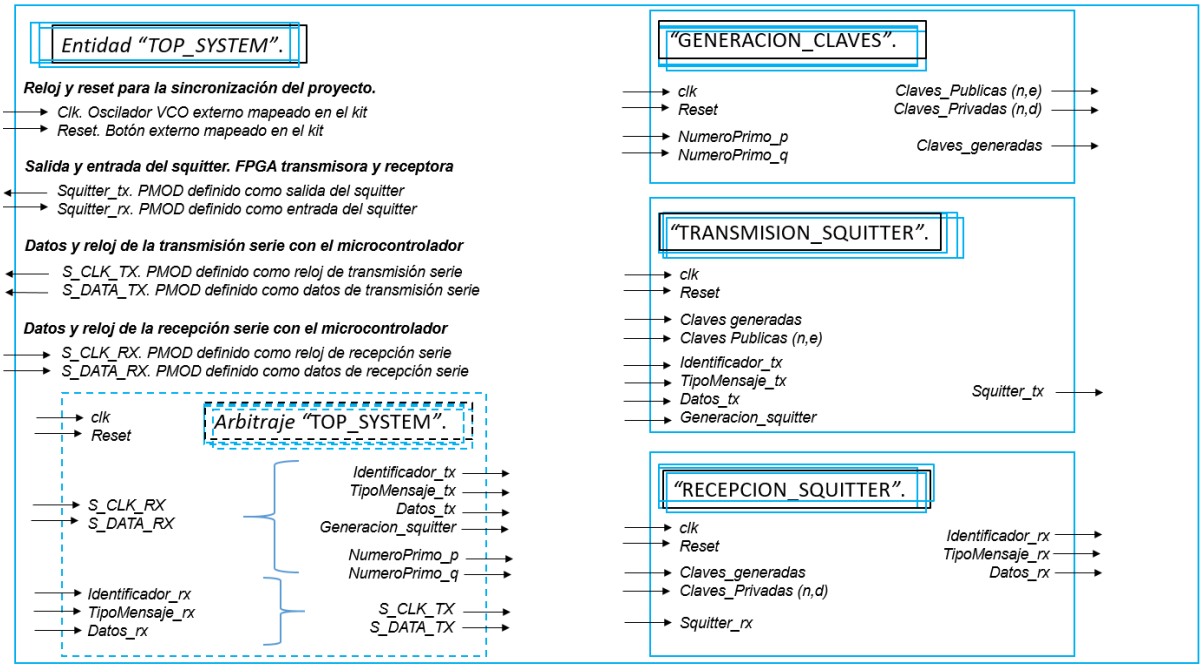


Ilustración 31. Diagrama de bloques de la entidad "TOP_SYSTEM"

En la siguiente parte del capítulo se analiza por separado, de manera exhaustiva cada una de las entidades firmware desarrolladas.

4.1 Entidad Top_System.

Se trata de la entidad superior del proyecto. Tal y como se ha comentado previamente, tiene dos funciones asociadas.

La primera y más importante, asociada a todas las entidades marcadas como *"top level"* de cualquier proyecto firmware, consiste en mapear mediante señales intermedias todas las entidades incluidas.

La segunda función, es implementar mediante procesos síncronos el arbitraje de la comunicación entre la FPGA y el microcontrolador en ambas direcciones, es decir, tanto para los puertos de entrada como de salida, en otras palabras, para implementar la transmisión y la recepción de datos desde la FPGA.

Para las etapas de transmisión y recepción entre ambos dispositivos se implementa una comunicación serie, para ello se transmiten cadenas de datos de bits de forma secuencial en un mismo pin (S_DATA) sincronizado mediante una señal de reloj asociado (S_CLK). Con el objetivo de blindar la comunicación entre ambos dispositivos, las escrituras son realizadas en los flancos de subida, mientras que las lecturas son realizas en los flancos de bajada.

Para la recepción de datos desde el microcontrolador a la FPGA (FPGA transmisora) se implementa el siguiente proceso. Este proceso recibe en la FPGA los números primos y los datos sin encriptar con los que se generara el squitter.

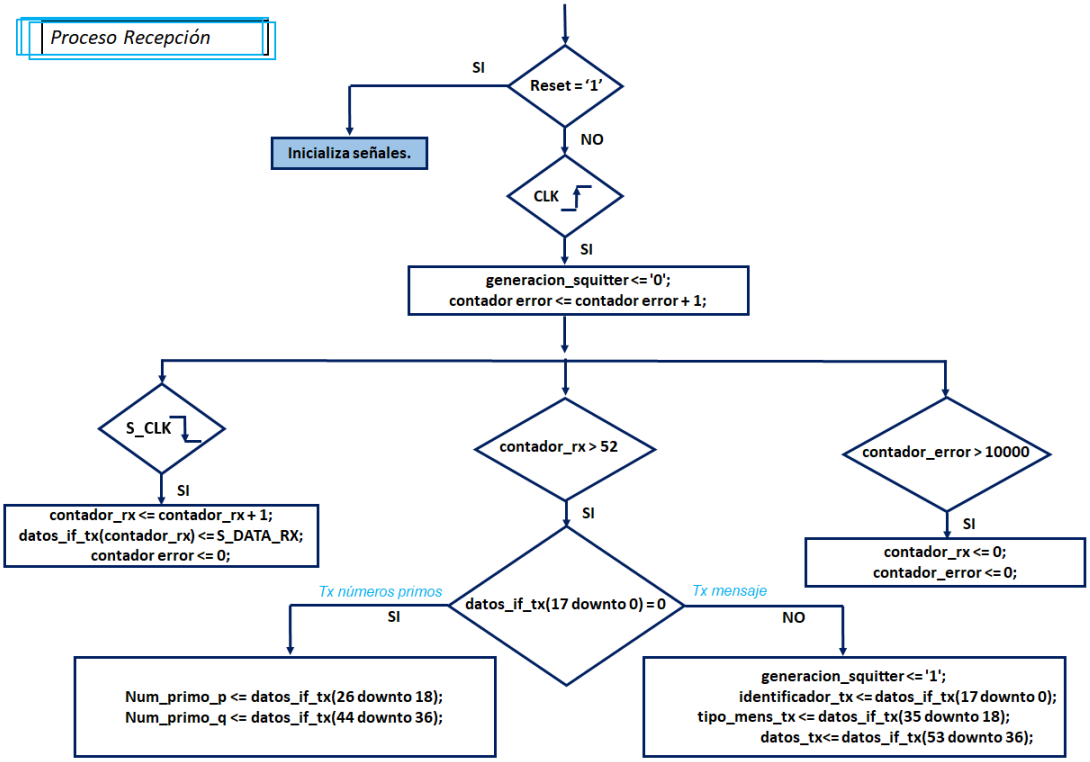


Ilustración 32. Flujograma del proceso de arbitración para la recepción.

Se trata de un proceso síncrono con el flanco de subida del reloj de trabajo de la FPGA, el cual es generado en el oscilador del kit de desarrollo. El proceso está formado por tres sentencias condicionales claves para la recepción correcta de datos (es importante tener presente que al ser la FPGA más rápida que el microprocesador, no hace falta meter ciclos de espera en este proceso). Se puede observar como los números primos, son transmitidos con un valor de identificador nulo.

La primera de ellas detecta el flanco de bajada en la señal de reloj, con el objetivo de almacenar el dato transmitido desde el microcontrolador, la segunda condición confirma la detección de la trama completa, almacenando los datos en las señales mapeadas a los demás bloques, por último, la tercera condición tiene la función de eliminar posibles detecciones erróneas, reiniciando el contador de datos recibidos tras un valor máximo de contador asociado.

Para la transmisión de datos desde la FPGA al microprocesador (FPGA receptora) se genera el siguiente proceso. Este proceso transmite desde la FPGA los números primos con sus respectivas claves públicas y privadas asociados, y los datos desencriptados que se han recibido en el último squitter. Es importante destacar que existen dos procesos sincronizados, uno para la transmisión inicial de los números primos junto con las claves asociadas y otro para la transmisión de los squitter recibidos. Las señales de salida son la suma lógica de las salidas de ambos procesos.

4.1.1 Proceso de transmisión de números primos y claves RSA.

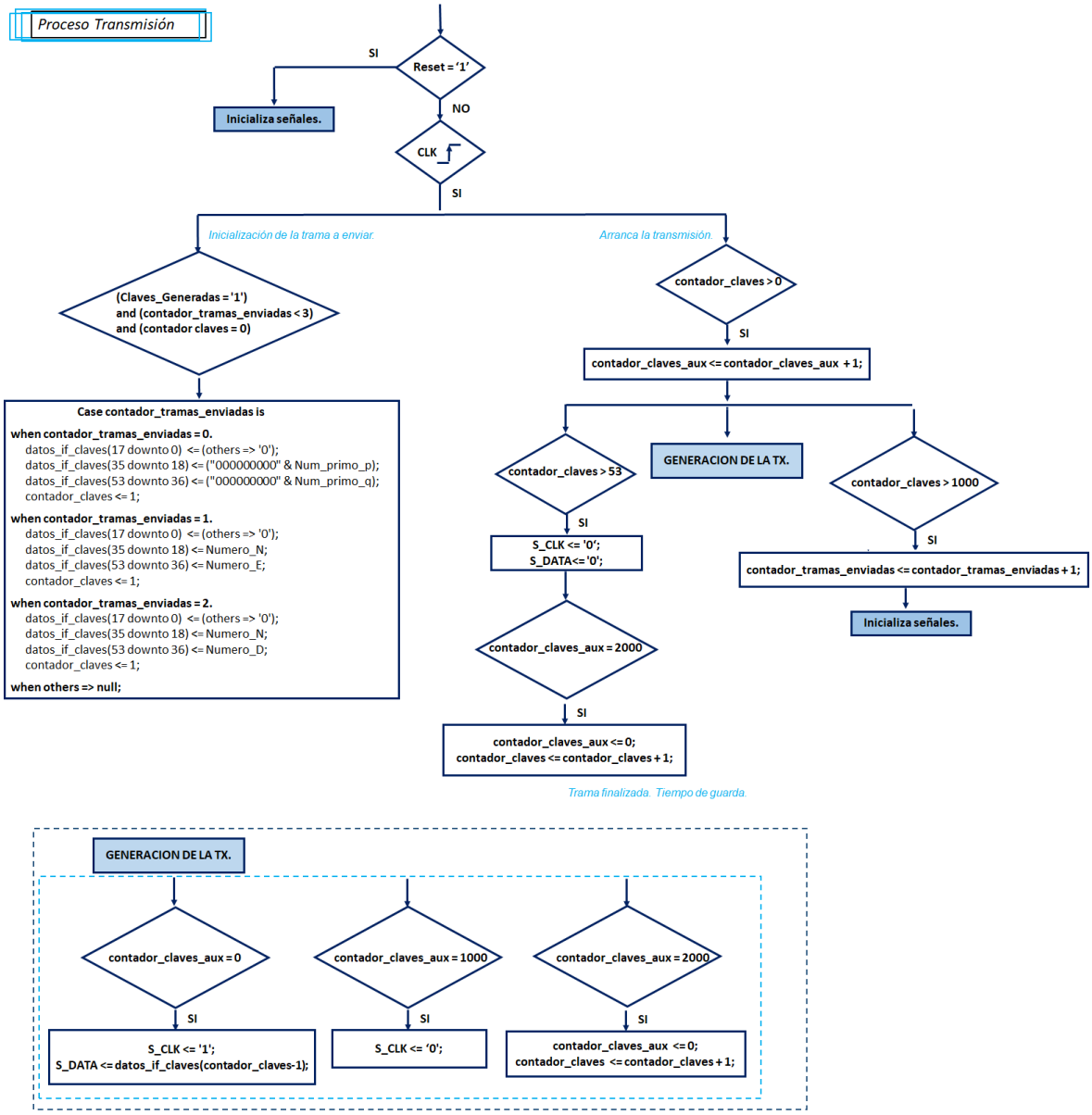


Ilustración 33. Flujograma del proceso de arbitración para la transmisión de datos RSA

4.1.2 Proceso de transmisión de squitters recibidos.

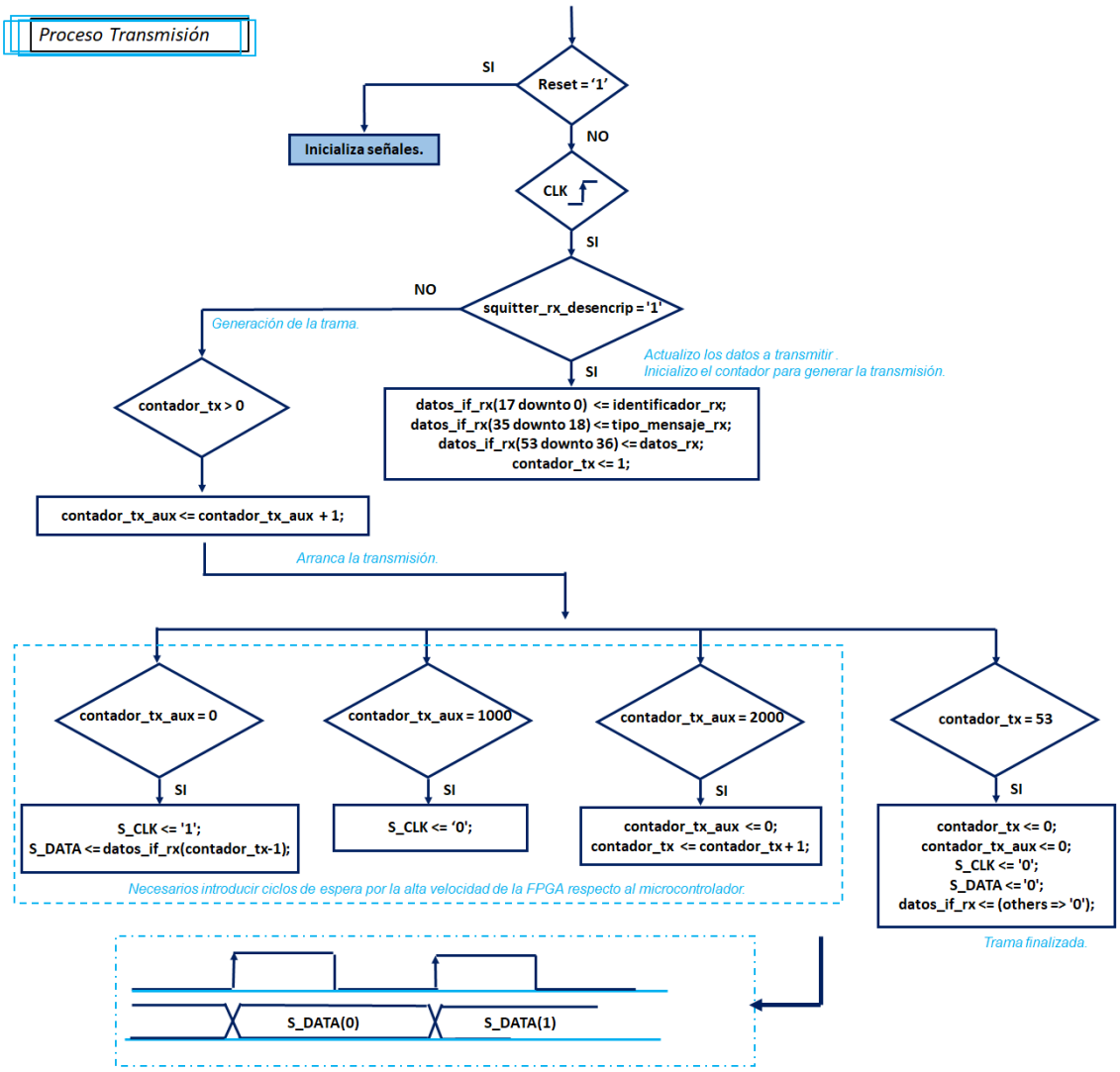


Ilustración 34. Flujograma del proceso de arbitración para la transmisión de squitters.

Al igual que en la transmisión, se trata de un proceso síncrono con el flanco de subida del reloj de trabajo de la FPGA. Los procesos se inicializan con la activación de la señal en forma de flag, en el primer proceso, es las claves generadas, por el contrario, en el segundo proceso hace referencia a la desencriptación de un squitter recibido. Este flag inicializa el contador asociado a la trama y el vector que se va a transmitir con los datos. Con el contador principal inicializado se utiliza un nuevo contador auxiliar con el objetivo de introducir ciclos de esperar y poder conseguir una comunicación entre dispositivos con diferentes velocidades. Mediante este contador auxiliar se modela la comunicación serie asociada a los pines de reloj y datos. Cuando el contador principal llega al valor máximo, la trama se ha enviado por completo y se inicializan todos los contadores.

El primer proceso, tiene un numero de transmisiones fijas (tres transmisiones) marcadas por la instrucción case. Por el contrario, el segundo proceso, se inicializa cada vez que la FPGA desencripta un squitter. para poder identificar los números primos frente a los datos asociados a los squitters recibidos, la transmisión de los números primos lleva asociada un identificador constante a valor cero.

4.2 Entidad Generacion_Claves.

Esta entidad recoge los números primos transmitidos desde el microcontrolador por el puerto serie y calcula las claves públicas y privadas del algoritmo RSA. Es importante recordar que estas claves corresponden al equipo receptor, por lo que el transmisor cifrara el mensaje con la clave pública del receptor, mientras que el receptor descifrara el mensaje con su clave privada.

La siguiente figura muestra el diagrama de bloques de la entidad descrita.

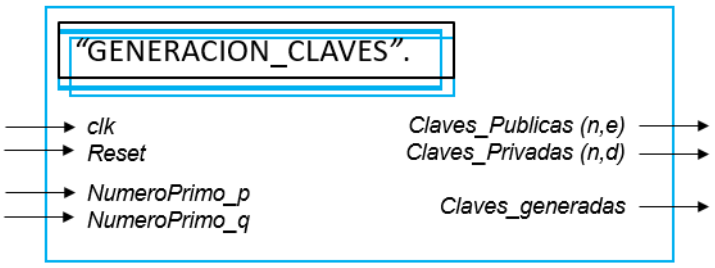


Ilustración 35. Diagrama de bloque de la entidad "Generacion_Claves"

La entidad se encuentra constituida por dos procesos, cada uno de los procesos tiene la función de generar cada una de las exponenciales de cifrado y descifrado. El resto del epígrafe realiza un análisis exhaustivo de cómo son estos procesos y, en conclusión, como se generan cada una de las claves.

La entidad tiene la siguiente estructura firmware, donde se puede apreciar el uso de dos tipos de IP Cores, utilizados para realizar el producto y la división de los diferentes apartados del algoritmo de generación, estas entidades se encuentran prediseñados y desarrolladas, por lo que solo necesitan ser instanciadas y mapeadas tantas veces como se utilicen en la entidad desarrollada. La entidad fusiona los procesos síncronos de generación, con los Cores comentados previamente, con el objetivo de poder generar las claves del algoritmo RSA. La señal de salida referente a las claves generadas, es el producto lógico de las señales referentes a las generaciones de las claves públicas y privadas.

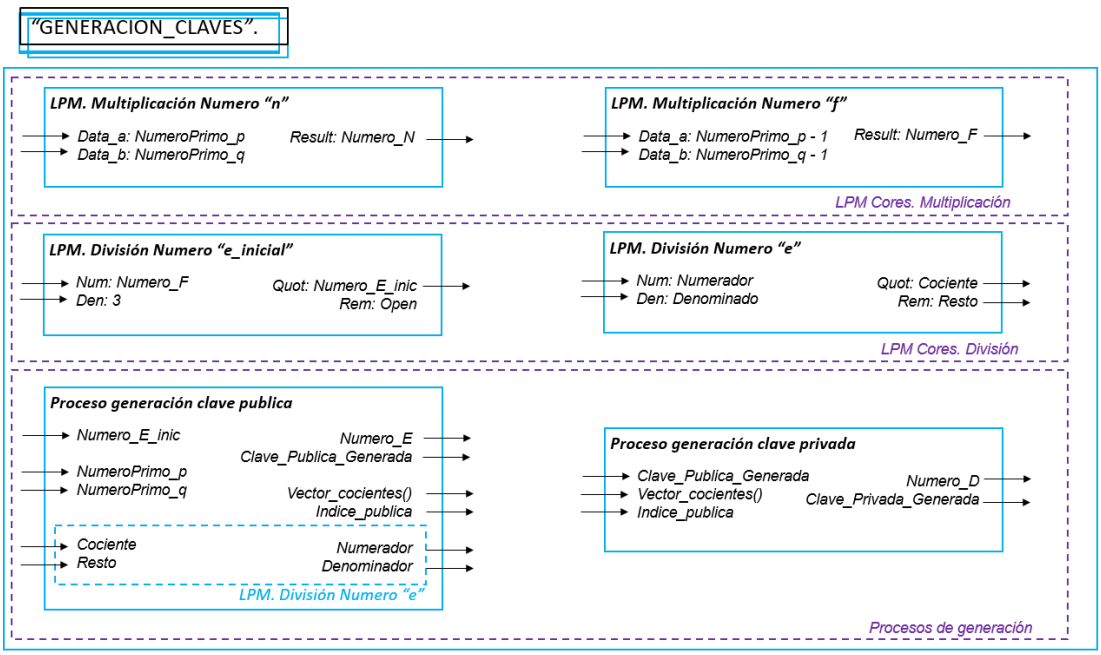


Ilustración 36. Diagrama de bloque interno de la entidad "Generacion_Claves"

Entre los principales problemas encontrados en el desarrollo de la entidad, fue descubrir que el reloj del oscilador externo y en consecuencia la frecuencia de trabajo de los dos procesos era mayor que el tiempo necesario para realizar las operaciones en los Cores. El problema se soluciona, introduciendo ciclos de espera en los procesos, mediante una señal contador.

A continuación, analizamos cada uno de los procesos de generación de forma individual.

4.2.1 Proceso de generación de la clave pública.

La generación del exponente de cifrado de la clave pública se basa en el algoritmo de Euclides, mediante el cual, aplicando una serie de pasos, se obtiene el máximo común divisor de dos números. El algoritmo se postula y resume en la siguiente tabla.

ALGORITMO DE EUCLIDES

- *POSTULADO, DESCRIPCIÓN DEL ALGORITMO.*

Sean a,b enteros no nulos, con a mayor que b. Entonces "mcd(a,b) = mcd(b,r)" donde r es el único 0<r<b tal que existe un entero q con a = bq + r (esto es, que r es el resto de la división de a por b).

Esta proposición indica que es igual de válido calcular el mcd(a,b) que el mcd(b,r), con la ventaja de que r es un entero de menor tamaño que el original a.

- *DESARROLLO DEL ALGORITMO.*

$a = b\ q_1 + r_1\ (0<r_1<b)$
$b = r_1\ q_2 + r_2\ (0<r_2<r_1)$
$r_1 = r_2\ q_3 + r_3\ (0<r_3<r_2)$
....
$r_{k-3} = r_{k-2}\ q_{k-1} + r_{k-1}\ (0<r_{k-1}<r_{k-2})$
$r_{k-2} = r_{k-1}\ q_k\ (r_k=0)$

- *Conclusión de la proposición anterior:*

$mcd\ (a, b) = mcd\ (b, r_1) = mcd\ (r_1, r_2) = ... = mcd\ (r_{k-2}, rk_{-1}) = r_{k-1}$

Como detalle de interés, de cara a la velocidad de cálculo, se puede demostrar que el número de pasos necesarios para calcular el mcd de dos números es menor que 5 veces el número de dígitos del menor de ellos.

Con el marco teórico definido, la memoria avanza en el desarrollo firmware de la entidad. En primer lugar, calcula el módulo RSA, denominado con la letra "n" el cual se obtiene directamente del Core multiplicación. Para el cálculo del exponente de cifrado se elige un número *"e"* al azar inferior a *"f"*. El número *"e"* inicial elegido es *"f/3"*, siendo el divisor un numero al azar elegido por el desarrollador. Mediante el algoritmo de Euclides definido previamente se discrimina si el número *"e"* inicial elegido es coprimo al número *"f"*. En caso afirmativo, se escoge este número *"e"* inicial como el exponente de cifrado de la clave pública, en caso negativo, se incrementa en una unidad el número *"e"* inicial y se vuelve a comprobar si este número es coprimo al número *"f"*, repitiendo este incremento hasta que finalmente lo sean.

En el algoritmo, la discriminación que determina si el número *"e"* es coprimo al número *"f"* viene marcada por el valor del ultimo resto, ya que de forma indirecta está relacionado con el máximo común divisor de ambos números. Es decir, en caso de que el resto sea nulo, los números no son coprimos, mientras que si, por el contrario, el resto es de valor la unidad, los números si lo serán.

Elegido el exponente de cifrado, se almacena en un vector todos los cocientes obtenidos en cada iteración en el algoritmo de Euclides ya que serán utilizados en el cálculo de la clave privada.

Para la generación del exponente de cifrado de la clave pública *(número e)* se desarrolla la teoría explicada previamente generándose en el siguiente proceso síncrono.

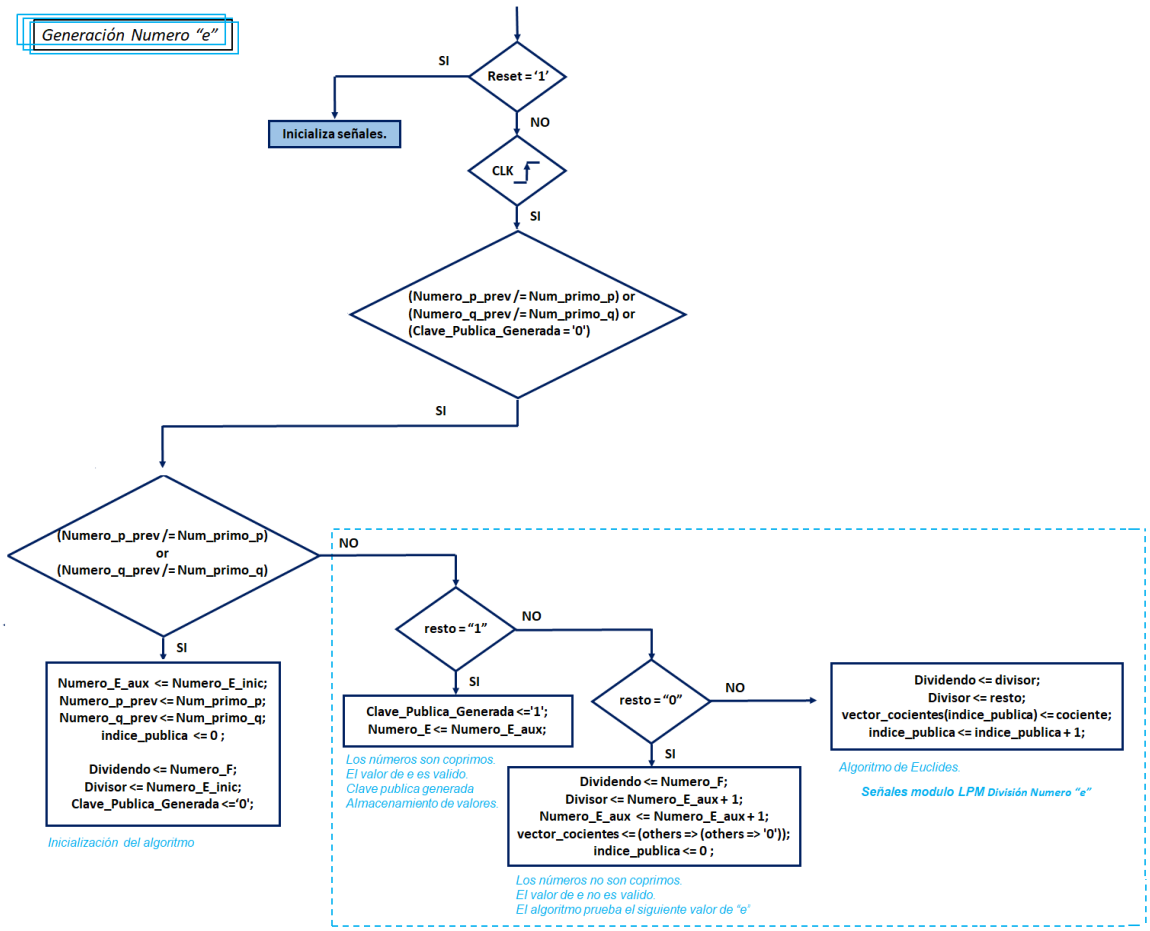


Ilustración 37. Flujograma del proceso de generación de la clave pública.

El diagrama de bloques, ofrece la estructura firmware desarrollada. Como todas las estructuras desarrolladas previamente, se trata de un proceso síncrono con el flanco de subida del reloj de trabajo de la FPGA. El proceso se inicializa con una variación en los valores de las señales de los números primos, para detectar una posible variación en estos números, se realiza la comparación de los valores actuales con los valores anteriores. El proceso avanza repetidamente mientras la clave pública no se encuentre generada (representado como un flag en dicha señal).
En la primera entrada del proceso, se inicializan los valores utilizados en el algoritmo, posteriormente se realiza una continua comparación en cada entrada al proceso con el resto obtenido en el Core que realiza la división del cálculo del número "e", si el resto es la unidad se genera correctamente la clave pública con estos valores, mientras que si el resto es cero se reinicia el algoritmo, probando el siguiente valor del número "e" inicial. En caso de que el resto no sea ninguno de los dos anteriores, se realiza el algoritmo actualizando los valores de dividendo y divisor, almacenando el valor del cociente para el cálculo de la clave privada.

4.2.2 Proceso de generación de la clave privada.

La generación del exponente de descifrado de la clave privada se basa en el algoritmo de Euclides extendido, mediante el cual, además de encontrar un máximo común divisor de dos números enteros, permite expresarlos como la mínima combinación lineal de esos dos números, en el desarrollo, de forma indirecta, se retorna sobre los pasos realizados en el cálculo de la clave pública llegando finalmente a la identidad de Bezout.

La identidad de Bezout comentada en el párrafo anterior se resume en el siguiente punto

IDENTIDAD DE BEZOUT.

- *POSTULADO, DESCRIPCIÓN DE LA ENTIDAD.*

Siendo a y b dos números enteros distintos de cero, y siendo d su máximo común divisor, existen dos enteros x e y de manera que ax + by = d.
Dicho de otra manera. ax + by = d -> ax + by = MCD (a, b)

El algoritmo extendido de Euclides se postula y resume mediante la siguiente secuencia de pasos.

ALGORITMO EXTENDIDO DE EUCLIDES

- *POSTULADO, DESCRIPCIÓN DEL ALGORITMO.*

Es posible probar que el máximo común divisor de dos enteros d = MCD (a, b) se puede escribir siempre como combinación lineal de ambos, es decir, existen dos enteros λ y μ tales que d = λ a + μb.

Particularizando en el algoritmo RSA, es decir, un MCD (e, f) = 1, se puede escribir:

$$af + de = mcd(e; f) = 1$$

- *DESARROLLO DEL ALGORITMO.*

Para encontrar dichos enteros basta recorrer los pasos del algoritmo de Euclides visto antes, al revés. El procedimiento es el siguiente:

En el paso i-ésimo se debe sustituir en el despeje del resto correspondiente a dicho paso; debe hacerse en uno de sus dos sumandos cada vez, y usando para ello la igualdad del paso i−2, excepto en el primero, en que se usa la igualdad del paso inmediatamente anterior.

- *Conclusión de la proposición anterior:*

Inicialización. (Primera vez)

Numero = - Cociente$_K$
Num_Anterior = 1.

Algoritmo. (Mismo número de iteraciones que las realizadas en la clave pública)

Numero = Num_Anterior – Numero · Cociente$_{K-1}$
Num_Anterior = Numero

Al igual que en la clave pública, con el marco teórico definido, la memoria avanza en el desarrollo firmware de la entidad. El módulo RSA de la clave privada coincide con el de la clave pública. Para el cálculo del exponente de descifrado se calcula el número *"d"* resolviendo el algoritmo extendido de Euclides, consiguiendo la mínima combinación lineal entre ambos exponentes de cifrado y descifrado, mediante el vector de cocientes almacenados en un vector previamente en el cálculo de la clave pública.

Para la generación del exponente de descifrado de la clave privada *(número d)* se desarrolla la teoría explicada previamente en el siguiente proceso síncrono.

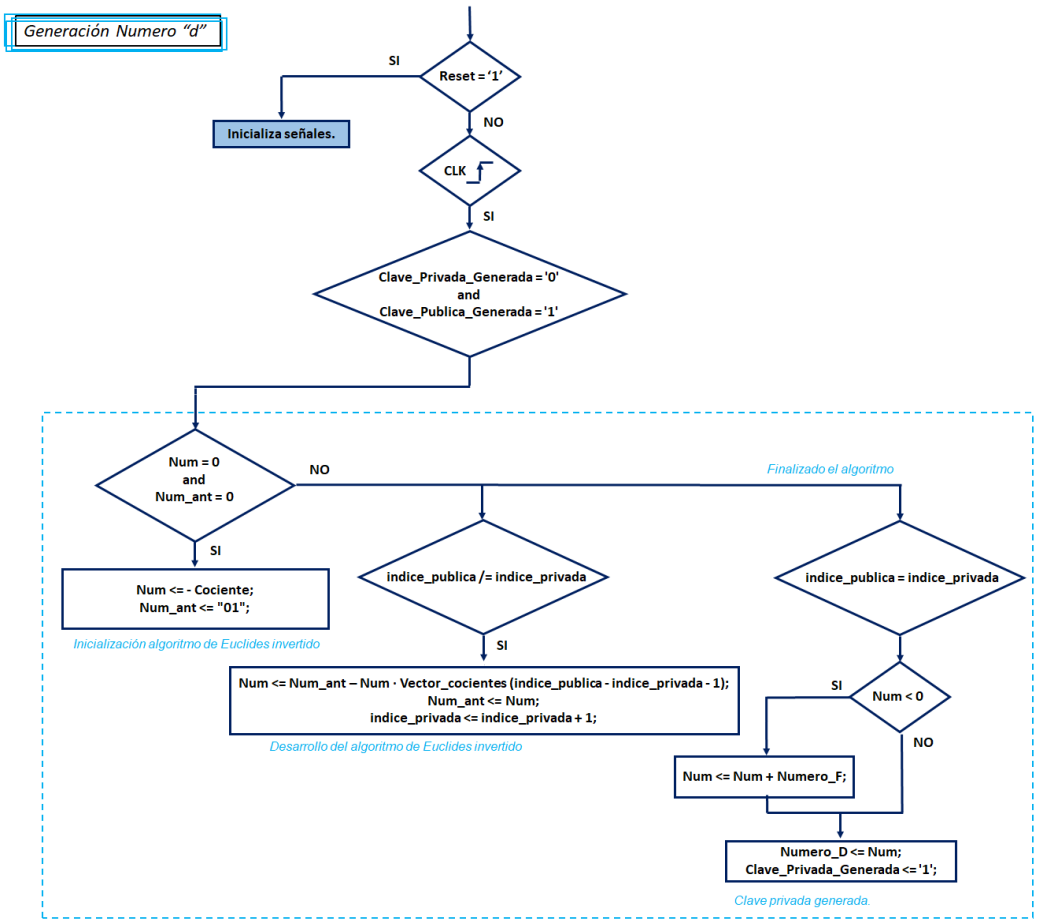


Ilustración 38. Diagrama de bloques del proceso de generación de la clave privada.

El algoritmo comienza cuando se activa el flag referente a la correcta generación de la clave pública, es decir, según finaliza el proceso anterior, arranca este, permaneciendo activo en ese estado, hasta que se genera la señal referente a la generación de la clave privada. Tal y como se ha explicado en el cuadro resumen anterior, el proceso trabaja con un número y ese mismo número, pero en su valor anterior.

La primera entrada en el proceso (con ambos valores inicializados a cero por el reset) inicializa los valores de ambos números. Las siguientes entradas en el proceso, implementa el algoritmo extendido, entrando en este proceso de cálculo de la clave privada, las mismas iteraciones que en el proceso relacionado con la clave pública. Una vez alcanzado el número máximo de entradas al proceso, se da por terminado el algoritmo, por lo que únicamente se necesitaría saber si el valor obtenido es positivo o negativo. En caso de que sea positivo, se asigna directamente, mientras que, si por el contrario el número es negativo, se le suma el número "f" antes de realizar la asignación.

En el Anexo III se ofrece un ejemplo ilustrativo para una mejor comprensión del lector.

4.3 Entidad Transmision_Squitter.

Esta entidad ubicada en la FPGA transmisora, tiene la función de generar y transmitir el squitter. La entidad recibe, por un lado, las claves públicas con las que encriptara el mensaje y, por otro lado, los tres mensajes que se transmitirán. La entidad se puede dividir en dos generaciones, la primera entidad genera el preámbulo, la segunda entidad genera los datos (como es de esperar, la segunda entidad actúa a continuación de la primera).

La siguiente figura muestra el diagrama de bloques de la entidad descrita.

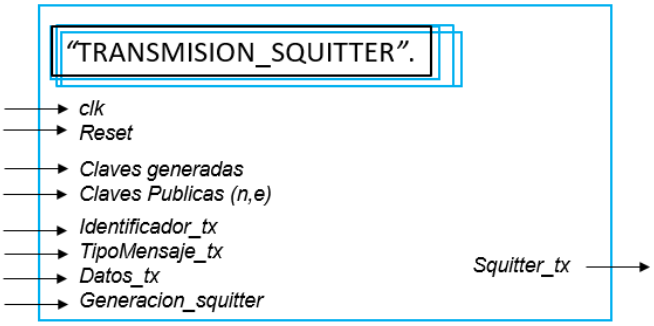


Ilustración 39. Diagrama de bloque de la entidad "transmision_squitter"

Existen tres conjuntos de señales de entrada definidas en la entidad transmisora. En primer lugar, se introducen las señales dedicadas de reloj y reset, necesarias para la correcta sincronización del proyecto, a continuación, se introducen las claves públicas del equipo receptor, necesarias para el cifrado de datos, en último lugar, se introducen los tres datos provenientes del microcontrolador, junto con una señal que marca de manera externa cuando se debe realizar la generación del squitter.

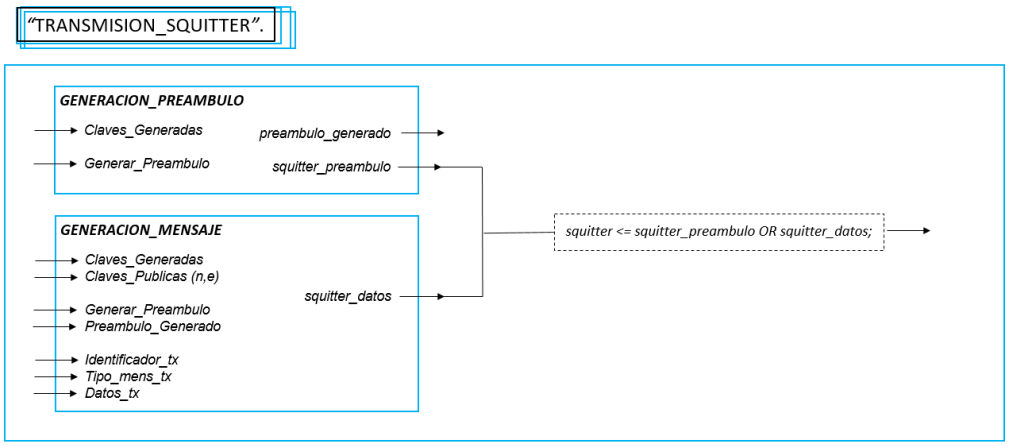


Ilustración 40. Diagrama de bloque interno de la entidad "transmision_squitter"

La entidad se encuentra constituida por dos entidades generadoras, mediante las cuales se genera el preámbulo y los datos modulados en PPM. La salida de la entidad, definida como *"Squitter_tx"* se obtiene como la suma lógica, de las salidas de ambas entidades generadoras. La generación del mensaje, comienza con la activación por flanco de la señal *"Generacion_squitter"*, activada en el proceso relacionado con la comunicación serie entre el microcontrolador y la FPGA. La interconexión entre ambas entidades generadoras, se lleva a cabo mediante la señal *"Preambulo_Generado"* esta señal define el final de la generación del preámbulo, asociado a la primera entidad, y, por consiguiente, marca el inicio de la generación de datos, asociado a la segunda entidad. El epígrafe, avanza en el marco técnico del análisis de ambas entidades generadoras.

4.3.1 Entidad Generacion_Preambulo.

Esta entidad se genera mediante un proceso síncrono formado principalmente por una máquina de estados, la transición entre estados del proceso, comienza a ejecutarse con la activación de la señal *"Generacion_squitter"*, mediante la cual se indica que se ha recibido una trama del microcontrolador y en consecuencia que la FPGA debe generar un squitter, comenzando por la generación del preámbulo. La transición entre estados se marca con el flag asociado al fin de cuenta de un contador externo.

La lógica seguida en el proceso, se emula en el siguiente diagrama de bloques, donde se puede observar como la máquina de estados se encuentra dentro del propio proceso síncrono, evaluándola en cada flanco de reloj ascendente de la señal de reloj.
Por otro lado, se encuentra la entidad del contador. Entre las entradas de la entidad, se pueden destacar las señales de reset, enable y el valor de fin de cuenta, mientras que, en las salidas, solo se mapea el valor de fin de cuenta del contador.

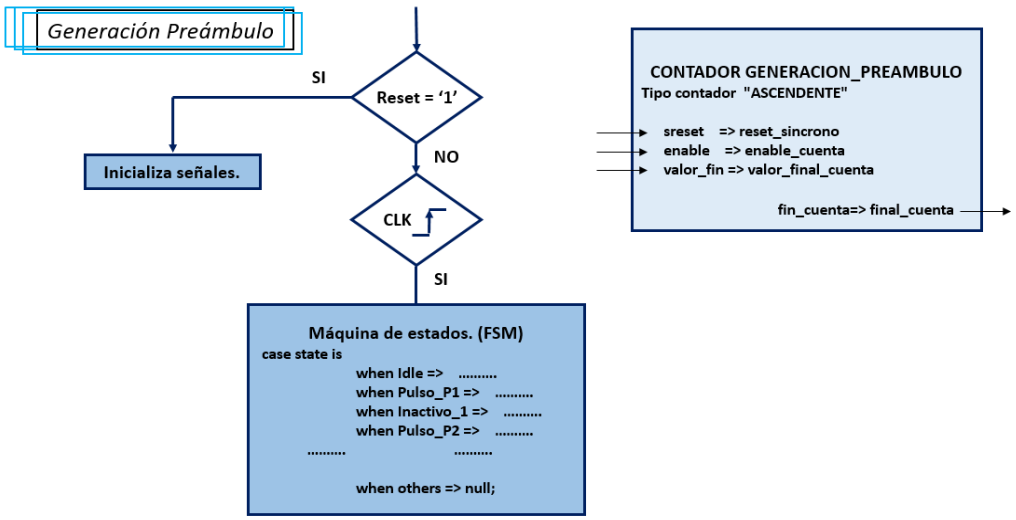


Ilustración 41. Flujograma del proceso de generación del preámbulo.

Conocida la frecuencia de trabajo del oscilador externo *(f_{CLK} = 50 MHz, T_{CLK} = 0.02 µs)* ubicado en el kit de desarrollo, el cual es mapeado al pin dedicado como reloj de la FPGA y los tiempos marcados por los protocolos IFF para la transmisión de squitters, es fácil identificar y marcar los valores de cuenta asociados al temporizador de cada estado.

La máquina de estados desarrollada para generar el preámbulo se ilustra en la siguiente imagen. La máquina de estados parte de un estado de reposo, esperando el nivel alto en la señal *"Generar_Preambulo"*, en caso de que previamente se hayan generado las claves, el proceso avanza entre los diferentes estados que generan los cambios de nivel alto y nivel bajo en la señal de salida, creando indirectamente el preámbulo de la comunicación. Como se ha comentado previamente, la transición entre estados se realiza cuando el temporizador alcanza el valor final de la cuenta propia de cada estado, estos valores se encuentran resumidos en la tabla anterior

En la siguiente página, se resume en una tabla el valor asociado a los valores de las señales final de cuenta del temporizador de cada estado de la FSM.

Tabla resumen de tiempos y números de ciclos de estados del squitter.

Nombre (constante). Definición	*Duración tiempo.*	*Numero de ciclos.*
DURACION_PULSO	*0.5 µs*	*25 Ciclos*
DURACION_INACTIVO_P1_P2	*0.5 µs*	*25 Ciclos*
DURACION_INACTIVO_P2_P3	*2 µs*	*100 Ciclos*
DURACION_INACTIVO_P3_P4	*0.5 µs*	*25 Ciclos*
DURACION_INACTIVO_P4_DAT	*3 µs*	*150 Ciclos*
DURACION_DATOS	*56 µs*	*2800 Ciclos*

Tabla 6. Tabla resumen de tiempos asociados a los estados de la generación del preámbulo.

La máquina de estados desarrollada en el proceso síncrono, se esboza en el siguiente diagrama de bloques

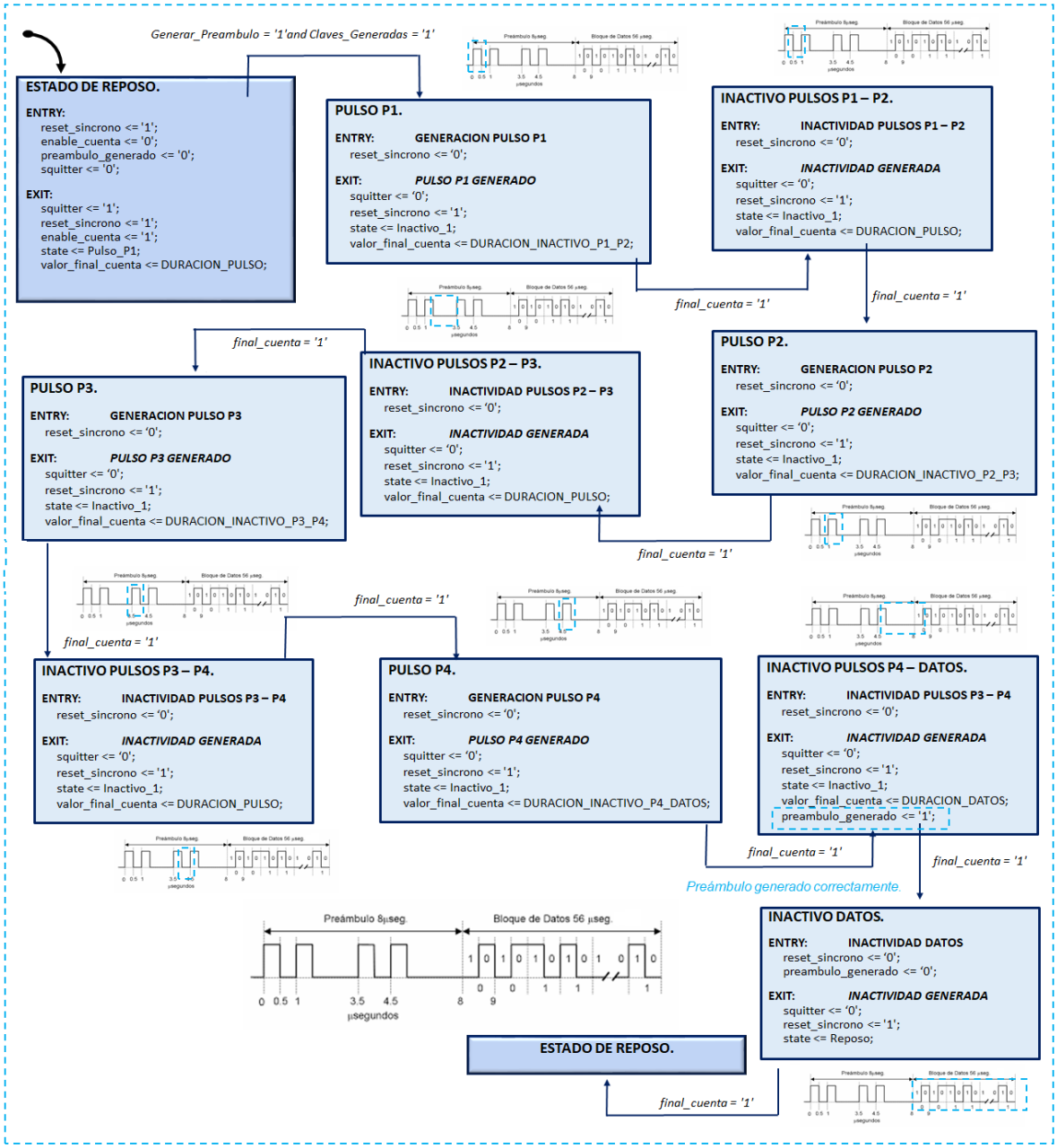


Ilustración 42. Diagrama de bloques de la máquina de estados de generación del preámbulo.

4.3.2 Entidad Generacion_Mensaje.

Esta entidad tiene múltiples funciones, en primer lugar, en paralelo a la generación del preámbulo, comienza realizando la encriptación por separado de los tres mensajes que forman el squitter, con los paquetes de datos encriptados, la entidad avanza realizando la concatenación en una única señal, de las tres señales de salida de los tres bloques encriptadores. Concluida la generación del preámbulo, esta entidad transmite el vector generado en la concatenación con una modulación PPM. La encriptación comienza mediante la activación de la señal *"Generacion_squitter"*, realizando la encriptación en el tiempo en el que se transmite el preámbulo. Mientras que, la transmisión de los datos comienza mediante la activación de la señal de salida generada en la entidad anterior (Generacion_Preambulo) denominada *"Preambulo_Generado"*.

Para la encriptación del mensaje se utilizan las claves públicas del receptor, en esta entidad se introducen las señales de sincronismo, la señal mediante la cual se indica el comienzo de la encriptación, el mensaje que se quiere cifrar y las respectivas claves de cifrado. Como es de esperar, al tener tres mensajes que encriptar, se instancia tres veces esta entidad.

El siguiente diagrama de bloques esboza el mapeado entre las diferentes entidades que se han comentado previamente.

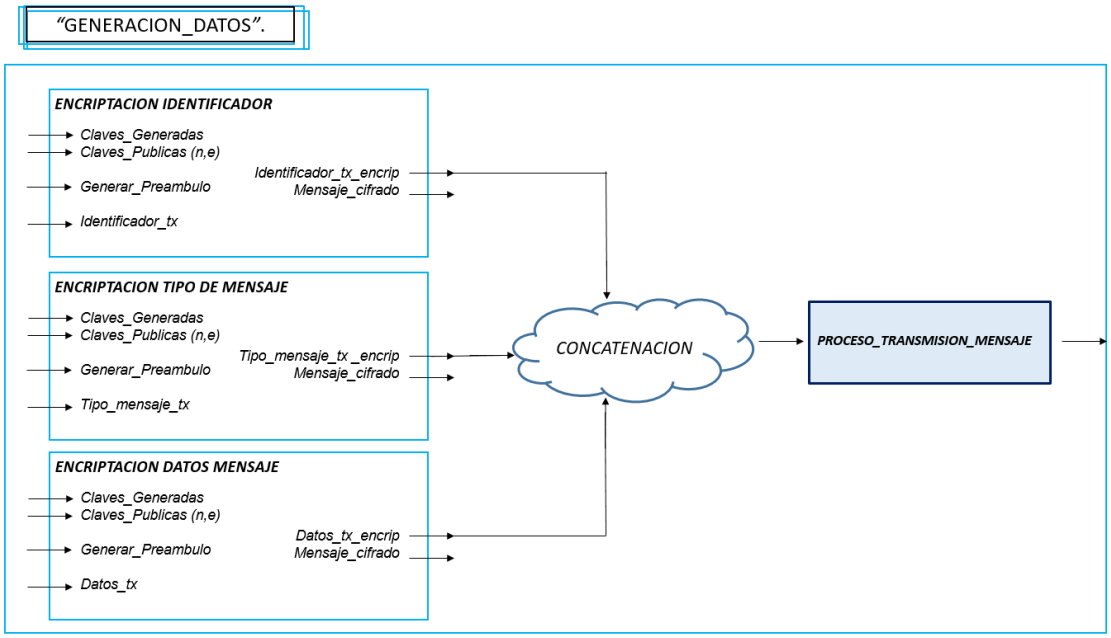


Ilustración 43. Diagrama de bloque interno de la entidad "Generacion_Mensaje"

Junto con el mapeado de las entidades de encriptación en esta entidad destaca el proceso de transmisión del mensaje Squitter en una modulación PPM. Es importante recordar que, en este tipo de modulación, dependiendo del valor del bit a transmitir (nivel alto o bajo) se transmite un pulso de 0.5 µseg. de duración, en la primera o segunda mitad del intervalo.

Trabajando con el reloj del oscilador *(f_{CLK} = 50 MHz, T_{CLK} = 0.02 µs)* y unos pulsos de longitud 0.5 µseg. de duración aparece la necesidad de un contador auxiliar que introduzca ciclos de esperar por valor de veinticinco, con el fin de conseguir las anchuras deseadas.

4.3.2.1 Proceso de transmisión del mensaje (Squitter).

En el siguiente diagrama de bloques se representa el proceso que genera la transmisión de datos en modulación PPM, transmitiendo la señal obtenida de la concatenación de los tres mensajes encriptados.

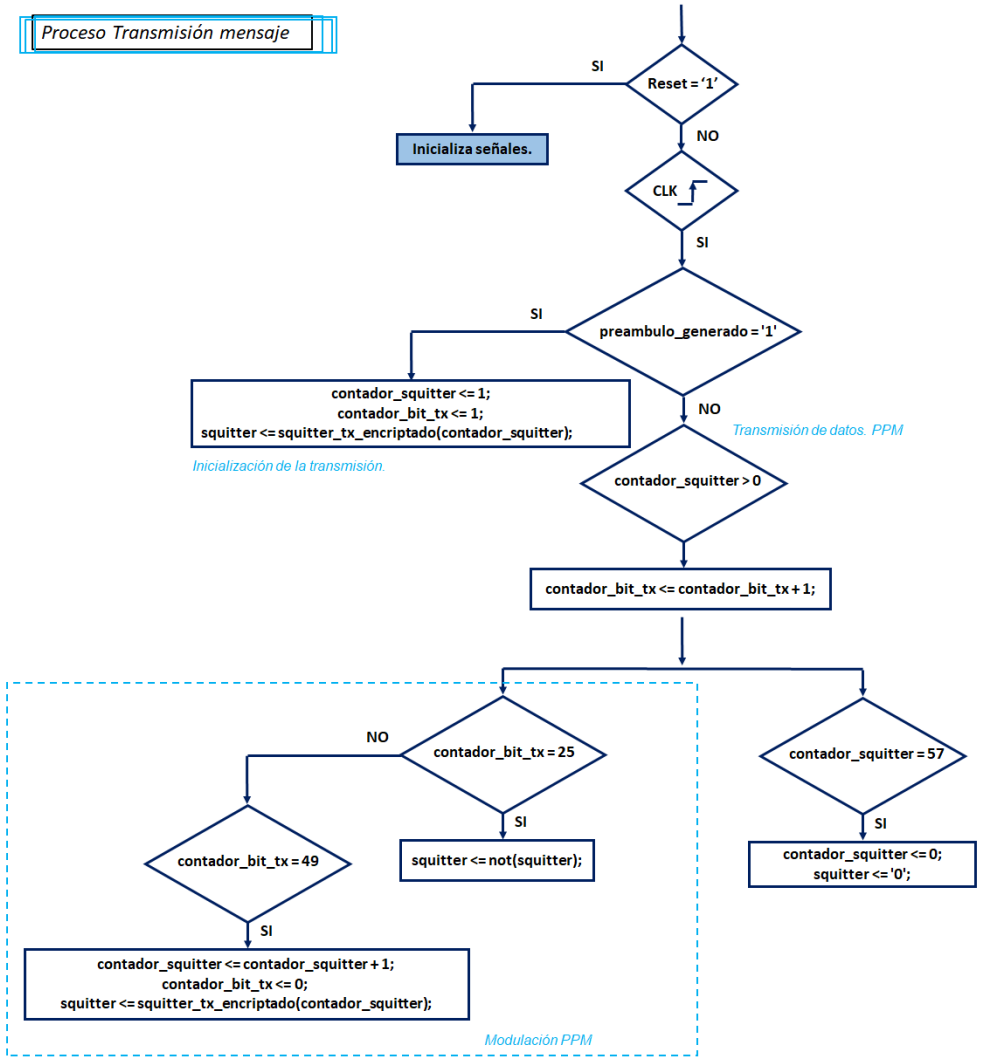


Ilustración 44. Flujograma del proceso de transmisión squitter modulado en PPM.

El formato del squitter encriptado transmitido es el siguiente.

Estructura del squitter transmitido.

squitter_tx_encriptado <= datos_encrip & tip_mensaje_encrip & identificador_encrip & "111"

TIPO DE MENSAJE
3 bits a nivel alto constantes al principio de la trama
18 bits a continuación: identificador de aeronave.
18 bits a continuación: tipo de mensaje.
18 bits a continuación: mensaje enviado. *Total bits transmitidos = 56*

En la siguiente parte del epígrafe, se analiza la entidad que realiza la encriptación del mensaje introducido.

4.3.2.2 Entidad Encriptacion_Mensaje.

La estructura de la entidad que realiza la encriptación del mensaje es la siguiente, es importante destacar que como se ha comentado previamente, la entidad se instancia tres veces, debido a que se encriptan tres mensajes. El nombre genérico de la entidad es *"Encriptacion_Mensaje"* para posteriormente, como se ha explicado previamente, instanciar tres veces esta misma entidad con los tres nombres referenciados al paquete de datos mapeado en la entrada y por lo tanto encriptado.

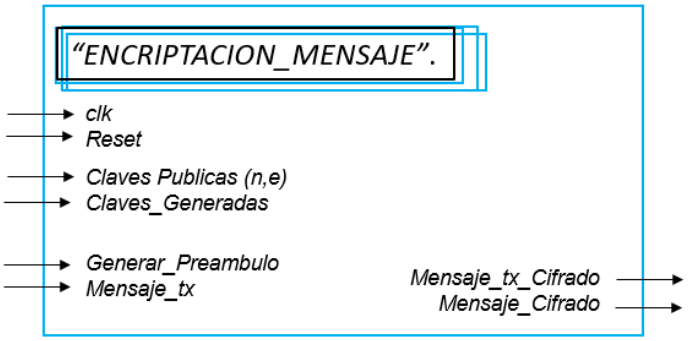


Ilustración 45. Diagrama de bloque de la entidad "Encriptacion_Mensaje"

La encriptación de los mensajes aplicando la metodología de encriptación RSA se basa en el cálculo de aritmética modular. La identidad de aritmética modular se puede resumir en el siguiente cuadro.

$$a^n \ (mod\ m\ ) = resto(\frac{a^n}{m})$$

Los problemas que presentan estos cálculos de encriptación son los valores de los numeradores, pudiendo llegar a ser números de cientos de miles de cifras decimales. La complejidad y seguridad del algoritmo reside en la complejidad de trabajar con estos números tan elevados, la clave del sistema criptográfico son las funciones matemáticas unidireccionales cuyo cálculo directo es sencillo, pero muy complejo a la inversa, ya que requiere muchas operaciones computacionales. Para resolver este problema, con el objetivo de trabajar con números más pequeños se utilizan las propiedades de este tipo de aritmética, en concreto se estudia la exponenciación binaria, también denominado, método de los cuadrados repetitivos.

A continuación, se resumen las propiedades más importantes de este tipo de aritmética.

ARITMETICA MODULAR. EXPONENCIACIÓN BINARIA

- *POSTULADO, DESCRIPCIÓN DEL ALGORITMO.*

  La idea consiste en obtener la representación del exponente n en base binaria, mediante las propiedades de las potencias hallar de forma sucesiva los cuadrados que aparecen en el exponente, para después multiplicar módulo m las potencias a^{2i} correspondientes a los dígitos binarios que han aparecido en la *exponencial binaria.*

  Las tres propiedades de las potencias fundamentales son:

  Propiedad 1: $x^1 = x$
  Propiedad 2: $x^{a+b} = x^a x^b$
  Propiedad 3: $x^{a\,b} = (x^a)^b$

- *DESARROLLO DEL ALGORITMO.*

1. *Calculo del exponente en base binaria:*

$n(decimal) = b_k\ b_{k-1} \ldots . b_4\ b_3\ b_2\ b_1\ b_0\ (binario)$

2. *Rescribir el exponente en forma binaria:*

$n = 2^{b_k} + 2^{b_{k-1}} \ldots \ldots 2^{b_4} + 2^{b_3} + 2^{b_2} + 2^{b_1} + 2^{b_0}$

3. *Rescribir la base con el exponente en forma binaria:*

$a^n = a^{2^{b_k}+2^{b_{k-1}}\ldots\ldots 2^{b_4}+2^{b_3}+2^{b_2}+2^{b_1}+2^{b_0}}$

$a^n = a^{2^{b_k}}\ a^{2^{b_{k-1}}} \ldots \ldots \ldots a^{2^{b_4}}\ a^{2^{b_3}} a^{2^{b_2}} a^{2^{b_1}} a^{2^{b_0}}$

4. *Rescribir la base con el exponente en forma binaria:*

$a^n (mod\ m) = a^{2^{b_k}+2^{b_{k-1}}\ldots\ldots 2^{b_4}+2^{b_3}+2^{b_2}+2^{b_1}+2^{b_0}}\ (mod\ m)$

$a^n (mod\ m) = a^{2^{b_k}}\ (mod\ m)\ a^{2^{b_{k-1}}}\ (mod\ m) \ldots\ a^{2^{b_1}}\ (mod\ m)\ a^{2^{b_0}}\ (mod\ m)$

De esta forma con este tipo de exponencial, se puede garantizar como el sistema trabaja siempre con números inferiores al valor *"n"* al cuadrado.

El resumen de la exponenciación binaria se resume en la siguiente tabla resumen.

- ***Conclusión de la proposición referente a la exponenciación binaria**:*

$a^{2^{b_0}}\ (mod\ m) = a\ (mod\ m) = r_0$
$a^{2^{b_1}}\ (mod\ m) = r_0^2\ (mod\ m) = r_1$
$a^{2^{b_2}}\ (mod\ m) = r_1^2\ (mod\ m) = r_2$
.....
$a^{2^{b_{k-1}}}\ (mod\ m) = r_{k-2}^2\ (mod\ m) = r_{k-1}$
$a^{2^{b_k}}\ (mod\ m) = r_{k-1}^2\ (mod\ m) = r_k$

Conclusión final obtenida

$$a^n (mod\ m) = \prod_0^k \left[r_{k_{|b_k=1}}\ r_{k-1_{|b_{k-1}=1}} \right]\ mod(n)$$

$$= \prod_0^k \left[\left[\left[r_{k_{|b_k=1}}\ r_{k-1_{|b_{k-1}=1}} \right]\ mod(n) \right] r_{k-2_{|b_{k-2}=1}} mod(n) \right] r_{k-3_{|b_{k-3}=1}} mod(n) \ldots$$

Con el marco teórico de la aritmética modular y en concreto la exponenciación binaria definidos, el epígrafe avanza en la explicación del desarrollo firmware de la entidad. Extrapolando las bases teóricas definidas previamente, el desarrolla obtiene el mensaje cifrado como:

$$mensaje\_cifrado \equiv mensaje^e\ (mod\ n)$$

La entidad se encuentra constituida por módulos IP Cores que realizan las divisiones, multiplicaciones y cuadrados en el campo de operaciones matemáticas elementales y por dos procesos síncronos. El primer proceso calcula los sucesivos valores definidos como *"r_k"* almacenando los resultados en un vector. Por otro lado, cuando el primer proceso ha concluido, el segundo proceso realiza el producto de las posiciones del vector recalculando los nuevos módulos de las soluciones que se van obteniendo de forma dinámica. El resto del epígrafe realiza un análisis exhaustivo de cómo son estos dos procesos y, en conclusión, como se generan el mensaje encriptado.

La entidad tiene la siguiente estructura firmware, donde se puede apreciar el uso de tres tipos de IP Cores, utilizados para realizar el producto, la operación cuadrado y la división de los diferentes apartados del algoritmo de encriptación, como se ha explicado en la generación de las claves, estas entidades se encuentran prediseñados y desarrolladas, por lo que solo necesitan ser instanciadas y mapeadas tantas veces como se utilicen en la entidad desarrollada. La entidad fusiona los procesos síncronos de encriptación, con los Cores comentados previamente, con el objetivo de poder implementar la encriptación del algoritmo RSA.

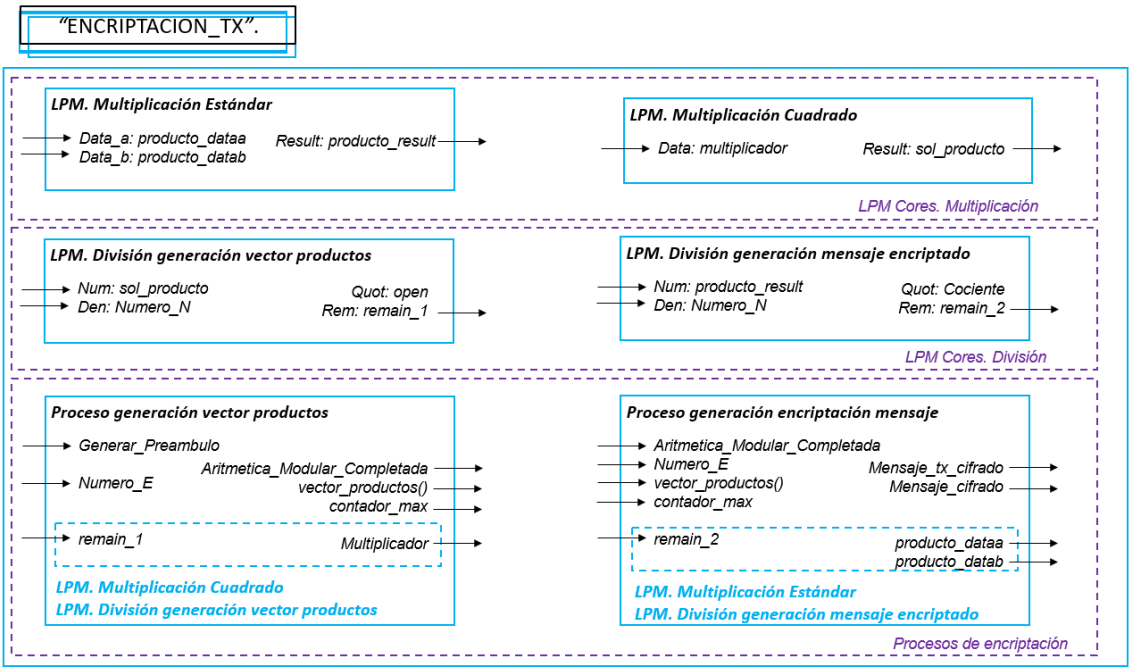


Ilustración 46. . Diagrama de bloque interno de la entidad "Encriptacion_Tx"

Como detalle de interés y uno de los principales problemas de identificar por parte del desarrollador durante el proceso del diseño y la implementación, es que al igual que en la generación de las claves aparece la necesidad de introducir ciclos de espera en el propio proceso debido a la alta velocidad del reloj de trabajo de la FPGA en comparación con el tiempo que tardan los IP Cores en realizar las operaciones.

La siguiente parte del epígrafe avanza en la explicación mediante diagramas de bloques de los dos procesos principales de encriptación. Es importante recordar, que el primero se inicia con la generación del preámbulo y el segundo cuando termina el primero. Por otro lado, el tiempo máximo de encriptación no puede ser superior al tiempo asociado a la transmisión del preámbulo (8 microsegundos).

4.3.2.2.1 Proceso de encriptación de generación vector productos.

El diagrama de bloques del primer proceso de encriptación es el siguiente:

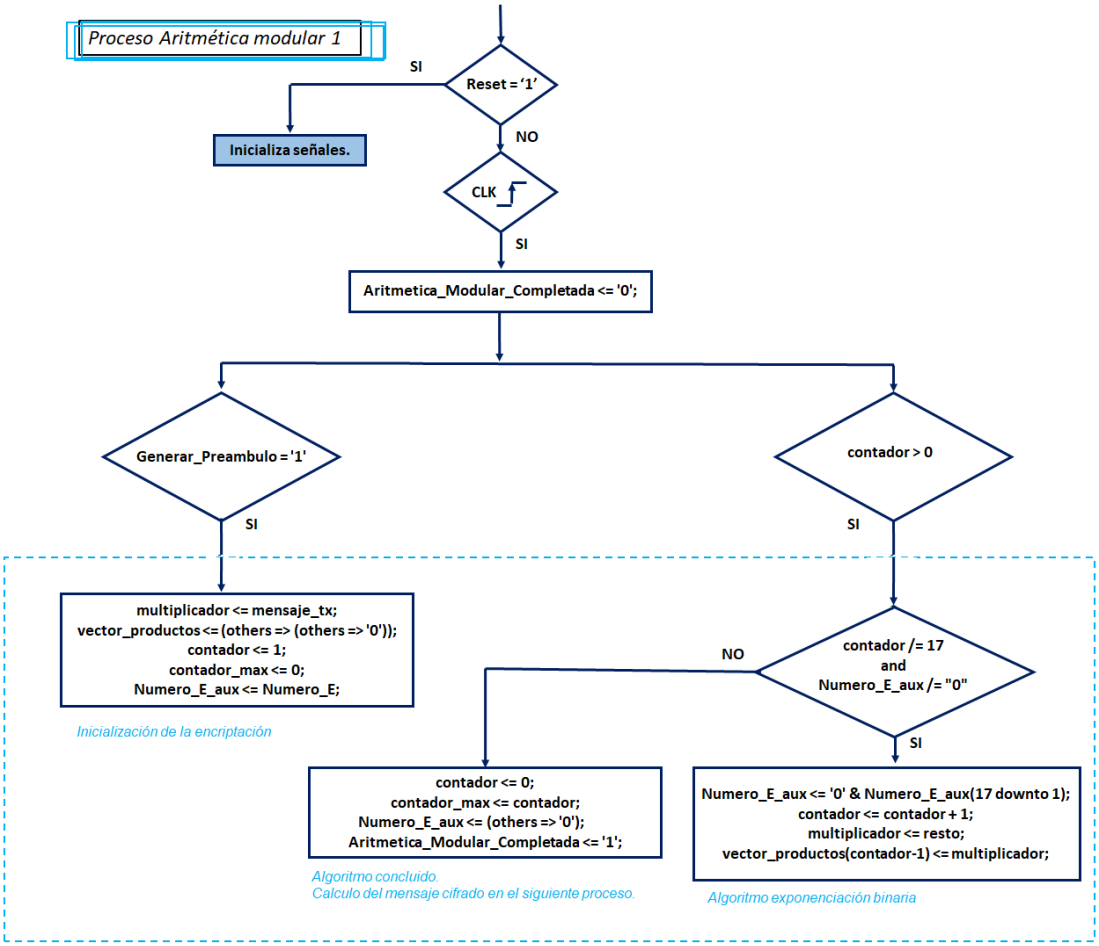


Ilustración 47. Flujograma del proceso de encriptación (primera parte)

Tal y como se ha comentado al comienzo del epígrafe, este proceso se inicializa con la detección del flanco de subida de la señal referente a la generación del preámbulo *(Generar_Preambulo)*. La detección inicializa el algoritmo de la exponenciación binaria, con esta detección se inicializan los valores de las señales utilizadas. El proceso evalúa dos señales. Por un lado, el valor del contador principal del proceso, realizando el algoritmo mientras este sea inferior a la longitud en bits del *"numero e"*. Por otro lado, el valor auxiliar de esta señal, con la que se discrimina si existe algún nivel alto en los bits que quedan por evaluar y que serían utilizados en el siguiente proceso.

Mientras se cumplan estas dos condiciones el proceso avanza en su cálculo iterativo con el que se rellena una nueva posición en el vector calculado en los Cores mapeados en la entidad. Cuando el proceso concluye, activa un flanco de reloj a nivel alto la señal *"Aritmetica_Modular_Completada"* mediante la cual se inicia el siguiente proceso. También se almacena el valor máximo del contador, para aumentar la velocidad del siguiente proceso, definiendo de forma clara el valor máximo de cuenta.

En resumen, el objetivo de este proceso, es acumular en una señal definida como un vector denominado *"Vector_productos()"* cada uno de los resultados obtenidos en cada una de las iteraciones desarrolladas.

4.3.2.2.2 Proceso de encriptación de generación mensaje TX.

El diagrama de bloques del segundo proceso de encriptación es el siguiente:

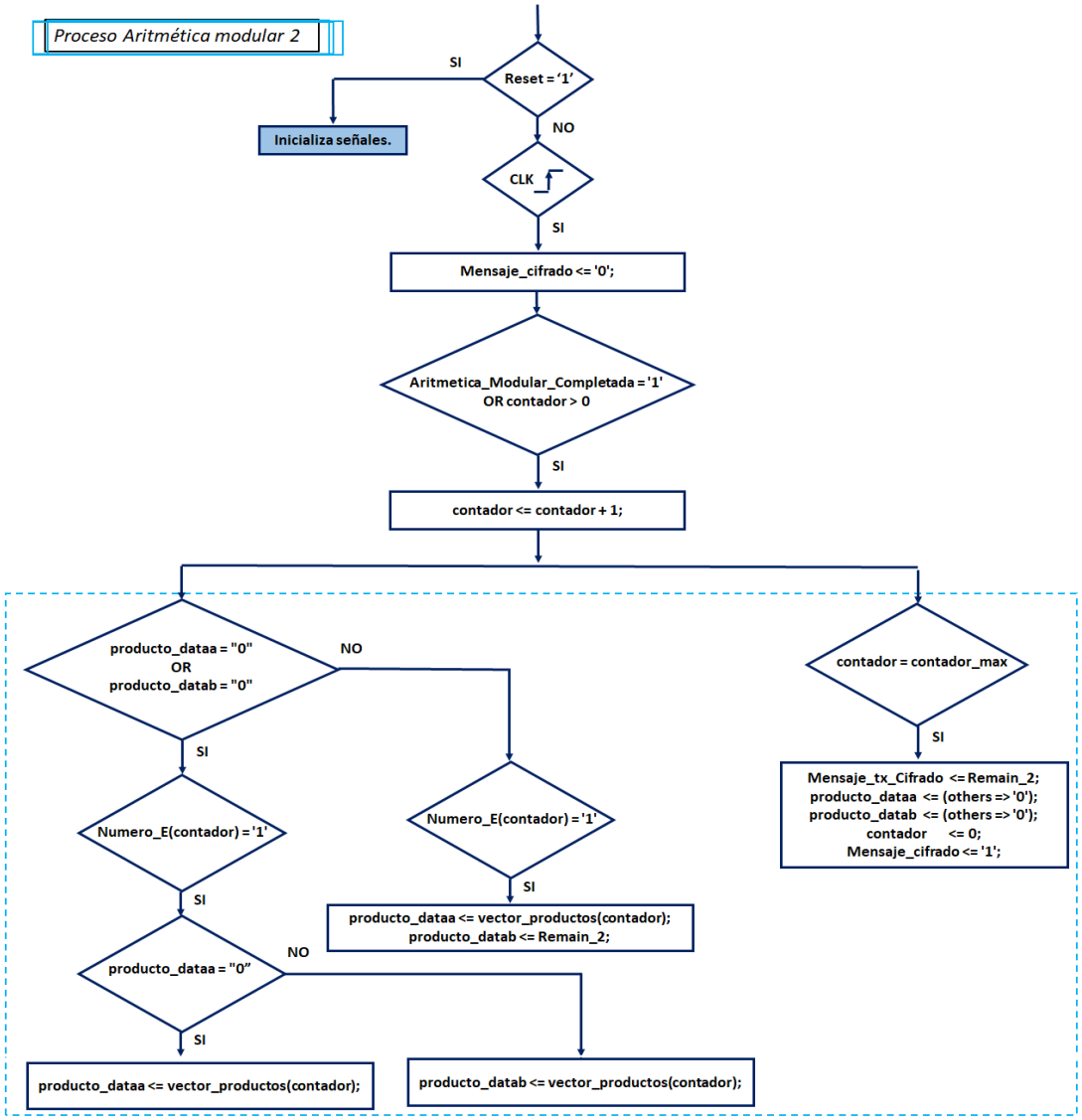


Ilustración 48. Flujograma del proceso de encriptación (segunda parte)

Este proceso se efectúa según termina el proceso anterior, teniendo una duración fija al valor máximo del contador del proceso anterior, este valor se almacena en el proceso anterior en la señal *"contador_max"*. El proceso actualiza los valores de las dos entradas del Core con el valor del vector de productos definido y desarrollado en el proceso anterior, la clave del proceso está en la condición para la introducción de estos valores en las señales de entrada al Core, la asignación se condiciona a que la posición del bit del *"numero e"* sea un nivel alto, para poder reflejar si la base se eleva a ese exponente o no debido al valor binario de esta última.

El proceso inicializa los valores de los dos multiplicadores, por el contrario, en caso de que estén inicializados, actualiza uno de ellos con el resto de la iteración anterior y el otro con el nuevo valor a introducir, siempre condicionados tal como se ha comentado en el párrafo anterior. Cuando el contador llega al valor máximo, el resto obtenido es el propio mensaje cifrado.

En el Anexo IV se ofrece un ejemplo ilustrativo para una mejor comprensión del lector.

4.4 Entidad Recepcion_Squitter.

Esta entidad ubicada en la FPGA receptora, tiene como función recibir e interpretar el squitter detectado. La entidad utiliza, por un lado, las claves privadas ubicadas ocultadas en el receptor con las que desencriptará el mensaje y, por otro lado, el propio squitter recibido y mapeado desde el pin de entrada de la FPGA. La entidad se puede dividir en dos recepciones, la primera recepción identifica correctamente el preámbulo, mientras que, la segunda entidad identifica los datos (como es de esperar, la segunda entidad actúa a continuación de la primera).

La siguiente figura muestra el diagrama de bloques de la entidad descrita.

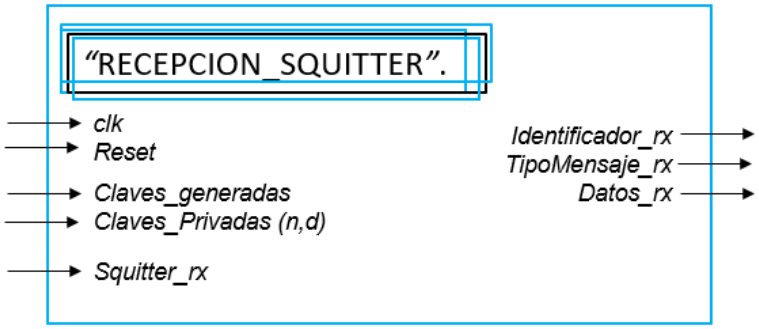


Ilustración 49. Diagrama de bloque de la entidad "recepcion_squitter"

La entidad se encuentra constituida por dos entidades receptoras, mediante las cuales se recibe el preámbulo y los datos modulados en PPM. Las tres salidas de la entidad se corresponden con cada uno de los tres mensajes recibidos y desencriptados del squitter.

La primera entidad receptora denominada *"Deteccion_Preambulo"* tiene como única función detectar correctamente el preámbulo transmitido, avisando mediante la señal *"Preambulo_Detectado"* a la segunda entidad receptora denominada *"Deteccion_Datos"* cuya función es detectar y desencriptar (mediante la clave privada del receptor) los tres mensajes transmitidos en el squitter.

La siguiente imagen muestra el mapeado interno de las dos entidades que constituyen la recepción del squitter. en resumen, se puede observar como la primera entidad, recibe únicamente el squitter y genera una única salida mediante la cual la siguiente entidad interpreta si el squitter se ha detectado correctamente. La segunda entidad receptora, recibe la señal detección de la anterior entidad, junto con el propio squitter y las claves privadas del sistema receptor, generando en tres salidas, las tres señales asociadas a los tres mensajes que se han transmitido.

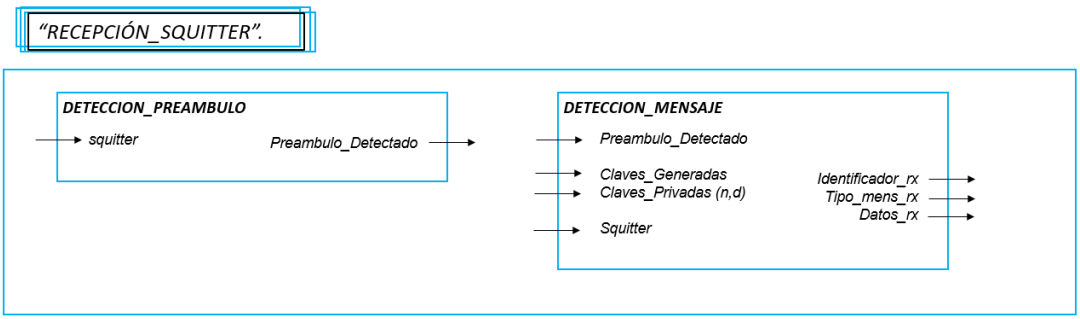


Ilustración 50. Diagrama de bloque interno de la entidad "recepcion_squitter"

El epígrafe, avanza en el marco técnico del análisis de ambas entidades receptoras.

4.4.1 Entidad Deteccion_Preambulo.

El algoritmo de funcionamiento de esta entidad es muy similar al de la transmisión del squitter. Esta entidad se genera mediante un proceso síncrono formado principalmente por una máquina de estados, la transición entre estados del proceso, comienza a ejecutarse con la detección de un flanco de subida del pin de entrada del squitter. la transición entre estados se marca con la detección de un nuevo flanco condicionado con el valor asociado a un contador interno al proceso.
La lógica seguida en el proceso, se emula en el siguiente diagrama de bloques, donde se puede observar como la máquina de estados se encuentra dentro del propio proceso síncrono, evaluándola en cada flanco de reloj ascendente de la señal de reloj.

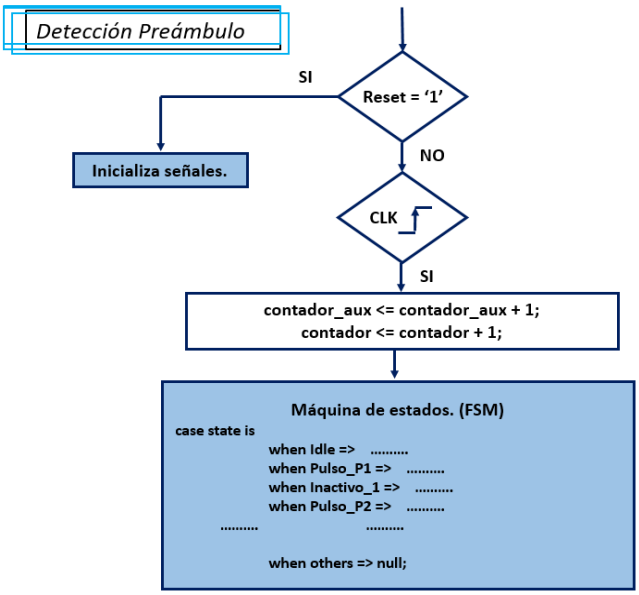


Ilustración 51. Flujograma del proceso de detección del preámbulo

Conocida la frecuencia de trabajo del oscilador externo *(f_{CLK} = 50 MHz, T_{CLK} = 0.02 µs)* ubicado en el kit de desarrollo, el cual es mapeado al pin dedicado como reloj de la FPGA y los tiempos marcados por los protocolos IFF para la transmisión de squitters, es fácil identificar y marcar los valores de cuenta asociados al contador de cada estado. Con el fin de poder blindar mejor el sistema, se otorgan los mismos márgenes de error utilizados en la tecnología IFF en los tiempos asociados a cada estado. Los tiempos definidos, por un lado, son el mínimo y máximo valor de estado, con unos márgenes del 10% respecto al valor ideal y, por otro lado, un mínimo de ciclos al nivel del valor en el que debería de estar el estado.

La máquina de estados desarrollada para generar el preámbulo se ilustra en la siguiente imagen. La máquina de estados parte de un estado de reposo, esperando la detección de un flanco ascendente en la señal del squitter. la detección del flanco se lleva a cabo mediante la puerta lógica XOR. Los cambios de estados de la FSM se condicionan a los valores de tres contadores. Su función se resume a continuación.

- **Contador.** Este contador, discrimina si el tiempo en cada estado se encuentra entre el máximo y mínimo permitido.
- **Contador_val.** Este contador, discrimina si el squitter cumple un numero de ciclos mínimos con el valor que debería en el estado.
- **Contador_aux.** Este contador se utiliza para garantizar que se cumple el tiempo de un preámbulo tanto si se detecta correctamente como si falla en algún estado de la FSM, su función es blindar la detección, descartando posible ruido en el sistema.

En la siguiente tabla, se resumen los valores asociado a los diferentes contadores en función del estado en el que se encuentra la máquina de estados.

Nombre (constante). Definición	*Duración tiempo.*	*Numero de ciclos.*
DURACION_PULSO	***0.5 µs***	***25 Ciclos***
DURACION_PULSO_MAX	*0.5 µs + 10%*	*27 Ciclos*
DURACION_PULSO_MIN	*0.5 µs - 10%*	*23 Ciclos*
VALOR_MIN_PULSO	*80% de 0.5 µs*	*20 Ciclos*
DURACION_INACTIVO_P1_P2	***0.5 µs***	***25 Ciclos***
DURACION_INACTIVO_P1_P2_MAX	*0.5 µs + 10%*	*27 Ciclos*
DURACION_INACTIVO_P1_P2_MIN	*0.5 µs - 10%*	*23 Ciclos*
VALOR_MIN_INACTIVO_P1_P2	*80% de 0.5 µs*	*20 Ciclos*
DURACION_INACTIVO_P2_P3	***2 µs***	***100 Ciclos***
DURACION_INACTIVO_P2_P3_MAX	*2 µs + 10%*	*110 Ciclos*
DURACION_INACTIVO_P2_P3_MIN	*2 µs - 10%*	*90 Ciclos*
VALOR_MIN_INACTIVO_P2_P3	*80% de 2 µs*	*80 Ciclos*
DURACION_INACTIVO_P3_P4	***0.5 µs***	***25 Ciclos***
DURACION_INACTIVO_P3_P4_MAX	*0.5 µs + 10%*	*27 Ciclos*
DURACION_INACTIVO_P3_P4_MIN	*0.5 µs - 10%*	*23 Ciclos*
VALOR_MIN_INACTIVO_P3_P4	*80% de 0.5 µs*	*20 Ciclos*
DURACION_INACTIVO_P4_DAT	***3 µs***	***150 Ciclos***
DURACION_INACTIVO_P4_DAT_MAX	*3 µs + 10%*	*165 Ciclos*
DURACION_INACTIVO_P4_DAT_MIN	*3 µs - 10%*	*135 Ciclos*
VALOR_MIN_INACTIVO_P4_DAT	*80% de 3 µs*	*120 Ciclos*
DURACION_PREAMBULO	***8 µs***	***420 Ciclos***

Tabla 7. Tabla resumen de tiempos asociados a los estados de la detección del preámbulo

Tal y como se ha comentado al comienzo del apartado, la detección del flanco de subida se detecta mediante la puerta lógica XOR, las señales de entrada a esta puerta son el valor actual del squitter y el valor anterior (registrado en una nueva señal a la entrada del proceso). El algoritmo implementado en cada estado de la máquina de estados consiste en condicionar de forma continua los valores de los contadores para poder avanzar al siguiente estado, la lógica de un estado genérico se esboza en la siguiente imagen.

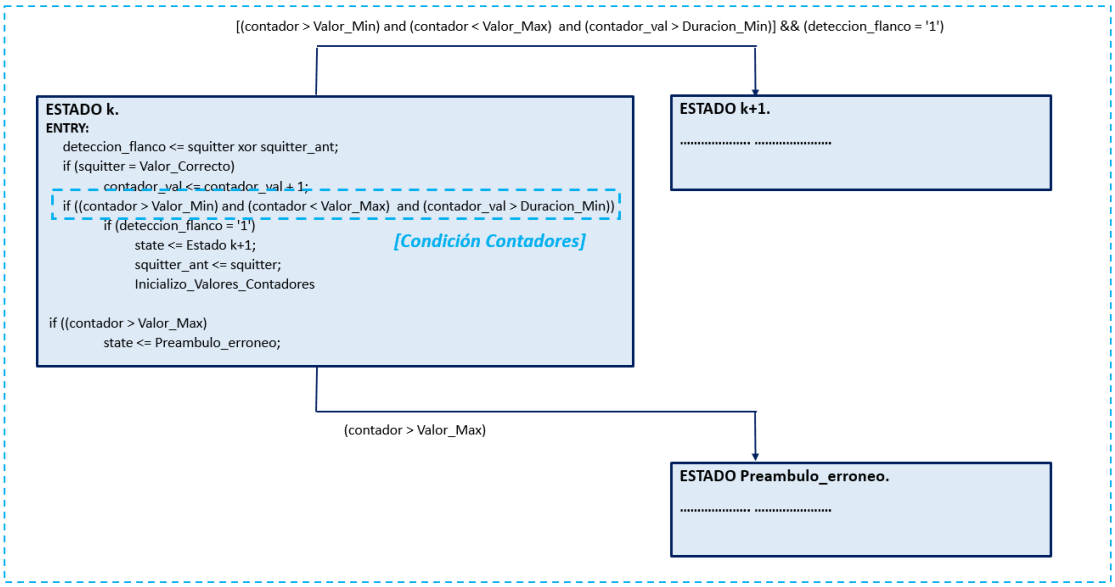


Ilustración 52. Estructura de un estado de la detección de un preámbulo.

La máquina de estados desarrollada en el proceso síncrono, se esboza en el siguiente diagrama de bloques

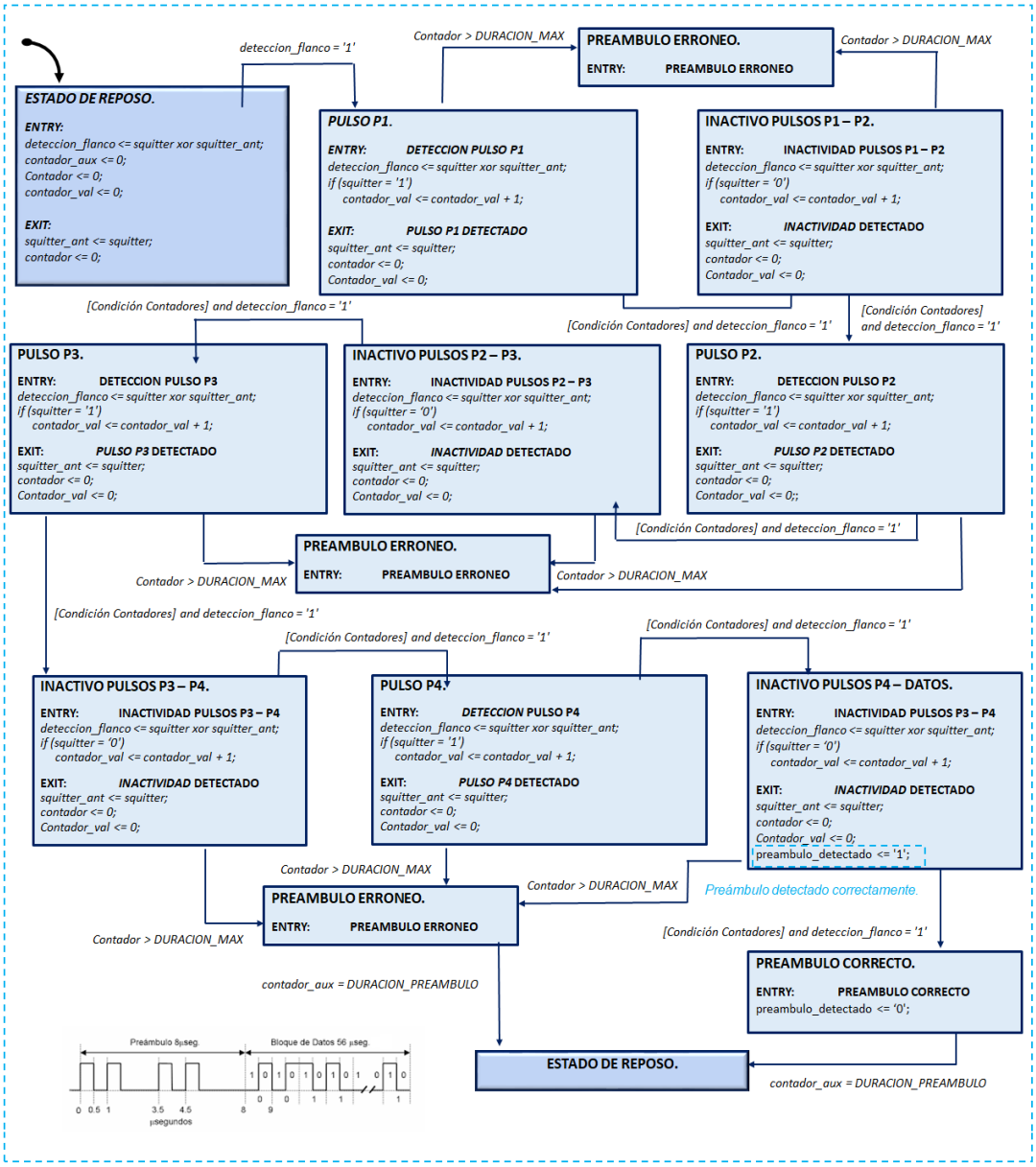


Ilustración 53. Diagrama de bloques de la máquina de estados de detección del preámbulo

La respectiva condición contadores que aparece en cada una de las transiciones entre estados, se encuentra asociada a poder garantizar dos parámetros.

- En primer lugar, poder garantizar que el preámbulo tiene una duración en tiempo correcta, se le otorga un margen de error (asociado a un tiempo de guarda) de un 10%, para ello se aplican las siguientes condiciones:

  (contador > DURACION_MIN) and (contador < DURACION_MAX)

- En segundo lugar, un mínimo de ciclos en el valor (alto o bajo) correspondiente al estado de la FSM, para ello se aplican la siguiente condición:

  (contador_val > VALOR_MIN)

4.4.2 Entidad Deteccion_Mensaje.

Al igual que en la entidad que lleva a cabo la generación de los datos, esta entidad tiene múltiples funciones. En primer lugar, mediante un proceso síncrono inicializado con un flanco ascendente en la señal de salida de la entidad *"Deteccion_Preambulo"* la entidad detecta los datos recibidos mediante una modulación PPM. El proceso almacena los datos del squitter completo en un vector, para a continuación, de forma concurrente y conociendo la posición de cada mensaje del vector conseguido en el proceso, poder extraer cada mensaje encriptado del vector completo. Mediante una señal denominada *"mensaje_detectado"* el proceso indica a las demás entidades que se han recibido correctamente los tres mensajes que conforman el squitter, los cuales se encuentran almacenados en tres señales diferentes y están listos para ser desencriptados.

Para la desencriptación del mensaje se utilizan las claves privadas del receptor. En esta entidad se introducen las señales de sincronismo, la señal mediante la cual se indica la correcta detección de un squitter, las tres señales referentes a los tres mensajes extraídos del squitter y las respectivas claves de descifrado. Al igual que en la parte de la encriptación, al tener tres mensajes que desencriptar por separado, se instancia tres veces esta entidad.

El siguiente diagrama de bloques esboza el mapeado entre las diferentes entidades que se han comentado previamente.

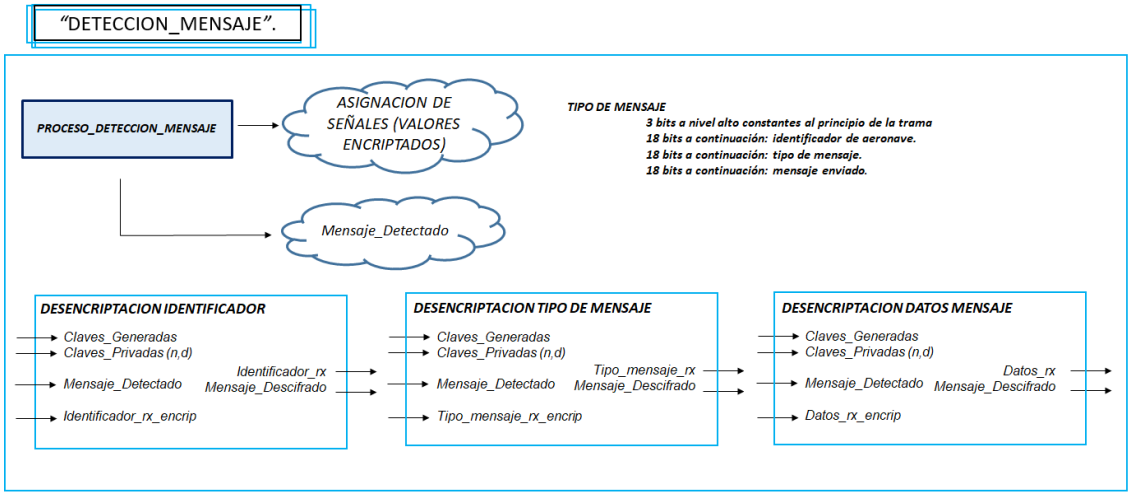


Ilustración 54. Diagrama de bloque interno de la entidad "Deteccion_Mensaje"

Junto con el mapeado de las entidades de desencriptación en esta entidad destaca el proceso de detección del mensaje Squitter en una modulación PPM. Es importante recordar que, en este tipo de modulación, dependiendo del valor del bit a transmitir (nivel alto o bajo) se transmite un pulso de 0.5 µseg. de duración, en la primera o segunda mitad del intervalo.

Trabajando con el reloj del oscilador *(f_{CLK} = 50 MHz, T_{CLK} = 0.02 µs)* y unos pulsos de longitud 0.5 µseg. de duración, como en el proceso de transmisión, con el objetivo de cumplir los tiempos utilizados en la tecnología IFF, aparece la necesidad de un contador auxiliar que introduzca ciclos de esperar por valor de veinticinco, marcando la mitad del periodo de recepción y consiguiendo así, poder medir las anchuras deseadas.

4.4.2.1 Proceso de detección del mensaje (Squitter).

En el siguiente diagrama de bloques se representa el proceso que de detección de datos en modulación PPM, asignando en la salida los valores de cada señal encriptada. Es importante destacar que con el fin de poder sincronizar el proyecto correctamente, aparece la necesidad de registrar algunas señales del mismo.

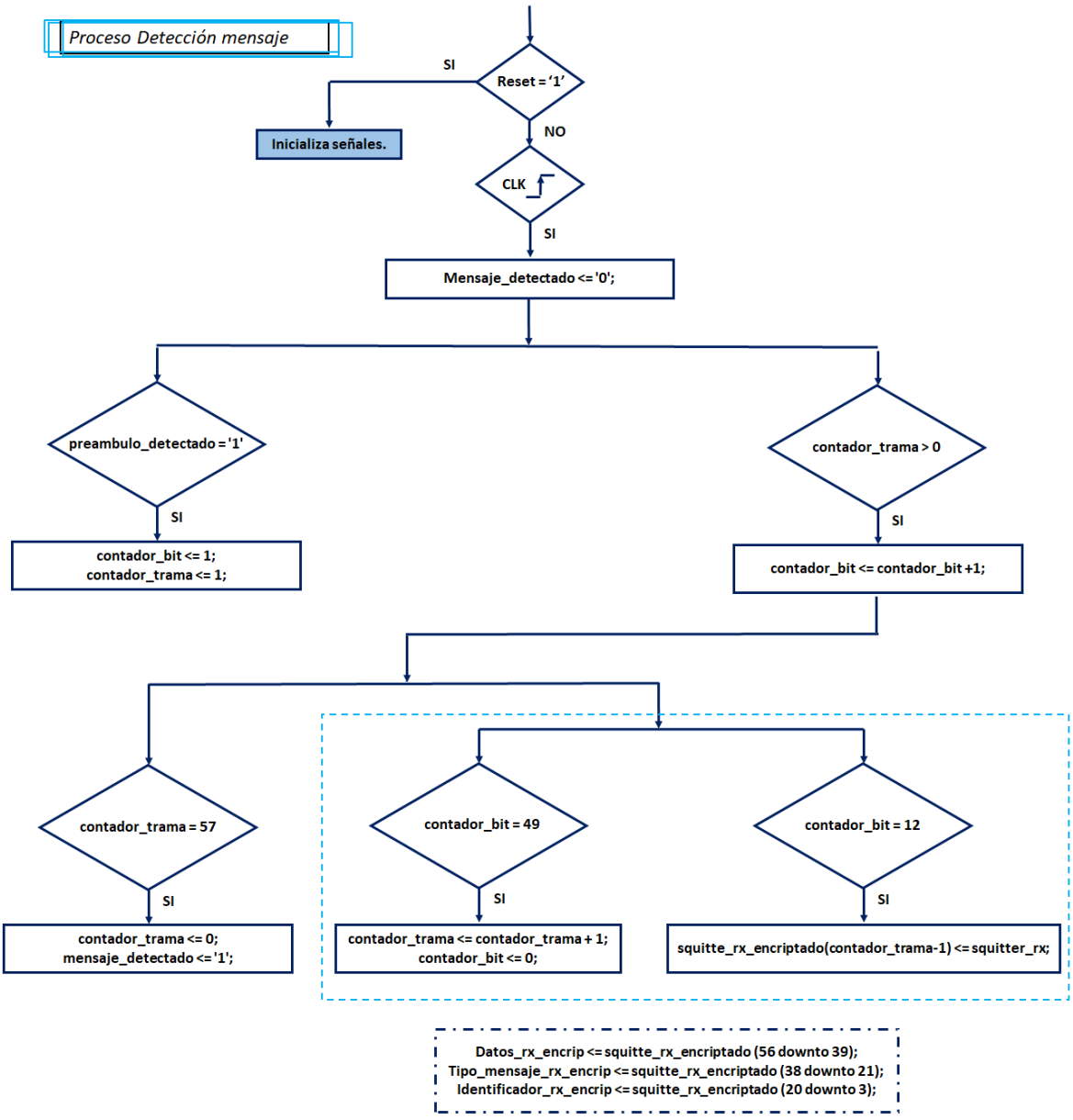


Ilustración 55. Flujograma del proceso de recepción squitter modulado en PPM

El proceso se inicializa cuando se activa la señal *"Preambulo_detectado"* asociada al proceso anterior y marcando la correcta detección en la recepción del preámbulo de un squitter. Mediante el contador de trama, se detecta el bit que se ha recibido de toda la transmisión, mientras que, con el contador de bit, se identifica si el nivel alto aparece en la primera mitad o en la segunda del semiciclo, con el objetivo de poder detectar si se ha transmitido un nivel alto o un nivel bajo.

Los datos recibidos, se encuentran encriptados, la siguiente parte de la cadena de recepción, consiste en la desencriptación para su futura transmisión al microcontrolador.

4.4.2.2 Entidad Desencriptacion_Mensaje.

La estructura de la entidad que realiza la desencriptación del mensaje es muy similar a la entidad que lo encripta. Entre las dos entidades, de cara a la declaración de la entidad presentan únicamente dos diferencias, la primera reside en las claves de cifrado, en una se cifra con la clave pública, mientras que en la otra se descifra con la clave privada, la segunda diferencia es la señal de inicio de encriptación o desencriptación, el comienzo de la encriptación comienza con la generación del preámbulo, mientras que en la desencriptación comienza con la detección del mensaje squitter. Es importante destacar que como se ha comentado previamente, la entidad se instancia tres veces, debido a que se desencriptan las tres partes del squitter, equivalente a tres mensajes recibidos.

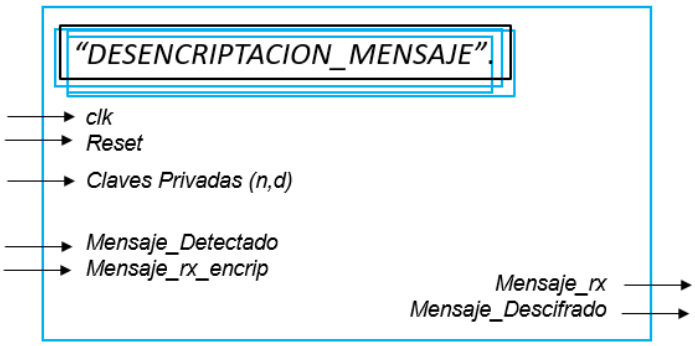


Ilustración 56. Diagrama de bloque de la entidad "Desencriptacion_Mensaje"

Al igual que en la parte de la encriptacion, la desencriptación de los mensajes aplicando la metodología de encriptación RSA se basa en el cálculo de aritmética modular. La identidad de aritmética modular se puede resumir en el siguiente cuadro.

$$a^n \ (mod\ m\ ) = resto(\frac{a^n}{m})$$

Como se ha explicado en el estado del arte, en concreto en el epígrafe donde se estudia el algoritmo de encriptación RSA, la clave para conseguir la desencriptación del mensaje recibido son las siguientes ecuaciones.

$$mensaje\_cifrado \ \equiv \ mensaje\_plano^e \ (mod\ n\ )$$
$$mensaje\_plano \ \equiv \ mensaje\_cifrado^d \ (mod\ n\ )$$

En otras palabras, la clave es la siguiente identidad.

$$mensaje \ \equiv \ mensaje^{ed} \ (mod\ n\ )$$

Se puede identificar dos claras y únicas diferencias de esta entidad frente a la entidad referente a la encriptación.

- En primer lugar, la base de la operación. La entidad desencriptadora trabaja con el mensaje cifrado, mientras que la encriptadora lo hace con el mensaje en texto plano.
- En segundo lugar, el exponente de la operación. La entidad desencriptadora utiliza la clave privada (exponente de descifrado *"numero d"*), mientras que la encriptadora utiliza la clave pública (exponente de cifrado *"numero e"*).

Los problemas que presentan estos cálculos de desencriptación son los mismos que los identificados en la parte de la encriptación, de hecho, en estos problemas que se presentan reside la seguridad del algoritmo, tal y como se ha comentado previamente, se trabaja con funciones matemáticas unidireccionales cuyo cálculo directo es sencillo, pero muy complejo a la inversa, presentando la necesidad de realizar un gran número de operaciones computacionales. El problema se resuelve aplicando la misma lógica que en la encriptación, evitando así, trabajar con números tan elevados.

Los algoritmos utilizados en esta parte son los mismos que en la anterior, destacando la aritmética modular, en concreto la exponenciación binaria y la resolución mediante el método de los cuadrados repetitivos.

La estructura de la entidad es muy similar a la entidad *"Encriptacion_Mensaje"* ya que como se ha comentado previamente, presenta dos diferencias en la entidad en cuanto a números utilizados y una nueva señal de inicio de desencriptación, pero una misma lógica matemática de los procesos. Las partes recicladas y explicadas previamente en la parte de la encriptación se resumirá en esta parte de la memoria.

La entidad tiene la siguiente estructura firmware, la entidad se encuentra constituida por módulos IP Cores que realizan las divisiones, multiplicaciones y cuadrados en el campo de operaciones matemáticas elementales y por dos procesos síncronos. El primer proceso calcula los sucesivos valores definidos como *"r_k"* almacenando los resultados en un vector. Por otro lado, cuando el primer proceso ha concluido, el segundo proceso realiza el producto de las posiciones del vector recalculando los nuevos módulos de las soluciones que se van obteniendo de forma dinámica. En resumen, La entidad fusiona los procesos síncronos de desencriptación, con los Cores comentados previamente, con el objetivo de poder implementar la desencriptación del algoritmo RSA. El resto del epígrafe realiza un análisis exhaustivo de cómo son estos dos procesos y, en conclusión, como se generan el mensaje desencriptado.

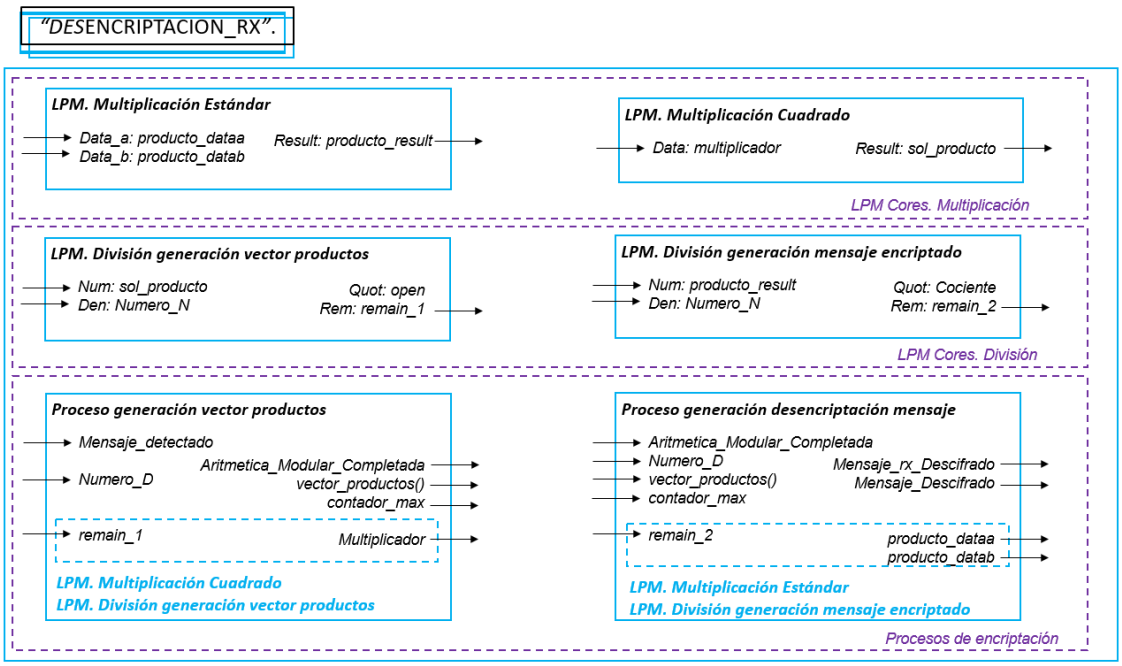


Ilustración 57. Diagrama de bloque interno de la entidad "Desencriptacion_Rx"

La siguiente parte del epígrafe avanza en la explicación mediante diagramas de bloques de los dos procesos principales de desencriptación.

4.4.2.2.1 Proceso desencriptación de generación vector productos.

El diagrama de bloques del primer proceso de desencriptación es el siguiente:

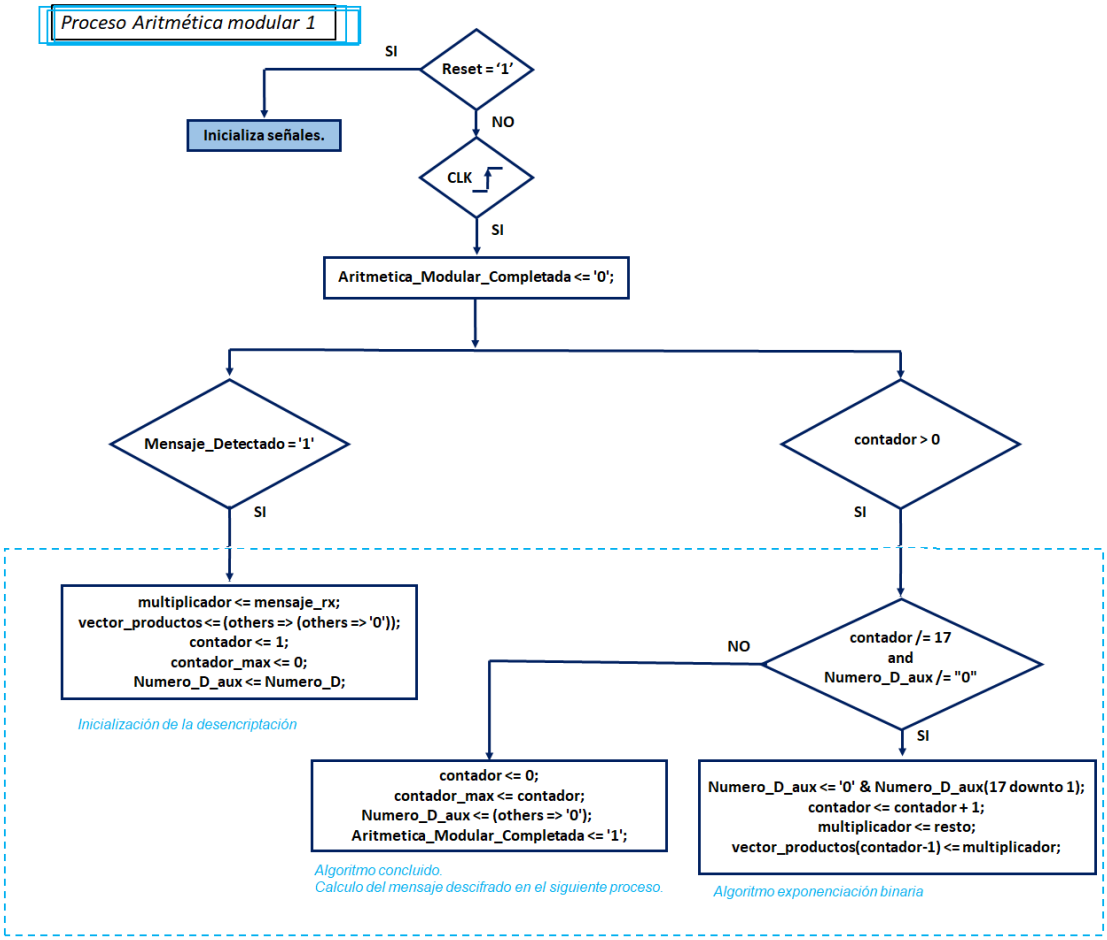


Ilustración 58. Flujograma del proceso de desencriptación (primera parte)

Tal y como se ha comentado al comienzo del epígrafe y en diferencia a la entidad de encriptación este proceso se inicializa con la detección del flanco de subida de la señal referente a la detección del mensaje *(Mensaje_Detectado)*. La detección inicializa el algoritmo de la exponenciación binaria, con esta detección se inicializan los valores de las señales utilizadas y el proceso avanza con la misma lógica que en la encriptación. De esta forma, el proceso rellena en cada iteración o entrada, las posiciones del vector denominado *"Vector_productos()"* con las salidas de los Cores que han sido mapeados en la entidad. Cuando el proceso concluye, activa un flanco de reloj a nivel alto la señal *"Aritmetica_Modular_Completada"* mediante la cual se inicia el siguiente proceso de desencriptación. También se almacena el valor máximo del contador, para aumentar la velocidad del siguiente proceso, definiendo de forma clara el valor máximo de cuenta.

Concluido el proceso referente a la primera parte de la desencriptación. Se almacena en el vector *"Vector_productos()"* todos los productos almacenados en cada iteración del proceso, el contador máximo *"Contador_max"* referente a todos los productos realizados y una señal *"Aritmetica_Modular_Completada"* mediante la cual se inicializa el siguiente proceso con un nivel alto durante un ciclo de reloj.

4.4.2.2.2 Proceso desencriptación de detección del mensaje RX.

El diagrama de bloques del segundo proceso de desencriptación es el siguiente:

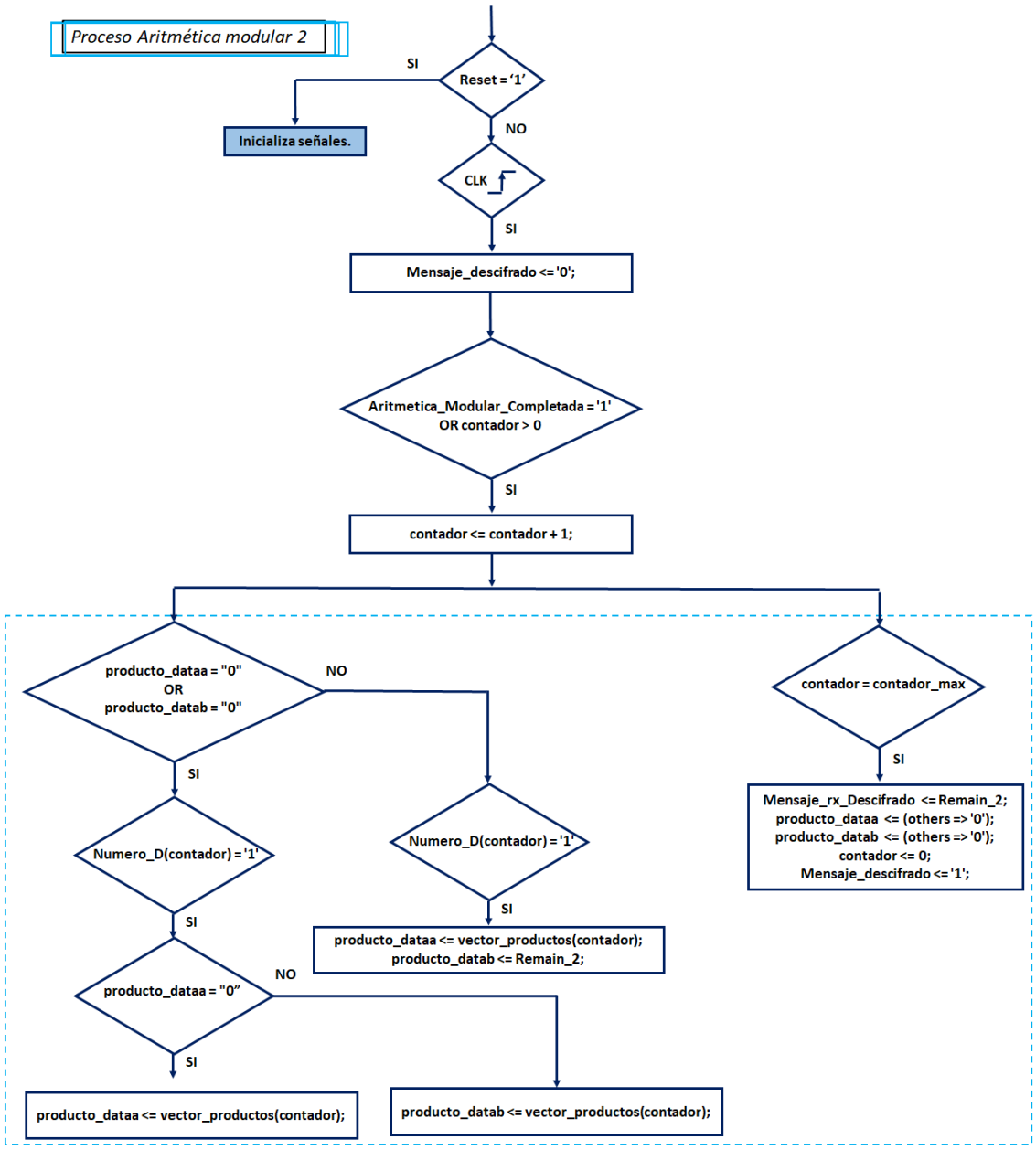


Ilustración 59. Flujograma del proceso de desencriptación (segunda parte)

Este proceso se efectúa según termina el proceso anterior, teniendo una duración fija al valor máximo del contador del proceso anterior, este valor se almacena en el proceso anterior en la señal *"contador_max"*. De forma análoga a la entidad anterior, el proceso actualiza los valores de las dos entradas del Core con el valor del vector de productos definido y desarrollado en el proceso anterior. La asignación se condiciona a que la posición del bit del *"numero d"* sea un nivel alto, para poder reflejar si la base se eleva a ese exponente o no debido al valor binario de esta última señal. Cuando el contador llega al valor máximo, el resto obtenido en el Core es el propio mensaje descifrado.

4.5 Capturas de simulación. Resultados obtenidos del capítulo.

Esta parte final del capítulo, busca ofrecer mediante capturas de simulación de los entornos de desarrollo utilizados, una demostración de los resultados obtenidos.

Para una mejor visualización de las capturas de simulación, todas ellas se encuentran ampliadas en el anexo V incrustadas en apaisado.

- ***Capturas de pantallas del proceso de generación de claves.*** *Estas capturas ofrecen una visualización del entorno de desarrollo de los procesos de cálculo de las claves públicas y privadas con los números primos elegidos previamente. Para una mejor estructuración, se divide el apartado en dos subapartados para cada una de las claves generadas.*

Clave pública: (e, n)	*Clave privada: (d, n)*

- ***Capturas de pantallas del proceso de generación de la clave pública.***

Para el cálculo del exponente de cifrado se elige un número *"e"* al azar inferior a *"f"*. El número *"e"* inicial elegido es *"f/3"*, siendo el divisor (numero 3) un numero al azar elegido por el desarrollador debido a que no ser ni un número muy pequeño ni muy grande relativo al número *"f"*. Mediante el algoritmo de Euclides se discrimina si el número *"e"* inicial elegido es coprimo al número *"f"*. En caso afirmativo, se escoge este número *"e"* inicial como el exponente de cifrado de la clave pública, en caso negativo, se incrementa en una unidad el número *"e"* inicial y se vuelve a comprobar si este número es coprimo al número *"f"*, repitiendo este incremento hasta que finalmente lo sean. En el algoritmo, la discriminación que determina si el número *"e"* es coprimo al número *"f"* viene marcada por el valor del ultimo resto, ya que de forma indirecta está relacionado con el M.C.D. de ambos números. En caso de que el resto sea cero, los números no son coprimos, mientras que si, por el contrario, el resto es un uno, los números si lo serán.
Elegido el exponente de cifrado, se almacena en un vector todos los cocientes obtenidos en cada iteración en el algoritmo de Euclides ya que serán utilizados en el cálculo de la clave privada.

- ***Capturas de pantallas del proceso de generación de la clave privada.***

Para el cálculo del exponente de descifrado se calcula el número *"d"* resolviendo de manera inversa el algoritmo de Euclides, mediante los cocientes almacenados en un vector previamente en el cálculo de la clave pública. La condición de este número secreto es que el producto *"d"* por *"e"* menos uno, sea dividido exactamente por *"f"*.
La implementación más sencilla y más lenta, seria realizar un bucle de *"f"* iteraciones hasta obtener un resto de valor la unidad. El proyecto utiliza el algoritmo extendido de Euclides, apoyándose en la identidad de Bezout particularizando un máximo común divisor de valor unidad. Mediante estas propiedades se acelera descomunalmente el cálculo de la clave privada a muy pocas iteraciones en un proceso, evitando así realizar un bucle de *"f"* iteraciones. Este algoritmo, parte del resto con valor unidad obtenido en la última división de la clave pública para recorrer el vector de forma inversa aplicando las operaciones inversas y acabar llegando al exponente de descifrado *"d"*.

Finalizados los cálculos de las claves públicas y privadas se almacenan en la FPGA y se activa a nivel alto una señal mediante la cual se habilitará en un futuro la encriptación y desencriptación de los mensajes transmitidos y recibidos.

- ***Capturas de pantallas del proceso de generación de la clave pública.***

Ilustración 60. Capturas de pantallas del proceso de generación de la clave pública (1)

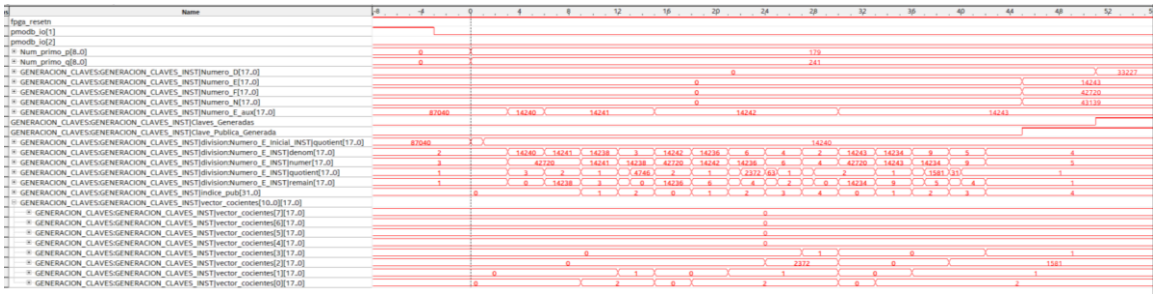


Ilustración 61. Capturas de pantallas del proceso de generación de la clave pública (2)

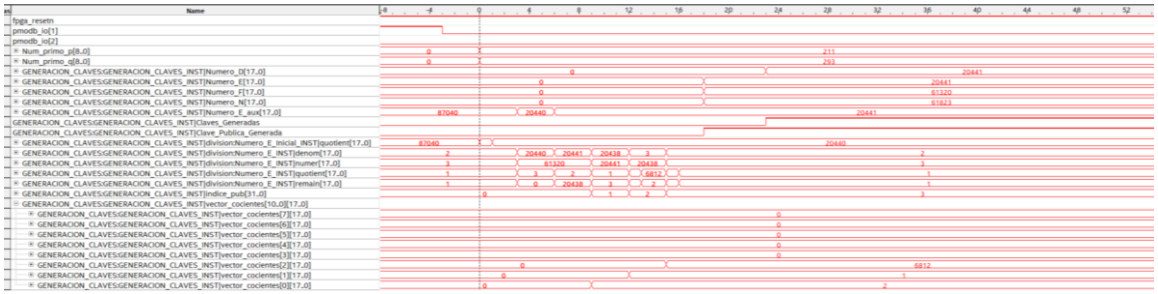


Ilustración 62. Capturas de pantallas del proceso de generación de la clave pública (3)

Ilustración 63. Capturas de pantallas del proceso de generación de la clave pública (4)

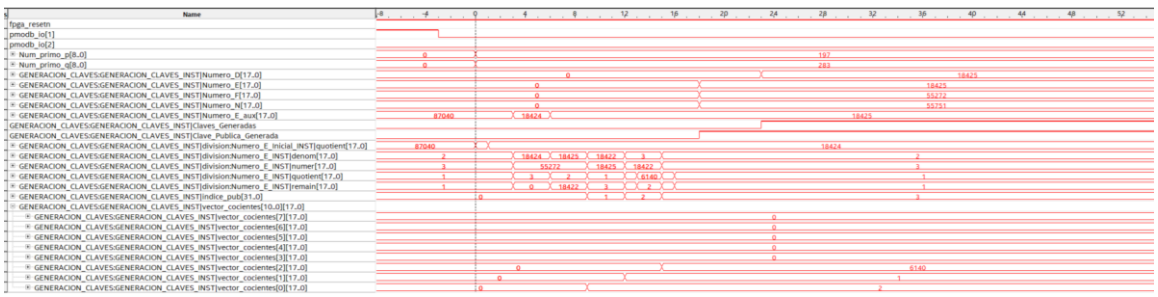


Ilustración 64. Capturas de pantallas del proceso de generación de la clave pública (5)

Explicación de las capturas de simulación obtenidas.
Introducida previamente la idea genérica del funcionamiento de la entidad se analiza de forma breve dos de las simulaciones realizadas al azar para explicar la lógica generada.

Es de especial interés destacar que, en algunos casos, los primeros valores obtenidos en las señales asociadas a los cocientes y restos de los IP Cores mapeados, ofrecen soluciones erróneas. El motivo es que la velocidad del proceso por el reloj asociado es superior al tiempo necesario del Core para realizar la división.

Tal y como se ha podido observar en los capítulos anteriores, existen dos Cores mapeados. El primero tiene como función calcular el número E inicial con el que se probara la clave pública, con el objetivo de que sea más impredecible, no se ajusta este problema y se coge la primera solución obtenida, el segundo tiene como función realizar el algoritmo de Euclides, para obtener los valores exactos, se decide introducir en el proceso un contador que genere pequeños ciclos de espera en la toma de datos de este Core.

Capturas de pantallas del proceso de generación de la clave pública (1).

Para este ejemplo se comienza trabajando con el "número E inicial" 17.023, siendo este la primera aproximación de la división entre el "número F" y el constante número 3. El resultado exacto de la división aparece en el cociente unas muestras más adelante, dando un resultado inicial que hasta cierto punto podría ser impredecible.

A continuación, se implementa el algoritmo de Euclides ofreciendo este primer número como solución para la clave de cifrado ya que 17023 es coprimo con respecto al "número F" 50.686. Se puede comprobar esta conclusión con el valor del ultimo resto siendo este el M.C.D. de ambos números.

Los valores de los cocientes se almacenan en el vector de forma correcta y se activa la señal que hace referencia a la generación de las claves públicas.

Para concluir, se comprueba que los valores almacenados en el vector cociente son coherentes y coinciden con los desarrollados en el algoritmo en la fila cocientes del Core mapeado inicialmente.

Capturas de pantallas del proceso de generación de la clave pública (2)

Para este ejemplo se comienza trabajando con el "número E inicial" 14.240, siendo este el valor exacto de la división entre el "número F" y el constante número 3.

El proceso comienza en la búsqueda del número coprimo al número F 42.720. El primer número elegido es 14.240, se descarta de forma inmediata puesto que el número inicial es el número E elegido multiplicado por el factor constante número 3. A continuación, el algoritmo incrementa en una unidad el numero E y comprueba el número 14.241, descartando este también por ser ambos números divisibles por tres. El algoritmo vuelve a incrementar el número E y busca comprobar el número 14.242, descartando también esta opción por ser divisibles por dos. Finalmente, el algoritmo incrementa en una unidad el numero E y busca comprobar el número 14.243, siendo este, un número coprimo al número F.

De especial interés es comprobar como en todas las iteraciones anteriores, justo antes de llegar al resto nulo, el valor del resto coincide con el valor del M.C.D. del número F y del valor del número E esa iteración.

Los valores del vector cocientes, se van reseteando a valores ceros cada vez que el número escogido resulta no ser coprimo, comprobando como al final, contiene solo los últimos valores de la última iteración (número 14.243).

- ***Capturas de pantallas del proceso de generación de la clave privada.***

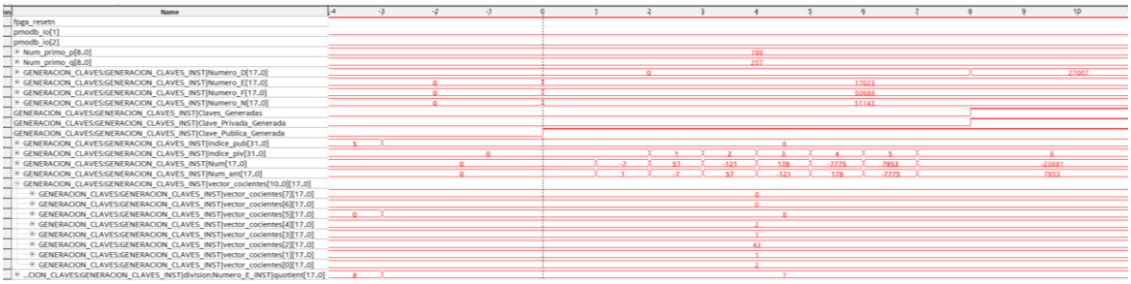


Ilustración 65. Capturas de pantallas del proceso de generación de la clave privada (1).

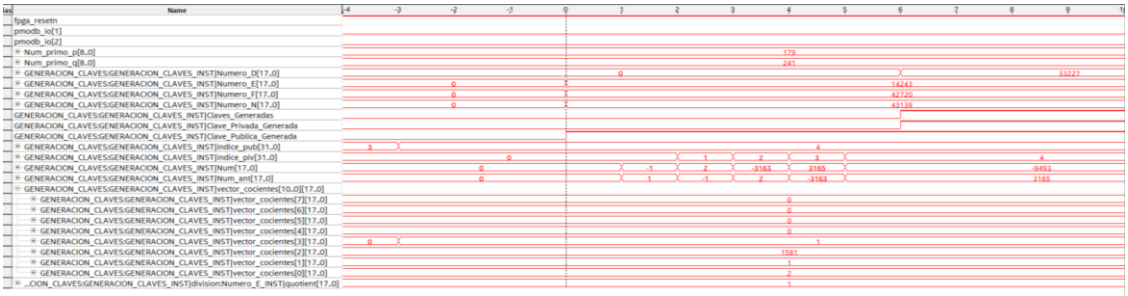


Ilustración 66. Capturas de pantallas del proceso de generación de la clave privada (2).

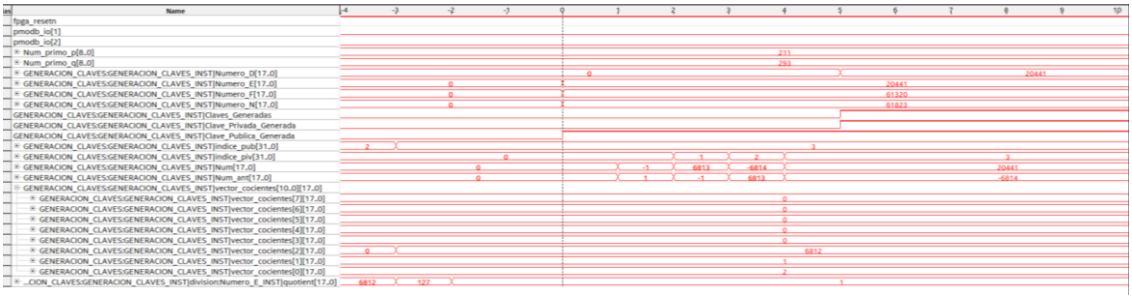


Ilustración 67. Capturas de pantallas del proceso de generación de la clave privada (3).

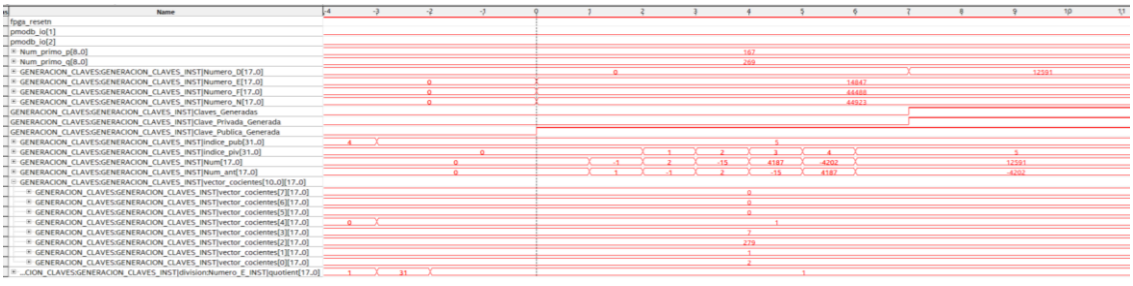


Ilustración 68. Capturas de pantallas del proceso de generación de la clave privada (4).

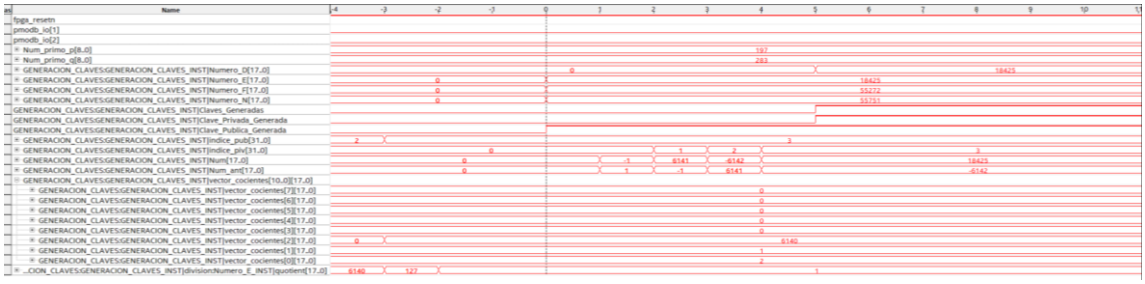


Ilustración 69. Capturas de pantallas del proceso de generación de la clave privada (5).

Explicación de las capturas de simulación obtenidas.
Introducida previamente la idea genérica del funcionamiento de la entidad se analiza de forma breve la primera de las simulaciones realizadas al azar para explicar la lógica generada

La clave del cálculo de la clave privada, es resolver de manera inversa el algoritmo de Euclides. Para ello es de vital importancia disponer de los cocientes que se han ido obteniendo en el cálculo recursivo, estos vectores se encuentran almacenados en el vector denominado "Vector_Cocientes".

El algoritmo implementado se muestra en el siguiente diagrama de flujos.

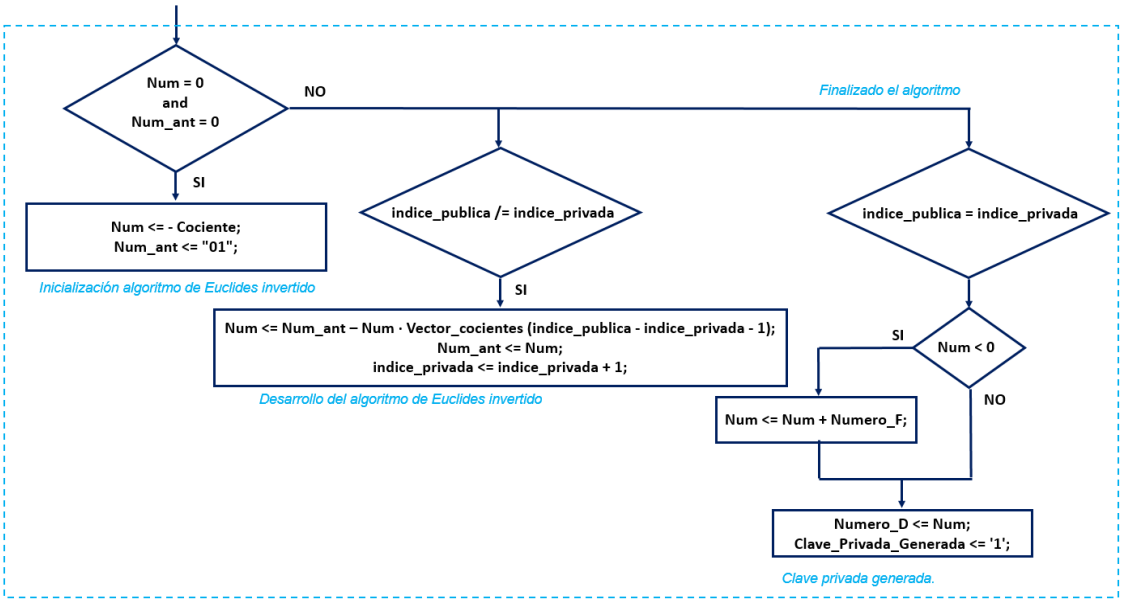


Ilustración 70. Flujograma del algoritmo del cálculo de la clave privada.

Iteración 1.

$Num_{ant} = 1$

$Num = -Cociente = -1$

Iteración 2.

$Num_{ant} = Num = -1$

$Num = Num_{ant} - Num \cdot Vector\_cocientes(2) = 1 + 6812 = 6813$

Iteración 3.

$Num_{ant} = Num = 6813$

$Num = Num_{ant} - Num \cdot Vector\_cocientes(1) = -1 - 6813 = -6814$

Iteración 4.

$Num_{ant} = Num = 6814$

$Num = Num_{ant} - Num \cdot Vector\_cocientes(0) = 6813 + 2 \cdot 6814 = 20441$

Iteración 5.

$Numero\_D = Num = 20441$

Conclusión personal de los algoritmos de cálculos de las claves de cifrado RSA.
Como detalle de interés, poniendo en valor el desarrollo. Me gustaría destacar la velocidad de los procesos implementados del cálculo de las claves públicas y privadas. El peor caso de todos, tarda aproximadamente un total de 50 ciclos de reloj, lo que equivaldría a un total de un microsegundo.

Potencia del algoritmo desarrollado: Mismos números primos elegidos por el usuario, generan distintas claves de cifrado RSA.

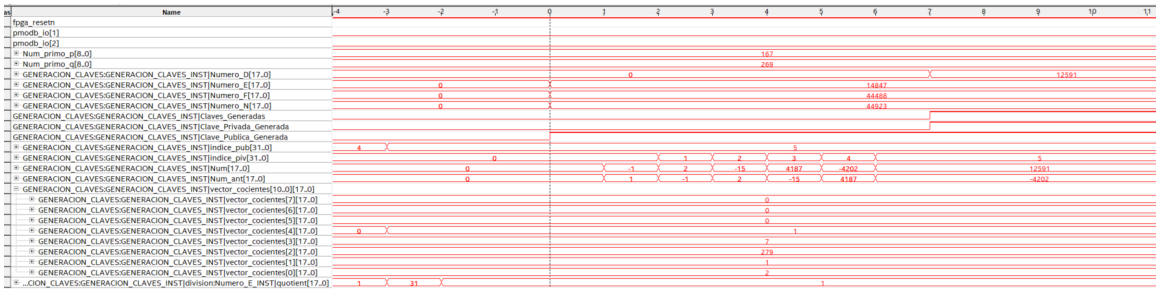


Ilustración 71. Capturas de pantallas del proceso de generación de claves (1)

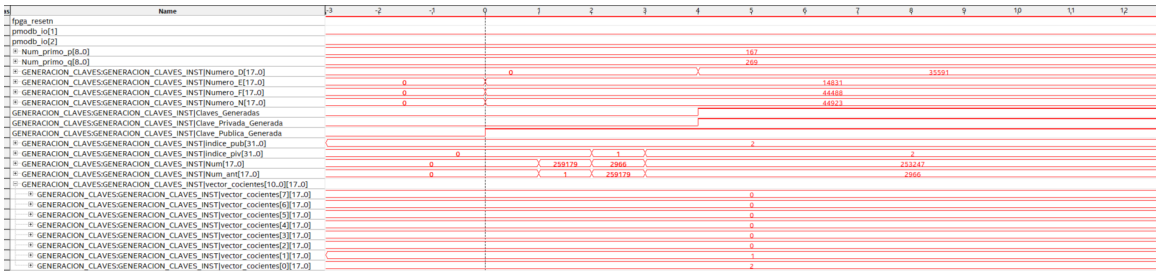


Ilustración 72. Capturas de pantallas del proceso de generación de claves (2)

Explicación de las capturas de simulación obtenidas.

La clave del algoritmo RSA, como la de cualquier otro algoritmo de encriptación es conseguir unas claves de encriptación lo más impredecibles posibles, con el objetivo de aumentar la dificultas de que estas sean calculadas por parte de una persona externa. En las capturas anteriores, se puede observar como con los mismos números primos elegidos por el usuario, los exponentes de cifrado y descifrado son diferentes.

El motivo es el valor del número E inicial, como se ha comentado, los módulos Core, no ofrecen la solución exacta de la división de forma inmediata en la primera lectura del proceso, tardando algunos flancos más.

El algoritmo dispone de un contador interno para obtener con exactitud este valor cuando se aplica el algoritmo de Euclides, un contador que funciona de manera continua desde que arranca la FPGA, el instante de actualización del valor del cociente de la división, es función del valor del contador interno asociado al proceso del cálculo del algoritmo, pudiendo trabajar con el primer resultado que ofrece, siendo en algunos casos el valor exacto y en otros muchos una aproximación de la división y además un valor totalmente impredecible. Al variar el valor inicial del número E, este se ve modificado por completo el del número D.

Como es de esperar, ambas soluciones encriptan y desencriptan perfectamente.

- ***Captura de pantalla de la transmisión y recepción de squitter.*** *Estas capturas ofrecen una visualización del entorno de desarrollo de la transmisión y recepción del squitter.*

Ilustración 73. Captura de pantalla de la transmisión y recepción de squitter.

Para una mejor visualización se amplía en la captura las partes del inicio del squitter (relacionada con el generación y recepción del preámbulo) y del final (desencriptación de los datos recibidos).

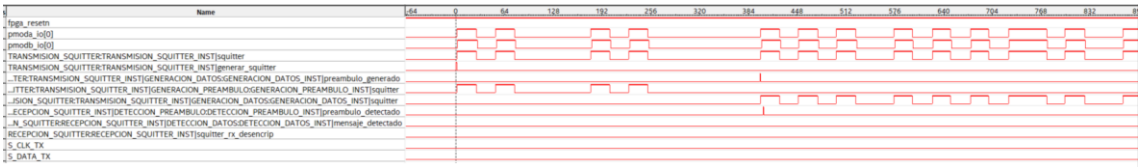


Ilustración 74. Captura de pantalla de la transmisión y recepción de squitter ampliado inicio

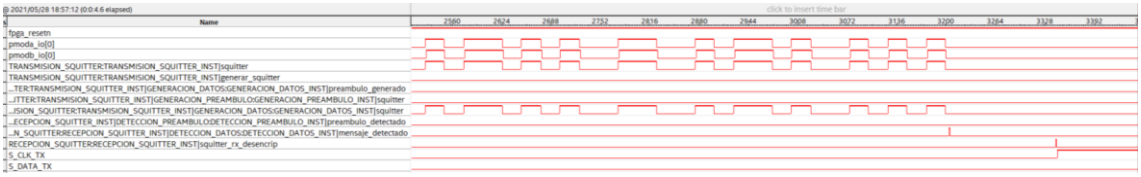


Ilustración 75. Captura de pantalla de la transmisión y recepción de squitter ampliado final

Explicación de las capturas de simulación obtenidas.

Se puede observar la parte de la transmisión y recepción del mismo squitter en los pines de entrada y salida de la entidad superior definidos como pines PMOD en la entidad.

Referente a la transmisión. Se puede identificar las dos señales con las que se genera el squitter, la primera asociada al preámbulo y la segunda asociada a los datos. De manera conjunta a estas dos señales, se pueden identificar las señales referentes a la generación del preámbulo y la referente al preámbulo generado con la que se genera la transmisión de los datos.

Referente a la recepción. Se puede identificar la señal asociada al pin PMOD definido como entrada, de manera conjunta a esta señal (squitter), aparecen también las señales referentes a las detecciones del preámbulo y del mensaje.

La captura del entorno de desarrollo también identifica el flanco asociado a la señal que marca la correcta desencriptación de los tres paquetes de datos recibidos.
Con los tres mensajes desencriptados y activada la señal asociada, se activa el proceso referente a la comunicación serie entre la FPGA y el microcontrolador, con las señales de reloj y datos.

- ***Captura de pantalla de los procesos de encriptación y desencriptación.*** *Estas capturas ofrecen una visualización del entorno de desarrollo en conjunto de la encriptación y transmisión del squitter, junto con la desencriptación y recepción.*

Las primeras capturas de simulación ofrecen una visualización conjunta de la parte de la encriptación y transmisión y la parte de la desencriptación y recepción.

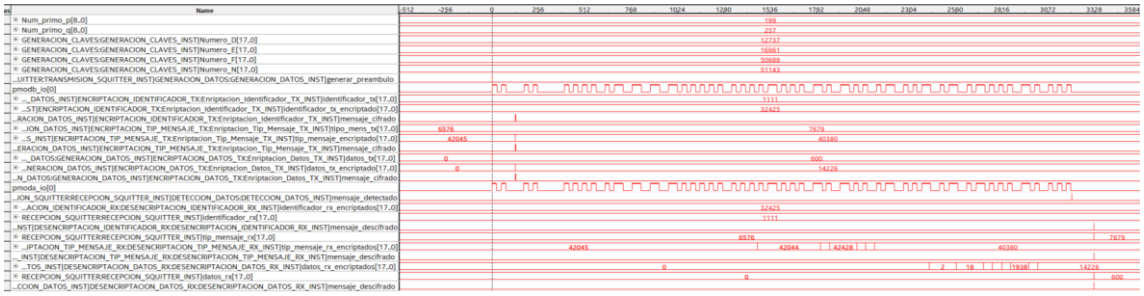


Ilustración 76. Captura de pantalla de los procesos de encriptación y desencriptación (1)

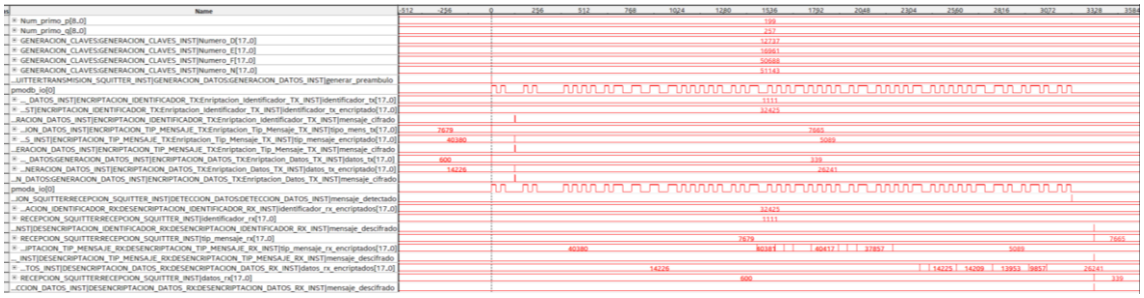


Ilustración 77. Captura de pantalla de los procesos de encriptación y desencriptación (2)

Destacar de estas primeras figuras, únicamente el funcionamiento desde un alto nivel, destacando los siguientes detalles conceptuales.

- *La encriptación termina mucho antes de que termine el preámbulo, incluso antes de generar la mitad de este.*
- *Los mensajes encriptados y desencriptados coinciden, además de los mensajes en texto plano transmitidos y recibidos.*
- *Los flags asociados a la encriptación y desencriptación coindicen exactamente con la actualización de los valores de las señales encriptadas y desencriptadas respectivamente.*

De manera conjunta a las primeras figuras, se ofrece un zoom al inicio y al final de la última captura de simulación, para una mejor visualización de ambas partes del proyecto comprendiendo así, mejor las encriptaciones y desencriptaciones.

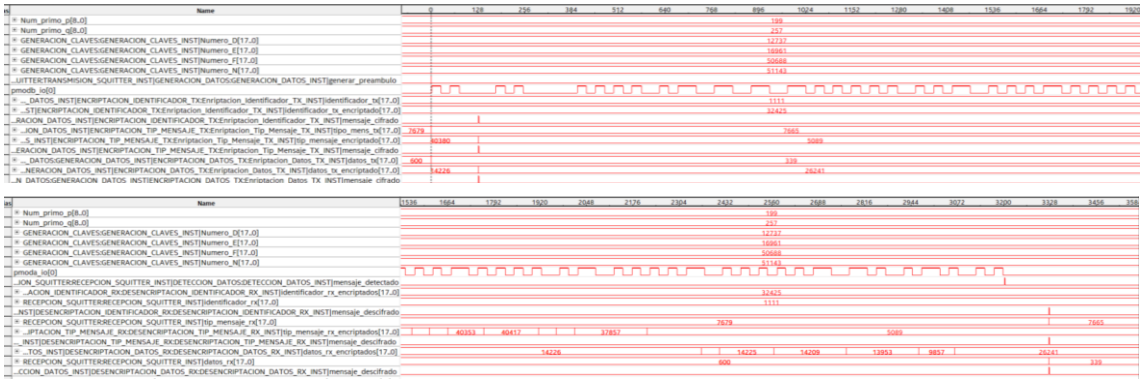


Ilustración 78. Captura de pantalla de los procesos de encriptación y desencriptación zoom inicio y fin

Si se profundiza un poco más en los procesos de encriptación y desencriptación de las respectivas entidades, se encuentran los procesos referenciados a la aritmética modular y la exponenciación binaria, aplicando el método de los cuadrados repetitivos.

Es importante recordar que los procesos de encriptación y desencriptación tienen la misma lógica, lo único que se cambia es el mensaje introducido y el obtenido junto con la respectiva clave utilizada para cifrar o descifrar. Habiendo un total de tres entidades encriptadoras (una por cada mensaje) y otras tres entidades desencriptadoras (al igual que en la encriptación, una por cada mensaje).

Únicamente se analiza el funcionamiento de una entidad elegida completamente al azar. Se elige la entidad "ENCRIPTACION_DATOS_TX" mediante la cual el proyecto realiza la encriptación de datos transmitidos, en concreto se estudia y captura, la encriptación del valor inicial de latitud, con un valor numérico de 385.

La etapa de encriptación se divide en dos procesos síncronos. El primero se activa con la señal de generación de preámbulo y el segundo según termina el primero. El primer proceso tiene como objetivo, almacenar en un vector todos los valores obtenidos por el método de los cuadrados repetitivos, el segundo realiza el producto de los que aparecen según la base binaria del exponente de cifrado.

- *Primer proceso de encriptación.*

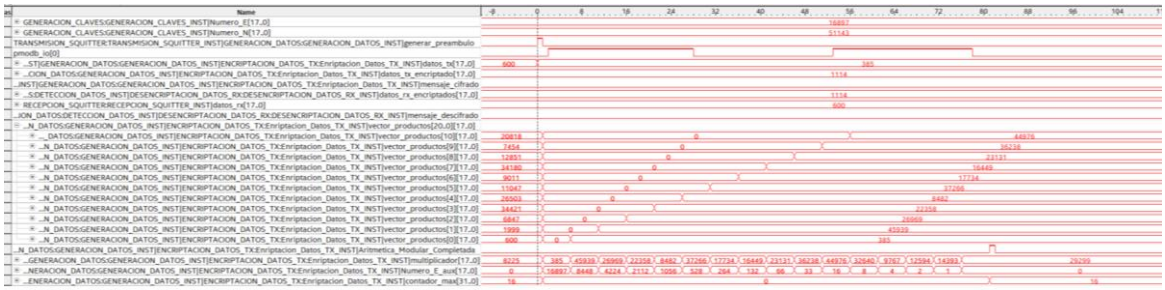


Ilustración 79. Captura de pantalla del proceso de encriptación de datos (1).

Destacar de este proceso como comienza la encriptación con la señal de generación del preámbulo, como según avanzan el tiempo se va rellenando el vector con los valores obtenidos en el multiplicador. Se pueden comprobar como las soluciones introducidas en el vector también son valores coherentes. El proceso termina cuando el numero E auxiliar llega al valor cero, almacenando el número de cuentas en el contador asociado y activando la señal "Aritmetica_Modular_Completa".

El algoritmo implementado se muestra en el siguiente diagrama de flujos.

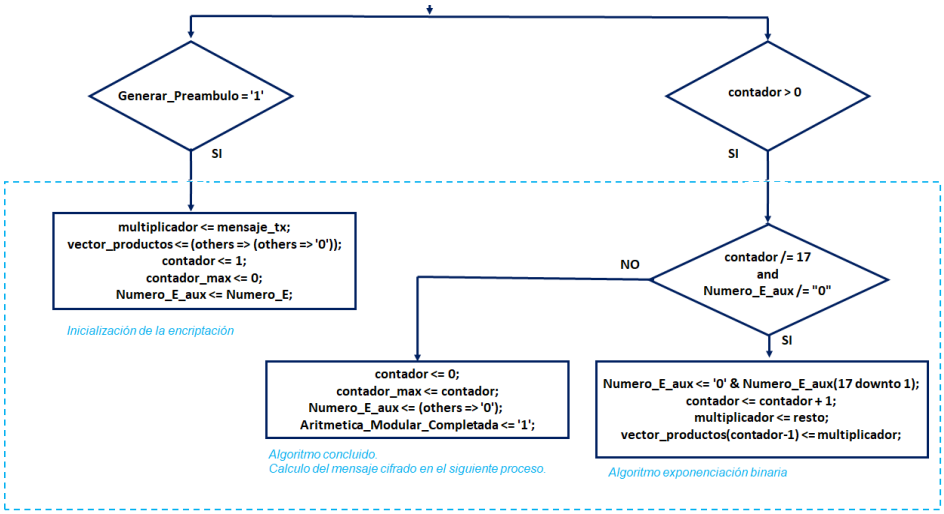


Ilustración 80. Flujograma del algoritmo de encriptación parte 1.

Iteración 1.

$$Vector\_Productos(0) = 385 \;\; Valor\ inicial$$

Iteración 2.

$$Vector\_Productos(1) = 385^2 \; mod(51143) \; = 45939$$

Iteración 3.

$$Vector\_Productos(1) = 45939^2 \; mod(51143) \; = 26969$$

Iteración 4.

$$Vector\_Productos(1) = 26969^2 \; mod(51143) \; = 22358$$

Iteración …….

- *Segundo proceso de encriptación.*

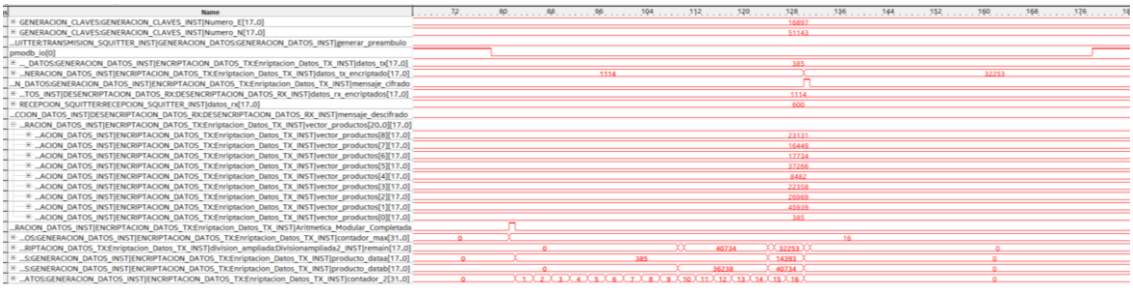


Ilustración 81. Captura de pantalla del proceso de encriptación de datos (2).

Destacar de este proceso síncrono, el comienzo con la finalización del proceso anterior, mediante la señal "Aritmetica_Modular_Completa". El número de iteraciones realizadas se encuentra condicionado al valor máximo del contador del proceso anterior (para este caso un valor 16). Ilustrar como el algoritmo inicializa ambos multiplicadores de datos y como actualiza los valores con el resto anterior y con el nuevo valor del vector.

Aparece un problema de espacio en la FPGA y se han eliminado las posiciones más altas del vector calculado en el proceso anterior.

El algoritmo implementado se muestra en el siguiente diagrama de flujos.

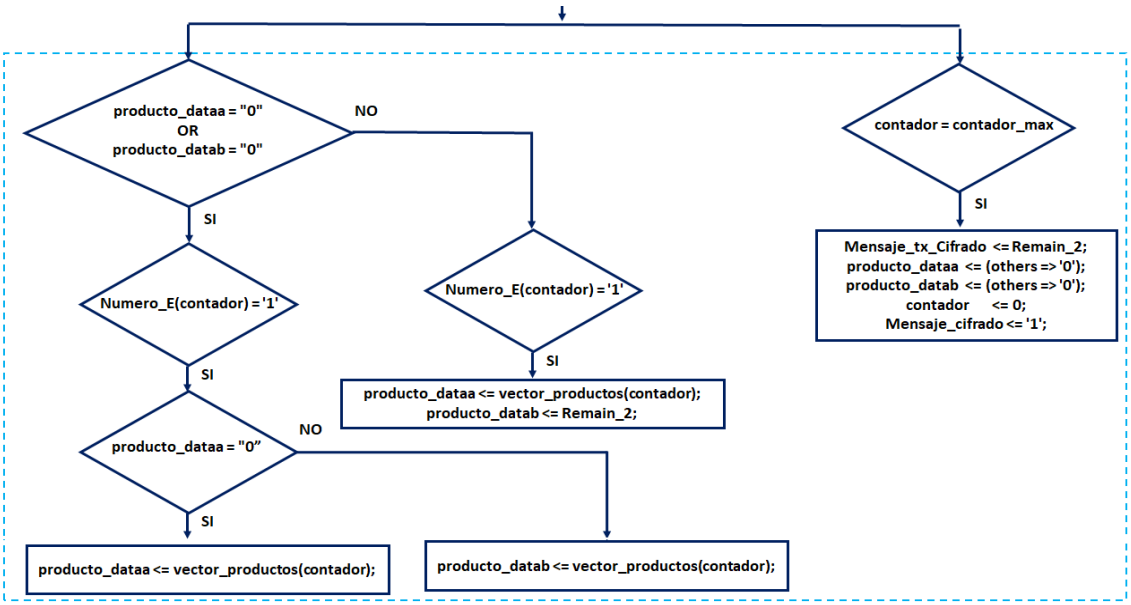


Ilustración 82. Flujograma del algoritmo de encriptación parte 2.

Numero E = 16.897 decimal = 100 0010 0000 0001 binario

Iteración 1.

$$producto\_dataa = 385\ \ Numero\ E\ (0)$$
$$producto\_datab = 0\ \ Valor\ inicial$$

Iteración 2.

$$producto\_dataa = 385\ \ Numero\ E\ (0)$$
$$producto\_datab = 36.238\ \ Numero\ E\ (9)$$

Iteración 3.

$$producto\_dataa = 14.393\ Numero\ E\ (9)$$
$$producto\_datab = 385 \cdot 36.238\ mod(51.143) = 40.734$$

Iteración 4.

$$solucion = datos\_tx\_encriptados = 14.393 \cdot 40.734\ mod(51.143) = 32.253$$

De forma conjunta, se visualizan ambos procesos en la siguiente captura

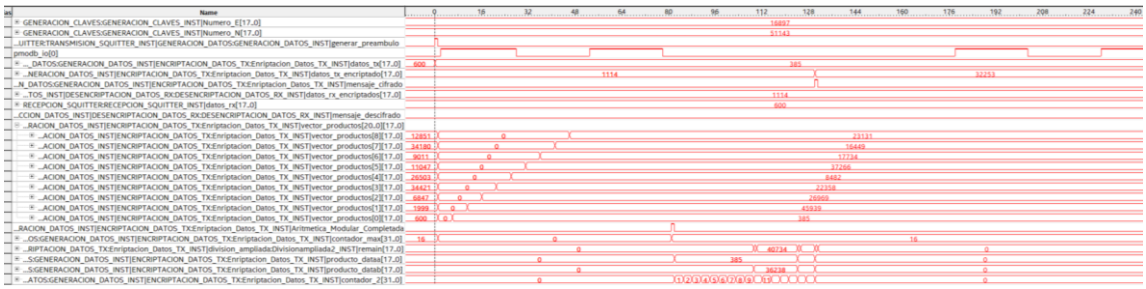


Ilustración 83. Captura de pantalla del proceso de encriptación de datos (3).

Conclusión personal de los algoritmos de encriptación y desencriptación. *Al igual que en la generación de claves, como detalle de interés, poniendo en valor el desarrollo. Me gustaría destacar la velocidad de los procesos implementados del cálculo de encriptación y desencriptación, pudiendo hacerse siempre en menos de la mitad del tiempo de transmisión de un preámbulo. Tarda aproximadamente un total de 130 ciclos de reloj, lo que equivaldría a un total aproximado de dos microsegundos y medio. Por otro lado, destacar también la potencia que se desarrolla gracias el uso de la aritmética modular, mediante estos teoremas se trabajan con números relativamente normales, de forma aproximada con el valor del Numero N, por el contrario, se tendría que elevar al exponente de cifrado el valor del dato, para hacer una estimación, en este ejemplo nos iríamos a un numero de 43.687 digitos.*

- ***Captura de pantalla de los procesos de interfaz con el microcontrolador.*** *Estas capturas ofrecen una visualización del entorno de desarrollo de la transmisión y recepción del puerto serie mediante el que se conectan los micros y la FPGA.*

Es importante recordar cómo se ha comentado previamente, que para conseguir una comunicación entre la FPGA y el microcontrolador es necesario adaptar el dispositivo más rápido a la velocidad del más lento. Este detalle se traduce en la necesidad de tener que introducir ciclos de espera en la FPGA con un contador de valores de mil ciclos de reloj. Sera de especial interés este detalle, para poder entender que, debido a la limitación de la FPGA con el número máximo de profundidad de muestras tomadas con el proyecto cargado, y sobretodos, los ciclos de espera comentados previamente, va a resultar imposible una visualización completa de la transmisión.

Respecto a los pines de entrada:	***Respecto a los pines de salida:***
S_CLK_RX <= pmodb_io(1);	**pmoda_io(1) <= (S_CLK_TX or S_CLK_CLAVES);**
S_DATA_RX <= pmodb_io(2);	**pmoda_io(2) <= (S_DATA_TX or S_DATA_CLAVES);**

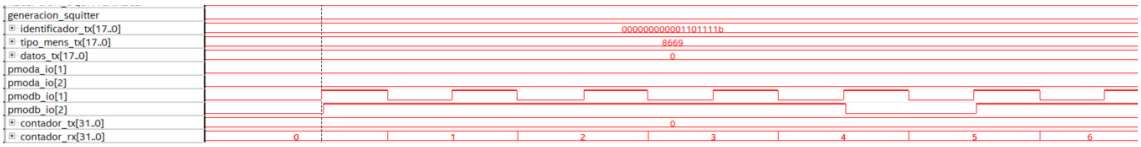


Ilustración 84. Captura de pantalla de los procesos de interfaz con el microcontrolador ampliado inicio

Ilustración 85. Captura de pantalla de los procesos de interfaz con el microcontrolador

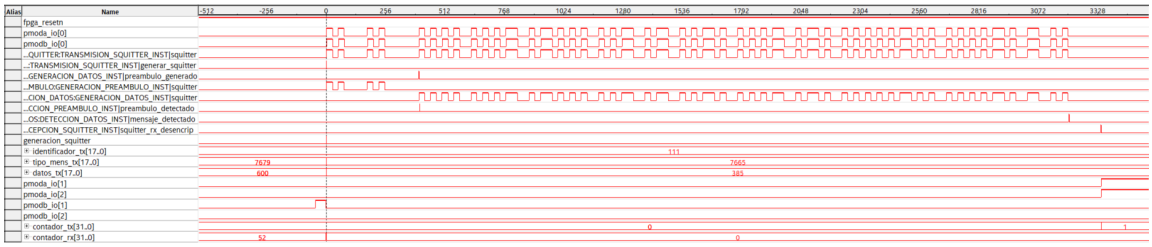


Ilustración 86. Captura de pantalla de los procesos de interfaz con el microcontrolador ampliado

Explicación de las capturas de simulación obtenidas.

En la primera captura de simulación, se puede observar el inicio de una transmisión desde el microcontrolador. Destacaría en el comienzo, la visualización de la correcta recepción de la parte más baja del identificador además de un correcto funcionamiento del protocolo, realizando escrituras desde el microcontrolador en los flancos de subida de la señal de reloj y lecturas por parte de la FPGA en los flancos de bajada del mismo.

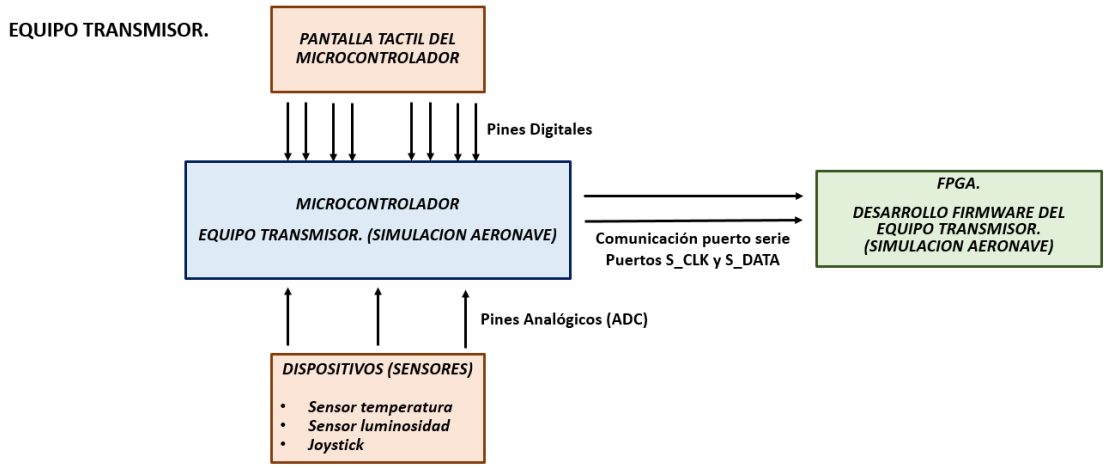
En la segunda y tercera captura de simulación, se visualiza un correcto funcionamiento del protocolo conjunto. Es decir, cuando se recibe por completo los tres paquetes desde el microcontrolador transmisor en la FPGA, la FPGA transmisora genera la transmisión del squitter, por otro lado, la FPGA receptora recibe el squitter, detecta el mensaje por completo, lo desencripta y comienza la transmisión por el puerto serie hacia el microcontrolador receptor. De especial interés son las señales intermedias, mediante las cuales se relacionan todos los procesos.

5. Diseño y desarrollo de la interface externa de visualización junto con el control de la aeronave desarrollado en el microcontrolador.

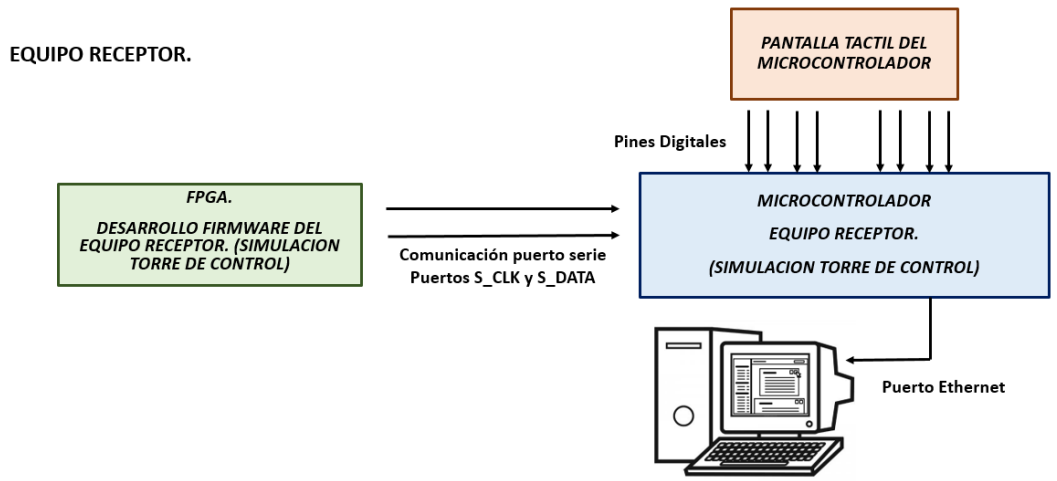
Para este capítulo se realiza por separado el análisis del microcontrolador ubicado en el equipo transmisor, cuya principal función es la de ofrecer al usuario un control de la aeronave y del microcontrolador ubicado en el equipo receptor, cuya principal función es ofrecer en un entorno de visualización dinámico un control del espacio aéreo al usuario.

A modo de introducción de este capítulo, capitulo relacionado al desarrollo software del microcontrolador, se realiza un diagrama de bloques de ambas etapas.

- **Etapa de diseño de transmisor. (Simulación aeronave).** Para esta etapa se conecta la pantalla táctil al microcontrolador por una serie de pines digitales, los sensores analógicos por los pines referentes al ADC, la conexión a la FPGA se realiza mediante pines de propósito general.



- **Etapa de diseño de receptor. (Simulación torre de control).** Para esta etapa se conecta la FPGA con el microcontrolador por pines de propósito general, la pantalla táctil al microcontrolador igual que la transmisión, por último, mediante una conexión ethernet, se conecta el microcontrolador a un ordenador.



5.1 Equipo transmisor. Control de la aeronave

El equipo transmisor formado por un microcontrolador conectado a la FPGA, tiene como principal objetivo realizar el interfaz entre el usuario y la FPGA. Mediante un joystick y una pantalla táctil el usuario puede controlar la aeronave indicando a la FPGA que procesos tiene que activar y en consecuencia que datos debe de encriptar y transmitir.

Este proyecto consiste en un conjunto de funciones e interrupciones periódicas que se habilitan y deshabilitan en función del estado en el que se encuentra el equipo. El programa principal se encuentra constituido por un bucle while infinito, en el que se van produciendo las interrupciones periódicas habilitadas en el estado en el que se encuentre el equipo transmisor.

Para ofrecer una idea global del proyecto, se realiza un análisis desde un alto nivel del funcionamiento global del microcontrolador.

En primer lugar, según arranca el microcontrolador, la función main del sistema inicializa el Statechart del sistema, los pines utilizados y los cuatro temporizadores. El resto de la función main, es un bucle while infinito que evalúa constantemente la situación en la que se encuentra el Statechart.

El estado inicial del Statechart ofrece al usuario la elección de los números primos utilizados en el algoritmo RSA, cuando el usuario ha elegido los números primos, se activa el temporizador número 1 y se envían por el puerto serie a la FPGA para que genere las claves públicas y privadas. Concluida la transmisión, el sistema apaga el temporizador y el Statechart avanza al siguiente estado que ofrece al usuario la elección del identificador de la aeronave, cuando el usuario ha confirmado el identificador elegido, el Statechart avanza a un estado en el que permite al usuario realizar un control total de la aeronave. En paralelo se activa el temporizador número 2, este temporizador envía de forma periódica activando y desactivando al primer temporizador, las tramas con los valores de longitud, latitud, velocidad y altitud. Cada vez que el microcontrolador envía una trama periódica, debida a una interrupción completa del temporizador número 2 se actualizan los valores de longitud y latitud mediante cálculos trigonométricos de la velocidad y la orientación.

Estos nuevos estados del Statechart ofrecen al usuario, la posibilidad de controlar la aeronave, pudiendo modificar el rumbo y la velocidad a su gusto mediante un joystick, leído en los pines del ADC y asignado a las respectivas variables globales en la handler del temporizador número 0. Mediante los diferentes botones que aparecen en la pantalla táctil se abre un árbol de expansión que ofrece un abanico de posibilidades de mensajes a enviar por el usuario.

El sistema está preparado para cambiar el identificador de la aeronave, apagando la aeronave simulada, almacenando los datos referentes a esta por si se quiere retomar en un futuro y arrancando con un nuevo valor de identificador a elegir por el usuario. En caso de retomar un identificador trabajado previamente, el Statechart inicializa el tercer temporizador y envía secuencialmente todos los valores que tiene almacenados respectivos a este identificador.

Para terminar este epígrafe, se realiza un análisis entre las funciones e interrupciones más importantes de cara al código desarrollado en el microcontrolador de este equipo, para en la siguiente parte del epígrafe, analizar cada uno de estas secciones de una forma más exhaustiva.

- **ESTRUCTURAS DE LAS PANTALLAS DE LA FSM.**

Mediante una librería, se calibra y detecta las posibles pulsaciones en la pantalla táctil del dispositivo. La parte desarrollada es el interfaz de los cuadros que aparecen en la pantalla táctil, junto con las funciones asignadas a cada zona previamente definida con el objetivo de que el usuario sepa donde presionar, en función de lo que quiera que haga la aeronave.

- **INICIALIZACION DE LOS PINES GPIO.**

La comunicación entre el microcontrolador y la FPGA se realiza mediante un conjunto de pines definidos previamente como pines de propósito general. La lectura de los pines conectados a los diferentes sensores utilizados en el sistema, se lleva a cabo mediante pines dedicados del dispositivo a la conversión analógico digital (ADC).

- **INICIALIZACION DE LOS TEMPORIZADORES TIMMERS.**

Este fragmento del proyecto tiene como objetivo configurar los temporizadores de los que dispone el microcontrolador, para que realice interrupciones periódicas con un periodo de interrupción previamente asignado. La configuración se realiza mediante un conjunto de registros configurándose un total de cuatro temporizadores.

- **CALCULOS DE LAS POSICIONES DE LONGITUD Y LATITUD.**

Mediante trigonometría, conocido el ángulo y el módulo de un vector, siendo estos valores los asociados a la dirección y velocidad de la aeronave, se puede calcular la descomposición de este vector en dos componentes, asociados a los incrementos de los valores de longitud y latitud de la aeronave.

- **MAQUINA DE ESTADOS DEL MENÚ.**

Debido a la gran dimensión del proyecto, se implementa un Statechart del sistema, el usuario avanza y retrocede entre los diferentes estados del Statechart en función de las zonas presionadas en la pantalla táctil, las cuales han sido previamente definidas.

- **TIMER0_IRQHANDLER**

La función handler de este temporizador, se utiliza para realizar las actualizaciones de las lecturas de los pines del ADC en las respectivas variables globales del código.

- **TIMER1_IRQHANDLER**

La función handler de este temporizador, se utiliza para realizar la transmisión serie desde el microcontrolador a la FPGA, configura los pines asociados al reloj y a los datos, transmitiendo a la FPGA el squitter que esta última deberá de encriptar y transmitir.

- **TIMER2_IRQHANDLER**

La función handler de este temporizador, se utiliza para indicar a la handler del TIMER1 en que momento tiene que comenzar la transmisión por el puerto serie, es decir, la principal función de este temporizador es llevar el control de la transmisión automática y periódica de squitters.

- **TIMER3_IRQHANDLER**

La función handler de este temporizador, se utiliza para realizar actualización de los datos de una aeronave, que ya se haya inicializado previamente, indicando al TIMER1 que transmita de forma secuencial y en un orden, todos los parámetros asociados al identificador de la aeronave que se acaba de inicializar. La necesidad de esta transmisión, aparece debido a que el receptor no conoce el estado de los mensajes de carácter especial enviados previamente por la aeronave ya que se recuerda, que este tipo de mensaje no se envía de forma periódica.

Con una idea global del proyecto definida. La siguiente parte de este apartado de la memoria, analiza más en detalle cada una de las partes citadas en el apartado.

5.1.1. Estructuras de las pantallas de la FSM.

Mediante una librería, se calibra y detecta las posibles pulsaciones en la pantalla táctil del dispositivo. La parte desarrollada es el interfaz de los cuadros que aparecen en la pantalla táctil, mediante una estructura, se define una zona de la pantalla estructurada en forma de cuadrado. Esta estructura contiene los valores de inicio y tamaño en coordenadas cartesianas *(x, y)*. Estas estructuras contienen funciones asignadas con el objetivo de que el usuario sepa donde presionar, en función de lo que quiera que haga la aeronave, estas estructuras permiten una comunicación con el microcontrolador y este posteriormente y de forma invisible al usuario con la FPGA.

La transición de los estados del Statechart se condiciona a las estructuras presionadas por el usuario en la pantalla táctil. En la siguiente imagen, se ilustran algunas de las pantallas diseñadas mediante estas estructuras, es importante destacar para una mejor comprensión, que cada pantalla se encuentra asociada a un estado del Statechart.

Se resume la configuración en las siguientes imágenes.

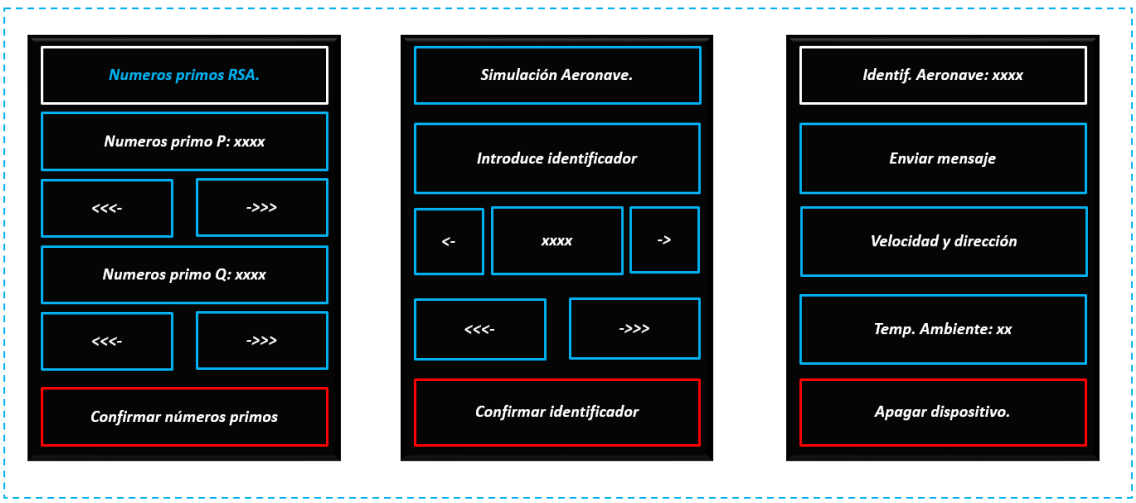


Ilustración 87. Ejemplos de los cuadros de dialogo de la pantalla táctil (1)

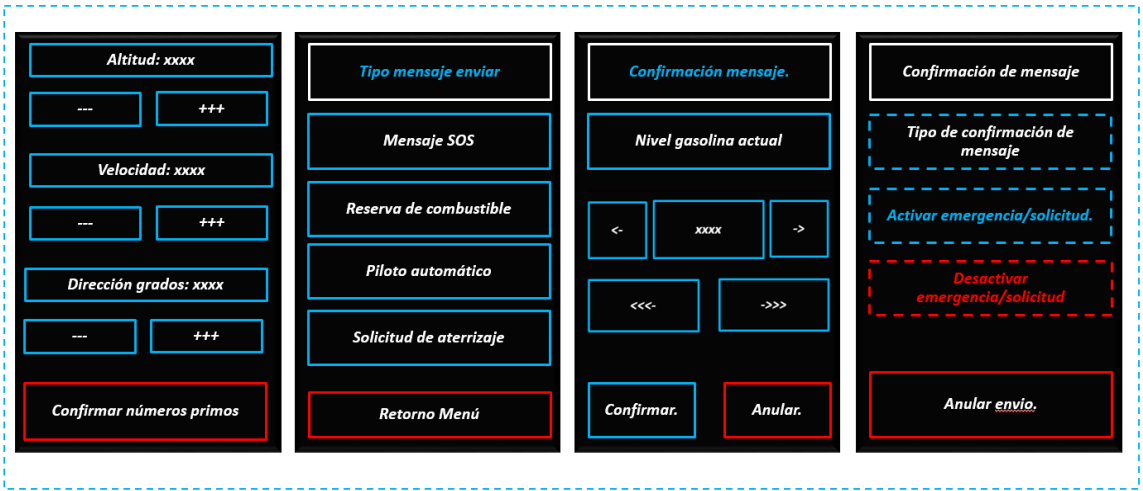


Ilustración 88. Ejemplos de los cuadros de dialogo de la pantalla táctil (2)

5.1.2. Inicialización de los pines GPIO y dedicados del ADC.

Para la parte de la configuración de los pines de entrada y salida del microcontrolador, por un lado, los pines de salida, los cuales se configuran los pines de propósito general con el objetivo de realizar la comunicación serie con la FPGA, por otro lado, los pines de entrada, los cuales se configuran como entradas de ADC.

Estas configuraciones se consiguen atacando a los registros asociados a los pines, para ello es necesario trabajar realizando operaciones a nivel de bit, al más bajo nivel posible.

Analizando el manual del dispositivo ofrecido por el fabricante, se pueden destacar los siguientes registros de configuración de pines, con la respectiva captura ofrecida por el fabricante del microcontrolador.

Registro de selección de función de pin (PINSELx). Los registros PINSEL controlan las funciones de los pines del dispositivo. Cada pin está relacionado con dos bits de cada registro. Cada par de bits corresponden a un pin especifico.

Pin function select register bits

PINSEL0 to PINSEL9 Values	Function	Value after Reset
00	Primary (default) function, typically GPIO port	00
01	First alternate function	
10	Second alternate function	
11	Third alternate function	

Ilustración 89. Registro de selección de función de pin (PINSELx).

Registro de selección de modo pin (PINMODEx). Este registro controla el modo de entrada de todos los puertos. Con este registro se puede conectar a la entrada de cada pin una resistencia interna de *pull-up*, de *pull-down*, ninguna de las dos resistencias o por ultimo seleccionar el modo repetidor (esto permite mantener el último valor aplicado).

Pin Mode Select register Bits

PINMODE0 to PINMODE9 Values	Function	Value after Reset
00	Pin has an on-chip pull-up resistor enabled.	00
01	Repeater mode (see text below).	
10	Pin has neither pull-up nor pull-down resistor enabled.	
11	Pin has an on-chip pull-down resistor enabled.	

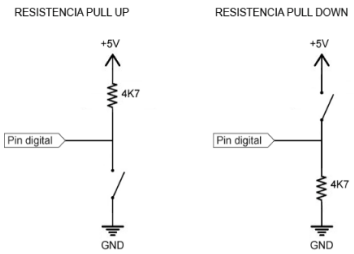


Ilustración 90. Registro de selección de modo pin (PINMODEx).

Registro de selección de modo pin (PINMODE_ODx). Este registro se utiliza para configurar la salida del pin en formato colector abierto o por el contrario en forma normal.

Open Drain Pin Mode Select register Bits

PINMODE_OD0 to PINMODE_OD4 Values	Function	Value after Reset
0	Pin is in the normal (not open drain) mode.	00
1	Pin is in the open drain mode.	

Ilustración 91. Registro de selección de modo pin (PINMODE_ODx).

Puertos definidos como GPIO. Los puertos que previamente han sido definidos como GPIO se controlan mediante los siguientes registros.

GPIO. Registro de Dirección (FIOxDIR) El contenido de este registro determina si los pines que funcionan como GPIOs son de entrada o de salida.

Fast GPIO port Direction register FIO0DIR to FIO4DIR - addresses 0x2009 C000 to 0x2009 C080) bit description

Bit	Symbol	Value	Description	Reset value
31:0	FIO0DIR FIO1DIR FIO2DIR FIO3DIR FIO4DIR		Fast GPIO Direction PORTx control bits. Bit 0 in FIOxDIR controls pin Px.0, bit 31 in FIOxDIR controls pin Px.31.	0x0
		0	Controlled pin is input.	
		1	Controlled pin is output.	

Ilustración 92. Registro de Dirección (FIOxDIR)

GPIO. Registro de Lectura y escritura. Registro de Datos (FIOxPIN) La lectura de éste registro entrega el valor del pin al que hace referencia cada bit. La escritura en este puerto transmite el valor escrito en cada bit a su pin correspondiente

Fast GPIO port Pin value byte and half-word accessible register description

Generic Register name	Description	Register length (bits) & access	Reset value	PORTn Register Address & Name
FIOxPIN0	Fast GPIO Port x Pin value register 0. Bit 0 in FIOxPIN0 register corresponds to pin Px.0 … bit 7 to pin Px.7.	8 (byte) R/W	0x00	FIO0PIN0 - 0x2009 C014 FIO1PIN0 - 0x2009 C034 FIO2PIN0 - 0x2009 C054 FIO3PIN0 - 0x2009 C074 FIO4PIN0 - 0x2009 C094

Ilustración 93. Registro de Lectura y Escritura. Registro de Datos (FIOxPIN)

GPIO. Registro de Escritura. Registro de Set (FIOxSET) Este registro se utiliza para escribir un nivel alto en los pies del puerto configurados como salida.

Fast GPIO port output Set register (FIO0SET to FIO4SET - addresses 0x2009 C018 to 0x2009 C098) bit description

Bit	Symbol	Value	Description	Reset value
31:0	FIO0SET FIO1SET FIO2SET FIO3SET FIO4SET		Fast GPIO output value Set bits. Bit 0 in FIOxSET controls pin Px.0, bit 31 in FIOxSET controls pin Px.31.	0x0
		0	Controlled pin output is unchanged.	
		1	Controlled pin output is set to HIGH.	

Ilustración 94. Registro de Escritura. Registro de Set (FIOxSET)

GPIO. Registro de Escritura. Registro de CLEAR (FIOxCLR) Este registro se utiliza para escribir un nivel bajo en los pines del puerto configurados como salida.

Fast GPIO port output Clear register (FIO0CLR to FIO4CLR- addresses 0x2009 C01C to 0x2009 C09C) bit description

Bit	Symbol	Value	Description	Reset value
31:0	FIO0CLR FIO1CLR FIO2CLR FIO3CLR FIO4CLR		Fast GPIO output value Clear bits. Bit 0 in FIOxCLR controls pin Px.0, bit 31 controls pin Px.31.	0x0
		0	Controlled pin output is unchanged.	
		1	Controlled pin output is set to LOW.	

Ilustración 95. Registro de Escritura. Registro de CLEAR (FIOxCLR)

Es importante destacar que como se ha comentado previamente, se acceden a los bits de los registros de configuración, por lo que aparece una pequeña complejidad en la necesidad de trabajar realizando operaciones lógicas de desplazamientos, productos y sumas a nivel de bit.

Realizado un análisis por los diferentes registros de configuración de los pines. Se analizan por separado los pines de salida de propósito general GPIO y los pines de entrada dedicados ADC.

- **Pines de salida de propósito general GPIO. Comunicación serie con FPGA**

Para realizar la configuración de los pines que realizan la comunicación serie, en concreto el pin dedicado a la transmisión de datos *SDA* y el pin dedicado a la transmisión de reloj *SCL*, se realizan los siguientes pasos. En primer lugar, se accede a los registros de selección de función de pin (PINSELx) con el objetivo de definir los pines como GPIO, al ser pines de salida, no es necesario definir resistencias internas asociadas. Una vez definidos como propósito general, mediante el registro de Dirección (FIOxDIR) se define el pin como salida. Para concluir, se inicializan a niveles bajo ambos pines.

Se resume la configuración en el siguiente cuadro resumen.

```
LPC_PINCON->PINSELx &= ~(3<< SDA);
LPC_PINCON->PINSELx &= ~(3<< SCL);
LPC_GPIOx->FIODIR|=(1<<SDA)|(1<<SCL);
LPC_GPIOx->FIOCLR=(1<<SCL);
LPC_GPIOx->FIOCLR=(1<<SDA);
```

Más adelante, para realizar la escritura en los pines utilizados en la comunicación serie se utiliza el registro de Set (FIOxSET) y el registro de CLEAR (FIOxCLR).

Se resume las sentencias de escritura en el siguiente cuadro resumen.

```
LPC_GPIOx->FIOSET=(1<<SCL);  -> Nivel Alto en SCL
LPC_GPIOx->FIOSET=(1<<SDA);  -> Nivel Alto en SDA
LPC_GPIOx->FIOCLR=(1<<SCL);  -> Nivel Bajo en SCL
LPC_GPIOx->FIOCLR=(1<<SDA);  -> Nivel Bajo en SDA
```

- **Pines de entrada dedicados ADC. Lectura de los sensores.**

En primer lugar, destacar que el ADC se encuentra configurado en modo ráfaga, este detalle, ofrece una lectura de los pines dedicados analógicos, sin cargar de trabajo la CPU del microcontrolador. Mediante el registro LPC_ADC->ADCR se habilitan cuatro canales analógicos que ofrece el microcontrolador. La idea es que, de forma continua,

se realice la lectura del pin analógico almacenando su valor en los registros asociados del ADC, para que en la interrupción asociada al temporizador número 0 se actualicen las variables globales asociadas a estos parámetros. Por otro lado, destacar que el brillo de la pantalla del microcontrolador, es proporcional al ciclo de trabajo de una señal PWM generada de manera interna en el propio microcontrolador.

Los dispositivos conectados a los cuatro canales de los pines asociados a la conversión son los siguientes. En el canal 0 el sensor de temperatura LM35, en el canal 1 el sensor de iluminación LDR y en los canales 2 y 3 los valores *Vx* y *Vy* del joystick.

En la siguiente tabla ofrecida por el fabricante, se muestran los pines utilizados con su respectiva configuración en el registro de selección de función de pin (PINSELx) para la conversión analógica.

Pin function select register 1 (PINSEL1 - address 0x4002 C004) bit description

PINSEL1	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
1:0	P0.16	GPIO Port 0.16	RXD1	SSEL0	SSEL	00
3:2	P0.17	GPIO Port 0.17	CTS1	MISO0	MISO	00
5:4	P0.18	GPIO Port 0.18	DCD1	MOSI0	MOSI	00
7:6	P0.19[1]	GPIO Port 0.19	DSR1	Reserved	SDA1	00
9:8	P0.20[1]	GPIO Port 0.20	DTR1	Reserved	SCL1	00
11:10	P0.21[1]	GPIO Port 0.21	RI1	Reserved	RD1	00
13:12	P0.22	GPIO Port 0.22	RTS1	Reserved	TD1	00
15:14	P0.23[1]	GPIO Port 0.23	AD0.0	I2SRX_CLK	CAP3.0	00
17:16	P0.24[1]	GPIO Port 0.24	AD0.1	I2SRX_WS	CAP3.1	00
19:18	P0.25	GPIO Port 0.25	AD0.2	I2SRX_SDA	TXD3	00
21:20	P0.26	GPIO Port 0.26	AD0.3	AOUT	RXD3	00
23:22	P0.27[1][2]	GPIO Port 0.27	SDA0	USB_SDA	Reserved	00

Ilustración 96. Registro de selección de función de pin (PINSELx) de los pines ADC.

Para realizar la configuración de los pines asociados a las lecturas de los dispositivos conectados al microcontrolador, se realizan los siguientes pasos. En primer lugar, se accede a los registros de selección de función de pin (PINSELx) con el objetivo de definir los pines previamente elegidos como pines analógicos ADC. Al tratarse de pines de entrada, es necesario definir resistencias internas asociadas mediante el registro de selección de modo pin (PINMODEx). En este caso, al ser pines dedicados predefinidos por el microcontrolador, se decide no introducir ningún tipo de resistencia interna.

Se resume la configuración en el siguiente cuadro resumen.

```
LPC_PINCON->PINSELx|= (1<< LM35);
LPC_PINCON->PINSELx|= (1<<LDR);
LPC_PINCON->PINSELx|= (1<<Vx_ JOYSTICK);
LPC_PINCON->PINSELx|= (1<<Vy_ JOYSTICK);
LPC_PINCON->PINMODEx|= (2<< LM35);
LPC_PINCON->PINMODEx|= (2<< LDR);
LPC_PINCON->PINMODEx|= (2<< Vx_ JOYSTICK);
LPC_PINCON->PINMODEx|= (2<< Vy_ JOYSTICK);
```

5.1.3. Inicialización de los temporizadores timers.

Este apartado describe los registros configurados para conseguir inicializar y modelar los cuatro temporizadores utilizados en el sistema.

Como punto de partida se habilitan las interrupciones de los temporizadores, asignando un nivel de prioridad a cada interrupción. La estructura es la siguiente.

```
NVIC_EnableIRQ (TIMERx_IRQn);
NVIC_SetPriority (TIMERx_IRQn, Nivel_Prioridad);
```

Cada temporizador tiene asociado una serie de registros para ofrecer al desarrollador un control de las propiedades del mismo, entre estos registros destacan los siguientes, se ofrece una captura del manual del dispositivo ofrecido por el fabricante.

Generic Name	Description	Access	Reset Value[1]	TIMERn Register/ Name & Address
IR	Interrupt Register. The IR can be written to clear interrupts. The IR can be read to identify which of eight possible interrupt sources are pending.	R/W	0	T0IR - 0x4000 4000 T1IR - 0x4000 8000 T2IR - 0x4009 0000 T3IR - 0x4009 4000
TCR	Timer Control Register. The TCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the TCR.	R/W	0	T0TCR - 0x4000 4004 T1TCR - 0x4000 8004 T2TCR - 0x4009 0004 T3TCR - 0x4009 4004
TC	Timer Counter. The 32-bit TC is incremented every PR+1 cycles of PCLK. The TC is controlled through the TCR.	R/W	0	T0TC - 0x4000 4008 T1TC - 0x4000 8008 T2TC - 0x4009 0008 T3TC - 0x4009 4008
PR	Prescale Register. When the Prescale Counter (below) is equal to this value, the next clock increments the TC and clears the PC.	R/W	0	T0PR - 0x4000 400C T1PR - 0x4000 800C T2PR - 0x4009 000C T3PR - 0x4009 400C
PC	Prescale Counter. The 32-bit PC is a counter which is incremented to the value stored in PR. When the value in PR is reached, the TC is incremented and the PC is cleared. The PC is observable and controllable through the bus interface.	R/W	0	T0PC - 0x4000 4010 T1PC - 0x4000 8010 T2PC - 0x4009 0010 T3PC - 0x4009 4010
MCR	Match Control Register. The MCR is used to control if an interrupt is generated and if the TC is reset when a Match occurs.	R/W	0	T0MCR - 0x4000 4014 T1MCR - 0x4000 8014 T2MCR - 0x4009 0014 T3MCR - 0x4009 4014
MR0	Match Register 0. MR0 can be enabled through the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt every time MR0 matches the TC.	R/W	0	T0MR0 - 0x4000 4018 T1MR0 - 0x4000 8018 T2MR0 - 0x4009 0018 T3MR0 - 0x4009 4018
CTCR	Count Control Register. The CTCR selects between Timer and Counter mode, and in Counter mode selects the signal and edge(s) for counting.	R/W	0	T0CTCR - 0x4000 4070 T1CTCR - 0x4000 8070 T2CTCR - 0x4009 0070 T3CTCR - 0x4009 4070

Ilustración 97. Registros de configuración de los temporizadores

Para realizar la configuración de los temporizadores utilizados por el microcontrolador, tras habilitar y asignar un nivel de prioridad a cada interrupción se realizan los siguientes pasos para cada uno de los temporizadores utilizados.

En primer lugar, mediante el registro TCR se habilita la cuenta del temporizador asociado, en segundo lugar, el registro CTCR otorga al temporizador un trabajo en modo *timmer* en lugar de en modo *capture*, en tercer lugar, el registro MCR habilita la entrada en la handler cuando el valor del contador del temporizador coincide con el valor del MRx, también se indica un reinicio de cuenta del valor del contador, tras producirse esta interrupción, con el objetivo de que se repita de forma periódica el proceso.

Por último, se asigna los valores al PR y MR0 con el fin de ajustar el tiempo que tarda en producirse la interrupción. El reloj interno del microcontrolador, trabaja a 25 MHz, con el fin de que las interrupciones sean en valores enteros, el valor del prescaler se fija a un valor constante de 25 unidades, con un periodo de incremento de unidad en las unidades MR de un microsegundo.

Los valores de los registros MR del temporizador son los siguientes.

MR0 Timer0. *La actualización de los datos leídos desde los pines del ADC no es una lectura crítica, por lo que se fija en valor de 500.000 interrumpiendo cada medio segundo.*

MR0 Timer1. *La transmisión por el puerto serie, tampoco es especialmente crítica, la FPGA es más rápida que el microcontrolador, por lo que se asigna un valor de MR de valor 10 interrumpiendo cada diez microsegundos.*

MR0 Timer2. *El tiempo de espera, asociado a cada transmisión que se envía por el puerto serie debería de poder garantizar una transmisión de todos los datos anteriores y una correcta generación de encriptación y transmisión en la FPGA, por lo que se asigna un valor de MR de valor 250000interrumpiendo cada cuarto de segundo.*

MR0 Timer3. *Mismo razonamiento y valor que en el timer2.*

Es importante destacar, que la habilitación de la cuenta del temporizador se realiza mediante el registro TCR. El temporizador número cero, asociado a los pines del ADC es el único que se inicializa según arranca el microcontrolador. Por el contrario, los otros temporizadores, los cuales están asociados a la transmisión de squitters, se habilitan y deshabilitan en función del estado en el que se encuentre el Statechart.

La estructura interna de los tres registros utilizados de control, es la siguiente.

Registro de control de temporizador (TCR): *Sólo los dos primeros bits de este registro son útiles (el resto reservados). El bit cero habilita el contador del timer (Counter Enable) y el contador de prescale. El bit uno indica el caso de que el contador del timer y el contador de prescaler sean reseteados en el próximo flanco positivo de PCLK.*

Registro de control de cuenta (CTCR): *Sólo los dos primeros bits de este registro son útiles (el resto reservados). Los dos primeros bits sirven para especificar si funcionará como temporizador (00) o como contador. Si se da el último caso, además se especifica en qué flanco se producirá la cuenta: subida (01), bajada (10), o ambos (11).*

Registro de control de match (MCR): *Se usan sólo los 12 primeros bits, siendo cada grupo de tres bits relativo a un determinado registro de match. El primer bit de cada grupo de tres habilita que se produce una interrupción cuando ocurre un match. El segundo bit produce el reseteo del timer tras el match. El tercer bit hace que el contador del temporizador y el contador de prescaler se detengan al producirse un match.*

Se resume la configuración en el siguiente cuadro resumen. Es importante destacar, que se diferencia la asignación del registro TCR para el temporizador número 0 de los demás temporizadores.

```
LPC_TIM0->TCR = 0x01; LPC_TIMx->TCR = 0x00;
LPC_TIMx->PR = 25;
LPC_TIMx->MR0 = Valor_MR
LPC_TIMx->MCR |= 0x3;
LPC_TIM0->CTCR = 0x0;
```

Es importante destacar el registro que almacena el flag de interrupción, denominado IR. Es de especial interés borrar el flag en la entrada de la interrupción con el objetivo de que no se quede activado de forma continua.

Registro de interrupciones (IR): *Los cuatro primeros bits de éste registro se corresponden con los flags de las interrupciones de match, desde el MR0 hasta el MR3. Si se genera una interrupción como resultado de un match se reflejará con un nivel alto el bit correspondiente. Escribiendo un nivel alto se resetea la interrupción.*

Se muestran este registro en la siguiente captura del mismo resumen ofrecido por el fabricante.

Generic Name	Description	Access	Reset Value	TIMERn Register/ Name & Address
IR	Interrupt Register. The IR can be written to clear interrupts. The IR can be read to identify which of eight possible interrupt sources are pending.	R/W	0	T0IR - 0x4000 4000 T1IR - 0x4000 8000 T2IR - 0x4009 0000 T3IR - 0x4009 4000

Se resume la operación de borrado de flag en el siguiente cuadro resumen.

```
LPC_TIMx->IR |= 1 << 0;
```

5.1.4. Cálculos de las posiciones de longitud y latitud.

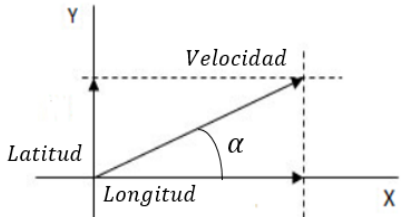
Este apartado describe las funciones que realizan los incrementos de forma automática y periódica de los valores de longitud y latitud a partir de la velocidad y dirección de la aeronave.

La función se basa en calcular la descomposición vectorial de la velocidad en dos componentes rectangulares mediante las proyecciones de un vector sobre sus respectivos ejes cartesianos aplicando conceptos trigonométricos. El proceso consiste en proyectar el vector velocidad sobre los ejes X e Y, reemplazándolo de esta manera por dos nuevos vectores asociados a las longitudes y latitudes perpendiculares entre sí que sumadas dan el vector original.

Debido a que entre el vector velocidad de la aeronave y los ejes cartesianos se forma un triángulo rectángulo, descomponer la velocidad consiste en hallar dos catetos a partir del valor de la hipotenusa y del ángulo marcado por la dirección o rumbo. Por lo tanto, como se ha comentado previamente, para llevar a cabo esta descomposición se aplican relaciones trigonométricas.

En el proyecto la componente de la longitud, se corresponde con el eje X como cateto contiguo al ángulo, mientras que la componente latitud se corresponde con el eje Y con un cateto opuesto.

En la siguiente imagen se ilustra el concepto desarrollado en este epígrafe.



$$cos(\alpha) = \frac{C.Contiguo}{Hipotenuda} \longrightarrow Longitud = Velocidad \cdot cos(\alpha)$$

$$sen(\alpha) = \frac{C.Opuesto}{Hipotenuda} \longrightarrow Latitud = Velocidad \cdot sen(\alpha)$$

Ilustración 98. Calculo de la longitud y latitud a partir de la velocidad y dirección

Se han desarrollado dos funciones, una función desarrolla el cálculo del incremento de la longitud y la otra el cálculo del incremento de la latitud. La estructura de la función es muy similar, ambas reciben como argumentos de entrada la velocidad y la dirección y retornan un valor entero, la diferencia existe la propiedad trigonométrica utilizada en la función y por lo tanto en el valor retornado. Para llevar a cabo estas funciones trigonométricas, previamente se han inicializado dos arrays con los valores de los senos y los cosenos de todos los ángulos múltiplos de 15º hasta un valor máximo de 360º, por lo que únicamente hay que realizar el producto de la velocidad por una de las posiciones de estos arrays.

En resumen, las funciones realizan las siguientes operaciones.

```
int Calcular_posicion_x (int velocidad, int ubicacion_grados)

    posicion_array = ubicacion_grados/15;
    posicion_x = (velocidad/50) * seno_alpha[posicion_array];
    return (posicion_x);
```

```
int Calcular_posicion_y (int velocidad, int ubicacion_grados)

    posicion_array = ubicacion_grados/15;
    posicion_y = (velocidad/50) * coseno_alpha[posicion_array];
    return (posicion_y);
```

5.1.5. Statechart del sistema.

El Statechart del sistema describe un diagrama con el comportamiento del sistema los diagramas de estados requieren que el sistema descrito esté compuesto por un número finito de estados que interaccionan entre ellos. Los diagramas de estado se utilizan para dar una descripción abstracta del comportamiento del sistema. Este comportamiento es analizado y representado por una serie de eventos que pueden ocurrir en uno o más estados posibles.

La lógica desarrollada consiste en tener asociados siempre pares de estados en el Statechart, es decir, por cada situación estudiada, se tienen dos estados en el Statechart de esta forma, el primer par del estado realiza las funciones de inicialización, mientras que el segundo par del estado, se queda a la espera de que ocurra un evento.

Se dispone de un total de ocho pares de estado, más un estado independiente, es decir, diecisiete estados en total. Se ilustra el funcionamiento del Statechart con un resumen de la función de cada estado y la transición entre ellos en la siguiente imagen.

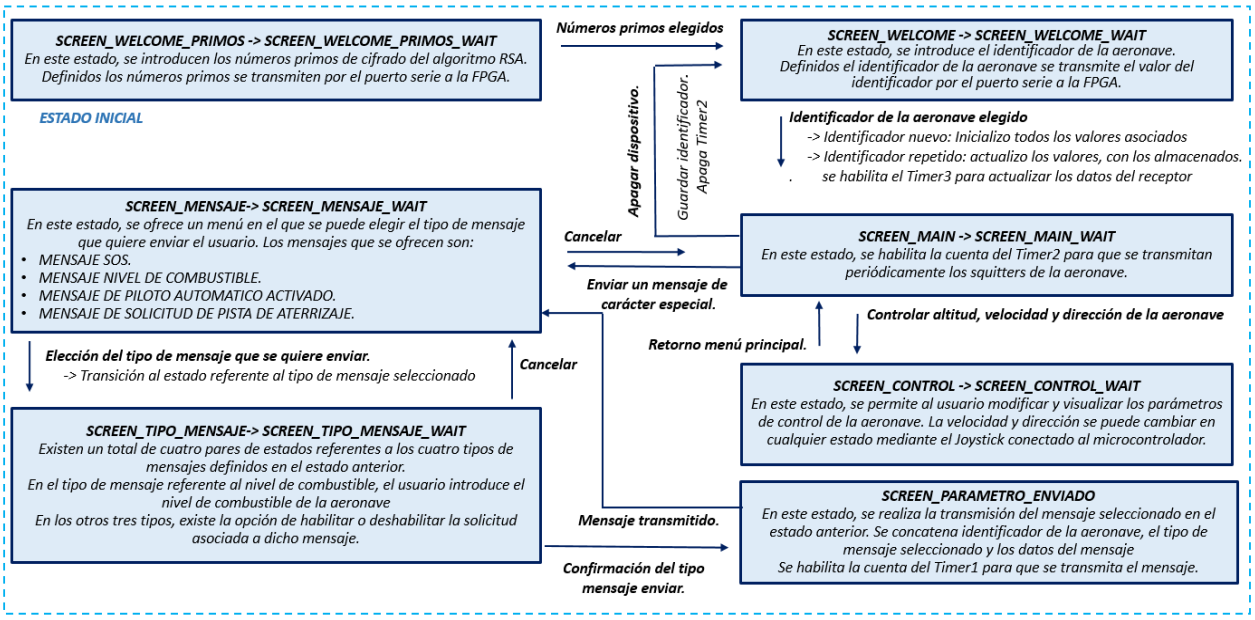


Ilustración 99. Diagrama de bloques del Statechart del sistema.

La idea en alto nivel implementada en el Statechart es la siguiente. Entre las funciones de inicialización, se encuentra una que inicializa el Statechart en el estado que pide los números primos al usuario *"SCREEN_WELCOME_PRIMOS"*, mediante los cuadros que aparecen en la pantalla táctil de este estado, se introducen los números primos elegidos para la encriptación del algoritmo RSA. Elegidos los números primos, se envían ambos a la FPGA mediante la comunicación serie y con un valor de identificador constante a cero. El estado del Statechart avanza al siguiente estado en el que se le pide al usuario el identificador de la aeronave, estado *"SCREEN_WELCOME"*, misma lógica que en el estado anterior, mediante los cuadros se elige el número de identificador. Elegido el identificador por el usuario, se compara si el identificador es nuevo y por lo tanto inicializa la aeronave en una ubicación inicial o si, por el contrario, es un identificador ya inicializado y por lo tanto se deben de cargar los valores almacenados a dicho identificador. Si la aeronave es nueva, se envía una transmisión con el identificador inicializado, mientras que si, por el contrario, la aeronave se encuentra inicializada, se actualizan los datos en el microcontrolador y se envía una transmisión con todos los valores asociados a este identificador mediante el tercer temporizador.

Una vez inicializada la aeronave con un identificador elegido por el usuario, el Statechart se encuentra en el menú principal denominado como *"SCREEN_MAIN"*. La primera instrucción de este estado es habilitar el segundo temporizador con el objetivo de realizar transmisiones periódicas con los valores de longitud, latitud, velocidad y altitud. Los valores asociados a la posición de la aeronave se actualizan en la propia handler del temporizador. En este menú, el usuario tiene la opción de poder elegir que hacer en la aeronave, ofreciendo dos opciones diferentes en el menú. En primer lugar, la opción de controlar el rumbo, velocidad y altitud de la aeronave. En segundo lugar, aparece la opción de enviar algún mensaje de carácter especial. Como última opción en la pantalla táctil se ofrece la posibilidad de apagar el dispositivo, en caso de optar por esta opción, se termina de transmitir el squitter, se apaga el temporizador y se guarda en un array el identificador con todos los parámetros que tiene actualmente con el objetivo de que si en un futuro, el usuario vuelve a introducir el mismo identificador se retome la situación actual. A continuación, se analiza y por lo tanto se avanza en el análisis de las primeras dos opciones comentadas en este estado.

En caso de querer controlar el rumbo, velocidad y altitud de la aeronave, se avanza al estado *"SCREEN_CONTROL"*. Este estado del Statechart del sistema, únicamente ofrece una nueva pantalla táctil en la que se puede modificar mediante cuadros predefinidos, los valores de altitud, velocidad y dirección. Como mejora implementada, el microcontrolador dispone de un joystick conectado, que permite modificar la velocidad y la dirección de la aeronave en cualquier estado del Statechart.

Por el contrario, en caso de querer enviar algún mensaje de carácter especial, se avanza al estado *"SCREEN_MENSAJE"*. Al igual que en el estado anterior de control, este estado del Statechart, únicamente ofrece una pantalla táctil mediante la que se puede seleccionar el tipo de mensaje que se quiere enviar. El usuario presiona en la pantalla táctil sobre el mensaje que quiere transmitir, avanzando por lo tanto al estado asociado al mensaje que se ha querido transmitir.

Entre los estados del Statechart asociados a los mensajes de carácter especial se encuentran los siguientes mensajes con sus respectivos estados asociados.

- *Mensaje de alarma SOS con el estado "SCREEN_SOS"*
- *Mensaje referente al nivel de gasolina con el estado "SCREEN_GASOLINA"*
- *Mensaje de piloto automático con el estado "SCREEN_PILOTO_AUTOMATICO"*
- *Mensaje de solitud de aterrizaje con el estado "SCREEN_ATERRIZAJE"*

En estos estados se ofrece la opción de activar (transmitiendo un nivel alto) o desactivar (transmitiendo un nivel bajo) el mensaje de carácter especial que se ha seleccionado previamente. Por último, aparece la opción de retornar al menú anterior. Para la parte de la transmisión de un mensaje (tanto para la activación o desactivación), se evalúa la posibilidad de que el sistema se encuentre en mitad de la transmisión de un squitter, en caso de que se encuentre en plena transmisión, mediante un flag asociado al mensaje se indica la solicitud de transmisión, cuando la transmisión ha concluido, se evalúa el flag y se transmite el mensaje, por el contrario, si no se está transmitiendo un squitter la transmisión es inmediata habilitando el temporizador número uno.

Para concluir cuando se genera una transmisión, el Statechart avanza hasta su ultimo estado denominado *"PARAMETRO_ENVIADO"* cuya única función es indicar al usuario que se genera la transmisión, mediante el mensaje *"Enviando Mensaje"* en la pantalla táctil durante un periodo de 1 segundo y retornando al estado *"SCREEN_MENSAJE".*

5.1.6. Temporizador TIMER0_IRQHandler

La función handler de este temporizador, se utiliza para realizar las actualizaciones de las lecturas de los valores analógicos de los pines del ADC en las respectivas variables globales del código. La primera instrucción que debe de aparecer siempre en las handler de los temporizadores es la de borrar el flag de solicitud de interrupción, para ello, se escribe un nivel alto en la posición del match que ha producido la interrupción en el registro IR del temporizador.

En la configuración del ADC se ha activado el modo ráfaga de conversión *(modo burst)* en el registro de control ADCR. Este tipo de conversión utilizada, almacena en los bits comprendidos entre las posiciones 4 y 15 de los registros ADDRx los valores de las conversiones digitales del canal *"x"* asociado.

La siguiente imagen, ofrece una captura del manual del dispositivo ofrecido por el fabricante, referente a los registros más importantes del ADC.

Generic Name	Description	Access	Reset value[1]	AD0 Name & Address
ADCR	A/D Control Register. The ADCR register must be written to select the operating mode before A/D conversion can occur.	R/W	1	AD0CR - 0x4003 4000
ADGDR	A/D Global Data Register. This register contains the ADC's DONE bit and the result of the most recent A/D conversion.	R/W	NA	AD0GDR - 0x4003 4004
ADINTEN	A/D Interrupt Enable Register. This register contains enable bits that allow the DONE flag of each A/D channel to be included or excluded from contributing to the generation of an A/D interrupt.	R/W	0x100	AD0INTEN - 0x4003 400C
ADDR0	A/D Channel 0 Data Register. This register contains the result of the most recent conversion completed on channel 0.	RO	NA	AD0DR0 - 0x4003 4010
ADDR1	A/D Channel 1 Data Register. This register contains the result of the most recent conversion completed on channel 1.	RO	NA	AD0DR1 - 0x4003 4014
ADDR2	A/D Channel 2 Data Register. This register contains the result of the most recent conversion completed on channel 2.	RO	NA	AD0DR2 - 0x4003 4018
ADDR3	A/D Channel 3 Data Register. This register contains the result of the most recent conversion completed on channel 3.	RO	NA	AD0DR3 - 0x4003 401C

Ilustración 100. Registros de configuración del ADC.

La estructura de lectura de los pines dedicados es la siguiente.

```
Temperatura_Amb = (LPC_ADC->ADDR0>>4) & 0xfff; //Obtenemos el valor del pin 0.23
Brillo_Pant = (LPC_ADC->ADDR1>>4) & 0xfff; //Obtenemos el valor del pin 0.24
Joystick_x = (LPC_ADC->ADDR2>>4) & 0xfff; //Obtenemos el valor del pin 0.25
Joystick_y = (LPC_ADC->ADDR3>>4) & 0xfff; //Obtenemos el valor del pin 0.26
```

La lectura de los pines se asigna a las variables globales del fuente del microcontrolador, las lecturas de los sensores de luminosidad y temperatura, se dividen por un factor proporcional al número de bits codificados (14), por otro lado, la variables relacionadas a la velocidad y dirección de la aeronave se incrementan o decrementan en función de si se realizan lecturas de las posiciones extremas en el joystick.

5.1.7. Temporizador TIMER1_IRQHandler

La función handler de este temporizador, se utiliza para realizar la transmisión serie desde el microcontrolador a la FPGA, configura los pines asociados al reloj y a los datos, transmitiendo a la FPGA los datos del squitter que esta última deberá de encriptar y transmitir. Es importante recordar que la escritura se produce en los ciclos de subida de la señal SCL, mientras que la lectura por parte de la FPGA se producirá en los ciclos de bajada de esta misma señal.

La siguiente imagen, ilustra el diagrama de bloques del temporizador mediante el cual se realiza la transmisión de datos por el puerto serie. Esta función modela de manera conjunta la señal de reloj y de datos.

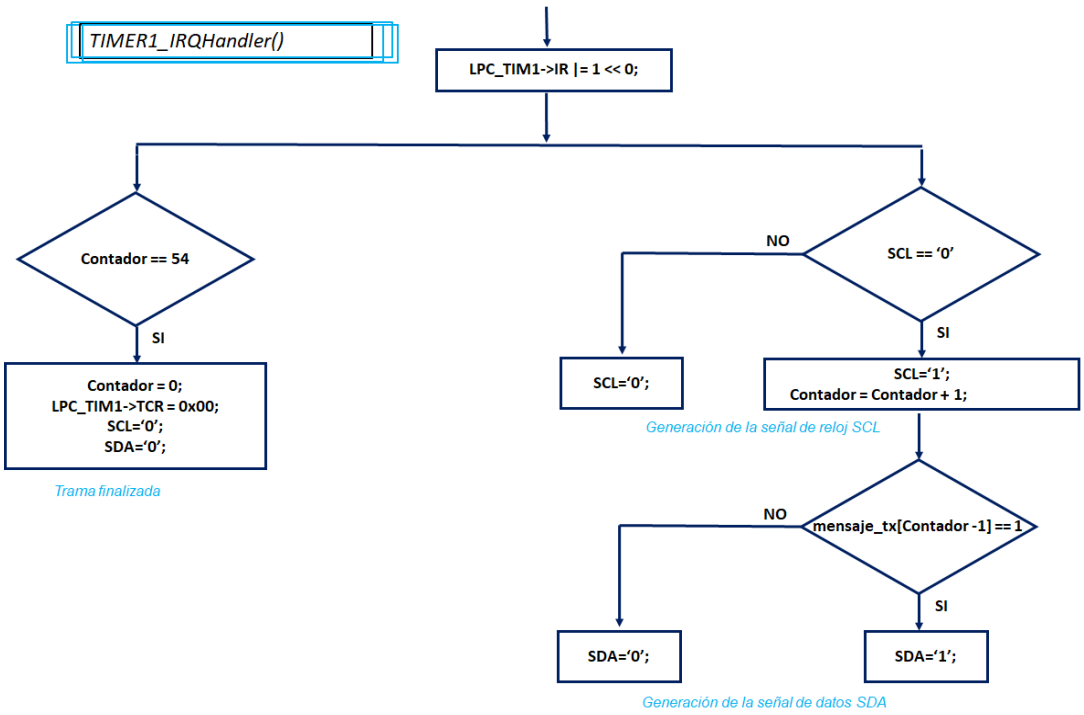


Ilustración 101. Flujograma del temporizador Timer1

En primer lugar, como se ha comentado en repetidas ocasiones el primer paso es borrar el flag de solicitud de interrupción. A continuación, mediante dos condiciones se evalúa la generación de la transmisión o el final de la misma. La transmisión de datos se actualiza con el valor del vector de datos que se quiere transmitir en los flancos de subida de la señal de reloj, es importante recordar que la transmisión de un nivel alto se realiza mediante el registro *"FIOSET"*, mientras que la transmisión de niveles bajo se hace con el registro *"FIOCLR"*.

La solución obtenida en el puerto serie, se ilustra en la siguiente imagen

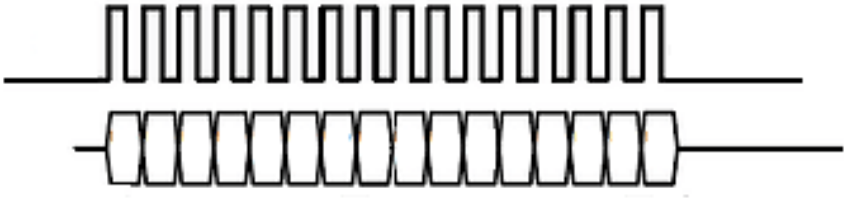


Ilustración 102. Transmisión de la señal SCL y SDA por el puerto serie.

5.1.8. Temporizador TIMER2_IRQHandler

La función handler del temporizador, se encuentra estrechamente ligada al temporizador anterior, se utiliza para controlar la habilitación y deshabilitación del contador interno de interrupción de la handler del temporizador anterior, controlando así en que momento tiene que comenzar la transmisión por el puerto serie. Es decir, la única función de este temporizador es llevar el control de que datos se transmiten a la FPGA.

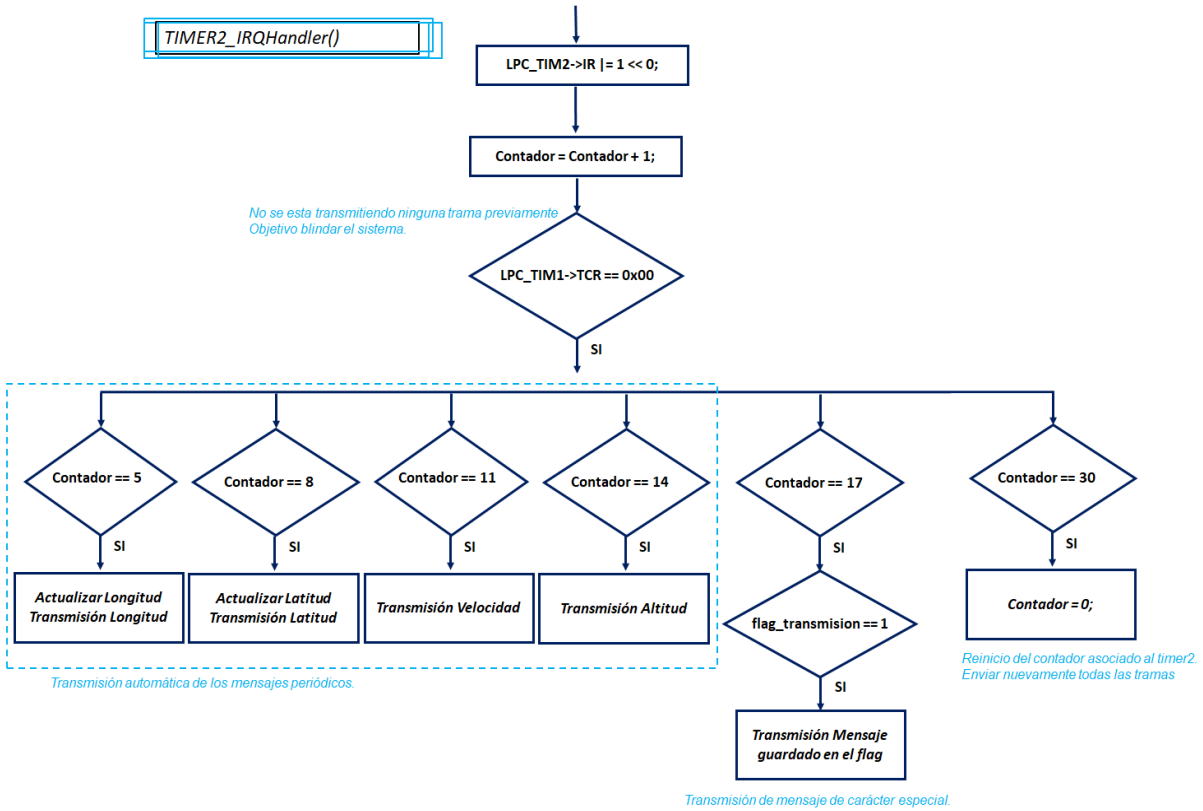


Ilustración 103. Flujograma del temporizador Timer2

La primera instrucción borra el flag asociado de solicitud. A continuación, la handler del temporizador incrementa de forma automática un contador asociado al temporizador, con el objetivo de no modificar la trama que se puede estar transmitiendo se condicionan todas las instrucciones a que el temporizador que genera las tramas este apagado. En caso de que no se esté generando ninguna trama, en función del valor del contador asociado, se realiza un tipo de transmisión u otra, de tal forma que cada tres cuentas se envían de forma automática los mensajes periódicos (longitud, latitud, velocidad y altitud) junto con el posible mensaje de carácter especial que se haya almacenado en el flag de interrupción generado por el usuario mientras se estaba transmitiendo algún otro mensaje. Recordar que este flag se activa cuando el usuario indica al micro que quiere transmitir un mensaje de carácter especial en algún instante de tiempo en el que el microcontrolador ya está transmitiendo una trama de los mensajes periódicos. Con el objetivo de no interrumpir esta trama, se guarda la solicitud en un flag y se envía cuando termine la transmisión. Por ultimo cuando el contador llega al valor máximo, se reinicia y vuelve a empezar el proceso descrito previamente.

En resumen, el temporizador realiza la transmisión de cuatro o cinco paquetes de datos en función del valor del flag. Un ejemplo de cómo sería la transmisión se muestra en la siguiente imagen.

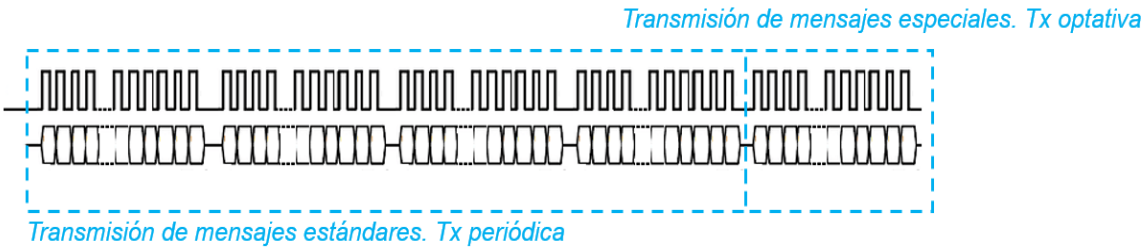


Ilustración 104. Ejemplo de una transmisión mediante el Timer2

La estructura de la transmisión es simple, se concatenan en un único array definido como variable global en el código, el valor del identificador, el tipo de mensaje y el propio mensaje que se quiere transmitir. Para esta parte del proyecto, destacan las funciones de conversión de decimal a binario y las funciones de concatenación implementadas por el usuario. La estructura es la siguiente.

GENERACIÓN DEL VECTOR TX.

```
identificador_tx = identificador;
tipo_mensaje_tx = "Tipo de mensaje definido"
datos_mensaje_tx = "Valor del mensaje";
conv_int_bin(identificador_tx, identificador_tx_vector);
conv_int_bin(tipo_mensaje_tx, tipo_mensaje_tx_vector);
conv_int_bin(datos_mensaje_tx, datos_mensaje_tx_vector);
crear_mensaje_tx (mensaje_tx, identificador_tx_vector, tipo_mensaje_tx_vector, datos_mensaje_tx_vector);
```

En la figura respectiva al temporizador, se puede apreciar como os valores de longitud y latitud, se actualizan justo antes de sus respectivas transmisiones, la actualización se basa en las funciones utilizadas en el apartado 5.1.4 referentes a los cálculos de longitud y latitud. La estructura es la siguiente.

LONGITUD.

```
aumento_posicion = Calcular_posicion_x(velocidad, ubicación);
valor_longitud = valor_longitud + aumento_posicion;
```

LATITUD.

```
aumento_posicion = Calcular_posicion_y(velocidad, ubicación);
valor_latitud = valor_latitud + aumento_posicion;
```

Es importante especificar o recordar que el temporizador se habilita cuando se inicializa la aeronave, es decir cuando el usuario confirma el identificador, cuando se apaga la aeronave se deshabilita el temporizador.

5.1.9. Temporizador TIMER3_IRQHandler

La función handler de este temporizador, se utiliza para realizar la actualización de los datos de una aeronave, que ya se haya inicializado previamente, indicando al TIMER1 que transmita de forma secuencial y en un orden, todos los parámetros asociados al identificador de la aeronave que se acaba de inicializar.

La necesidad de esta transmisión, aparece debido a que el receptor no conoce el estado de los mensajes de carácter especial enviados previamente por la aeronave ya que se recuerda, que este tipo de mensaje no se envía de forma periódica si no a petición del usuario.

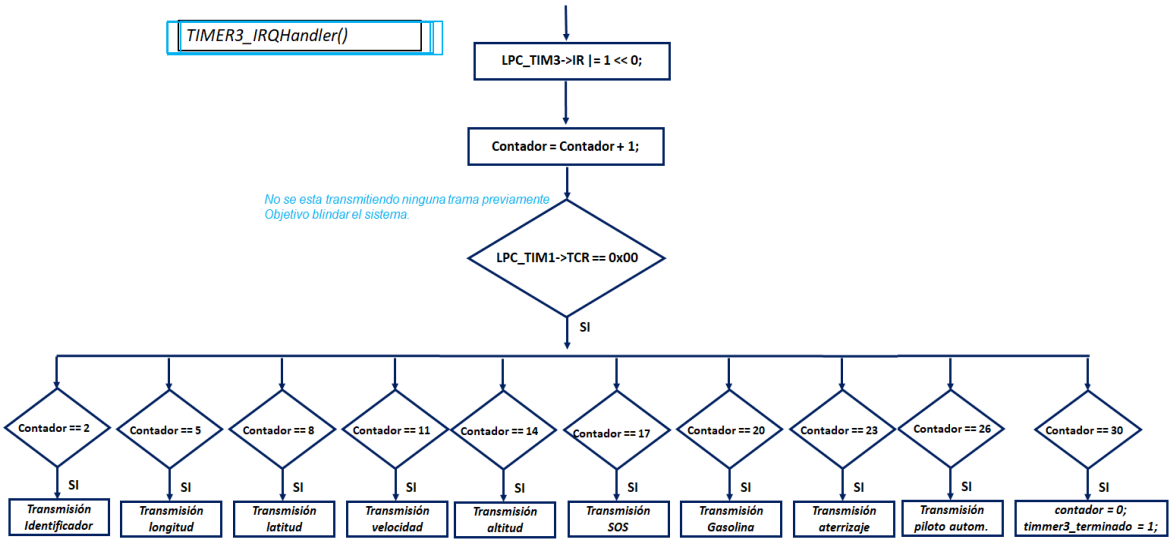


Ilustración 105. Flujograma del temporizador Timer3

Como se puede observar, la transmisión contempla todos los parámetros que tiene asociado un identificador. Se envía de forma secuencial y en orden de prioridad, el valor del identificador, de los mensajes asociados a la posición, velocidad y altitud, para concluir la transmisión con todos los mensajes de carácter especial.

Los datos son almacenados según el identificador en una estructura cuando el usuario apaga la aeronave. Por el contrario, cuando el identificador arranca la aeronave, introduciendo un nuevo identificador, el microcontrolador evalúa si se ha trabajado previamente con ese identificador, en caso de que sea nuevo se inicializan todos los parámetros, pero si coincide con algún valor almacenado previamente, se cargan todos los valores de la estructura y se envían al receptor mediante el tercer temporizador.

5.2 Equipo receptor. Interfaz de visualización.

El equipo receptor formado por un microcontrolador conectado a la FPGA, tiene como principal objetivo realizar el interfaz entre el usuario y la FPGA. Por un lado, mediante una conexión por puerto serie, la FPGA transmite al micro los squitters recibidos, por otro lado, mediante una conexión a internet por un puerto ethernet, se ofrece al usuario una visualización de la posición, el rumbo histórico seguido y el estado de control de todas las aeronaves que se encuentran en el espacio aéreo.

El equipo receptor consiste en un conjunto de funciones e interrupciones periódicas que se habilitan al comienzo del código. El programa principal se encuentra constituido por unas funciones de inicialización y un bucle while infinito, en el que se van produciendo las interrupciones periódicas habilitadas. Parte del proyecto se recicla de la etapa de transmisión, como por ejemplo la inicialización y calibración de la pantalla táctil, otra parte se recicla la estructura de una versión estándar ofrecida por el fabricante ARM, en la cual se implementa la comunicación con la página web. Entre las funciones de inicialización implementadas, destaca la inicialización de los cuadros de dialogo de la pantalla táctil, los pines y el temporizador, mediante estas dos funciones se inicializan los pines de entrada de la comunicación serie con la FPGA y el temporizador utilizado para la recepción de los datos.

Para ofrecer una idea global del proyecto, se realiza un análisis desde un alto nivel del funcionamiento global del microcontrolador.

En primer lugar, según arranca el microcontrolador, la función main del sistema inicializa el Statechart del sistema, los pines utilizados el temporizador y los parámetros relacionados con la comunicación de la página web. El resto de la función main, es un bucle while infinito que evalúa constantemente la situación en la que se encuentra el Statechart y las funciones timer_tick y TCPnet mediante los cuales se reciben los datos de la maquina remota.

El Statechart está formado solo un par de estados asociado a un único estado genérico que ofrecen al usuario, mediante la pantalla táctil del microcontrolador receptor, la posibilidad de habilitar los cuadros de identificación de las aeronaves y la habilitación de cara a la simulación de las aeronaves que no se están recibiendo información en el último refresco. Estos dos parámetros se asocian a dos variables globales que mediante un nivel alto o un nivel bajo representan los dos estados. Estas dos variables se actualizan de manera inmediata y son leídas desde las funciones asociadas al protocolo TCP de la página web.

Mediante un temporizador se controla la comunicación por el puerto serie con la FPGA, este proceso extrae los tres mensajes descifrados en la FPGA y en función de los campos del identificador y del tipo de mensaje, actualiza la variable global asociada al identificador recibido, para que las funciones relacionadas con el protocolo TCP realicen las lecturas de estas con su última actualización. El equipo receptor está preparado para recibir los números primos elegidos en el equipo transmisor y las claves generadas en la PFGA asociadas al algoritmo RSA, para identificar estos datos, se condiciona a que el identificador tenga un valor nulo.

Para terminar este epígrafe, se realiza un análisis entre las funciones e interrupciones más importantes de cara al código desarrollado en el microcontrolador de este equipo, para en la siguiente parte, analizar cada uno de estas secciones de una forma más exhaustiva.

- **ESTRUCTURAS DE LA PANTALLA DE LA FSM.**

Mediante una librería, se calibra y detecta las posibles pulsaciones en la pantalla táctil del dispositivo. La parte desarrollada es el interfaz de los cuadros que aparecen en la pantalla táctil, junto con las funciones asignadas a cada zona previamente definida.

- **INICIALIZACION DE LOS PINES GPIO.**

La comunicación entre el microcontrolador y la FPGA se realiza mediante un conjunto de dos pines definidos como salida previamente como pines de propósito general.

- **INICIALIZACION DEL TEMPORIZADOR TIMMER.**

Esta función de inicialización del proyecto tiene como objetivo configurar el temporizador para que realice interrupciones periódicas con un periodo de interrupción previamente asignado.

- **TIMER0_IRQHANDLER**

La función handler de este temporizador, se utiliza para realizar la recepción serie desde la FPGA hasta el microcontrolador, realiza la lectura de los pines asociados al reloj y a los datos, recibiendo por lo tanto los squitters recibidos y desencriptados de la FPGA.

- **HTTP.CGI**

Esta parte del proyecto recoge las funciones necesarias para la interactuación entre la página web y el sistema, debido a que la transferencia es unidireccional, desde el microcontrolador al servidor, únicamente se utiliza la función *"Cgi_func"*. Esta función interpreta el script TCPnet, para generar la parte dinámica de la respuesta HTTP.

- **INDEX.CGI**

Este fichero ofrece una simbiosis entre código HTML y JavaScript para implementar la página web cargando una dirección URL previamente definida. Es en este fichero donde se desarrolla toda la página web.

Con una idea global del proyecto definida. La siguiente parte de este apartado de la memoria, analiza más en detalle cada una de las partes citadas en el apartado. Es interesante, separar la parte del microcontrolador que realiza la comunicación con la FPGA y la comunicación con la página web, programado en C de la propia página web, programado en HTML y JavaScript.

En la memoria también se estudiará en función de esta división, la primera parte analiza todos los puntos de inicialización, el temporizador y el proyecto HTTP, la segunda parte del epígrafe, analizar el fichero que desarrolla la página web.

5.2.1. Interfaz de comunicación con la FPGA y la página web.

Esta parte del proyecto, analiza la parte referente a la comunicación con la FPGA, y la página web, es decir toda la parte desarrollada en C. por lo que se podría decir que este apartado abarca desde todas las funciones de configuraciones e inicializaciones, la función handler del temporizador que realiza la lectura de la FPGA y la parte referente a todas las funciones necesarias para la interactuación entre la página web y el sistema.

El resto del apartado, analiza cada una de estas partes implementadas por separado.

5.2.1.1 Funciones de inicialización.

Este apartado define todas las funciones de inicialización que aparecen en el programa principal *"Main"*. Estas funciones se incrustan antes del bucle while infinito y se realizan una única vez, con el objetivo de preparar todas las configuraciones del dispositivo. Las configuraciones realizadas son las siguientes.

- **Configuración e inicialización de los pines de entrada GPIO.**

Para realizar la configuración de los pines que realizan la comunicación serie, se reciclan las estructuras de registros de la etapa transmisora, en concreto el pin dedicado a la transmisión de datos *SDA* y el pin dedicado a la transmisión de reloj *SCL*. Para ello se realizan los siguientes pasos.

En primer lugar, se accede a los registros de selección de función de pin (PINSELx) con el objetivo de definir los pines como GPIO, al ser pines de entrada, es necesario definir si se quieren utilizar resistencias internas asociadas, debido a que la información aparece en la aparición de niveles altos, se decide introducir una resistencia interna a nivel bajo *"pull_down"*. Una vez definidos como propósito general, con su respectiva interna asociada, mediante el registro de Dirección (FIOxDIR) se define el pin como entrada.

Se resume la configuración en el siguiente cuadro resumen.

```
LPC_PINCON->PINSEL &= ~(3<< SDA);
LPC_PINCON->PINSEL &= ~(3<< SCL);
LPC_PINCON->PINMODE |= (3<< SDA);
LPC_PINCON->PINMODE |= (3<< SCL);
LPC_GPIO->FIODIR &= ~(1<<SDA);
LPC_GPIO->FIODIR &= ~(1<<SCL);
```

Más adelante, para realizar la lectura en los pines utilizados en la comunicación serie se utiliza el registro de lectura de datos (FIOxPIN), es importante recordar que la lectura de éste registro entrega el valor del pin al que hace referencia cada bit

Se resume las sentencias de escritura en el siguiente cuadro resumen

```
S_DATA = LPC_GPIO->FIOPIN & (1<<SDA);
S_CLK = LPC_GPIO->FIOPIN & (1<<SCL);
```

- **Configuración del temporizador Timer0.**

Para realizar la configuración del temporizador asociado a la lectura del puerto serie, al igual que en la configuración e inicialización de los pines de entrada, en este apartado se recicla la estructura de configuración del temporizador de la etapa de transmisión.

Como punto de partida se habilita la interrupción del temporizador, asignando un nivel de prioridad a la interrupción.

A continuación, mediante el registro TCR se habilita la cuenta del temporizador asociado, el registro CTCR otorga al temporizador un trabajo en modo *timmer* en lugar de en modo *capture*, el registro MCR habilita la entrada en la handler cuando el valor del contador del temporizador coincide con el valor del MRx, también se indica un reinicio de cuenta del valor del contador, tras producirse esta interrupción, con el objetivo de que se repita de forma periódica el proceso.

Por último, se asigna los valores al PR y MR0 con el fin de ajustar el tiempo que tarda en producirse la interrupción. El reloj interno del microcontrolador, trabaja a 25 MHz, con el fin de que las interrupciones sean en valores enteros, el valor del prescaler se fija a un valor constante de 25 unidades, con un periodo de incremento de unidad en las unidades MR de un microsegundo.

Se resume la configuración en el siguiente cuadro resumen.

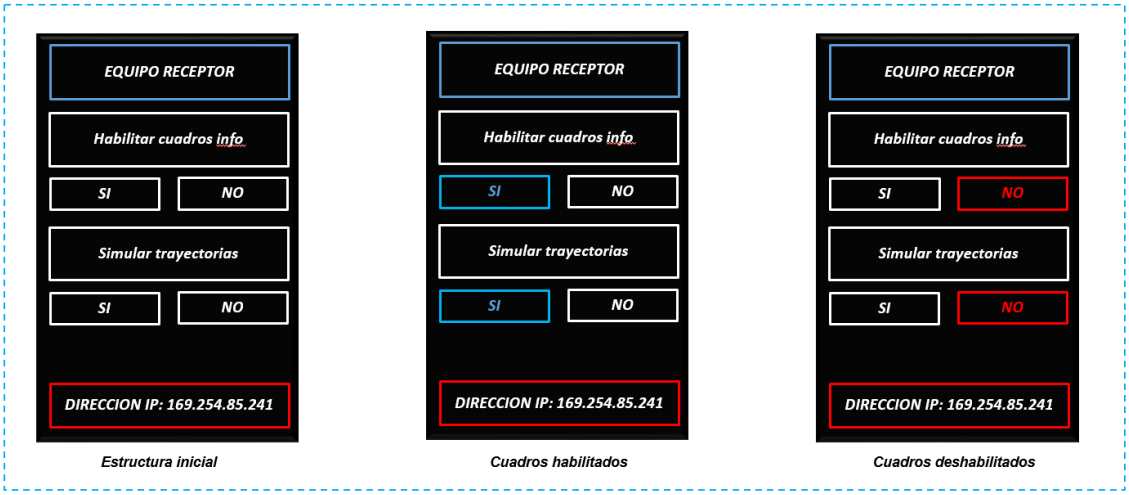
```
NVIC_EnableIRQ(TIMER0_IRQn);
NVIC_SetPriority(TIMER0_IRQn,0);
LPC_TIM0->TCR = 0x01;
LPC_TIM0->MCR |= 0x3;
LPC_TIM0->CTCR = 0x0;
LPC_TIM0->PR = 25;
LPC_TIM0->MR0 = Valor_MR0;
```

- **Configuración de la pantalla táctil.**

La última parte reciclada del proyecto anterior referente a la etapa de transmisión, corresponde a los cuadros implementados en la pantalla táctil de interfaz con el usuario. Mediante una librería, se calibra y detecta las posibles pulsaciones en la pantalla táctil del dispositivo. La parte desarrollada es el interfaz de los cuadros que aparecen en la pantalla táctil, mediante una estructura, se define una zona de la pantalla estructurada en forma de cuadrado. Esta estructura contiene los valores de inicio y tamaño en coordenadas cartesianas *(x, y)*. Estas estructuras representan los valores globales de las variables asociadas.

En este proyecto se implementa únicamente una pantalla de trabajo, esta pantalla ofrece la posibilidad de habilitar o deshabilitar los cuadros de dialogo referente a las aeronaves y la opción de simular las trayectorias de los equipos que no se recibe información.
En función de si las opciones se encuentran habilitadas o deshabilitadas se iluminarán de diferentes colores los cuadros respectivos. En caso de que, si se encuentre habilitada esa opción, el cuadro de dialogo de la opción “si” estará en color azul y el cuadro de dialogo de la opción “no” estará en color blanco. Por el contrario, si no están habilitados el cuadro de dialogo de la opción “si” estará en color blanco y el cuadro de dialogo de la opción “no” estará en color rojo.

Se resume la configuración en las siguientes imágenes.



- **Configuración de los protocolos TCP.**

Las configuraciones utilizadas para utilizar el protocolo TCP recicla la estructura de una versión estándar ofrecida por el fabricante ARM, con la estructura de esta versión se implementa la comunicación con la página web.

Entre las funciones TCP de inicialización recicladas, se puede destacar la siguiente función

- **init_TcpNet.** (ubicada en la biblioteca RL-TCPnet) esta función inicializa los recursos, protocolos y aplicaciones del sistema TCPnet. Como detalle de interés, esta función no devuelve ningún valor.

Entre las funciones TCP incluidas en el bucle while infinito del main, se pueden destacar las siguientes funciones.

- **timer_tick:** (ubicada en la biblioteca RL-TCPnet) esta función establece el indicador de tic del temporizador, que el sistema TCPnet utiliza para medir el tiempo. El sistema TCPnet no utiliza temporizadores de CPU. Por lo tanto, debe generar los eventos de temporización llamando periódicamente a la función timer_tick
- **main_TcpNet:** (ubicada en la biblioteca RL-TCPnet) pudiendo garantizar que se trata de la función principal TcpNet. Manejando los siguientes parámetros:
  - Tiempos de espera de protocolo
  - Caché de direcciones ARP
  - Sondeo del controlador ethernet para los datos recibidos.

  Cuando main_TcpNet recibe datos de la máquina remota, llama a las funciones apropiadas del protocolo TCPnet para procesar los datos y luego pasa los datos resultantes a la aplicación del usuario. Se debe llamar a la función main_TcpNet con frecuencia. De lo contrario, el sistema TCPnet no se ejecutará.
  La función main_TcpNet devuelve TRUE, si el sistema TCPnet está ocupado. Si el sistema TCPnet está inactivo, esta función devuelve FALSE. El valor de retorno se utiliza en el modo de operación controlado por eventos de TCPnet en el entorno RTX.

5.2.1.2 Comunicación con la FPGA TIMER0_IRQHandler

La función handler de este temporizador, se utiliza para realizar la recepción por el puerto serie desde la FPGA hasta el microcontrolador. Configurados previamente los pines asociados al reloj y a los datos, mediante este proceso se recibe desde la FPGA los datos del último squitter detectado. Es importante recordar que la escritura de los datos por parte de la FPGA se produce en los ciclos de subida de la señal SCL, mientras que la lectura de este proceso se producirá en los ciclos de bajada de esta misma señal.

La siguiente imagen, ilustra el diagrama de bloques del temporizador mediante el cual se realiza la recepción de datos por el puerto serie. Esta función modela de manera conjunta la señal de reloj y de datos.

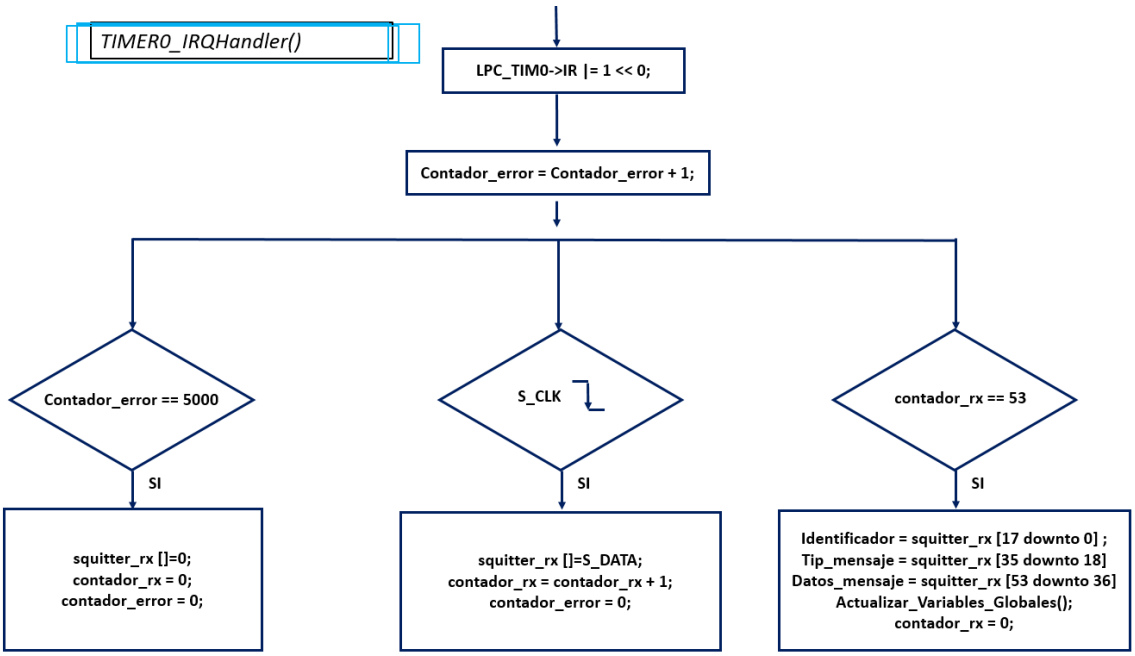


Ilustración 106. Flujograma del temporizador Timer0

La función asociada al temporizador timer0, comienza borrando el flag de interrupción ubicado en el registro IR. A continuación, aparecen tres condiciones que se evalúan constantemente según se produce la interrupción.

La primera condición, asociada al *"contador_error"*, pretende eliminar todas las posibles detecciones erróneas en la señal de reloj, para ello evalúa si ha trascurrido el suficiente tiempo para que hubiera detectado el siguiente flanco, en caso afirmativo, inicializa los parámetros del temporizador. La segunda condición, detecta el flanco de bajada de la señal de reloj almacenando el valor de la señal de datos del puerto serie, reiniciando el contador asociado a las detecciones erróneas e incrementa en una unidad el contador asociado a la recepción, se podría decir que en esta condición se realiza la detección de datos del puerto serie. Por último, la tercera condición, evalúa si se ha completado la detección de todos los datos transmitidos por el puerto serie, en caso afirmativo, almacena en las respectivas variables los datos referentes al identificador, al tipo de mensaje y a los datos del propio mensaje. Con estos datos actualizados, se realiza una llamada a la función "*Actualizar_Variables_Globales()"* con la que se actualizan todos los valores de la aeronave asociada al identificador recibido.

La función "*Actualizar_Variables_Globales()"* pretende actualizar las variables globales de cada identificador referentes a longitud, latitud, velocidad identificador junto con los valores de los parámetros de carácter especial, a partir de los valores recibidos del squitter (identificador, tipo de mensaje y datos del mensaje).

La primera parte de la función se resume en la siguiente imagen. Esta parte únicamente evalúa si los datos recibidos, tienen un identificador nulo, entendiendo que la transmisión se refiere a los números primos y las claves de cifrado del algoritmo RSA. O si, por el contrario, el identificador es distinto de cero y se han recibidos datos asociados a un identificador, relacionado con una aeronave.

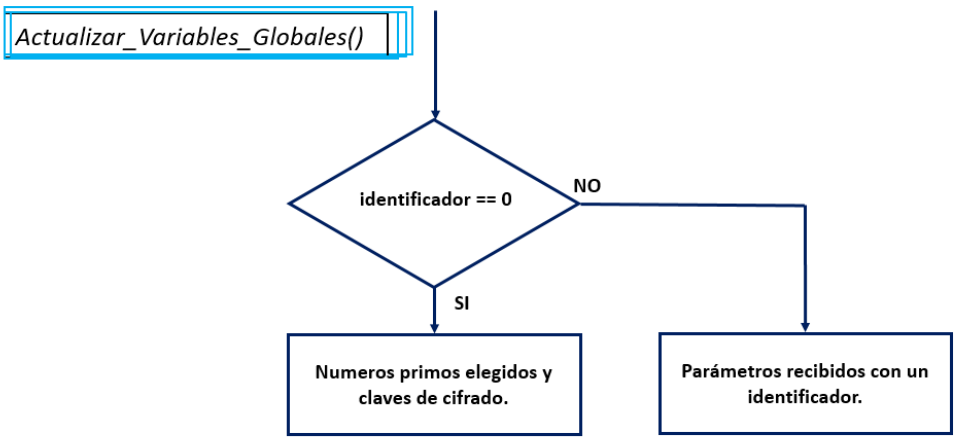


Ilustración 107. Flujograma de la función Actualizar_Variables_Globales()

A continuación, se estudian los dos bloques referentes a la función anterior.

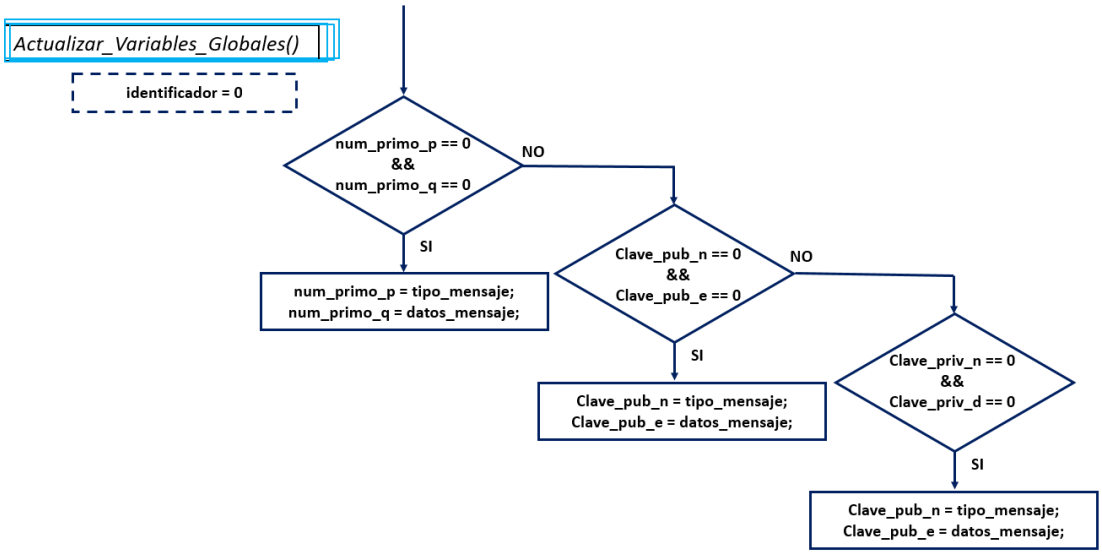


Ilustración 108. Flujograma de la función Actualizar_Variables_Globales(). Numeros primos

La primera condición de la función, evalúa un valor de identificador nulo, garantizando una recepción de parámetros referentes al algoritmo RSA. Se prepara la recepción en función de la transmisión, actualizando los valores que tienen el valor inicializado a cero, es decir primero se inicializarán los números primos, después las claves públicas y por ultimo las claves privadas.

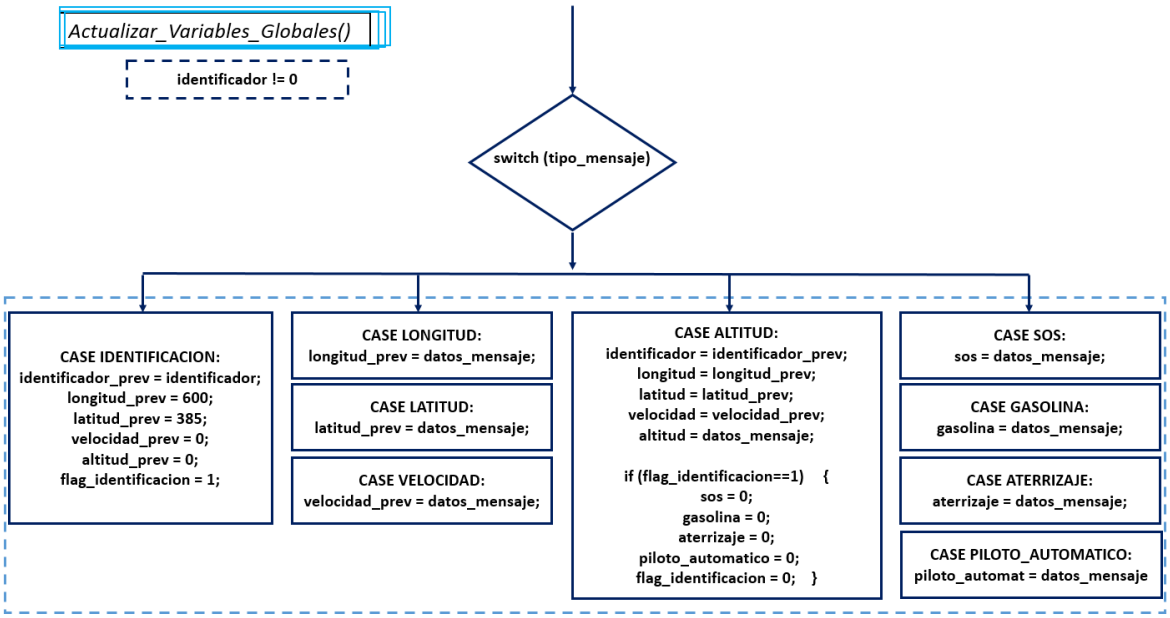


Ilustración 109. Flujograma de la función Actualizar_Variables_Globales(). Datos aeronaves.

La primera condición de la función, evalúa un valor de identificador no nulo, es decir un valor de identificador asociado a una aeronave.

La primera transmisión contiene únicamente el valor del identificador, en la recepción se evalúa esa posible recepción y se inicializan todos los valores previos de los valores estandarizados junto con un flag de que se ha producido este caso.
A continuación, se evalúan los tres primeros casos de los mensajes estándar, es decir los valores de longitud, latitud y velocidad en los que simplemente se actualizan los valores previos con los datos recibidos.
Para concluir los mensajes estandarizados, se evalúa la altitud, en caso de que se haya producido una recepción, se actualizan todas las variables globales con los valores previos asignados en los otros casos, se evalúa mediante el flag de identificación, la posibilidad de que el identificador sea nuevo y por lo tanto se inicializan las variables de los mensajes especiales.
Por otro lado, se evalúan los mensajes de carácter especial, en los siguientes casos, actualizando los valores de forma inmediata.

Es importante destacar que se utilizan los valores previos en los mensajes de carácter estándar con el fin de que la lectura desde la página web se realice con todos los valores actualizados y no solo una parte de ellos.

5.2.1.3 Comunicación con la página web HTTP.CGI

Esta parte del proyecto utiliza las funciones referenciadas al protocolo HTTP mediante las cuales se consigue implementar la comunicación entre el microcontrolador y la maquina remota.

En este fichero destacan dos funciones, cada una de ellas implementa la comunicación en un sentido. Debido a que nuestra comunicación es unidireccional, ya que solo se transmite información desde el microcontrolador hasta el servidor, solo se implementa la función *"cgi_func"* que realiza la comunicación en el sentido microcontrolador servidor.

La función llama al intérprete del script TCPnet, para generar la parte dinámica de la respuesta HTTP. El intérprete de secuencias de comandos llama a *"cgi_func"* para cada línea en la secuencia de comandos que comienza con el comando *"**c**"*.

De especial interés es el argumento "*env"* es un puntero a la cadena de entrada que usa *"cgi_func"* para crear la respuesta dinámica. Es la misma cadena de bytes que se especifica en el código de secuencia de comandos de TCPnet mediante el comando *"**c**"*. La función tiene la siguiente lista de argumentos.

```
U16 cgi_func (
   U8 *   env , / * Puntero para ingresar una cadena desde un script TCPnet. * /
   U8 *   buf , / * Ubicación donde escribir la cadena de respuesta HTTP. * /
   U16   buflen , / * Número de bytes en el búfer de salida. * /
   U32 * pcgi ); / * Puntero a una variable de almacenamiento. * /
```

La estructura de la función se muestra en el siguiente cuadro resumen.

```
U16 cgi_func (U8 *env, U8 *buf, U16 buflen, U32 *pcgi) {
U32 len = 0;
switch (env[0]) {
            case '1' :
               switch (env[2])          {
                        case 'a':
                        len = sprintf((char *)buf,(const char *)&env[4],valor_enviar1);
                        break;
                        case 'b':
                        len = sprintf((char *)buf,(const char *)&env[4],valor_enviar2);
                        break;
                        ……..    . ……..
                        }
 return ((U16) len);
```

Los paquetes enviados en los diferentes casos de los caracteres son los siguientes.

```
Case 'a': Valor del parámetro identificador
Case 'b': Valor del parámetro longitud
Case 'c': Valor del parámetro latitud
Case 'd': Valor del parámetro velocidad
Case 'e': Valor del parámetro altitud
Case 'f': Valor del parámetro sos
Case 'g': Valor del parámetro gasolina
Case 'h': Valor del parámetro aterrizaje
Case 'i': Valor del parámetro piloto automático
Case 'j': Numero primo p
Case 'k': Numero primo q
Case 'l': Clave publica numero "n"
Case 'm': Clave publica numero "e"
Case 'n': Clave privada numero "n"
Case 'o': Clave privada numero "d"
Case 'p': Habilitar Cuadros de identificación
Case 'q': Habilitar la simulación de las Trayectorias
```

5.2.2 Desarrollo de la página web.

Esta parte de la memoria analiza el fichero denominado *"Index.cgi"*, La página HTML propone mostrar de forma gráfica, los datos enviado desde el microcontrolador emisor al microcontrolador receptor, con el fin de ofrecer una mejor visualización por parte de un usuario externo. La página HTML hace peticiones periódicas de información al microcontrolador receptor, el cual funciona como servidor de datos, mostrando en la pantalla los datos recibidos, tanto de forma numérica, como de forma gráfica en un mapa de España previamente seleccionado.

Toda la página está maquetada con etiquetas HTML vacías y estáticas. Una vez que la página ha sido creada y cargada, se desencadena la ejecución de la parte variable y dinámica del programa, que se encarga de dar el contenido a dichas zonas.

El programa guarda de forma periodica la información recibida, en cookies (áreas de datos de almacenamiento en el ordenador de forma local). Cada vez que la página web solicita nueva información al servidor mediante una petición, automáticamente se lee la información almacenada de la petición anterior, y la implementa con la actual. Una vez que se tiene actualizada, y almacenada toda la información, se rellena de contenido variable y dinámico a las etiquetas HTML que están vacías, mediante etiquetas HTML, creadas con lenguaje JavaScript.

El documento HTML *"Index.cgi"*, está dividido en las siguientes partes:

- Identificación de página HTML, y definición de la URL del servidor.
- Creación de css, con los estilos, mediante los que mostrar la información cuantitativa.
- Funciones JavaScript para la creación de la parte dinámica de la página.
- Etiquetas HTML para el maquetado de la información en pantalla.

En la siguiente imagen se muestra la estructura del fichero desarrollado.

```
<html>
  <meta>
        Etiquetas con información de cabecera. Codificación y url del servidor
  <style>
        css con la definición de los estilos a utilizar
  </style>
  <script>
        Funciones programadas en JavaScript para la generación del código HTML
        dinámico
  </script>
  <body>
        Cuerpo principal de la página HTML, formado por
        <div>
            Capa para la zona del título y datos horarios
        </div>
        <div>
            <div>
                Capa para la presentación del mapa
            </div>
            <div>
                Capa para la presentación de los valores numéricos
            </div>
        </div>
  </body>
</html>
```

Ilustración 110. Estructura del fichero "index.cgi"

La página web tiene como objetivo principal ofrecer un entorno de visualización para la torre de control. La página web visualizara todas las aeronaves que se hayan detectado. Esta visualización ofrece un mapa de España con la ubicación actual y el histórico de la trayectoria seguida por la aeronave, de manera conjunta aparece un cuadro de texto con toda la información cuantitativa y cualitativa recibida por cada aeronave.

Como información complementaria, aparece toda la información referente al algoritmo RSA, en concreto los números primos y las claves públicas y privadas del algoritmo.

El desarrollo de la página web está constituido por una parte estática, desarrollada en HTML y una parte dinámica, desarrollada en JavaScript. Se vuelve a dividir el epígrafe de la memoria en estas dos partes desarrolladas.

La página web trabaja de forma conjunta entre el entorno HTML y JavaScript.

5.2.2.1 Desarrollo de la página web estática en HTML

En la página HTML, se ha dividido toda la pantalla en zonas mediante la instrucción DIV, para la presentación de cada una de las informaciones a mostrar. En concreto, la página se ha dividido en 4 zonas. La parte superior izquierda para la presentación del título, la parte superior derecha para mostrar la fecha y hora actual, y posteriormente dos grandes zonas donde se muestra la información más relevante, una parte izquierda que ocupa prácticamente toda la página, y que es la encargada de mostrar el mapa de España y mostrar las trayectorias de las aeronaves, y una zona a la derecha para ofrecer la información cualitativa y cuantitativa de los datos de cada uno de éstas. Todas las DIV, utilizadas, están en su carga original, sin contenido.

La siguiente imagen ofrece un cuadro de la estructura comentada.

Ilustración 111. Distribución de la página HTML estática.

La página HTML por sí sola, no muestra nada más que, la fecha, la hora actual, los números primos utilizados en el algoritmo RSA, y claves tanto privadas, como públicas obtenidas. HTML deja para JavaScript la tarea de rellenar todo el resto de la página.

El desarrollo de la parte de la visualización de la página estática HTML se ofrece en la siguiente estructura.

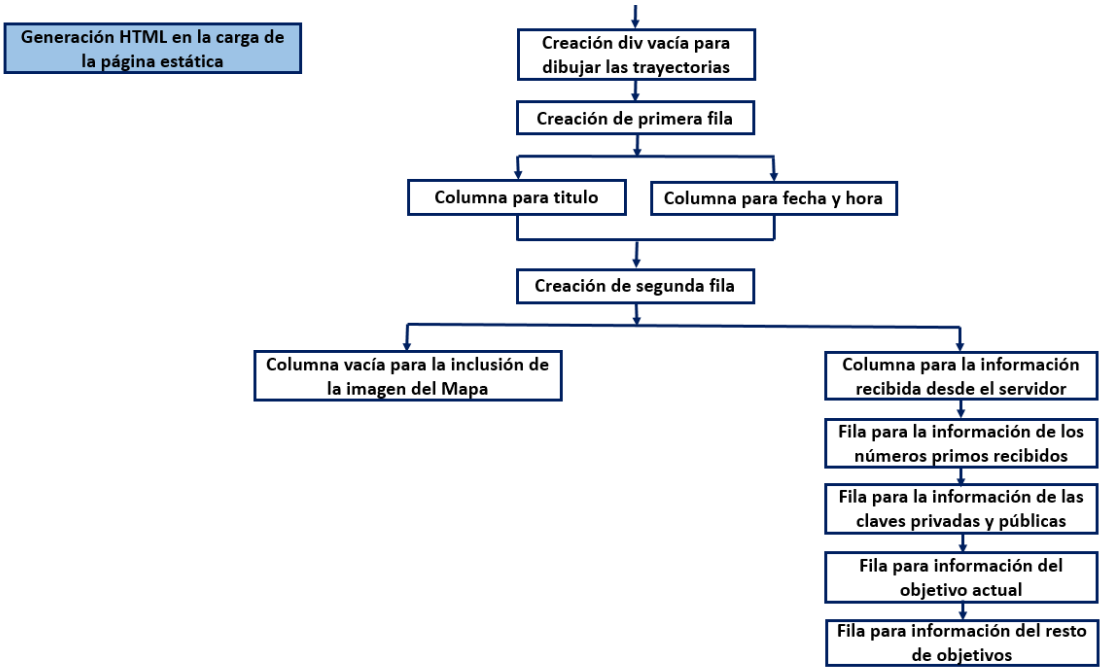


Ilustración 112. Generación HTML en la carga de la página estática

El código JavaScript es lanzado y ejecutado mediante el evento *"onload"* de la página HTML, y mediante asignaciones denominadas *"innerHTML"*, rellena el contenido de todas las DIV existentes que habían sido creadas en vacío.

En la zona destinada para el mapa de España, incluye una etiqueta IMG con la fuente del propio mapa de España, de una imagen *JPG* cargada en le microcontrolador previamente, dimensionando dicha imagen a las medidas necesarias para ocupar toda la parte de la pantalla destina para tal fin. Sobre esta imagen, se crea una zona con una nueva etiqueta DIV, en una dimensión sobre elevada respecto a la anterior utilizando para ello, el atributo *z-index*, esta nueva etiqueta permitirá dibujar las trayectorias de los distintos objetivos (aeronaves), incrustando una etiqueta SVG con la que poder dibujar trazos de líneas. La siguiente imagen resume las propiedades comentadas a lo largo del párrafo.

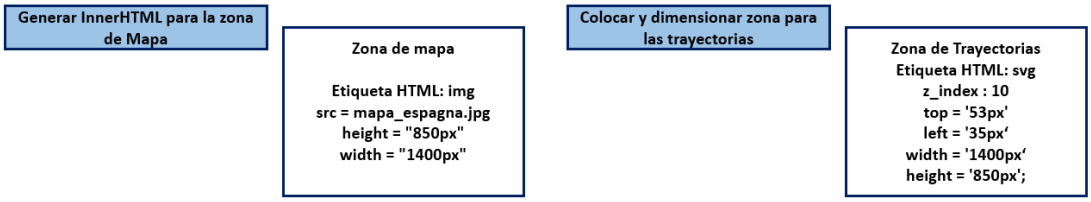


Ilustración 113. Propiedades del HTML para el mapa de España.

Para guardar la información en cada presentación de la información, la página HTML se apoya en el uso de cookies de almacenamiento en local. Existe una cookie principal con una denominación fija, y cuyo contenido es la información de todas las demás cookies utilizadas, existiendo tantas cookies secundarias como distintos objetivos hayan sido detectados.

5.2.2.2 Desarrollo de la página web dinámica en JavaScript.

Tras la carga de la página HTML estática, comienza la ejecución del programa dinámico desarrollado en JavaScript. La ejecución de esta parte de la página está dividida en 5 grandes bloques principales, resumidos en el siguiente párrafo.

Una primera parte, encargada de comprobar que es la primera vez que se ejecuta el programa, con el fin de inicializar todos los datos que serán guardados en el futuro. La segunda parte, encargada de guardar la información recibida del servidor relativa a la aeronave actual. La tercera parte, encargada de generar, solo si procede, la simulación de las trayectorias del resto de aeronaves existentes. La cuarta parte encargada de generar y mostrar de forma gráfica, las trayectorias de todas las aeronaves que se han inicializado previamente. Y la última parte, encargada de mostrar los datos cuantitativos en formato texto, de todas las aeronaves.

A continuación, se muestra la estructura genérica de la parte desarrollada en JavaScript.

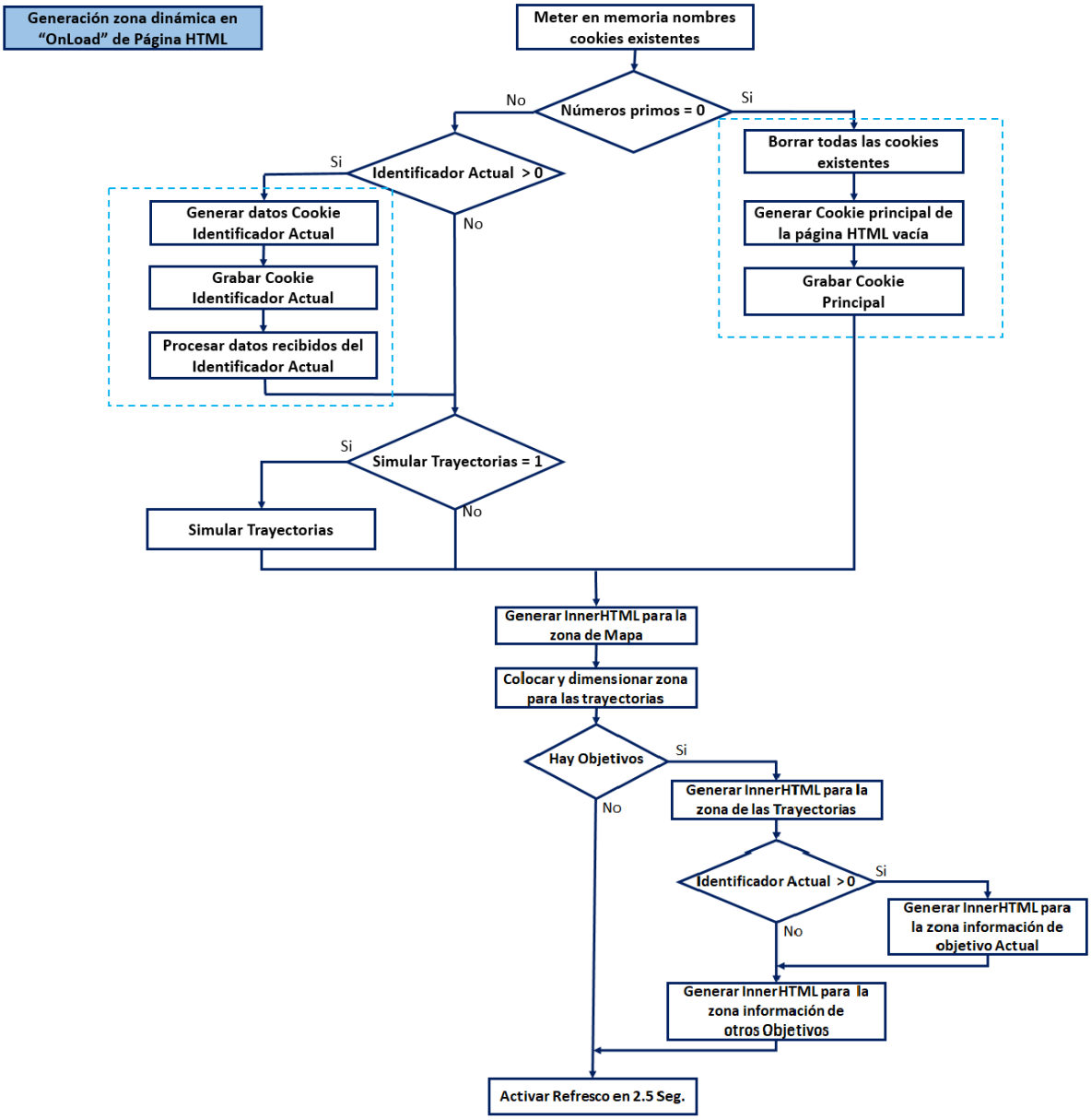


Ilustración 114. Flujograma de la generación zona dinámica en "OnLoad" de Página HTML

El programa JavaScript está estructurado mediante la definición de funciones, para cada parte ilustrada en la imagen anterior. De manera conjunta, se utilizan otras funciones de carácter general, para la obtención, o generación de la información recibida en cookies. Entre estas funciones generales caben destacar las siguientes:

- **Borrado de una cookie.** Esta función recibe el nombre de la cookie a borrar, se encarga de cambiar la fecha de expiración de ésta, a una fecha ya pasada.
- **Leer los datos de una cookie.** Esta función recibe como parámetro el nombre de la cookie que se desea leer y devuelve la información asociada a ésta.
- **Generar los datos de una cookie.** Devuelve los datos formateados relativos a una aeronave.
- **Guardar los datos de una cookie.** La función recibe el nombre de la cookie, y la información generada mediante la función anterior. Esta función graba la cookie, con una fecha prevista de expiración 2 horas después.

El resto de funciones empleadas en el programa JavaScript están pensadas, para que devuelvan las etiquetas HTML, que son incrustadas en las capas vacías de la página principal, mediante instrucciones de reemplazo del código.

La siguiente arte del epígrafe, realiza una visión desde un punto de vista más técnico, pudiendo resumir la parte dinámica desarrollada en JavaScript como se comenta en los siguientes párrafos.

El proceso principal desencadenado por el evento *"onload"* de la página HTML, empieza guardando en un Array dinámico, el nombre de todas las cookies existente en local, asociadas al proyecto. Para ello lee la cookie principal, y rastrea el nombre de cada una de las cookies almacenadas como contenido en ésta cookie principal.

Cuando el proyecto es arrancado por primera vez, la información asociada a los números primos las claves públicas y privadas, llega sin contenido. Éste hecho marca el inicio, indicando que debe borrar todas las posibles cookies que tenga en el anterior Array, ya que presupone que se tratan de cookies de una ejecución anterior. El programa crea entonces la cookie principal con su contenido en vacío, y guarda ésta cookie en local. Si por el contrario los números primos llegan con contenido distinto de cero, comprueba si llega información del identificador de un objetivo también distinto a cero, representando así la detección por parte del microcontrolador de un objetivo. En caso afirmativo, procesa la información de éste identificador, buscando si existe éste, dentro del Array de cookies secundarias. En caso de no existir crea la cookie asociada a éste identificador como nueva, pero con contenido vacío, para posteriormente hacer el proceso común, a todos los identificadores recibidos, que consta en regrabar el contenido de la cookie asociada, con la información recibida en esta ejecución.

La generación avanza condicionada al valor de la información recibida desde el servidor (microcontrolador), indicando si el usuario desea o no simular las trayectorias de los distintos objetivos que no son el actual. En caso afirmativo, lanza el proceso asociado para generar los nuevos puntos de longitud y latitud asociados a cada objetivo, según los últimos existentes para cada uno de ellos, excepto del objetivo actual, ya que éste se recibe en real.

Con todos los parámetros actualizados correctamente, el programa empieza a dibujar el contenido variable. Si existen objetivos en el Array existente para tal fin, genera los movimientos en el SVG creado previamente para cada uno de éstos objetivos, y genera la información cuantitativa de la zona derecha de la pantalla, diferenciando entre la zona destinada para el objetivo actual, y los que ya no lo son.

Para concluir, el último paso que realiza el programa, es lanzar una petición de SUBMIT contra la misma página en el servidor con una demora de 2.5 segundos.

5.2.2.2.1 Estructura de datos de las Cookies.

Como se ha comentado en la introducción, el programa se basa en el uso de cookies para almacenar la información histórica recibida desde el inicio en cada instrucción SUBMIT realizada contra el servidor.

Las cookies utilizadas son almacenadas en el área local del navegador, y son guardadas mediante la instrucción JavaScript *"document.cookie=parámetros"*. Los argumentos de esta instrucción nativa JavaScript, son pares de datos *key-value* para las siguientes informaciones:

- **Nombre = contenido.** Indica el nombre asignado a la cookie, y el contenido de ésta. Para evitar codificaciones extrañas y complejas, se codifica el contenido mediante la instrucción Javascript *"encodeURIComponent(contenido)"*. Por defecto todas las cookies, llevan en sus primeras quince posiciones de contenido la fecha y hora de creación, con formato *"yyyymmddhhmiss_"*
- **Expires = fecha.** Este campo almacena la fecha de expiración de la cookie, que como ya se indicó anteriormente, se inicializa a un valor de 2 horas más, con respecto a la hora actual. Esta fecha, debe ser indicada en formato *'UTC'*, para lo cual se emplea la instrucción JavaScript *".toUTCString()"*

5.2.2.2.1.1 Estructura de datos de la Cookie principal.

La cookie principal denominada *"Informacion_General"*, contiene como información fija la fecha y hora de creación de ésta ocupando un total de 15 posiciones, y siendo a partir de esta posición donde se ubica la información de los distintos objetivos que se han ido recibiendo. Con una separación fija de 5 caracteres de este contenido adicional, se hace referencia al identificador de cada uno de los objetivos recibidos.

La siguiente imagen muestra la estructura de la cookie principal.

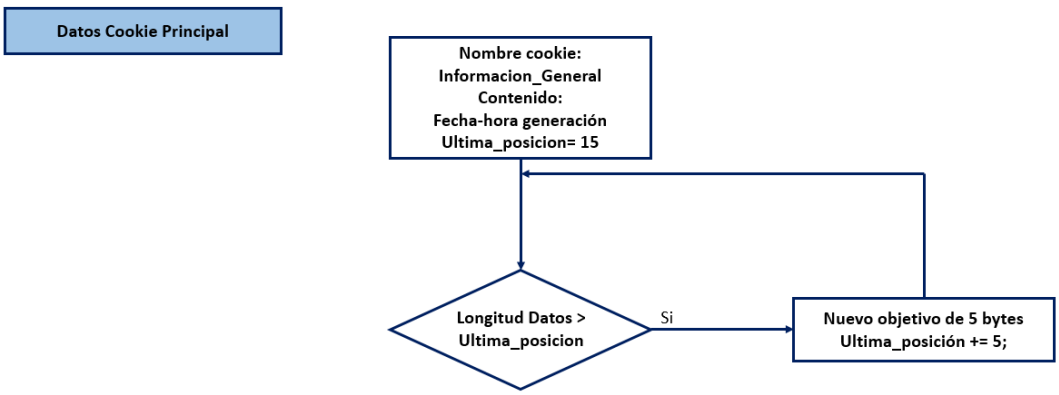


Ilustración 115. Datos y estructura de la Cookie Principal

5.2.2.2.1.2 Estructura de datos de las Cookies secundarias.

Para la información relativa a cada objetivo utilizado, el programa, utiliza una cookie con nombre igual al identificador de este objetivo. La estructura de las cookies secundarias, de cara al contenido tiene la misma parte fija que la principal, pero a partir de la posición número quince, guarda en siete bloques de cinco posiciones cada una de ellas, la información de la aeronave, relativa a su última velocidad, altitud, carburante, los indicadores de SOS, de solicitud de aterrizaje, de piloto automático, e indicador de si la aeronave se encuentra dentro del radio de acción. A partir de estas primeras quince y las siguientes treinta y cinco, se encuentra el histórico de todas las posiciones por las que ha estado posicionado el objetivo.

La siguiente imagen, muestra la estructura de datos de las cookies de cada objetivo.

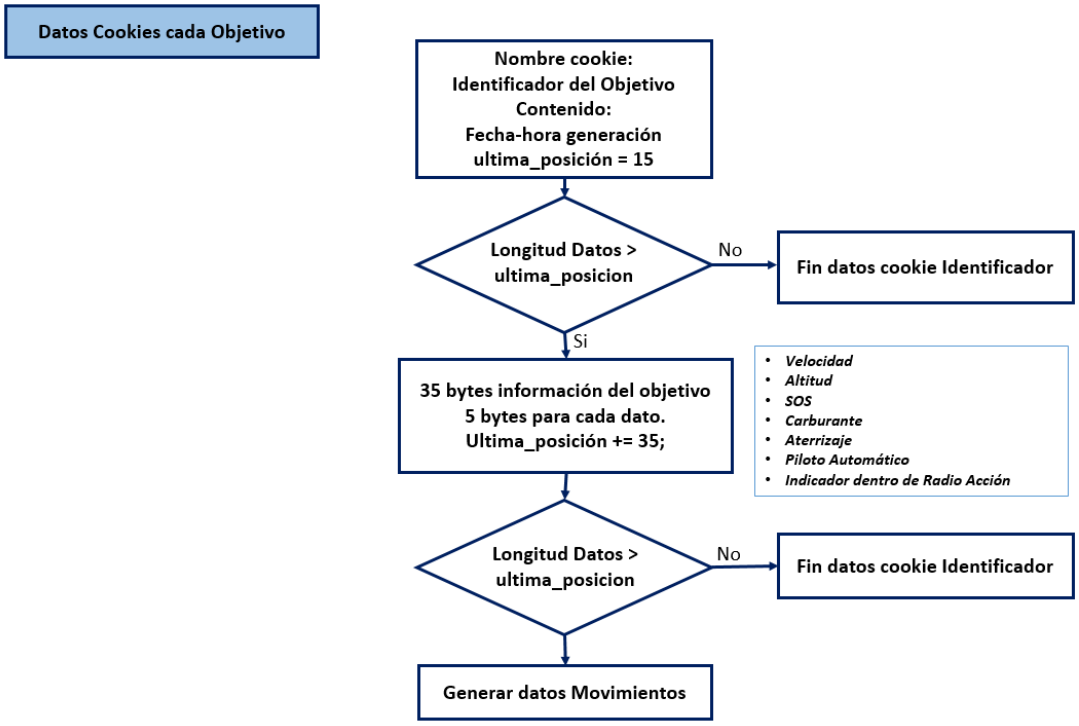


Ilustración 116. Datos y estructura de la Cookie de cada objetivo (aeronave).

Para cada posición se utilizan diez posiciones del contenido, dando información del tipo de movimiento *'R'* para movimiento reales, o *'F'* para movimientos ficticios, unidos a los valores de longitud y latitud de cada uno. Esta información es guardada en un Array dinámico de coordenadas del movimiento y del tipo de movimiento.

La siguiente imagen, muestra la estructura mediante la cual se generan los movimientos.

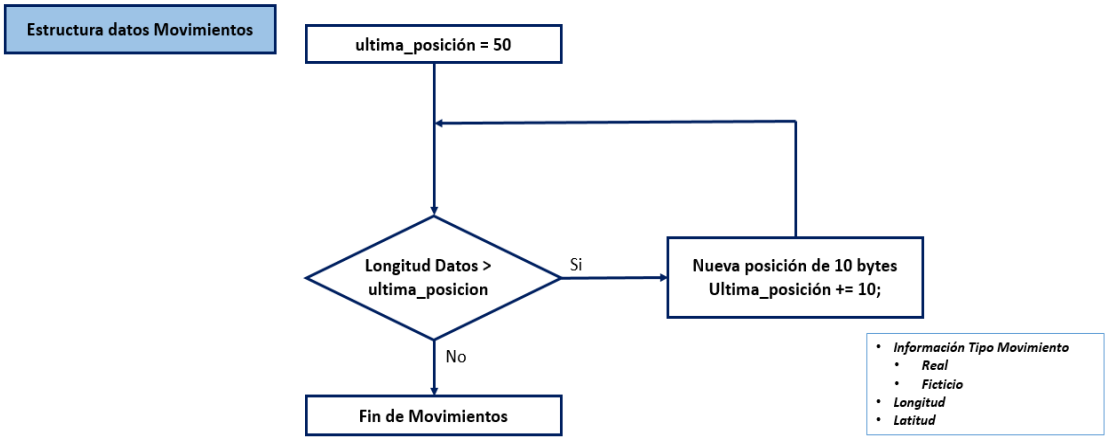


Ilustración 117. Datos y estructura de la generación de movimientos de los objetivos (aeronaves)

5.2.2.2.2 Generar, procesar y grabar el identificador actual.

Lo primero que se realiza es la generación de la cookie con el nombre correspondiente al identificador general. Para ello al igual que para la cookie principal, se genera una parte fija con el contenido de la fecha y hora de generación.

Una vez que la cookie ha sido grabada correctamente, se procede a su lectura, y con los datos recibidos del servidor, comienza la actualización de la información del objetivo actual. Para realizar la actualización del objetivo actual se realizan los siguientes pasos.

En primer lugar, se determina si la nueva posición se mantiene dentro del radio de acción, para lo cual se comprueba si el indicador de longitud y/o latitud, se encuentra entre los mínimos y máximos permitidos dentro de las coordenadas del mapa. En caso afirmativo se comprueba, si la nueva posición difiere respecto de la última posición guardada, con el fin de no guardar posiciones innecesarias. En caso de ser diferente, guarda como nuevo movimiento tipo 'R' real, la nueva ubicación en longitud, y latitud. De manera conjunta y automática, guardan los valores de los siete indicadores estándar que siempre son enviados. Por último, guarda toda la información que tiene en memoria en la cookie correspondiente al objetivo actual.

Una vez está actualizada la información en memoria, se procede a modificar los datos de la cookie con dicha información.

La siguiente imagen, muestra la estructura de datos mediante la cual se procesa el objetivo recibido.

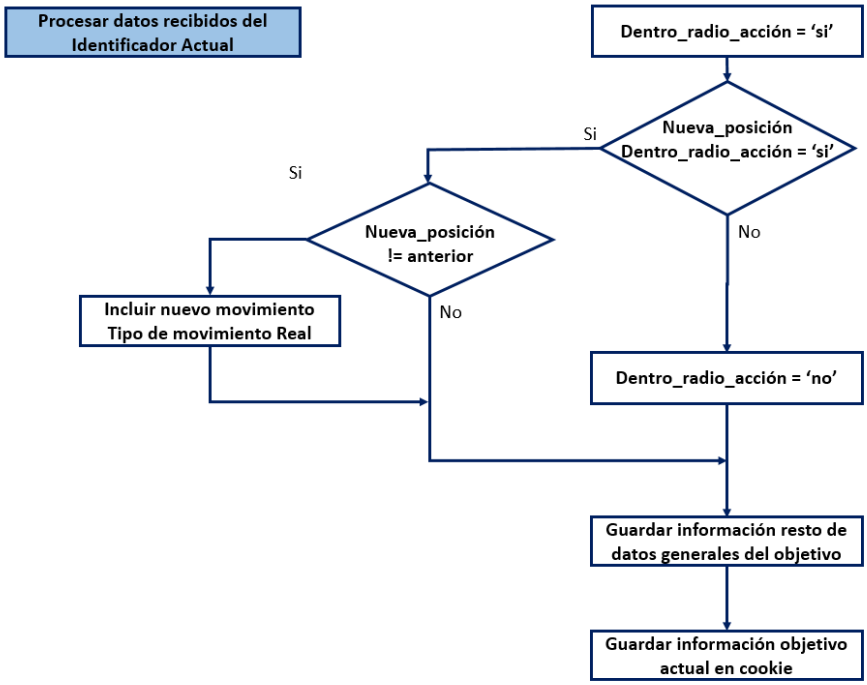


Ilustración 118. Flujograma para la generación de datos recibidos del Identificador Actual.

5.2.2.2.3 Simulación de las trayectorias

Para simular las trayectorias de los objetivos que no son el actual, y sólo si el servidor nos informa de esta petición generada por el usuario, el programa realiza para cada uno de los objetivos que no son el actual, el siguiente proceso.

En primer lugar, realiza la lectura de la información de la cookie del objetivo en cuestión, obteniendo la información del indicador de si aún se encuentra dentro el radio de acción y generando dinámicamente el Array con todos los movimientos históricos. Con el Array de los movimientos generado, y utilizando las dos últimas posiciones de la longitud y la latitud, determina la siguiente nueva posición, y guarda ésta, como un nuevo movimiento en el Array destinado a éstos. Este nuevo movimiento es guardado con el indicador de movimiento 'F' ficticio, para diferenciarlo de los movimientos que fueron generados como reales. La intención es que, una vez se pudiera volver al objetivo en cuestión por parte de la placa emisora, no se haga caso de estos movimientos, y solo se utilizaran los 'R' reales.

Una vez generado este nuevo movimiento, los datos de la cookie son almacenados de nuevo, convirtiendo el Array de movimientos a un String tan largo como sea necesario.

La siguiente imagen, muestra la estructura de la simulación de las trayectorias.

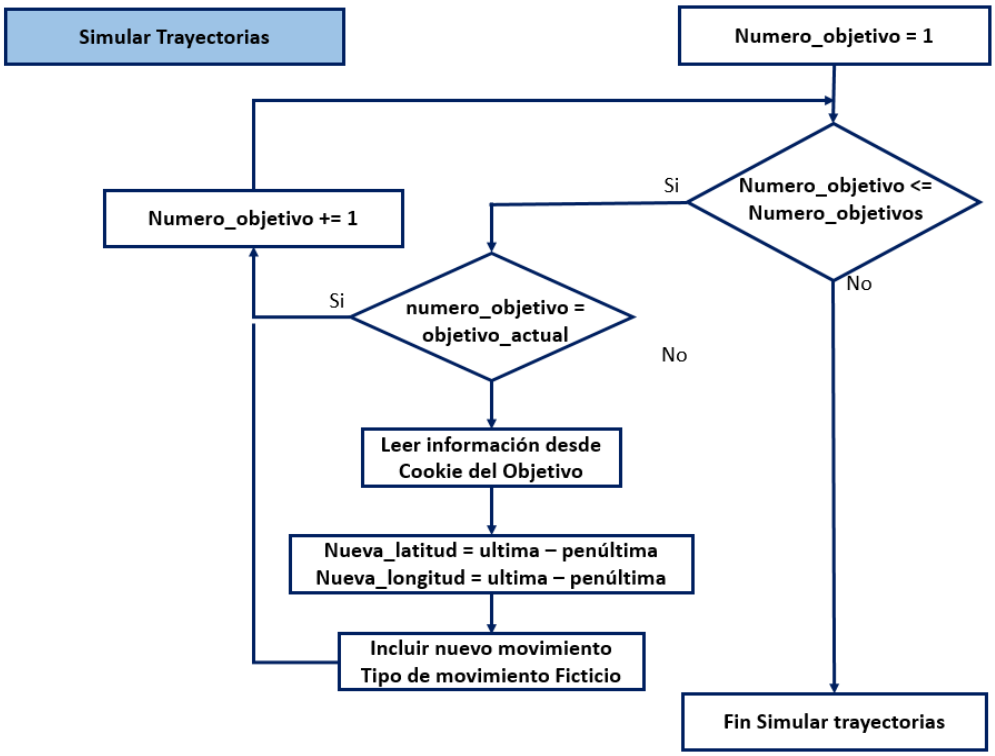


Ilustración 119. Flujograma de la función para simular trayectorias.

5.2.2.2.4 Generación dinámica de código HTML mediante JavaScript.

Como se ha recalcado en muchas ocasiones a lo largo del epígrafe. La información variable de la página web HTML, es generada mediante JavaScript. Para ello, realiza distintas llamadas a funciones que devuelven el contenido relativo a las etiquetas HTML a incrustar en cada una de las DIV creadas, que la generación estática dejó en vacío.

El procedimiento seguido es la llamada al método JavaScript *"innerHTML"*, que tiene por finalidad incrustar contenido a un objeto HTML el cual ha sido referenciado mediante el *'id'* en su creación accediendo mediante la instrucción *"document.getElementById(id)"*

Los siguientes puntos analizan por separado cada una de las funciones existentes con este fin:

5.2.2.2.4.1 Generación dinámica de las trayectorias.

Para generar las imágenes de las distintas trayectorias de todos los objetivos recibidos, el programa repite el mismo proceso para cada uno de estos. Comienza con una lectura de la información de la cookie correspondiente al objetivo, a continuación, comprueba si la información referente a que la aeronave se encontrarse dentro del radio de acción esta activada, en cuyo caso, genera una traza lineal en un color único con inicio y final según los valores en longitud y latitud de cada dos posiciones. De manera conjunta, determina la inclinación del objetivo respecto del eje cartesiano del plano del mapa, obteniendo una posible orientación de entre las ocho predeterminadas y almacenadas por defecto (norte, noreste, este, sureste, sur, suroeste, oeste, o noroeste). Al final de todas las líneas trazadas, coloca en una nueva DIV ubicada en una dimensión superior respecto que las propias líneas trazadas, la imagen del avión correspondiente a la inclinación determinada en el apartado anterior.

La siguiente imagen, muestra la estructura de la visualización de las aeronaves en la página web.

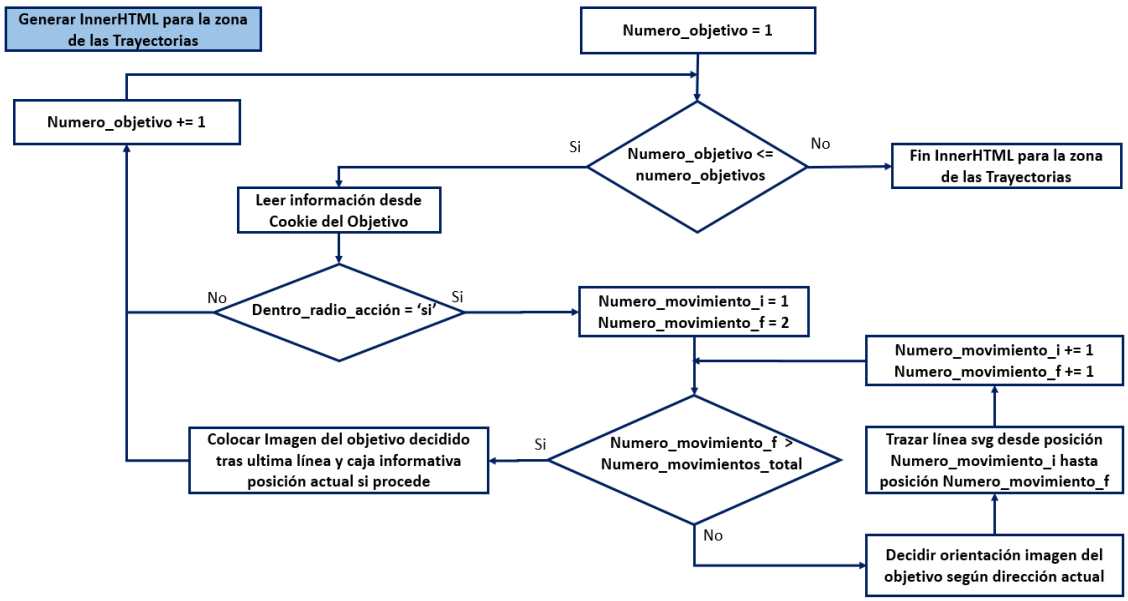


Ilustración 120. Flujograma de la Generación del "InnerHTML" para la zona de las Trayectorias

El código evalúa la posibilidad de que el usuario haya habilitado, la opción de indicar las coordenadas de la aeronave en un cuadro asociado a cada objetivo sobre el mapa, para ello crea una nueva DIV con marco del mismo color que la trayectoria del objetivo y con diferente grosor, en función de si se trata o no del objetivo actual, en la misma posición de longitud y latitud de la propia aeronave, colocando dentro de esta DIV, la imagen del avión junto con el valor de la longitud y latitud actual.
Para determinar la posición en grados, minutos y segundos de la longitud y latitud de la última posición se calcula en base al número de segundos reales existentes entre las cuatro esquinas del mapa de España, y mediante una sencilla regla de tres se determina según los valores de las coordenadas recibidas, la longitud y latitud a pintar.

El programa, evalúa si el objetivo con el que está trabajando y, por lo tanto, pintando en pantalla no es el objetivo actual. En este caso, los movimientos de la aeronave corresponden a posiciones creadas por el propio programa mediante la simulación de la trayectoria, siendo de carácter ficticio. Para crear trayectorias discontinuas, dibuja uno de cada 2 movimientos existentes, obteniendo así la traza de una línea discontinua.

5.2.2.2.4.2 Generación dinámica de la información del objetivo actual.

Concluida la generación de la parte visual referente a la parte visual, de las imágenes de las trayectorias y las imágenes de las aeronaves (ubicaciones en el mapa de España) el programa genera la parte en formato texto ubicada en la parte de la derecha.

Para ello primero inicializa con contenido a la DIV de los datos cuantitativos del objetivo actual, en caso de encontrarse dentro del radio de acción, muestra en filas y columnas la información de la latitud, longitud, velocidad, etc. En caso contrario simplemente, hace indicación de que se encuentra fuera del radio de acción.

La siguiente imagen, muestra la estructura de la información cualitativa y cuantitativa de las aeronaves en la página web.

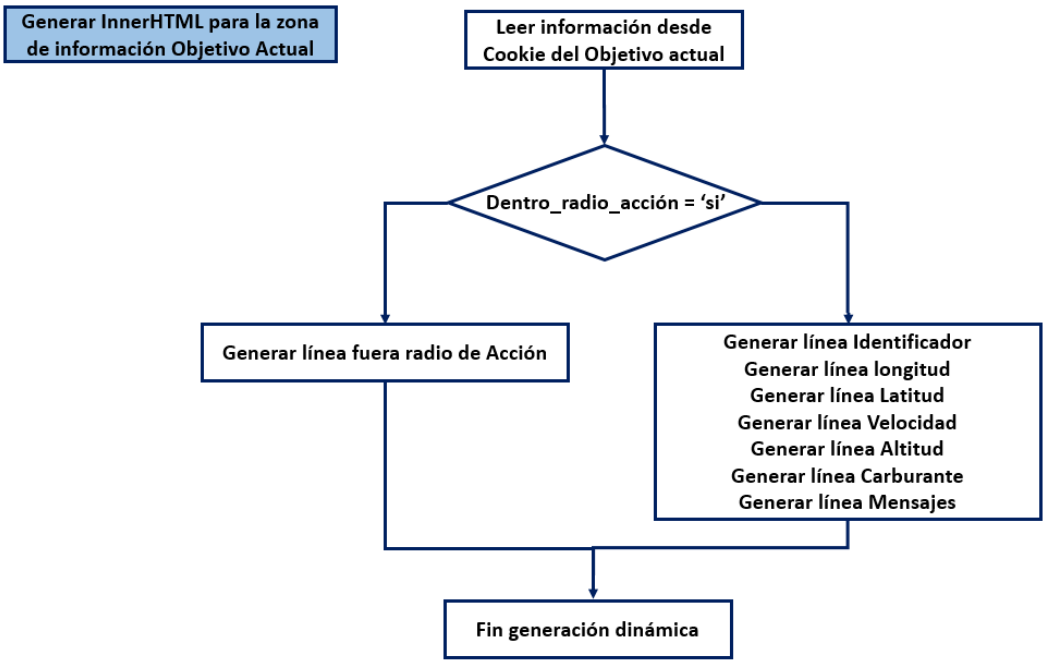


Ilustración 121. Flujograma de la Generación "InnerHTML" para la zona de información Objetivo Actual

5.2.2.2.4.3 Generación dinámica de la información de los demás objetivos.

Una vez concluida la parte referente al objetivo actual, realiza el mismo proceso para cada uno de los objetivos. En una iteración por cada una de las aeronaves, identificadas con un identificador de objetivo distinto, de entre los existentes en el Array de objetivos. Realiza esta condición para todos los objetivos, excepto para el objetivo actual, leyendo la información guardada en la cookie correspondiente y evaluando si la aeronave se encuentra aún dentro del radio de acción del mapa, para en caso negativo hacer mención de esta información, sin mostrar ninguna información más de ésta, y en caso afirmativo, mostrar la información relativa a la altitud, longitud, altura, velocidad, nivel de carburante, e indicadores tal cual estaban la última vez que se obtuvo información de esta aeronave, o de la última posición que habrá sido calculada en caso de estar el indicador de simular trayectorias activo.

Por cada aeronave de la que obtiene información, genera una etiqueta TABLE de HTML, con sus correspondientes TR y TD para maquetar la información.

La siguiente imagen, muestra cómo se realiza el barrido por los diferentes objetivos.

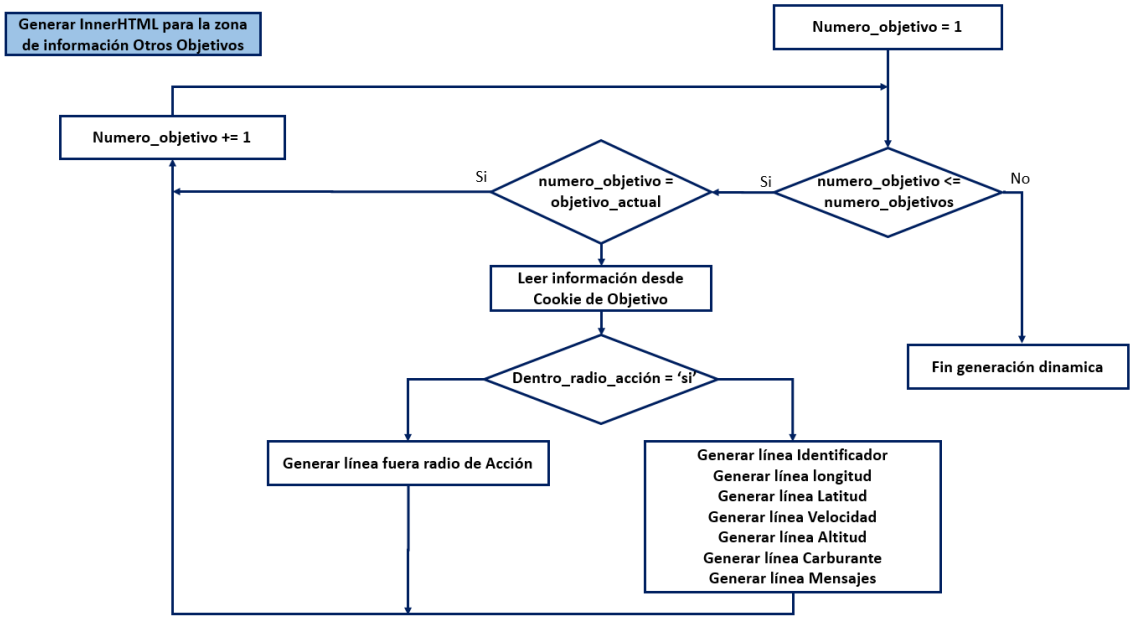


Ilustración 122. Flujograma de la Generación "InnerHTML" de la zona de información Otros Objetivos.

5.3 Capturas de simulación. Resultados obtenidos del capítulo.

Esta parte final del capítulo, busca ofrecer mediante capturas de simulación de los entornos de desarrollo utilizados, una demostración de los resultados obtenidos.

- ***Captura de pantalla de bienvenida de la página web.*** *Esta primera captura ofrece una visualización del entorno de bienvenida de la página web, sin ningún parámetro recibido.*

Ilustración 123. Captura de pantalla de bienvenida de la página web.

Explicación de la captura de simulación obtenida. *La captura ilustra la bienvenida de la página web con todas las cookies borradas y a la espera de la recepción de los números primos para comenzar a rellenar los primeros cuadros de texto de la derecha.*

- ***Captura de pantalla de la página web con los números primos y las claves del algoritmo RSA recibidos.*** *Esta captura ofrece una visualización de la página web, con la recepción de los números primos y las claves de cifrado a la espera de la recepción de alguna aeronave.*

Ilustración 124. Captura de pantalla de la página web con los números primos y las claves RSA.

Explicación de la captura de simulación obtenida. *La captura ilustra la página web con la recepción de los números primos y las claves de encriptación RSA asociadas al receptor. Actualmente, la página web se encuentra a la espera de la primera recepción de alguna aeronave con identificador no nulo.*

- ***Capturas de pantalla de la página web con recepciones de aeronaves.*** *Estas capturas ofrecen una visualización de la página web garantizando la recepción y visualización de múltiples aeronaves. Se pretende garantizar que funcionan los parámetros de habilitación asociados a la recepción, en concreto la habilitación de los cuadros de identificación y la simulación de las aeronaves de las que no se actualiza información.*

Ilustración 125. Capturas de pantalla de la página web con recepciones de aeronaves (1)

Explicación de la captura de simulación obtenida.
La captura ilustra la recepción de múltiples aeronaves, en esta primera captura se han habilitado los cuadros de información de las aeronaves, y deshabilitado la simulación de las trayectorias de los objetivos que no han actualizado su posición.

Ilustración 126. Capturas de pantalla de la página web con recepciones de aeronaves (2)

Explicación de la captura de simulación obtenida.
La captura ilustra la recepción de múltiples aeronaves, en esta segunda captura se han deshabilitado los cuadros de información de las aeronaves, y habilitado la simulación de las trayectorias de los objetivos que no han actualizado su posición.

- ***Capturas de pantalla de la página web con recepciones de múltiples aeronaves con diferentes identificadores y claves de cifrado.*** *El objetivo de este apartado de la memoria es visualizar distintos valores de claves y de posicionamientos de aeronaves.*

Primera captura de simulación, modificando los valores de las claves de encriptación y desencriptación junto con unos nuevos valores de identificadores de aeronaves.

Ilustración 127. Capturas de pantalla de la página web con recepciones de aeronaves (3)

Segunda captura, retornando los valores de todos los identificadores de las aeronaves previamente inicializados y modificando las rutas de todos ellos.

Ilustración 128. Capturas de pantalla de la página web con recepciones de aeronaves (4)

Tercera captura, habilitando la opción en el equipo receptor de realizar una simulación de trayectorias por parte de las aeronaves que no se han actualizado.

Ilustración 129. Capturas de pantalla de la página web con recepciones de aeronaves (5)

6. Estudio y diseño de la etapa RF implementada en PCB.

Esta parte del proyecto realiza el estudio y diseño para la futura implementación de la etapa de radiofrecuencia en una placa de circuito impreso (PCB). La función asociada al diseño de la PCB con un montaje RF es poder transmitir la señal del equipo transmisor al receptor por un medio no guiado (inalámbrico).

Debido a la falta de tiempo, en este apartado se realiza únicamente el estudio y diseño sin llegar a tener un montaje real. Por este motivo, se decide fusionar este capítulo con los capítulos referentes a líneas futuras y posibles mejoras del proyecto implementado.

En esta parte del proyecto, junto con el estudio de cara al diseño hardware de las etapas de transmisión y recepción, destaca la búsqueda por las páginas web de los diferentes fabricantes en búsqueda de los dispositivos necesarios para poder llevar a cabo una transferencia por radiofrecuencia con dispositivos al alcance del usuario.

A continuación, a modo de introducción antes de entrar en un análisis a fondo en ambas etapas, se ofrece una imagen y un resumen del diseño de cada una de las mismas.

- En la etapa de transmisión mediante un conmutador de alta frecuencia, haciendo uso de una portadora centrada en la banda ISM se modula una transmisión ASK con la señal moduladora obtenida en la salida de la FPGA. La salida del conmutador, será filtrada y amplificada para transmitirse vía RF hasta el receptor.

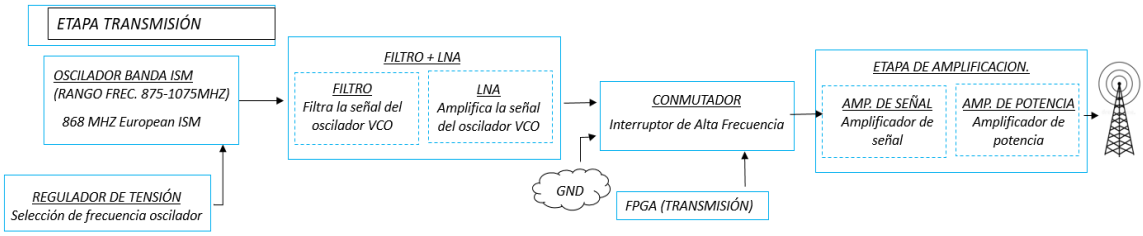


Ilustración 130. Resumen del diseño hardware etapa de transmisión

- En la etapa de recepción se recibe la señal RF, esta señal será filtrada con un filtro paso banda centrado en 868 MHz. La salida del filtro se introducirá en un mezclador con el objetivo de reducir la frecuencia de la portadora de la señal. Mediante un extractor de video, obtendremos la señal digitalizada con el fin de poder extraer los datos transmitidos.

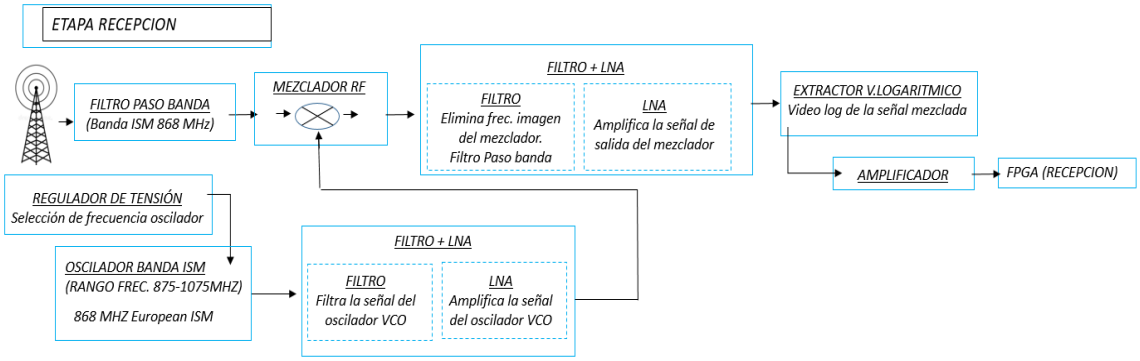


Ilustración 131. Resumen diseño hardware etapa de recepción

6.1 Estudio y diseño de la etapa RF transmisora

Este apartado de la memoria, realiza un estudio del diseño hardware de la etapa de transmisión. La siguiente imagen ofrece un diagrama de bloques más técnico con las conexiones y los dispositivos utilizados.

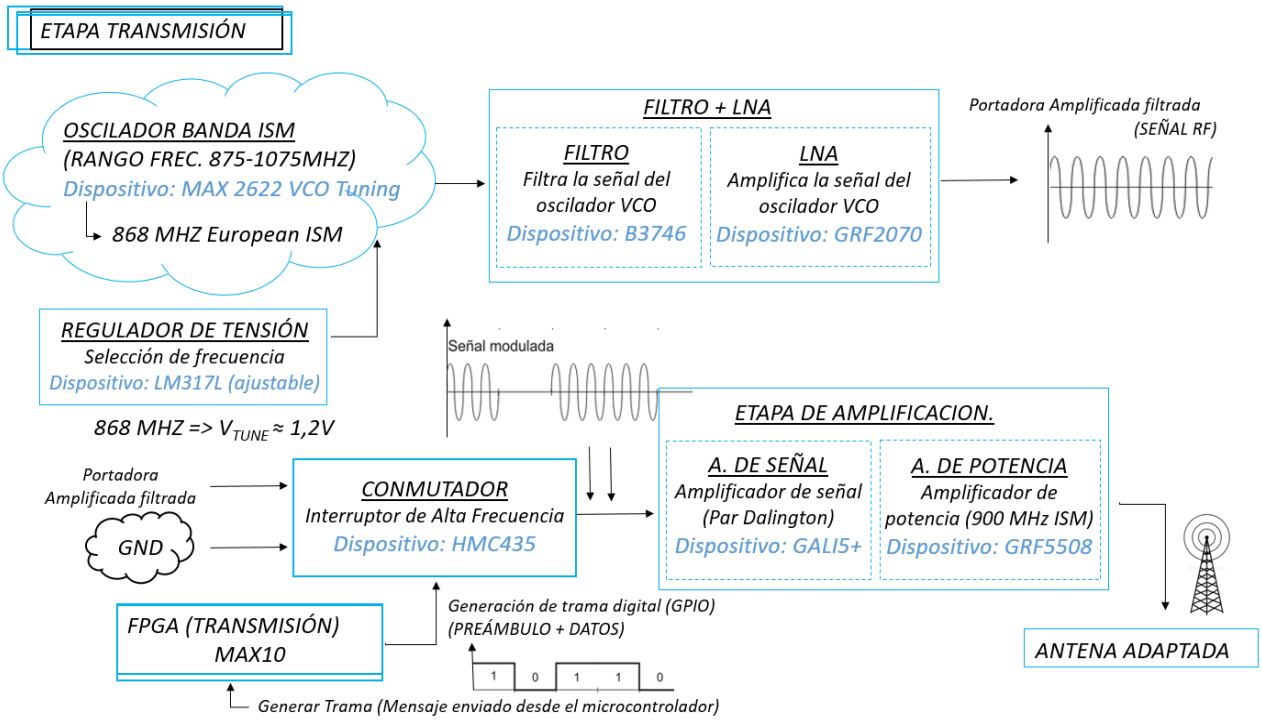


Ilustración 132. Resumen avanzado del diseño hardware etapa de transmisión

El funcionamiento del esquema hardware diseñado es el siguiente.

El oscilador se encuentra basado en un VCO *(Voltage-controlled oscillator)* este tipo de dispositivos genera una señal sinusoidal en función de la tensión introducida en uno de sus pines. En el diseño se ajusta la tensión de ese pin mediante un regulador de tensión. Este dispositivo generará la señal sinusoidal de forma ininterrumpida y, por lo tanto, será el encargado de generar la portadora de la señal RF. Con el fin de amplificar y filtrar posibles armónicos no deseados, se introduce la señal generada en el oscilador en una etapa de filtrado y amplificación adaptado a la frecuencia de trabajo.

La señal ASK se genera mediante un conmutador de alta frecuencia. El conmutador se controla mediante la señal moduladora obtenida en el pin digital de salida de la FPGA, conmutando por tanto entre la señal sinusoidal del oscilador o un nivel constante a cero.

La señal obtenida del conmutador antes de llevarla a una antena, se introduce en una nueva etapa de amplificación, realizando una amplificación en dos etapas, la primera de ellas referente a la señal y la segunda a la potencia, es importante que estos dispositivos se encuentren también adaptados a la frecuencia de trabajo.

En el anexo VI se puede observar una captura de la situación actúan en la que se encuentra actualmente el diseño en el entorno de desarrollo *EasyEDA* de la PCB de transmisión.

La siguiente parte del capítulo de la memoria, analiza de forma individual cada uno de los dispositivos claves utilizados en el diseño.

6.1.1 Oscilador controlado por tensión (VCO).

Un Oscilador controlado por tensión (Voltage-controlled oscillator) es un dispositivo electrónico que usa amplificación, realimentación y circuitos resonantes para ofrecer a su salida una señal eléctrica de frecuencia proporcional a la tensión de entrada.

En el proyecto se utiliza para generar la señal portadora necesaria en la comunicación por radiofrecuencia. La frecuencia utilizada es la banda libre europea ISM centrada en 868 MHz.

El dispositivo utilizado es un VCO con una frecuencia de operación en la banda ISM europea. El nombre del dispositivo es *MAX2622*. La conexión externa utilizada es la recomendada por el fabricante, tal y como se muestra en la siguiente captura.

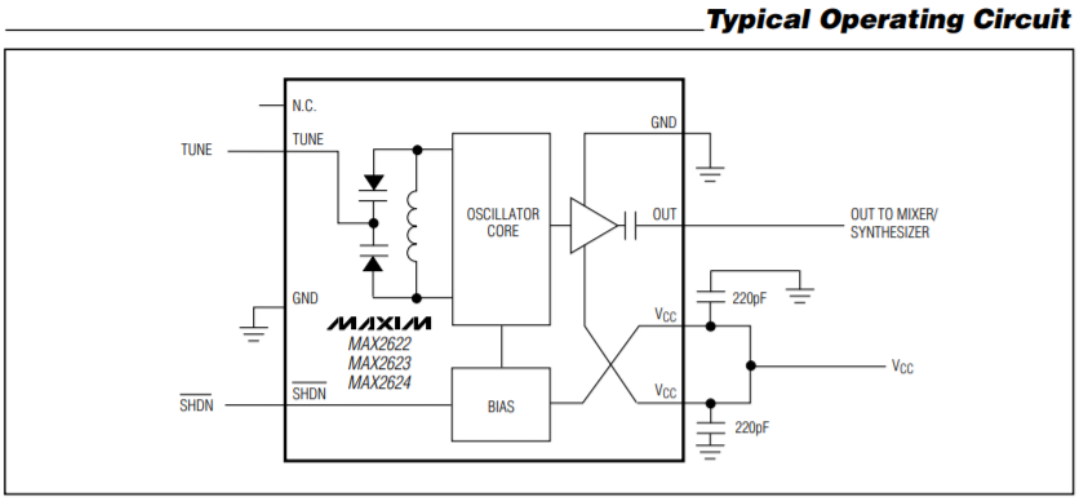


Ilustración 133. Conexión externa implementada del oscilador VCO.

De especial interés es el valor del pin denominado V_{TUNE}. Se trata de una entrada de voltaje para la sintonización de la frecuencia del oscilador. Se trata de una entrada de alta impedancia con un rango de entrada de voltaje de 0,4 V a 2,4 V.

La siguiente figura muestra tres graficas (una por cada dispositivo de la familia escogida) mediante las cuales se ilustra la relación entre la frecuencia de oscilación y la tensión introducida en el pin de sintonización. Se puede garantizar que para conseguir una frecuencia de operación próxima a la banda ISM el dispositivo que mejor se ajusta es el "*MAX2622"*. Y que, en este dispositivo elegido, el valor de tensión del pin debe de ser aproximadamente de 1.2 V.

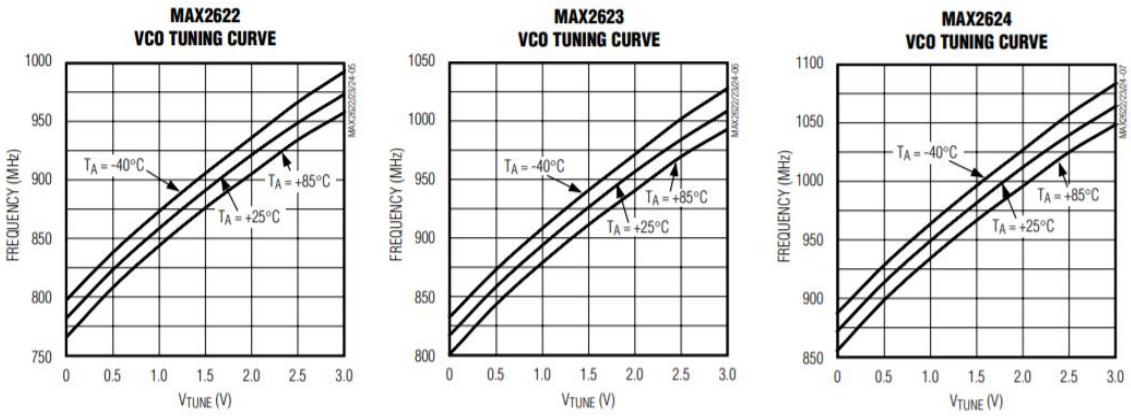


Ilustración 134. Graficas de funcionamiento del oscilador VCO.

6.1.2 Regulador de tensión.

Un regulador de tensión es un dispositivo electrónico diseñado para mantener un nivel de tensión constante, a partir de una alimentación de valor diferente.

En el proyecto tiene dos aplicaciones claramente diferenciadas, la primera es obtener las tensiones de alimentación recomendadas por los fabricantes, en los diferentes integrados del circuito, la segunda es conseguir los valores de tensión para los pines dedicados para los ajustes. El proyecto utiliza un total de cuatro reguladores de tensión.

El dispositivo utilizado es un regulador estándar que ofrece tensiones de salidas en un rango de 1.25 V hasta 32 V. El nombre del dispositivo es *LM317L.* La conexión externa utilizada es la recomendada por el fabricante, tal y como se muestra en la siguiente captura.

Typical Application

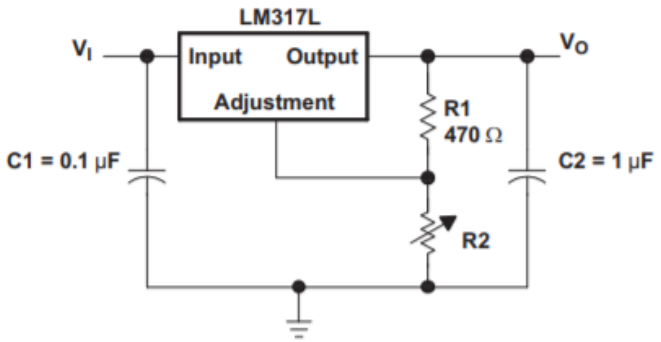


Ilustración 135. Conexión externa implementada del regulador de tensión.

Mediante el uso de un potenciómetro en la resistencia R2 se podrá ajustar la tensión de salida en el dispositivo. Esta tensión se modela mediante la siguiente formula:

$$V_{OUT} = V_{REF} \cdot \left(1 + \frac{R_2}{R_1}\right) + I_{ADJ} \cdot R_2$$

El valor típico de la corriente I_{ADJ} es de 50 µA, un valor despreciable en la mayoría de aplicaciones.

6.1.3 Filtro y amplificador LNA.

Este apartado estudia de forma conjunta el filtro y el amplificador LNA utilizados en esta etapa. Ambos dispositivos deberán de estar adaptados para trabajar en la frecuencia de operación (Banda ISM europea de 868 MHz).

- ***En primer lugar, se analiza el filtro utilizado en el proyecto.***

Un filtro paso banda es un tipo de filtro electrónico que deja pasar un determinado rango de frecuencias de una señal y atenúa el paso del resto.

El proyecto utiliza este tipo de filtros con el objetivo de poder eliminar todos los posibles tipos de armónicos no deseados que haya podido generar el oscilador. Puesto que el oscilador pretende generar un tono centrado en la banda ISM, la banda de pasa, deberá de estar centrada en esta misma banda.

El dispositivo utilizado es un filtro paso banda de tipo SAW componentes, con la banda centrada en la ISM europea (868 MHz) denominado *"B3746"*.

La siguiente captura muestra una imagen de la respuesta en frecuencia del filtro elegido.

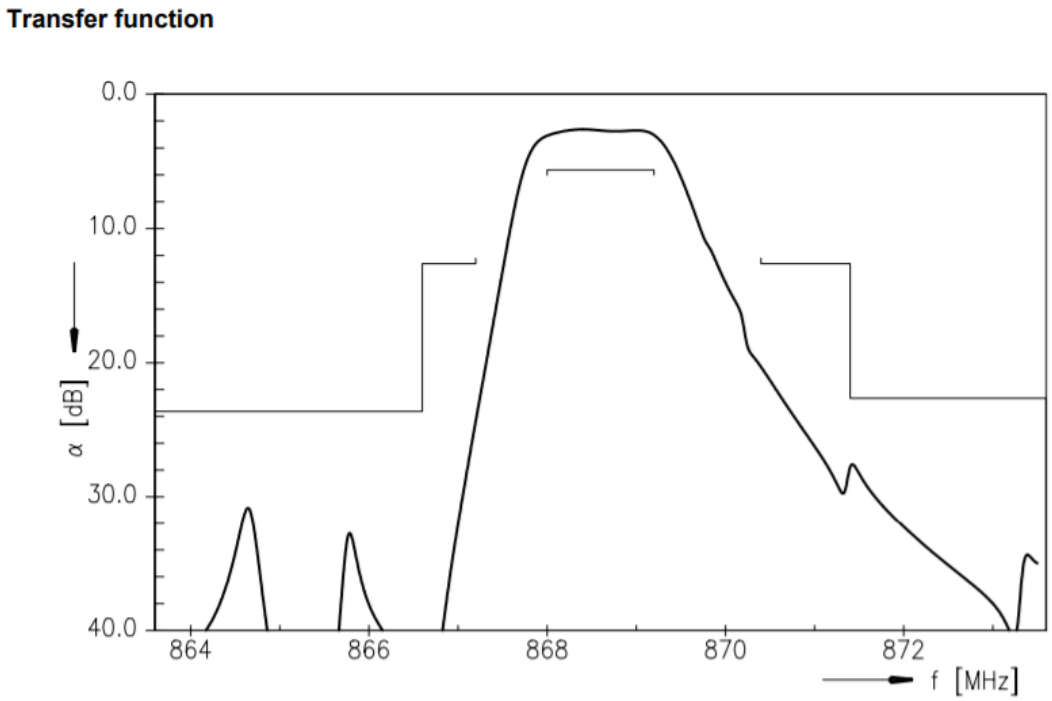


Ilustración 136. Respuesta en frecuencia del filtro paso banda, centrado en la banda ISM

La conexión externa adaptada a una impedancia de 50 Ω, utilizada es la recomendada por el fabricante, tal y como se muestra en la siguiente captura.

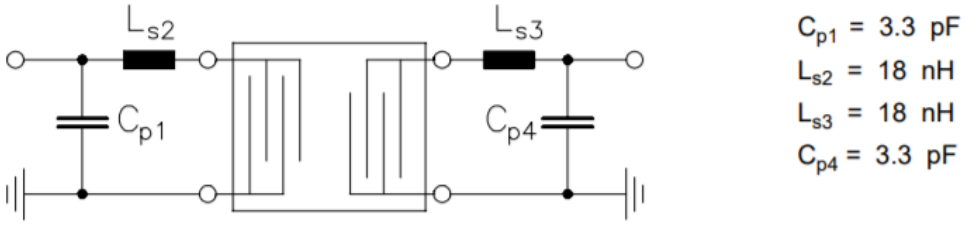


Ilustración 137. Conexión externa implementada del filtro paso banda, centrado en la banda ISM.

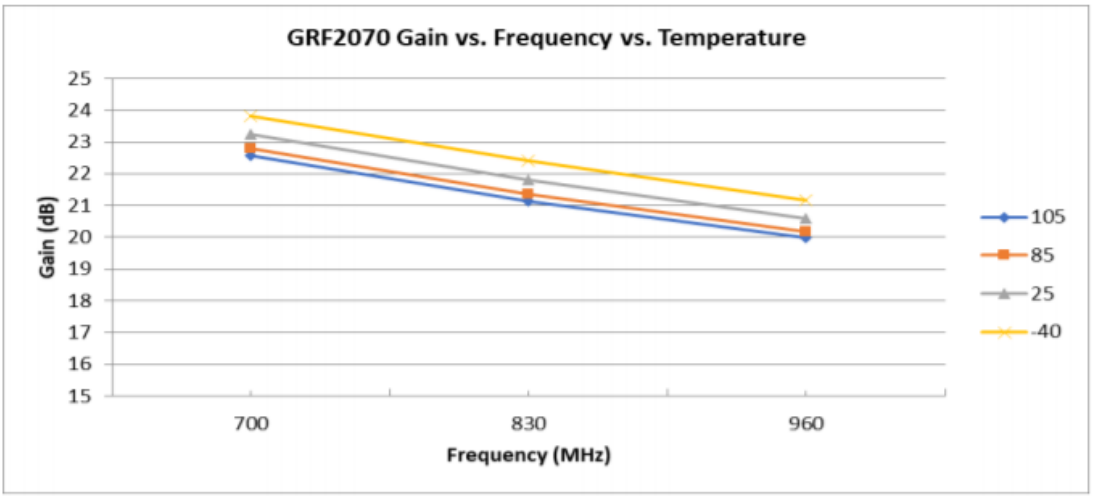
- ***En segundo lugar, se analiza el amplificador LNA utilizado en el proyecto.***

El amplificador utilizado es un amplificador LNA (Low Noise Amplifier) Un amplificador de bajo ruido es un amplificador electrónico que amplifica una señal de muy baja potencia sin degradar significativamente su relación señal/ruido.

Debido a las perdidas introducidas por el filtro paso banda anterior, aparece la necesidad de amplificar la señal referente a la portadora generada en el oscilador. Es importante elegir un dispositivo preparado para trabajar con señales de RF y aclimatado a la banda frecuencial de trabajo.

El dispositivo utilizado es un amplificador LNA, con una ganancia de 20.8 dBs y un rango de trabajo de 0.1 a 1.5 GHz denominado *"GRF2070"*.

Las siguientes figuras muestran primer lugar una representación de la ganancia y la figura del ruido en función de la frecuencia de trabajo y la temperatura y, en segundo lugar, una representación de los parámetros S del amplificador.



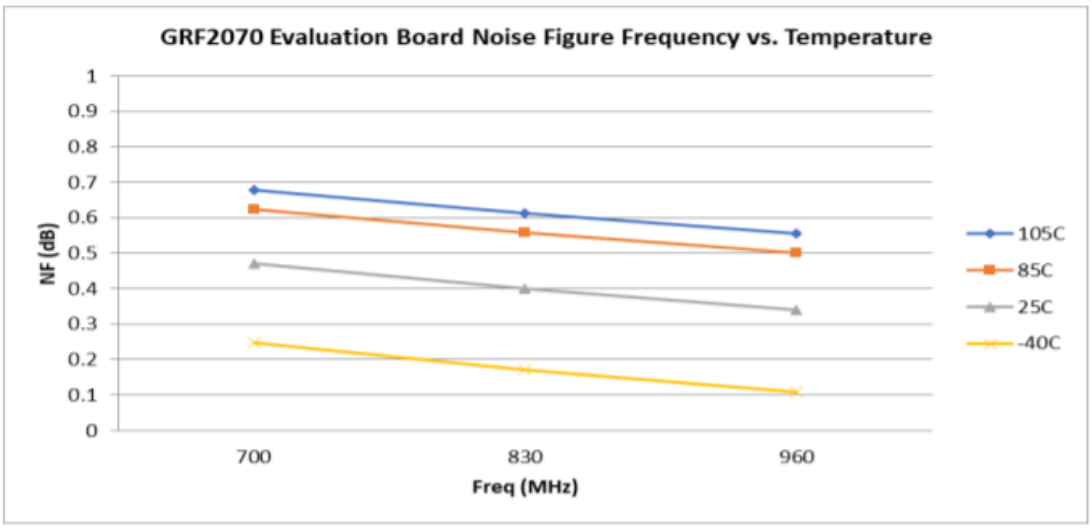


Ilustración 138. Respuesta en frecuencia del amplificador LNA

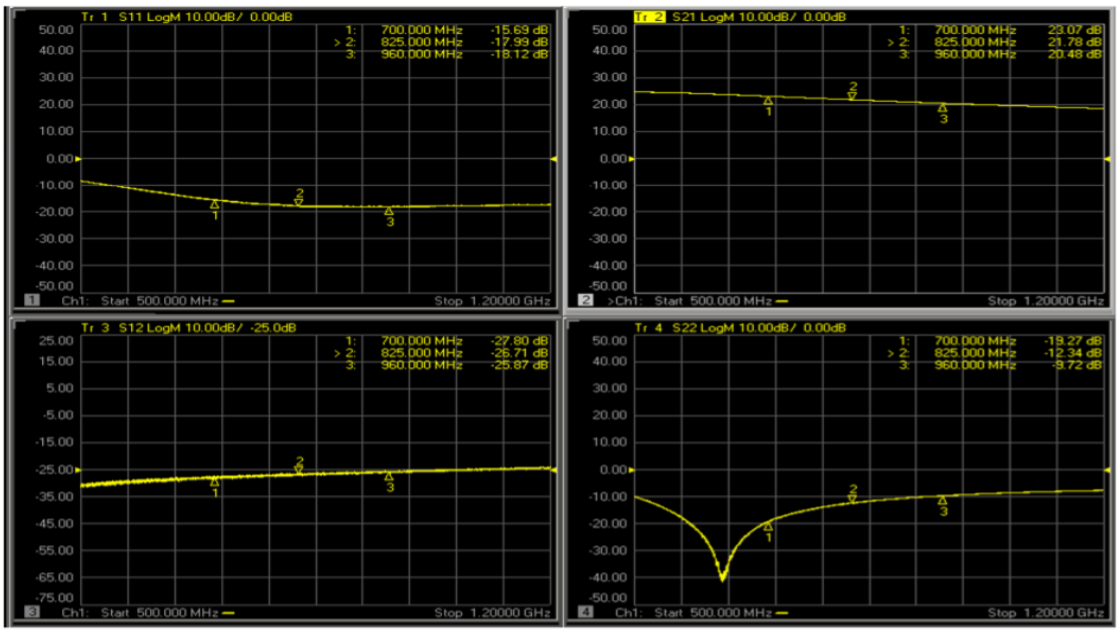
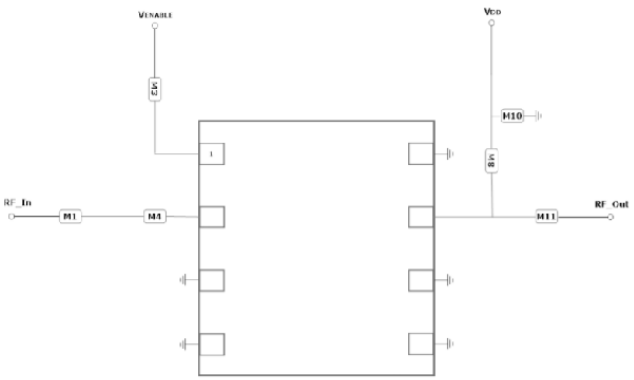


Ilustración 139. Parámetros S del amplificador LNA

Por último, se analiza la conexión externa del amplificador adaptado, por norma general, se utilizada la recomendada por el fabricante, tal y como se muestra en la siguiente captura extraída de la hoja de características.



Component	Type	Value
M1	Resistor (jumper)	0 Ohm
M3	Resistor	Sets Iddq
M4	Capacitor	47 pF
M8	Inductor	22 nH
M10	Capacitor	0.1 uF
M11	Capacitor	4.7 pF

Ilustración 140. Conexión externa implementada en el amplificador LNA

6.1.4 Conmutador.

Los conmutadores de RF se utilizan para seleccionar una señal deseada de entre varias fuentes disponibles, o para dirigir una señal a un canal que se desee.

El conmutador se utiliza en el proyecto para conseguir implementar la modulación ASK con la moduladora obtenida en la FPGA. Gracias al conmutado en función de los valores de salida de la FPGA, se podrá obtener la señal portadora o un valor constante a tierra consiguiendo así poder implementar la señal RF.

El conmutador utilizado es un dispositivo capaz de trabajar con frecuencias de hasta 4 GHz con valores típicos de pérdidas de inserción de 0.8 dBs El dispositivo se denomina con el nombre "*HMC435"*.

La siguiente captura muestra una imagen de la estructura interna del dispositivo junto con las pérdidas de inserción introducidas en función de la frecuencia de operación.

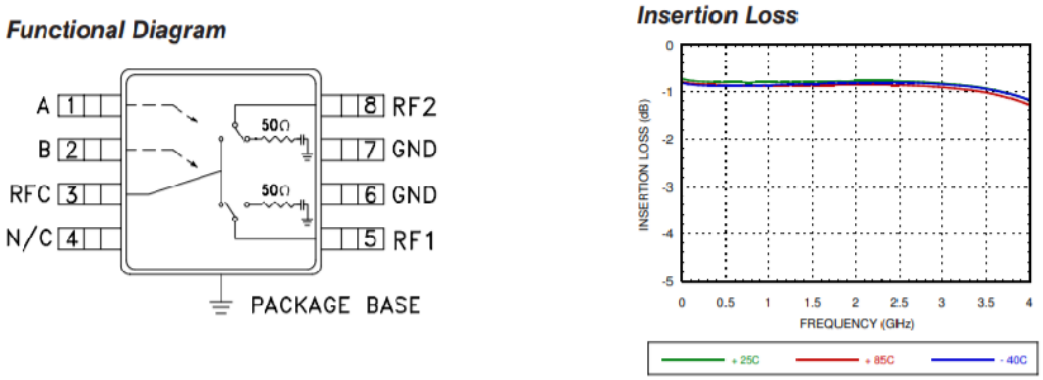


Ilustración 141. Estructura interna del conmutador y perdidas de inserción introducidas

De cara a la conexión del dispositivo, se utiliza la recomendada por el fabricante en la hoja de características. Es importante darse cuenta de que, con el tipo de conexión recomendada, aparece la necesidad de utilizar dos inversores digitales, para conseguir estos dos inversores digitales, en el proyecto se utiliza el integrado "*7404"*.

La conexión externa recomendada por el fabricante de muestra en la siguiente captura de la hoja de características.

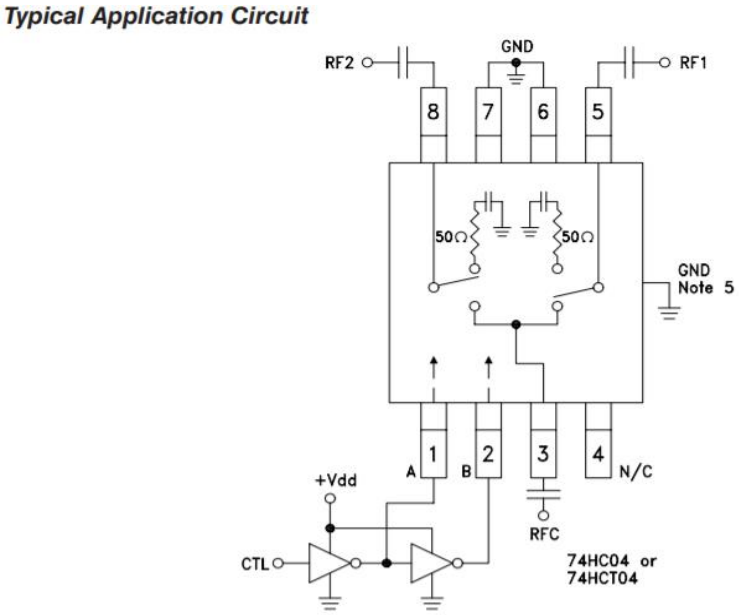


Ilustración 142. Conexión externa implementada en el conmutador.

6.1.5 Amplificación de señal y amplificación de potencia.

Este apartado estudia de forma conjunta el amplificador de señal y el amplificador de potencia utilizados en esta etapa. Ambos dispositivos deberán de estar adaptados para trabajar en la frecuencia de operación, se recuerda que se trata de la banda ISM europea (868 MHz).

El proyecto utiliza estos dispositivos para amplificar conjuntamente la señal y la potencia de la modulación generada en el conmutador de RF. Se pretende poder garantizar una amplitud y una potencia mínima para que la señal pueda llegar al receptor.

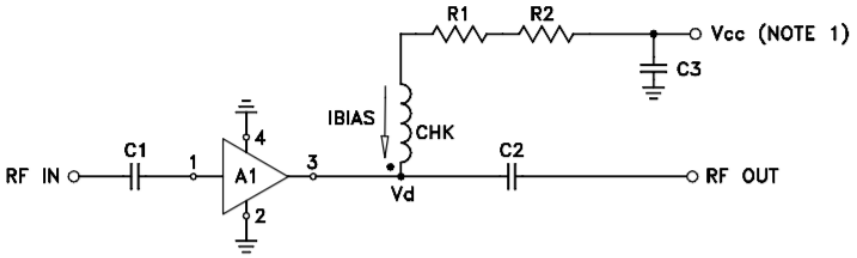
- ***En primer lugar, se analiza el amplificador de señal utilizado en el proyecto.***

Este primer amplificador, busca amplificar la señal de entrada, para ello se ha optado por utilizar un amplificador con una estructura interna de *"Par Darlington"*.
Esta configuración combina dos transistores bipolares en un mismo integrado, de esta forma el dispositivo es capaz de proporcionar una mayor ganancia de corriente.

El conmutador utilizado es un dispositivo capaz de trabajar con frecuencias de hasta 4 GHz con señales típicas de radiofrecuencia. El dispositivo se denomina con el nombre "*GALI 5+"*. La siguiente captura muestra un esquema interno del dispositivo.

Ilustración 143. Estructura interna del amplificador de señal.

De cara a la conexión del dispositivo, se utiliza la recomendada por el fabricante en la hoja de características, con los valores especificados en una tarjeta de evaluación.



COMPONENT	VALUE
A1	Gali-5(+)
C1 (NOTE 4)	2400 pF
C2 (NOTE 4)	2400 pF
C3 (bypass)	0.1 uF
R1	115 Ohms, 0.75W
R2	2.21 Ohms, 0.25W
CHK	Mini-Circuits TCCH-80+

Ilustración 144. Conexión externa implementada en el amplificador de señal.

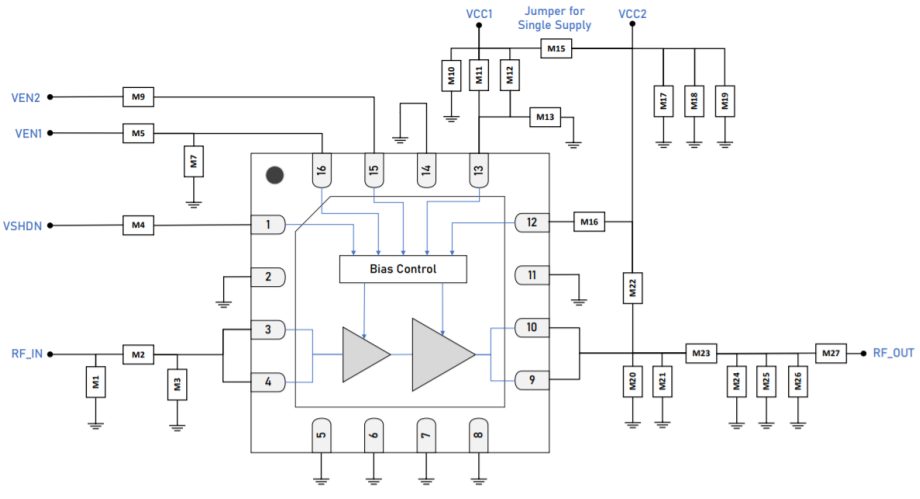
De especial interés es la bobina utilizada, siendo esta una bobina de choque, el objetivo de esta bobina es tener una reactancia muy grande a una frecuencia o rango de frecuencias determinadas.

- ***En segundo lugar, se analiza el amplificador de potencia desarrollado en el proyecto.***

Este segundo amplificador, busca amplificar la potencia entregada a la antena. Un amplificador de potencia es aquel cuya etapa de salida se ha diseñado para que sea capaz de generar unos rangos de tensión e intensidad más amplios, de forma que tenga capacidad de transferir a la carga la potencia que se requiere.

El dispositivo utilizado se encuentra adaptado a la frecuencia de operación, en concreto a la banda comprendida entre los 800 MHz y los 900 MHz. El amplificador lineal de potencia. El dispositivo se denomina con el nombre "*GRF5508"*.

De especial interés es la conexión externa del dispositivo. Recomendando el fabricante la mostrada en la siguiente imagen.



GRF5508 Application Schematic

Ilustración 145. Conexión externa implementada en el amplificador de potencia

Los valores de los componentes se identifican en la siguiente tabla.

GRF5508 Evaluation Board Assembly Diagram Reference

Component	Type	Manufacturer	Family	Value	Package Size	Substitution
M1	Inductor	Murata	LQG	8.2 nH	0402	ok
M2	Capacitor	Murata	GJM	6.8 pF	0402	ok
M3	DNP					
M4	Resistor	Various	5%	0 Ohm	0402	ok
M5	Resistor	Various	5%	1.7 kΩ	0402	ok
M7	Capacitor	Murata	GRM	100 pF	0402	ok
M9	Resistor	Various	5%	3.3 kΩ	0402	ok
M10	Capacitor	Murata	GRM	0.1 µF	0402	ok
M11	Inductor	Murata	LQG	6.8 nH	0402	ok
M12	DNP					
M13	DNP					
M15	Resistor	Various	5%	0 Ohm	0402	ok
M16	Resistor	Various	5%	0 Ohm	0402	ok
M17	DNP					ok
M18	Capacitor	Murata	GRM	10 µF	0402	ok
M19	Capacitor	Murata	GRM	100 pF	0402	ok
M20	DNP					
M21	DNP					
M22	Inductor: High Q	Murata	LQW	24 nH	0603	ok
M23	Inductor: High Q	Murata	LQW	2.4 nH	0402	ok
M24	Capacitor	Murata	GJM	8.2 pF	0402	ok
M25	DNP					
M26	DNP					
M27	Capacitor	Murata	GJM	47 pF	0402	ok

Con el análisis realizado por completo, realizando un barrido por cada uno de los dispositivos utilizados en la etapa, se añade en el anexo VII las capturas del entorno de diseño hardware (EasyEDA) de la etapa de transmisión realizada.

6.2 Estudio y diseño de la etapa RF receptora

Este apartado de la memoria, realiza un estudio del diseño hardware de la etapa de recepción. La siguiente imagen ofrece un diagrama de bloques más técnico con las conexiones y los dispositivos utilizados.

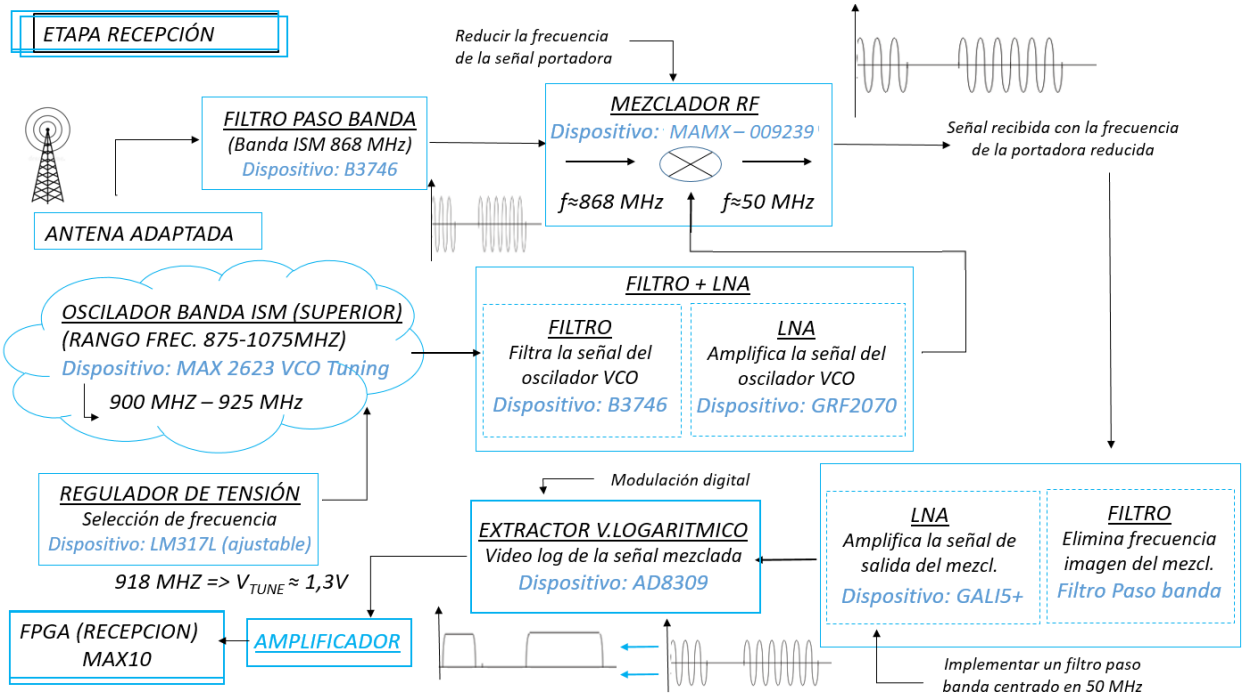


Ilustración 146. Resumen avanzado del diseño hardware etapa de recepción.

El funcionamiento del esquema hardware diseñado es el siguiente.

Mediante un filtro paso banda centrada en la banda ISM de trabajo, se discrimina y recibe únicamente las señales enviadas desde el equipo transmisor. Con el objetivo de disminuir la frecuencia de la portadora de trabajo, se introduce la señal en un sistema mezclador.

Las señales de entrada al mezclador son por un lado la señal de salida del filtro paso banda comentado en el párrafo anterior y la señal de salida filtrada y amplificada de un oscilador VCO similar al de la etapa de transmisión. La salida del mezclador se introduce en una nueva etapa de filtrado y amplificación, por la necesidad de eliminar la frecuencia imagen que aparece por el mezclador.

La salida de la etapa anterior, se introduce a un dispositivo denominado extractor de video logarítmico, este dispositivo produce una señal de salida de tensión proporcional a la potencia de la señal de entrada, por lo que podemos garantizar que este dispositivo conseguiría el paso de la señal analógica a la señal digital. La salida del extractor se introduce un amplificador estándar y la salida de este a un pin de entrada de la FPGA para su futuro procesamiento.

La siguiente parte del capítulo de la memoria, analiza de forma individual cada uno de los dispositivos claves utilizados en el diseño y que no se hayan reciclado de la etapa de transmisión.

6.2.1 Dispositivos reciclados del diseño de la etapa de transmisión.

Entre los dispositivos reciclados del proyecto de transmisión, se pueden destacar los siguientes dispositivos.

- Filtro paso banda, con banda centrada en 868 MHz (Banda ISM). Este dispositivo se utiliza para filtrar todas las señales RF del espectro aéreo, obteniendo únicamente las transmitidas en esa banda frecuencial. El dispositivo y la conexión externa es la misma que en la etapa de transmisión.
- Oscilador VCO. Este dispositivo se utiliza para generar la señal sinusoidal que se introducirá en el dispositivo mezclador. Es necesaria una señal con una frecuencia ligeramente superior, para que la mezcla de la recibida con la generada, ofrezca una señal con una componente frecuencial baja. El dispositivo es de la misma familia, pero un modelo diferente, en este caso se utiliza el modelo *"MAX2623"* generando así una señal sinusoidal de frecuencia superior a la del modelo *"MAX2622"*. La conexión externa es la misma que en la etapa de transmisión. La siguiente imagen muestra los tres dispositivos de la familia elegida para implementar los osciladores.

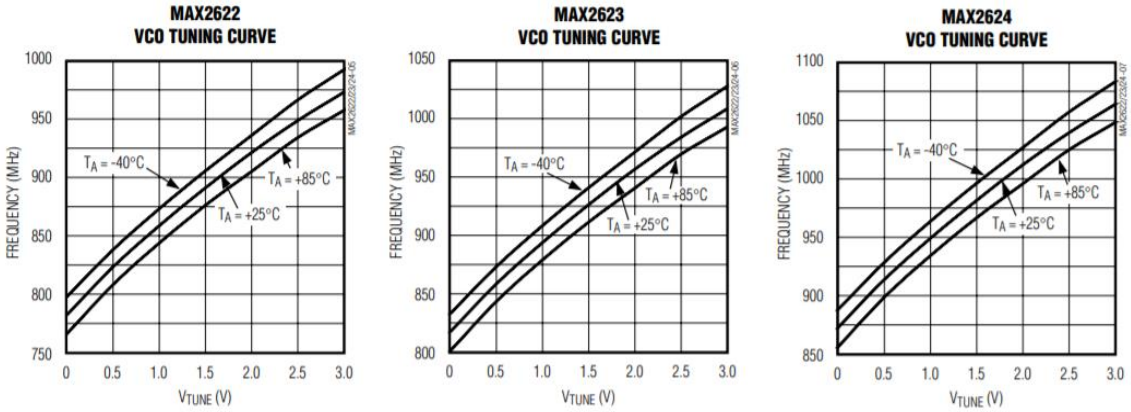


Ilustración 147. Graficas de funcionamiento del oscilador VCO

- Regulador de tensión. Este dispositivo tiene la misma función que en la etapa de transmisión, se utiliza igual y se conecta igual también.
- Etapa de filtrado y amplificación LNA. Esta etapa tiene la función de filtrar y amplificar la señal generada en el oscilador, en la etapa de transmisión, el oscilador se utilizaba para generar la portadora, mientras que en la recepción se utiliza para generar la señal mezcladora. La función del oscilador en el proyecto es irrelevante para la etapa de filtrado y amplificado estudiada en este punto. En conclusión, los dispositivos y las conexiones externas son las mismas que en la etapa de transmisión.
- Amplificador LNA de la señal mezclada. Este dispositivo tiene la misma función que en la etapa de transmisión, se utiliza igual y se conecta igual también

Con el desarrollo hardware estudiado hasta el momento en este epígrafe. Se dispone por un lado de la señal recibida en la antena (filtrada en la banda ISM) y por otro lado de una señal sinusoidal (filtrada y amplificada), con componente frecuencial ligeramente superior a la de la señal recibida en la antena.

Finalizado el estudio y resumen de los dispositivos reciclados de la etapa anterior, se analizan los introducidos nuevos de esta etapa.

6.2.2 Mezclador RF.

En el diseño de circuitos electrónicos, un mezclador es un dispositivo capaz de mezclar dos señales de entrada, generalmente de diferentes frecuencias, produciendo a su salida una mezcla de señales de diferentes frecuencias igual a una combinación lineal de las dos frecuencias de entrada.

El mezclador se utiliza en el proyecto para conseguir reducir la frecuencia portadora de la señal recibida. Se mezcla, la señal recibida en la antena (filtrada) con la generada en el oscilador local (filtrada y amplificada). Debido a que la señal generada en el oscilador, es ligeramente superior a la recibida, conseguimos trabajar en frecuencias de decenas de megahercios.

El mezclador utilizado es un dispositivo capaz de trabajar con frecuencias desde 10 MHz hasta 2500 MHz y valores de señales de RF superiores a los 13 dBm. El dispositivo se denomina con el nombre "*MAMX 009239"*.

La siguiente captura muestra una imagen extraída de las hojas de características del fabricante, representando las pérdidas de conversión en función de la frecuencia de operación.

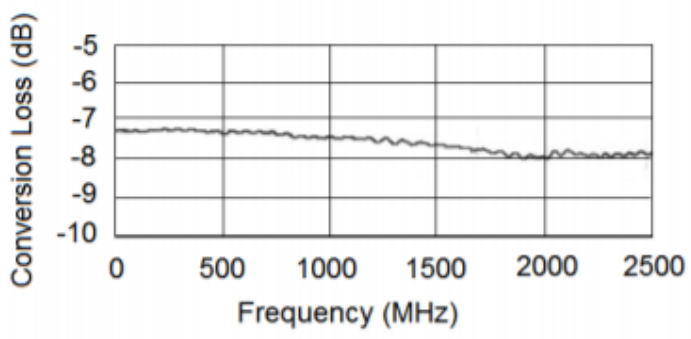


Ilustración 148. Perdidas de conversión del mezclador en función de la frecuencia de operación

6.2.3 Filtro y amplificador LNA en baja frecuencia.

Este apartado estudia de forma conjunta el filtro y el amplificador LNA utilizados en esta etapa. Ambos dispositivos deberán de estar adaptados para trabajar en la frecuencia reducida previamente por el mezclador (frecuencia de operación cercana a los 50 MHz).

- ***En primer lugar, se analiza el filtro utilizado en el proyecto.***

Se trata de un filtro paso banda, con una banda de paso centrada en los 50 MHz, al tratarse de un filtro particular, aparece la necesidad de tener que realizar el diseño del mismo mediante componentes pasivos de bobinas y condensadores. La función principal de este filtro es la de eliminar la frecuencia imagen que aparece en altas frecuencias en el mezclador.

Es importante entender que el mezclador, realiza la combinación lineal de ambas entradas, utilizando la operación suma y resta, es decir, al igual que aparece la señal deseada en bajas frecuencias, por la resta de ambas señales, también aparecerá una no deseada en altas por la suma de ambas componentes frecuenciales, la función de este filtro es la de eliminar esa componente.

Un detalle importante es la elección del filtro que se desea introducir en esta etapa de filtrado, los filtros previamente seleccionados para el diseño son los filtros elípticos, de Chebyshev, y de Butterworth.

En la siguiente imagen se puede observar una respuesta ideal de la respuesta en frecuencia para un filtro paso bajo de cada uno de los tipos de filtros previamente elegidos.

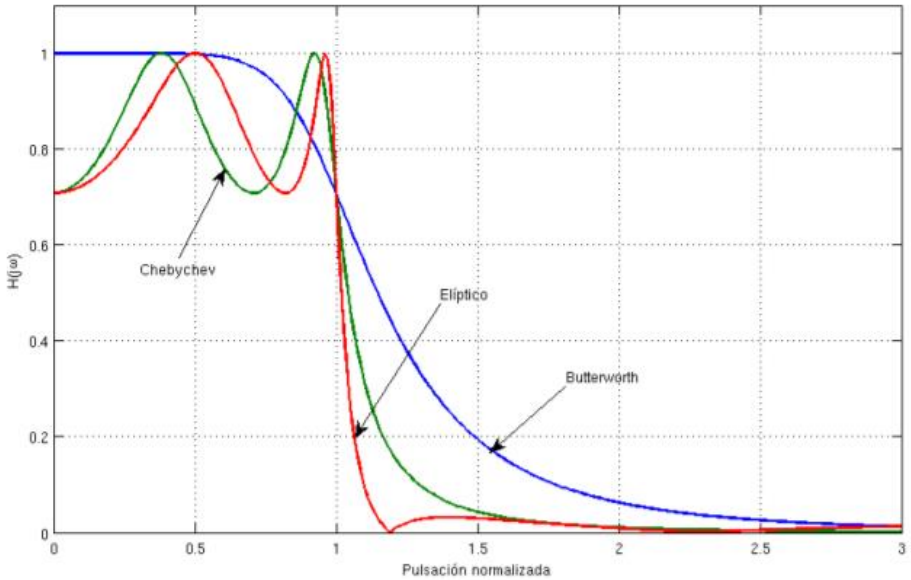


Ilustración 149. Comparación de filtros de Butterworth, Chebychev y elíptico de orden 4.

Obviamente nunca podremos conseguir estas respuestas en nuestros diseños, por lo que se profundiza y estudia cómo se comportan estos filtros con componentes reales y no ideales con el fin de obtener una elección más acertada.

Un detalle muy importante que aparece en este tipo de filtros es el retardo de grupo, que consiste en el retardo aproximado que introduce la envolvente de una onda al propagarse entre dos puntos. En la siguiente imagen se muestra una captura del efecto previamente descrito para cada uno de los filtros previamente seleccionados.

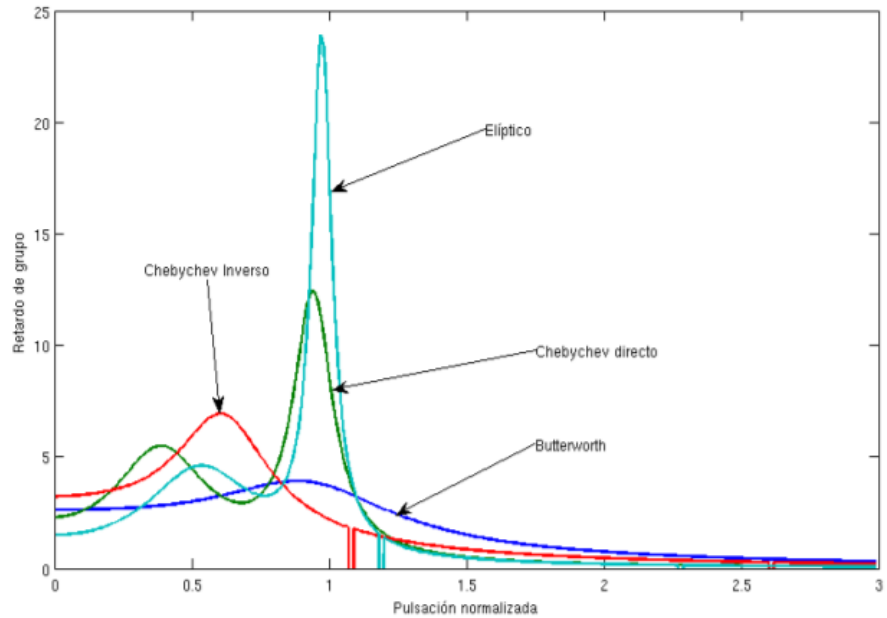


Ilustración 150. Comparación de las gráficas de retardo de grupo de filtros Butterworth, Chebychev Directo e inverso y elíptico de orden 4.

La elección del filtro se fundamenta en la búsqueda de la máxima atenuación posible en bandas que no sean las deseadas, el filtro de Butterworth introduce menos rizado en la banda de paso, pero una atenuación más lenta, motivo por el cual se descarta este filtro.

Los filtros elipticos y de Chebyshev se caracterizan por una caída muy pronunciada de la respuesta en frecuencia. Otro detalle característico es que aparece un alto rizado en la banda de paso. A diferencia del Filtro de Butterworth donde los polos se distribuyen sobre una circunferencia.

Por otro lado, se puede apreciar un mayor retardo de grupo en comparación con todos los demás en el filtro elíptico.

Se elige un filtro de Chebyshev en lugar de un filtro de Butterworth debido a que ofrece una caída más pronunciada al comenzar las bandas de atenuación. Por otro lado, también se elige este tipo de filtro frente al elíptico para poder eliminar el alto valor del retardo de grupo que aparece en este último.

Es importante recordar, que el objetivo de este filtro es obtener del mezclador, solo las señales que presenten una componente frecuencial cercana de 50MHz.

- ***En segundo lugar, se analiza el amplificador LNA utilizado en el proyecto.***

Tal y como se ha comentado previamente. Debido al alto ancho de banda que presenta este amplificador, se decide reciclar de la etapa de transmisión.

6.2.4 Extractor de video logarítmico.

La extracción de video logarítmico se lleva a cabo mediante un amplificador logarítmico. A continuación, de analiza en profundidad este tipo de amplificador.

Los amplificadores logarítmicos son amplificadores analógicos no lineales que producen una salida que es el logaritmo de la señal de entrada o la envoltura de la señal. Comprimen las señales de entrada que tienen un amplio rango dinámico en señales de salida con un rango de amplitud fijo. Esto se logra al proporcionar una alta ganancia para niveles de señal de entrada bajos y una ganancia progresivamente más baja para señales de nivel más alto

El amplificador logarítmico se utiliza en el proyecto para conseguir extraer la señal moduladora de la señal recibida, es importante entender que actualmente la señal recibida tiene una modulación por desplazamiento de amplitud (ASK), donde predomina claramente la señal de la portadora sinusoidal que utilizada en la transmisión.

Se ilustra mediante las siguientes figuras un ejemplo de las señales de entrada y salida del extractor, con el objetivo de aclarar el funcionamiento este dispositivo.

- *Señal de entrada al extractor de video logarítmico.*

$$v_{in}(t) = x(t) \cdot \cos(\omega t)$$

Ilustración 151. Representación de la señal de entrada del extractor de video logarítmico

- *Señal de salida del extractor de video logarítmico.*

$$v_{out}(t) = x(t)$$

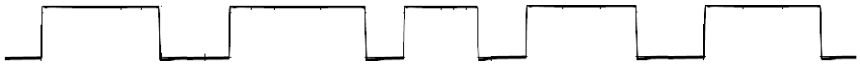


Ilustración 152. Representación de la señal de salida del extractor de video logarítmico.

El objetivo como se ha explicado previamente es conseguir eliminar la señal portadora, obteniendo solo la señal moduladora. En la siguiente imagen ofrecida por la hoja de características del fabricante, se puede apreciar claramente cómo se consigue en la salida el objetivo que hemos descrito previamente.

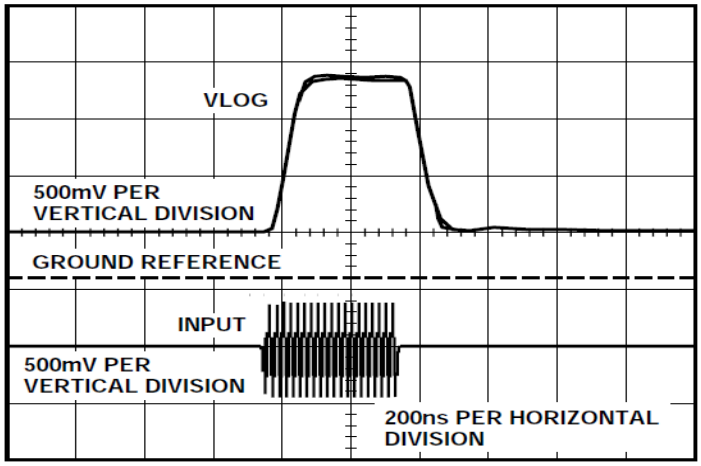


Ilustración 153. Ejemplo de extracción de video logarítmico.

La clave para la extracción del video, es que este proporciona una salida en tensión proporcional a la potencia de la señal que tenemos en la entrada. De esta forma, cuando aparece señal modulada en la entrada, el dispositivo genera una tensión en la salida proporcional a dicha amplitud, mientras que, si no aparece señal en la entrada, tampoco lo hará en la salida, debido a que la potencia de dicha señal seria nula.

Para concluir, se realiza un análisis de cómo evoluciona la señal de salida (V_{OUT}) en función del logaritmo de la tensión de entrada (V_{IN}).

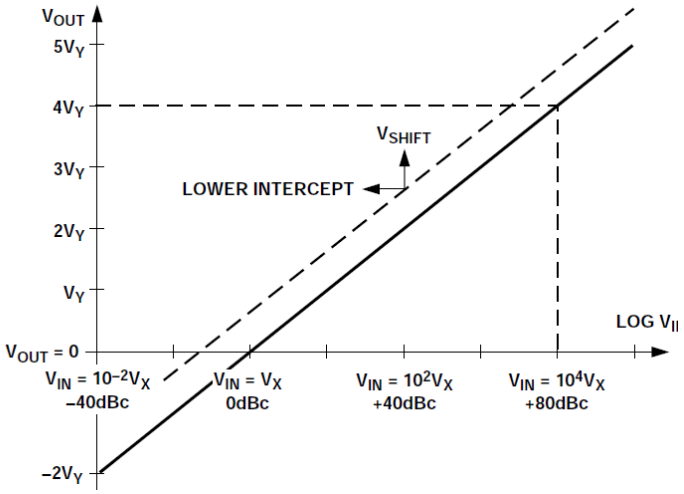


Ilustración 154. Tensión de salida en función de la tensión de entrada del extractor de video

Para comenzar se declara una tensión umbral, con la cual obtendremos una tensión de salida nula ante una entrada de ese valor, se recomienda introducir en este valor el ruido del sistema, con el fin de que el amplificador logarítmico lo elimine. Declarado este valor, obtendremos una constante proporcional al logaritmo en base diez del cociente entre la tensión de entrada y la tensión umbral. La tensión de salida será proporcional a la constante previamente obtenida la cual se utilizará como coeficiente del producto a la tensión utilizada como referencia en la salida.

De la hoja de características, se deduce la siguiente ecuación de salida en función de las constantes previamente descritas.

$$V_{OUT} = V_Y \log (V_{IN}/V_X)$$

Garantizando una pendiente, la cual responde a la siguiente ecuación.

$$\frac{\Delta V_{OUT}}{\Delta V_{IN}} = \frac{V_Y}{V_{IN}}$$

La pendiente es inversamente proporcional al valor de V_{IN}, gracias a este detalle, el amplificador se adapta mejor al ruido del sistema.

De especial interés es la conexión externa del dispositivo. Recomendando el fabricante la mostrada en la siguiente imagen.

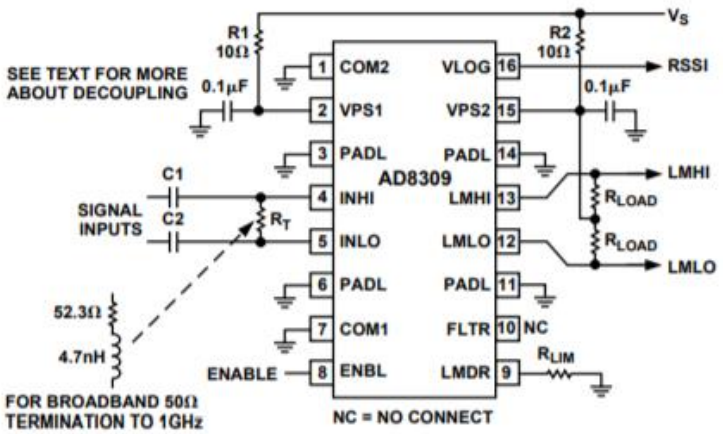


Ilustración 155. Conexión externa implementada en el amplificador logarítmico.

7. Valoración económica del trabajo.

Este apartado indica los gastos asociados al desarrollo y mantenimiento del trabajo realizado.

Realizar una estimación del proyecto en cuanto al coste de materiales es muy compleja, por lo que, en esta primera estimación, se decide tener solo en cuenta el precio del desarrollo en cuanto al coste de personal (recursos humanos).

Personal	***Precio/hora.***	***Nº Horas***	***Coste Total.***
Ingeniero gestión técnica/configuración/integración.	*75 €*	*500 horas*	*37.500 €*
2 ingenieros de desarrollo	*40 €*	*1500 horas*	*120.000 €*
2 Becarios de apoyo en desarrollo y pruebas	*20 €*	*1500 horas*	*60.000 €*
COSTE TOTAL	***--------***	***----------***	***217.500 €***

Por otro lado, se realiza en paralelo una valoración económica de los precios de los dispositivos utilizados hasta el momento. Para una mejor visualización se indican la valoración económica redondeada y aproximada en la siguiente tabla.

Dispositivo	***Precio/Unidad.***	***Nº Unidades***	***Coste Total.***
Intel DK-DEV-10M50A MAX 10 FPGA Development Board.	*200 €*	*1*	*200 €*
LPC1768 Mini-DK2 Development Board.	*55 €*	*2*	*110 €*
Ulink Cable conexión Micro	*35 €*	*1*	*35 €*
USB Blaster Cable conexión FPGA	*228 €*	*1*	*228 €*
Sensores externos y conexiones	*5 €*	*1*	*5 €*
COSTE TOTAL	***--------***	***----------***	***578 €***

No se han introducido los costes asociados a la producción de la PCB ya que se encuentra ubicado en las líneas futuras del proyecto.

8. Conclusión personal.

Este apartado busca resumir la sensación, opinión y conclusión que me ha transmitido haber realizado este proyecto.

Lo mejor será empezar por el principio, el motivo de haber elegido este proyecto fue el reto personal de poder diseñar un equipo real desde cero. Desarrollar un proyecto en el que poder identificar problemas y tener la habilidad personal de poder aplicar un diagnóstico al problema identificado y poder solucionarlo con una idea propia en una continua improvisación y no disponer de una ruta previamente marcada por una tercera persona como un tutor.

La elección de este proyecto en concreto es que considero que implementa gran parte de los conocimientos adquiridos en la carrera y en el master, entre estos conocimientos se pueden destacar los siguientes:

- ***Desarrollo Firmware.*** *Para esta parte, destacaría el proyecto desarrollado en la FPGA, desde la elección de compra, pasando por la comunicación y terminando en el propio desarrollo, el cual no considero que haya sido poco.*
- ***Diseño Hardware, desarrollo RF.*** *Este capítulo, quizás destacaría por la búsqueda realizada por múltiples páginas webs de dispositivos que cumplan las condiciones necesarias del proyecto desarrollado. También destacaría el prototipo de PCB desarrollado para la transmisión RF.*
- ***Encriptación asimétrica RSA.*** *Para la parte de la encriptación, destaco la comprensión y desarrollo en un campo con el que nunca había trabajado previamente. Diría que se trata de un método de encriptación, basado en operaciones matemáticas de elevada complejidad.*
- ***Desarrollo Software.*** *El desarrollo software trabaja con un microcontrolador, fusionando la parte firmware con la parte software. Se puede considerar tan importante la idea que soluciona el problema relacionado con el interfaz de visualización como el propio desarrollo.*
- ***Comunicación entre dispositivos y con servidor HTTP.*** *Para concluir el proyecto aparece la necesidad de conectar los dos microcontroladores con la FPGA. Con el objetivo de conseguir la mejor visualización posible, se conecta un microcontrolador por ethernet a un servidor aumentando la complejidad.*

De cara al trabajo durante estos cuatro meses de proyecto, destacar las horas muertas buscando información en internet, ya que como me ha enseñado mi vida universitaria, no hay nada en este mundo que no sepa GOOGLE. Por cada hora de desarrollo e implementación, había diez previas buscando información de cómo desarrollarlo en internet. Destacaría de la formación que he adquirido durante estos años, la capacidad que considero haber obtenido para poder enfrentarme a cualquier problema, dividirlo en partes y atacar cada una de ellas por separado. Mirando hacia atrás, en estos años, destacaría que una ingeniería te forma para eso, para poder enfrentarte a cualquier proyecto, todas las noches sin dormir, estudiando y trabajando, hoy puedo decir que tienen un significado.

La parte referente a la formación autodidacta del proyecto podría destacar toda la parte referente a la encriptación. Nunca jamás en mi vida había profundizado en este campo, mi carrera es una rama de la telecomunicación, pero con un alto nivel de especialización en la electrónica, motivo por el cual ha sido necesario formarme desde cero en esta parte.

Tengo que admitir que, a día de hoy, con lo que he aprendido en estos meses respecto al algoritmo RSA, me ha parecido un campo muy interesante. Los demás apartados desarrollados, han necesitado también de formación personal, pero tengo que admitir que en todos ellos llevaba una base previa.

En cuanto a la satisfacción personal tanto del master como del propio proyecto en sí mismo me llevo un muy buen sabor de boca. Soy consciente de que siempre hay que seguir formándose en este mundo y que no puedes parar porque enseguida te quedas atrás, nunca digas nunca, pero a día de hoy si considero que con este proyecto ponga punto y final en el capítulo referente a la formación en el libro de mi vida.

Terminado el proyecto, es importante echar la vista atrás y comprobar los logros de los objetivos planteados inicialmente. Como se ha comentado al principio, ha sido un proyecto de desarrollo personal, por lo que en gran medida considero que he ido improvisando las soluciones a los problemas que he ido identificando, haciendo autocritica es cierto que he podido sacrificar bastante desarrollo referente al diseño hardware de la PCB, pero a cambio considero una mejora descomunal en la parte referente al interfaz de visualización.

Por último, referente a las líneas futuras de trabajo las cuales no se han podido explotar y que han quedado pendientes destacaría el desarrollo de un prototipo hardware para una transmisión vía RF. Por otro lado, siendo honesto, debido a la falta de instrumentación considero muy difícil, por no decir prácticamente imposible, conseguir que la tarjeta funcionase.

9. Glosario.

Definición de los términos y acrónimos utilizados en la Memoria.

1090 ES: 1090 MHz Extended Squitter
ADS: Automatic Dependent Surveillance
ADS-B: Automatic Dependent Surveillance-Broadcast
EUROCAE: European Organization for Civil Aviation Equipment
FAA: Federal Aviation Administration
FL290: 29.000 pies de altitud.
FL285: 28.500 pies de altitud
FW: Firmware
IFF: Identification Friend or foe
IFR: Instrumental flight rules
KTAS: Knots true airspeed (Velocidad real)
RF: Radiofrecuencia
PCB: Printed Circuit Board
ISM: Industrial, Scientific & Medical
SIF: Selective Identification Feature
TXP: Transponder.
FPGA: field programmable gate array
VHSIC: (Very High Speed Integrated Circuit)
HDL: Hardware Description Language.
VHDL: VHSIC (Very High Speed Integrated Circuit) + HDL (Hardware Description Language).
HW: Hardware
SW: Software
RSA: Rivest, Shamir y Adleman
TX: Equipo transmisor
RX: Equipo receptor
FSM: Finite state machine (Maquina de estados finita)

10. Bibliografía.

En primer lugar, de cara a la idea global desarrollada, los protocolos y especificaciones de la comunicación IFF se extrae de la Documentación interna de INDRA SISTEMAS SA.

En segundo lugar, referente a la planificación y gestión del proyecto. En concreto los hitos y los planes de tiempo y riesgos. Se extrae la idea y la estructura de las asignaturas de dirección y gestión de proyectos cursadas en el master.

En tercer lugar, referente al estado del arte. Se realiza una exhaustiva búsqueda por internet para contrastar la información encontrada y poder tener una idea mucho más técnica de los campos desarrollados y los dispositivos utilizados. Entre las páginas más estudiadas y con más peso en el proyecto, se pueden listas las siguientes.

https://es.wikipedia.org/wiki/Sistema_de_Vigilancia_Dependiente_Autom%C3%A1tica

https://www.nytimes.com/2009/11/17/science/17air.html?ref=science&pagewanted=all

http://www.universalweather.com/blog/ads-b-for-2019-and-beyond/

https://es.wikipedia.org/wiki/Sistema_de_Vigilancia_Dependiente_Automátic

https://es.wikipedia.org/wiki/Field-programmable_gate_array

https://repositorio.espe.edu.ec/bitstream/21000/576/1/T-ESPE-019557.pdf

http://oa.upm.es/49976/1/PFC_ALICIA_PARRA_RUIZ.pdf

https://actualidadaeroespacial.com/18942-2/

https://es.wikipedia.org/wiki/Field-programmable_gate_array

https://planetachatbot.com/qu%C3%A9-es-una-fpga-y-por-qu%C3%A9-jugar%C3%A1n-un-papel-clave-en-el-futuro-e76667dbce3e

https://www.wired.com/2016/09/microsoft-bets-future-chip-reprogram-fly/

https://www.ionos.es/digitalguide/servidores/seguridad/todo-sobre-los-metodos-de-encriptado/

https://nextvision.com/todo-sobre-encriptacion-de-datos-para-empresas/#:~:text=La%20encriptaci%C3%B3n%20(traducci%C3%B3n%20de%20la,descifrar%20el%20c%C3%B3digo%20pueda%20leerla.

https://es.wikipedia.org/wiki/RSA

http://informatica.uv.es/iiguia/MC/Teoria/mc_capitulo12.pdf

https://es.wikipedia.org/wiki/Banda_ISM#Asignaci%C3%B3n_de_frecuencia

https://www.disk91.com/2017/technology/sigfox/all-what-you-need-to-know-about-regulation-on-rf-868mhz-for-lpwan/

https://radioslibres.net/10-como-se-crean-las-ondas-electromagneticas/

https://es.wikipedia.org/wiki/Modulaci%C3%B3n_por_desplazamiento_de_amplitud

https://definicion.de/radiofrecuencia/

https://d3.xlrs.eu/bandaism/#:~:text=Europa%3A%20banda%20868Mhz.,y%20posiblemente%20bajar%20la%20latencia.

http://scielo.sld.cu/pdf/eac/v33n3/eac06312.pdf

http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1815-59282012000300006

https://es.wikipedia.org/wiki/Modulaci%C3%B3n_por_posici%C3%B3n_de_pulso

https://medium.com/modulaciones-de-pulsos-muestreo-pam-ppm-pcm-y/modulaciones-de-pulsos-ppm-558c91689e7

http://www.itq.edu.mx/carreras/IngElectronica/archivos_contenido/Apuntes%20de%20materias/CDF1202_Comm_Digitales/5_PAM-PPM-PWM.pdf

https://hardzone.es/2018/03/17/pcb-importancia-dispositivos-electronicos/

https://es.wikipedia.org/wiki/Circuito_impreso

https://www.intel.com/content/www/us/en/processors/packaging-chapter-07-databook.html

https://www.youtube.com/watch?v=5Xd92df738I

https://www.youtube.com/watch?v=LFU3NY81PV4

https://www.youtube.com/watch?v=o78RxDOnAQg

En cuarto lugar, se estudia de forma individual las referencias de los dispositivos con mayor peso en el proyecto

Se adjunta la página web de compra y el manual de usuario del kit de desarrollo de la FPGA.

https://www.mouser.es/new/intel/intel-dk-dev-10m50a-board/

https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug-max10m50-fpga-dev-kit.pdf

Se adjunta la página web de compra y el manual de usuario microcontrolador.

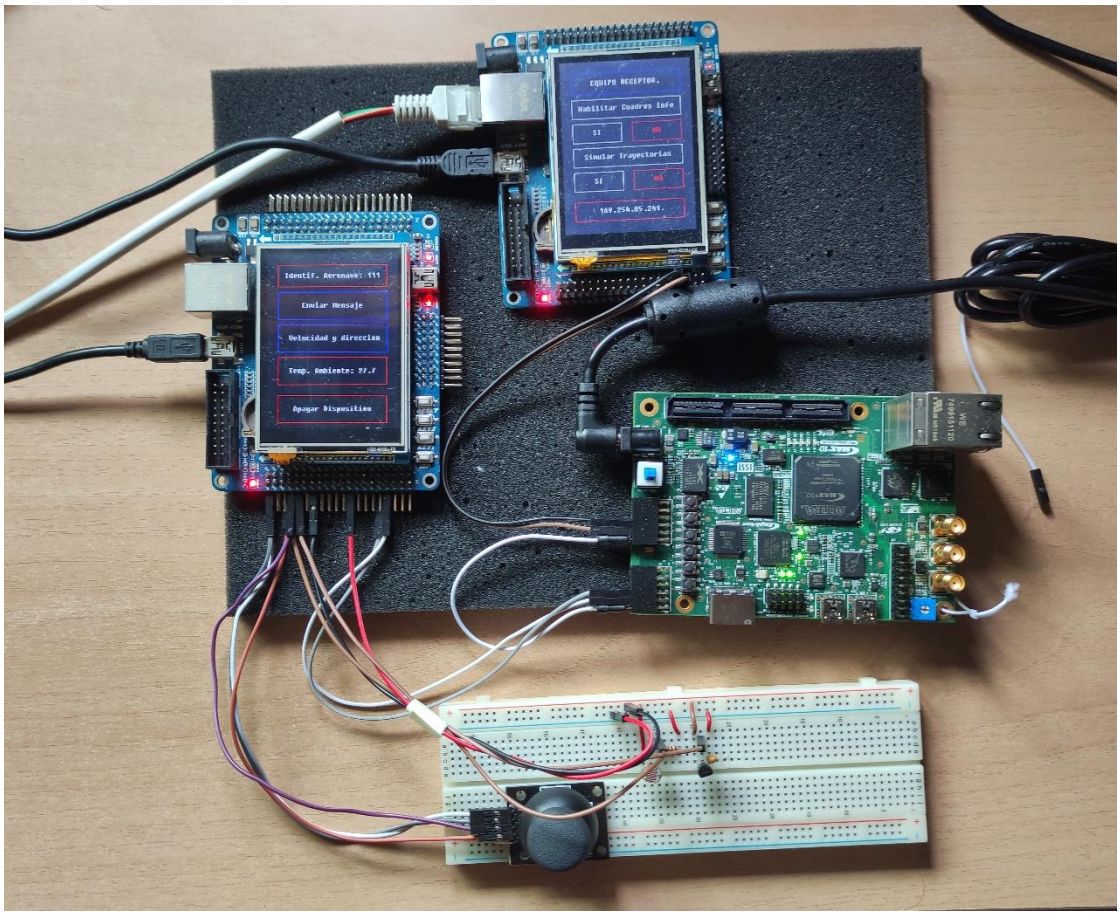
https://www.hotmcu.com/lpc1768minidk2-development-board-p-55.html

https://www.amazon.com/-/es/LPC1768-2-8/dp/B00GMF91ZM

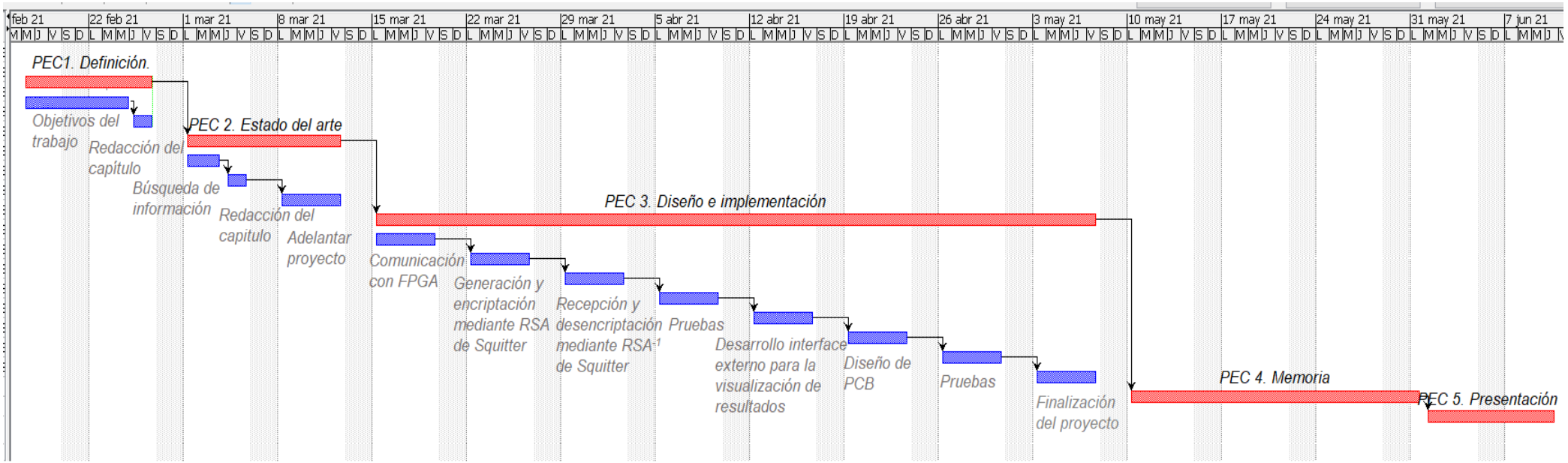
https://www.nxp.com/docs/en/user-guide/UM10360.pdf

11. Anexos.

ANEXO I. FOTOGRAFIA DEL PROYECTO.



ANEXO II. DIAGRAMA DE GANTT COMPLETO.



ANEXO III. EJEMPLO ILUSTRATIVO

Comprobación de si los números 721 y 448 son Coprimos.

$$721 = 448 \cdot 1 + 273$$
$$448 = 273 \cdot 1 + 175$$
$$273 = 175 \cdot 1 + 98$$
$$175 = 98 \cdot 1 + 77$$
$$98 = 77 \cdot 1 + 21$$
$$77 = 21 \cdot 3 + 14$$
$$21 = 14 \cdot 1 + 7$$
$$14 = 7 \cdot 2 + 0$$

Números NO Coprimos.
Ultimo resto no nulo con valor distinto a la unidad.

Comprobación de si los números 640 y 231 son Coprimos.

$$640 = 231 \cdot 2 + 178$$
$$231 = 178 \cdot 1 + 53$$
$$178 = 53 \cdot 3 + 19$$
$$53 = 19 \cdot 2 + 15$$
$$19 = 15 \cdot 1 + 4$$
$$15 = 4 \cdot 3 + 3$$
$$4 = 3 \cdot 1 + 1$$
$$3 = 1 \cdot 3 + 0$$

Números Coprimos.
Ultimo resto no nulo con valor unidad.

Obtención de la identidad de Bezout a partir del algoritmo extendido de Euclides de los números 640 y 231.

$$af + de = mcd(e; f) = 1 \qquad a \cdot 640 + d \cdot 231 = mcd(e; f) = 1$$

$$1 = 4 - 1 \cdot 3$$
$$1 = 4 - 1 \cdot (15 - 4 \cdot 3) = 4 \cdot 4 - 15$$
$$1 = 4 \cdot (19 - 15) - 15 = 4 \cdot 19 - 5 \cdot 15$$
$$1 = 4 \cdot 19 - 5 \cdot (53 - 19 \cdot 2) = 14 \cdot 19 - 5 \cdot 53$$
$$1 = 14 \cdot (178 - 53 \cdot 3) - 5 \cdot 53 = 14 \cdot 178 - 47 \cdot 53$$
$$1 = 14 \cdot 178 - 47 \cdot (231 - 178) = 61 \cdot 178 - 47 \cdot 231$$
$$1 = 61 \cdot (640 - 231 \cdot 2) - 47 \cdot 231 = 61 \cdot 640 - 169 \cdot 231$$

$$1 = 61 \cdot 640 - 169 \cdot 231 \rightarrow d = -169 = 640 - 169 = 471$$

ANEXO IV. EJEMPLO ILUSTRATIVO

Calcular mediante aritmética binaria la siguiente operación:

$$\mathbf{271^{321}\ mod(481)}$$

1. *Obtencion del exponente en binario*

$$321\ decimal = 1010\ 0001\ binario \quad contador\ maximo = 8$$

2. *Calculo de los cuadrados repetitivos*

$$271^{2^0}\ mod(481) = 271^1\ mod(481) = 271$$
$$271^{2^1}\ mod(481) = 271^2\ mod(481) = 329$$
$$271^{2^2}\ mod(481) = 329^2\ mod(481) = 16$$
$$271^{2^3}\ mod(481) = 16^2\ mod(481) = 256$$
$$271^{2^4}\ mod(481) = 256^2\ mod(481) = 120$$
$$271^{2^5}\ mod(481) = 120^2\ mod(481) = 451$$
$$271^{2^6}\ mod(481) = 451^2\ mod(481) = 419$$
$$271^{2^7}\ mod(481) = 419^2\ mod(481) = 477$$
$$271^{2^8}\ mod(481) = 477^2\ mod(481) = 16$$

3. *Obtencion de la solucion mediante aritmetica binaria*

3.1. *Solución estandar*

$$271^{321}\ mod(481) = 271^{2^0+2^6+2^8}\ mod(481) = 271^{2^0} \cdot 271^{2^6} \cdot 271^{2^8} mod(481)$$
$$271^{321}\ mod(481) = 271 \cdot 419 \cdot 16\ mod(481) = 1.816.784\ mod(481) = 47$$

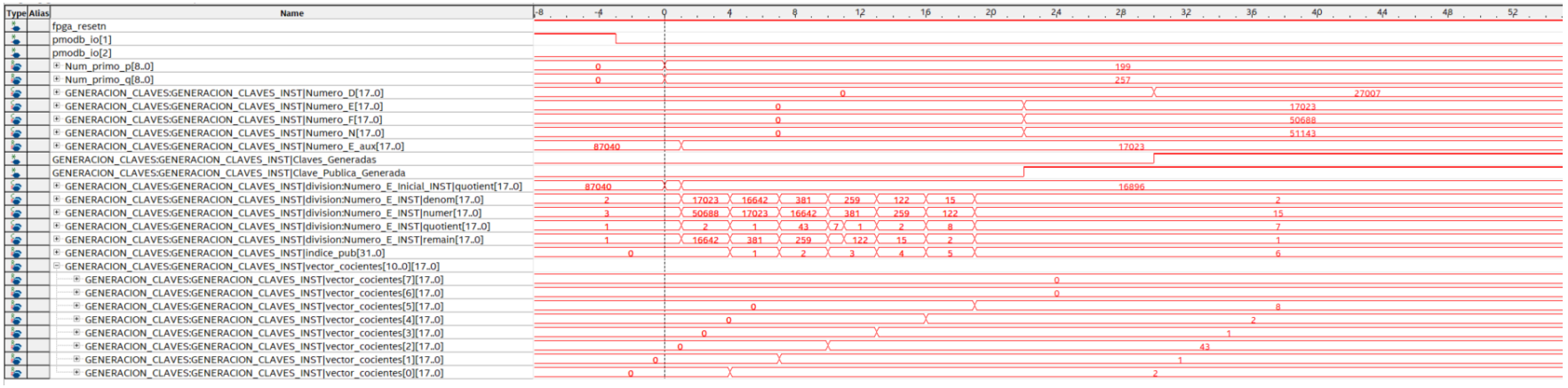
3.2. *Solución mejorada*

$$271^{321}\ mod(481) = 271^{2^0+2^6+2^8}\ mod(481) = 271^{2^0} \cdot 271^{2^6} \cdot 271^{2^8} mod(481)$$
$$271^{321}\ mod(481) = [271 \cdot 419\ mod(481)] \cdot 16\ mod(481)$$
$$271^{321}\ mod(481) = 33 \cdot 16\ mod(481) = 47$$

ANEXO V. CAPTURAS DE PANTALLA DEL ENTORNO DE DESARROLLO FIRMWARE.

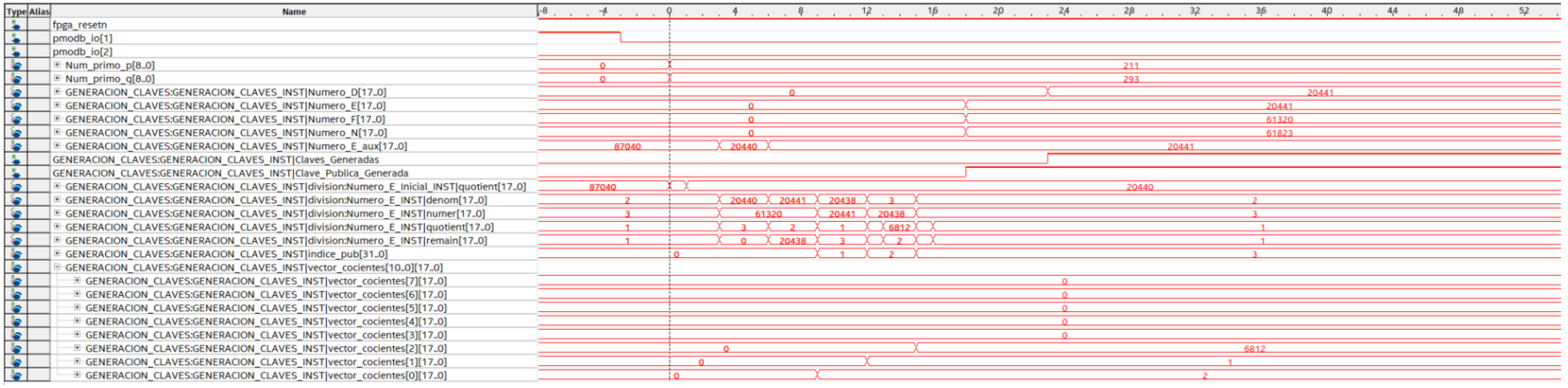
Capturas de pantallas del proceso de generación de claves.

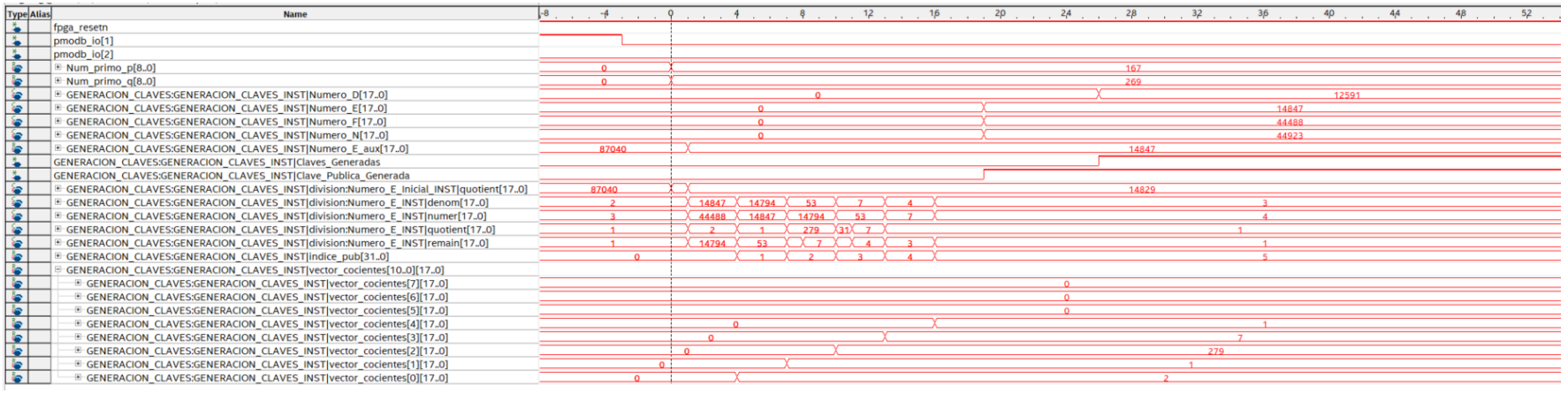
Capturas de pantallas del proceso de generación de claves públicas.



Capturas de pantallas del proceso de generación de claves.

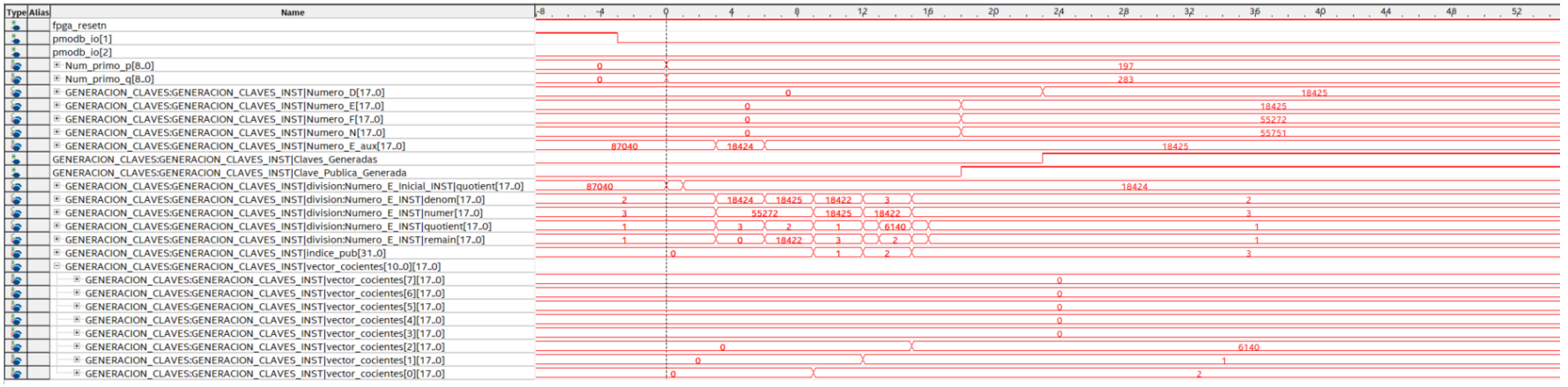
Capturas de pantallas del proceso de generación de claves públicas. Continuación (1)





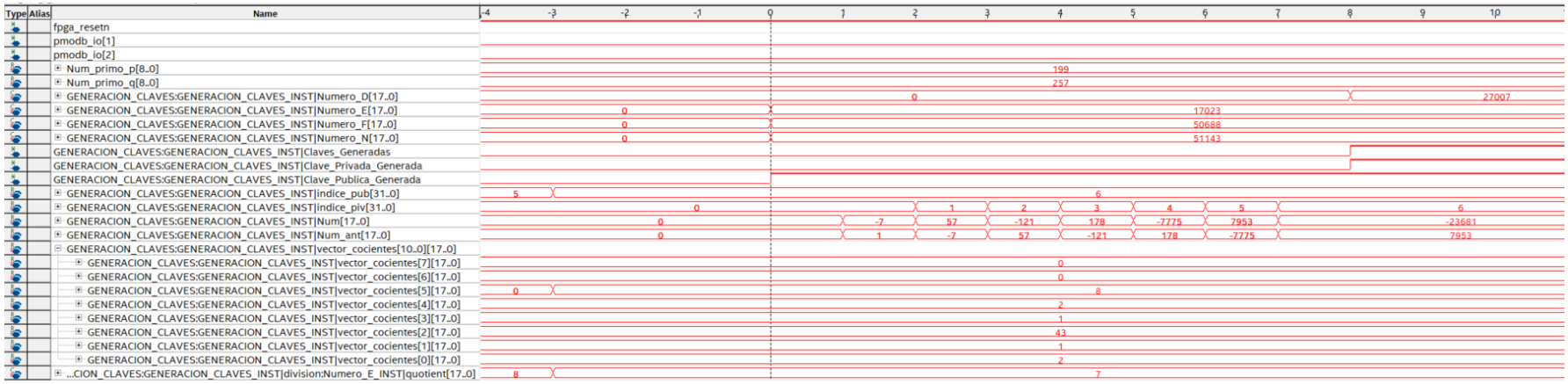
Capturas de pantallas del proceso de generación de claves.

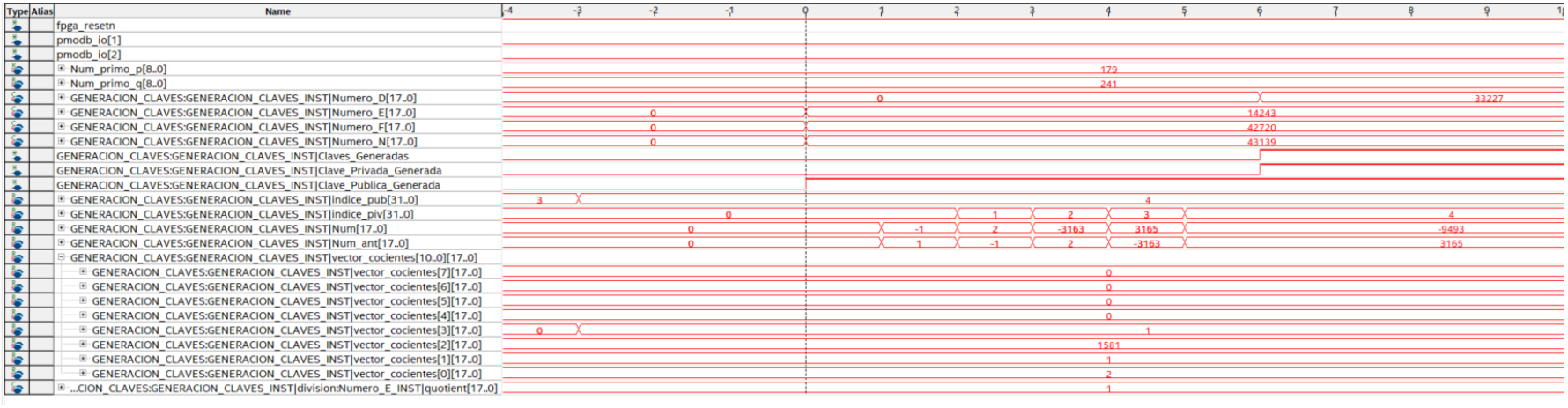
Capturas de pantallas del proceso de generación de claves públicas. Continuación (2)



Capturas de pantallas del proceso de generación de claves.

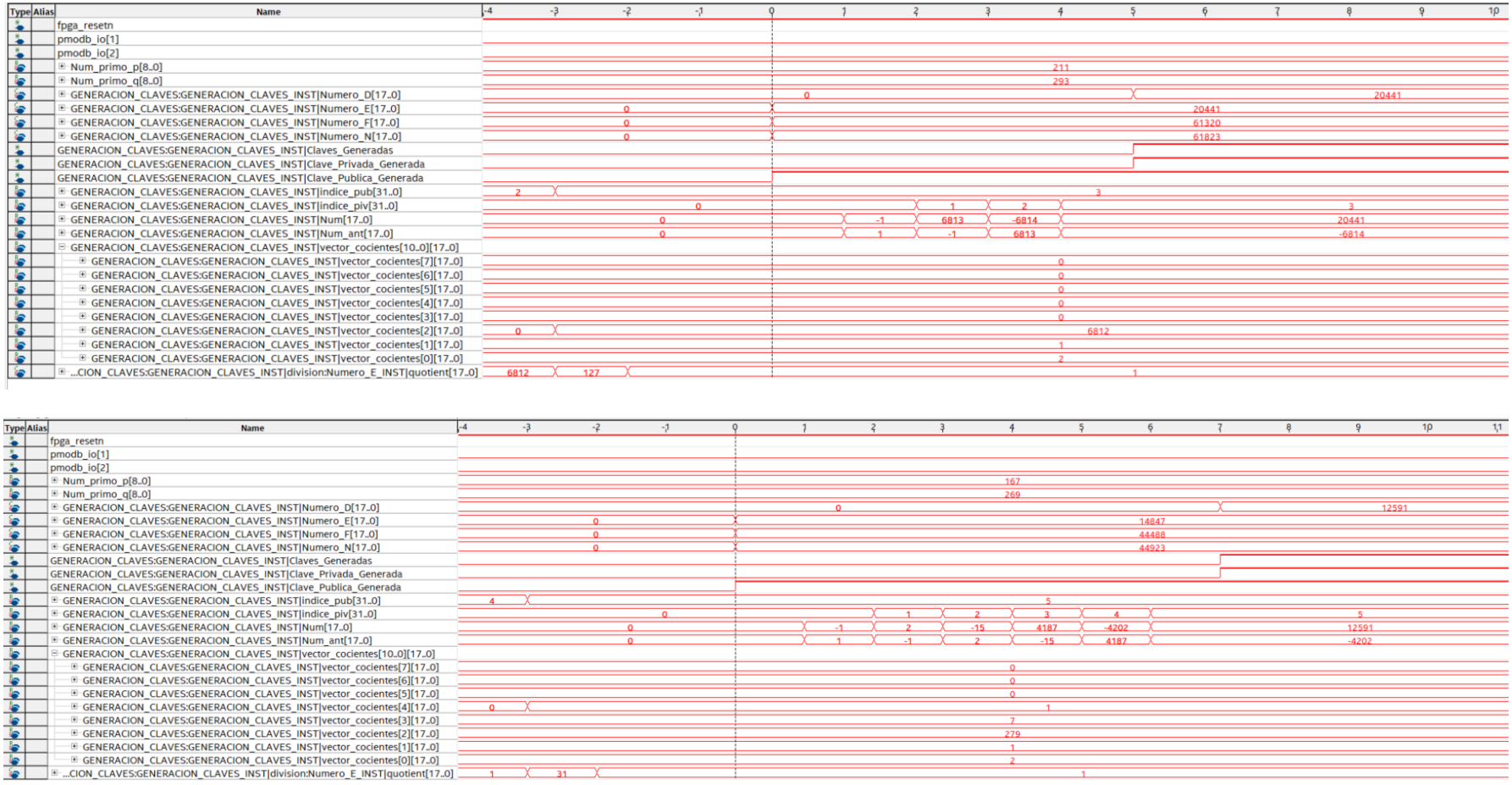
Capturas de pantallas del proceso de generación de claves privadas.





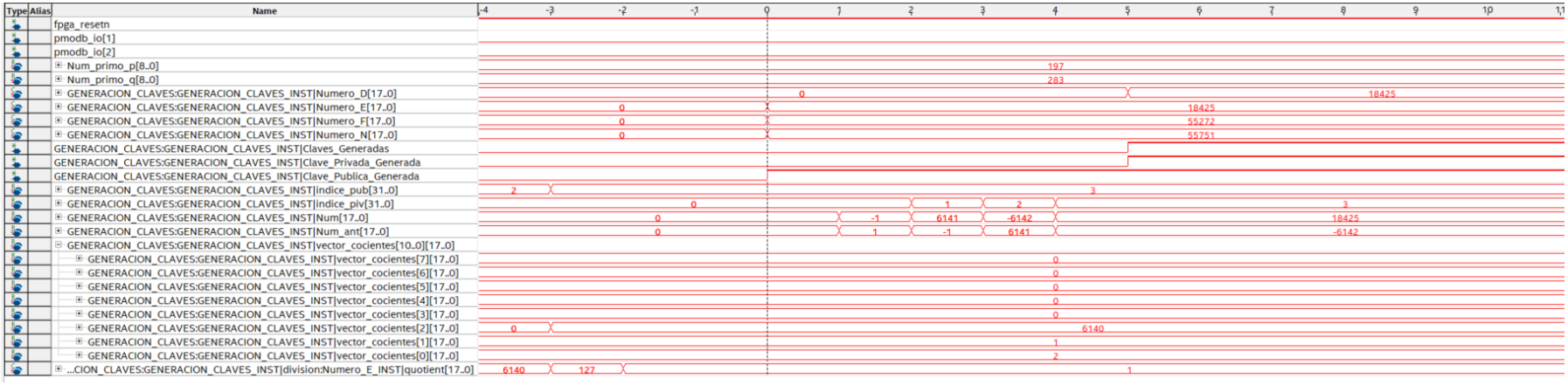
Capturas de pantallas del proceso de generación de claves.

Capturas de pantallas del proceso de generación de claves privadas. Continuación (1)

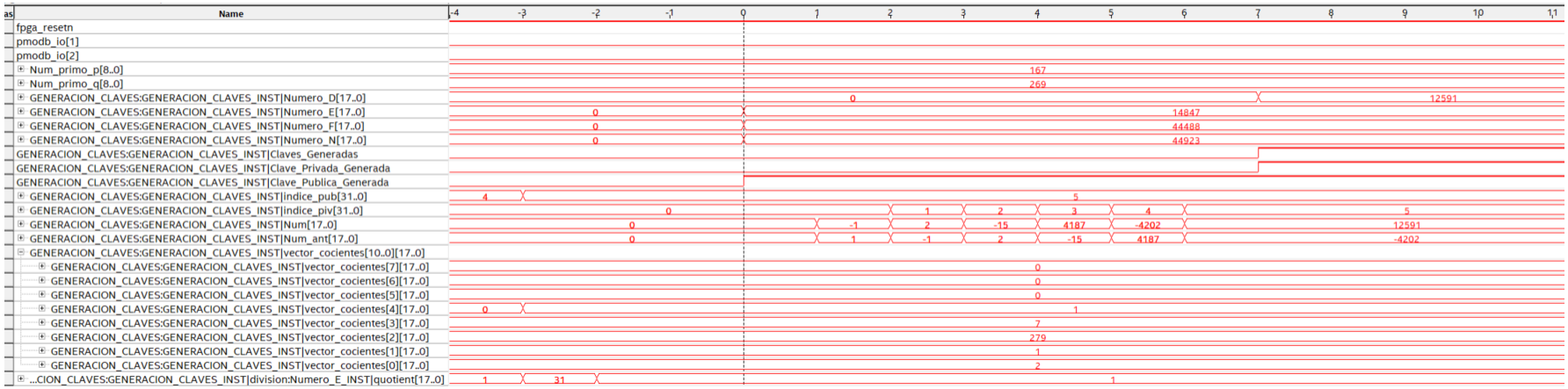


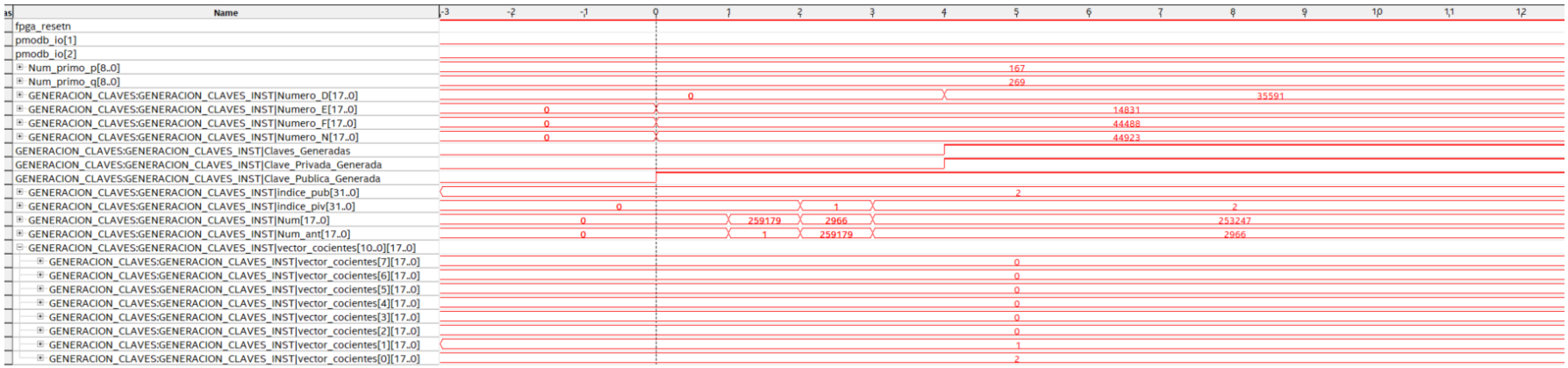
Capturas de pantallas del proceso de generación de claves.

Capturas de pantallas del proceso de generación de claves privadas. Continuación (2)



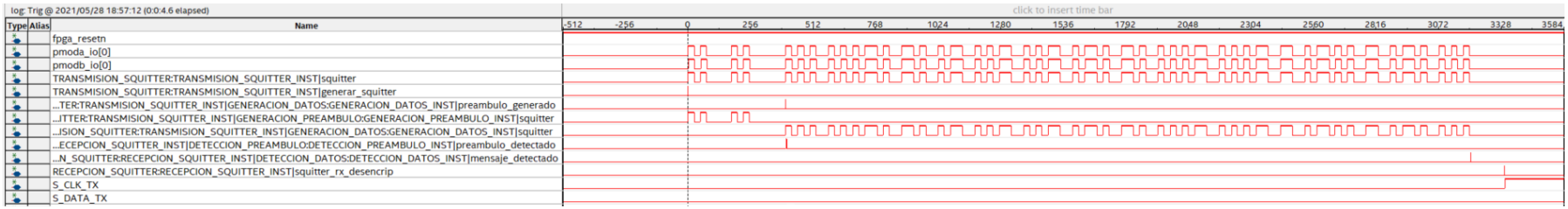
Potencia del algoritmo desarrollado: Mismos números primos elegidos por el usuario, generan distintas claves de cifrado RSA.

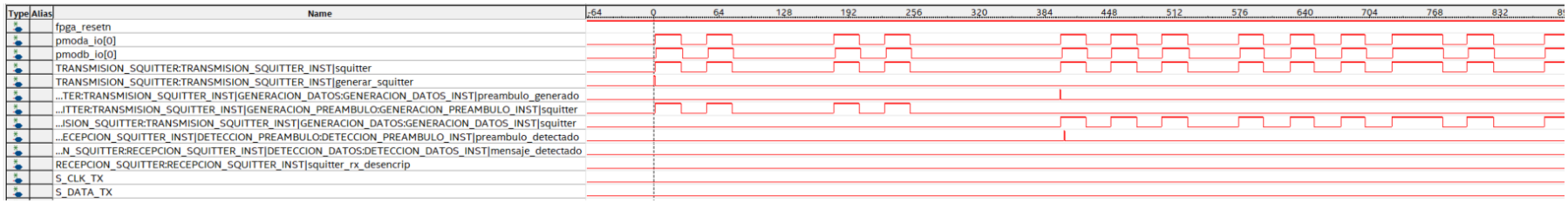


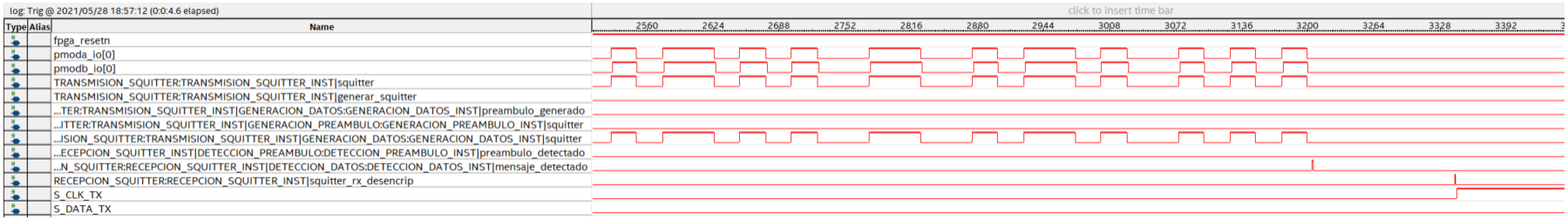


ANEXO V. CAPTURAS DE PANTALLA DEL ENTORNO DE DESARROLLO FIRMWARE.

Captura de pantalla de la transmisión y recepción de squitter

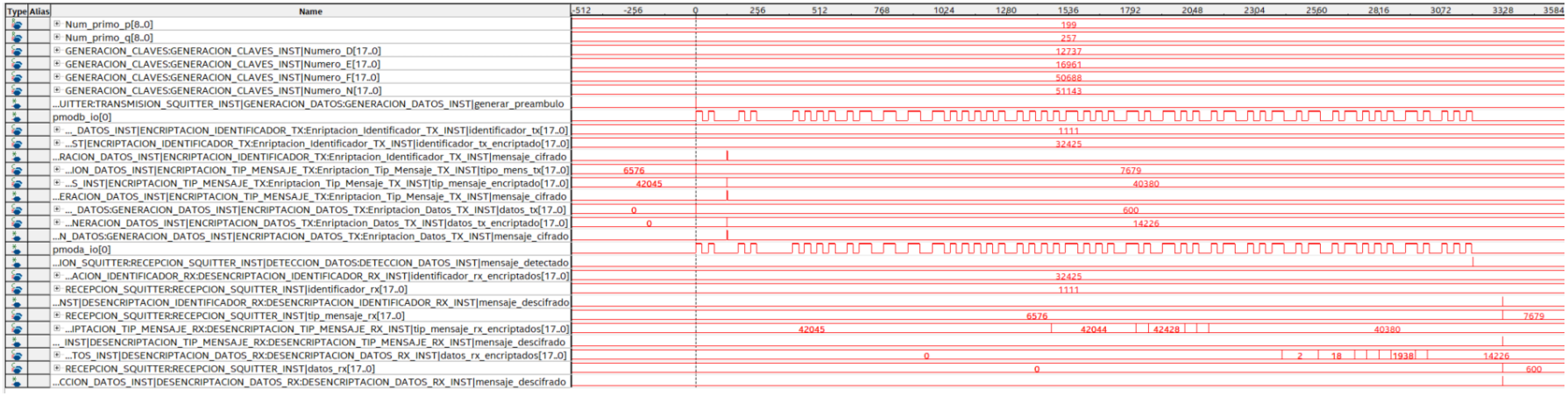


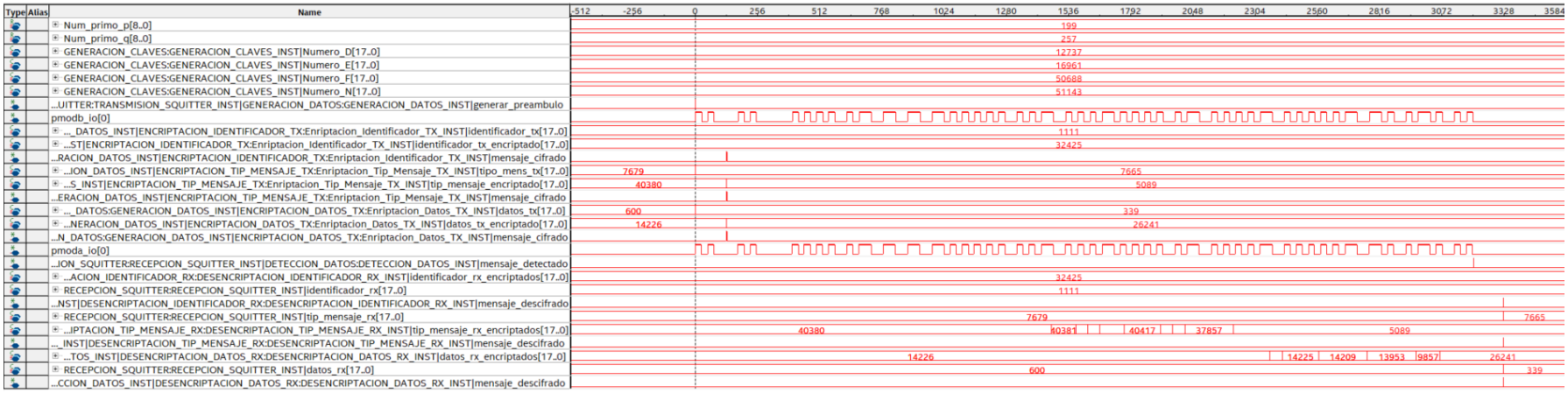




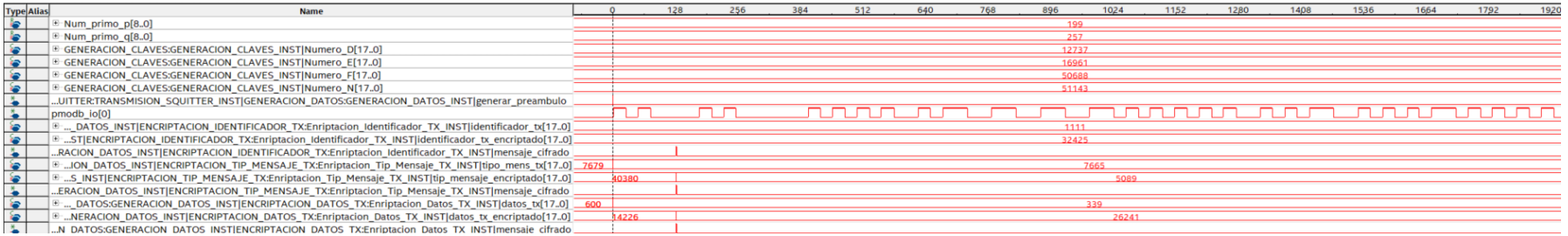
ANEXO V. CAPTURAS DE PANTALLA DEL ENTORNO DE DESARROLLO FIRMWARE.

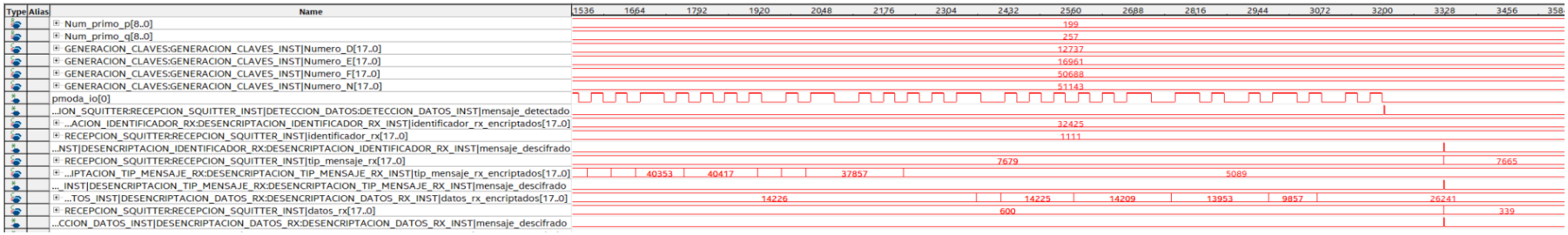
Captura de pantalla de los procesos de encriptación y desencriptación

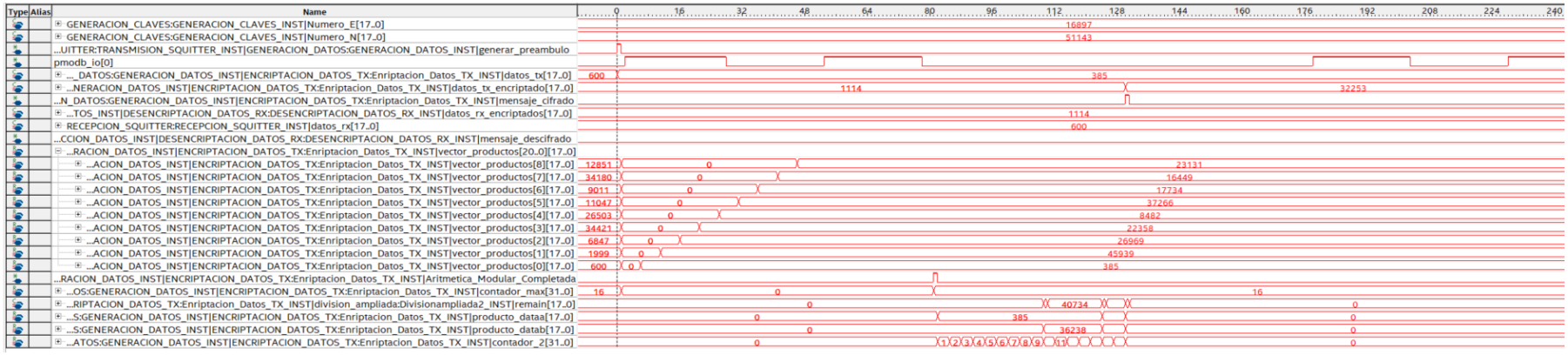




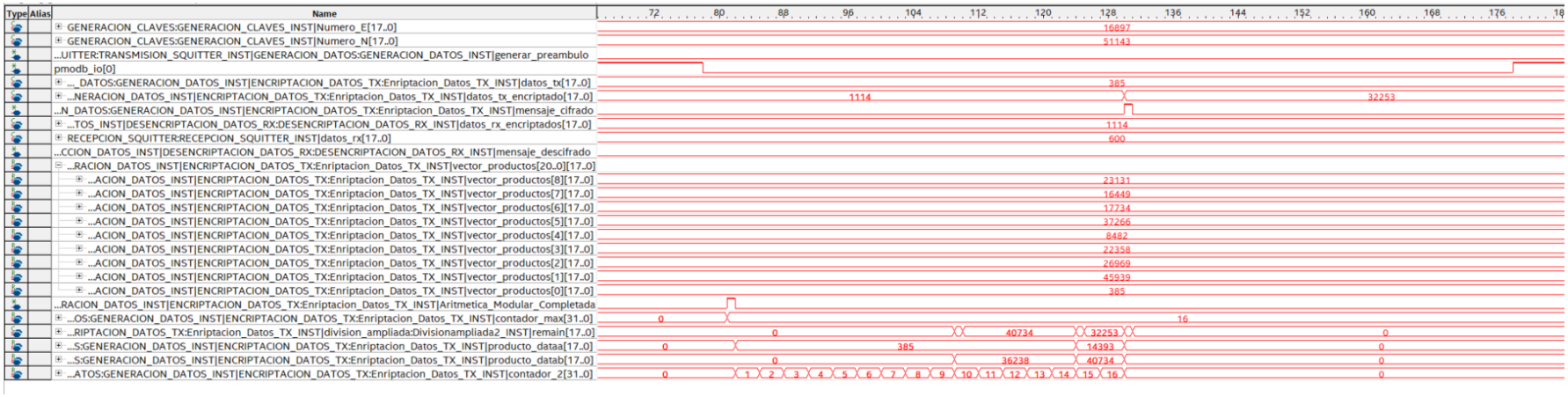
Capturas de pantalla de los procesos de encriptación y desencriptación (1).





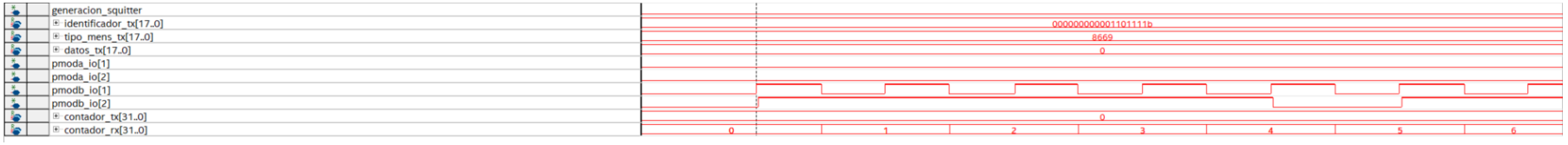


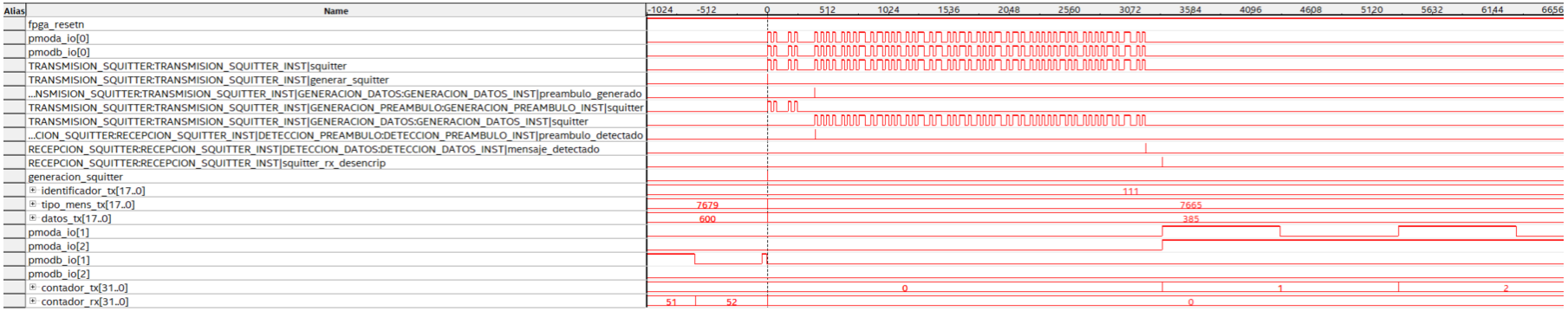
Capturas de pantalla de los procesos de encriptación y desencriptación (2).

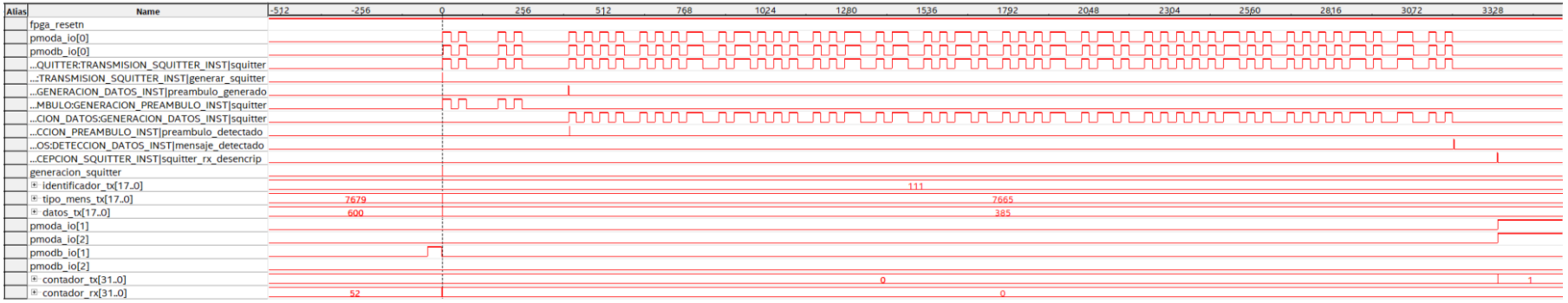


ANEXO V. CAPTURAS DE PANTALLA DEL ENTORNO DE DESARROLLO FIRMWARE.

Captura de pantalla de los procesos de interfaz con el microcontrolador



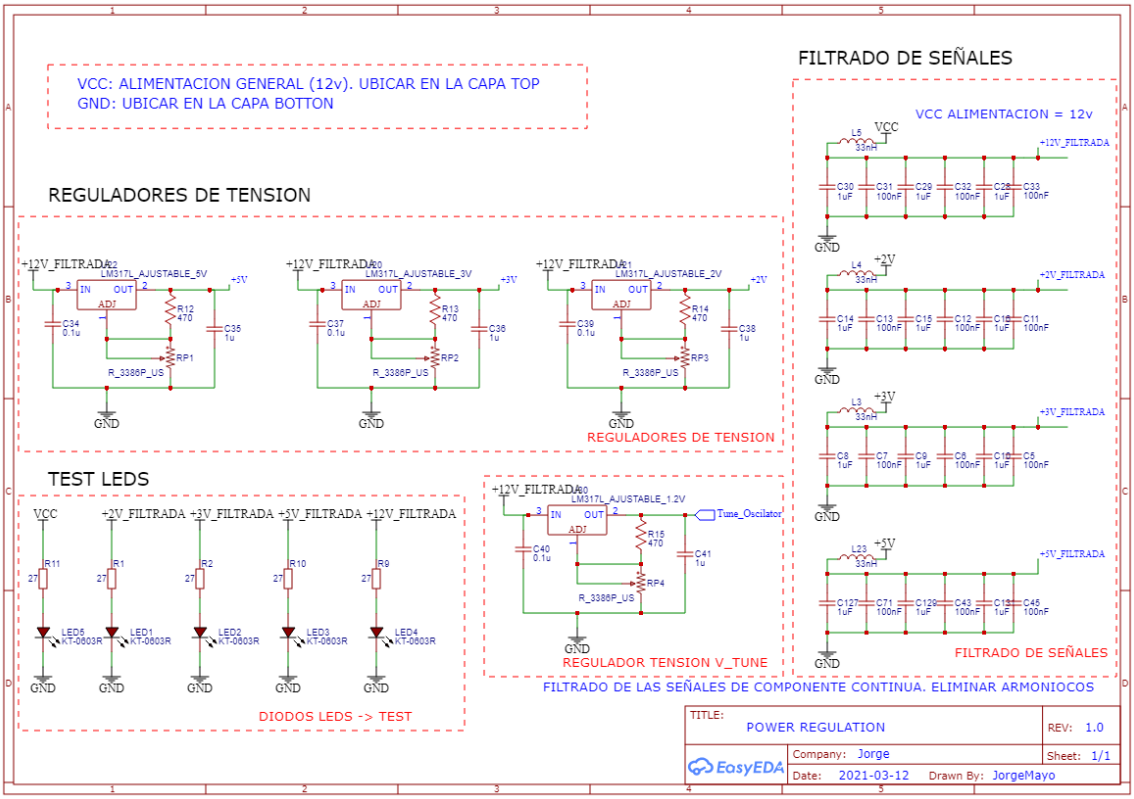




ANEXO VI. CAPTURA DEL DISEÑO HARDWARE DE LA ETAPA DE TRANSMISION.

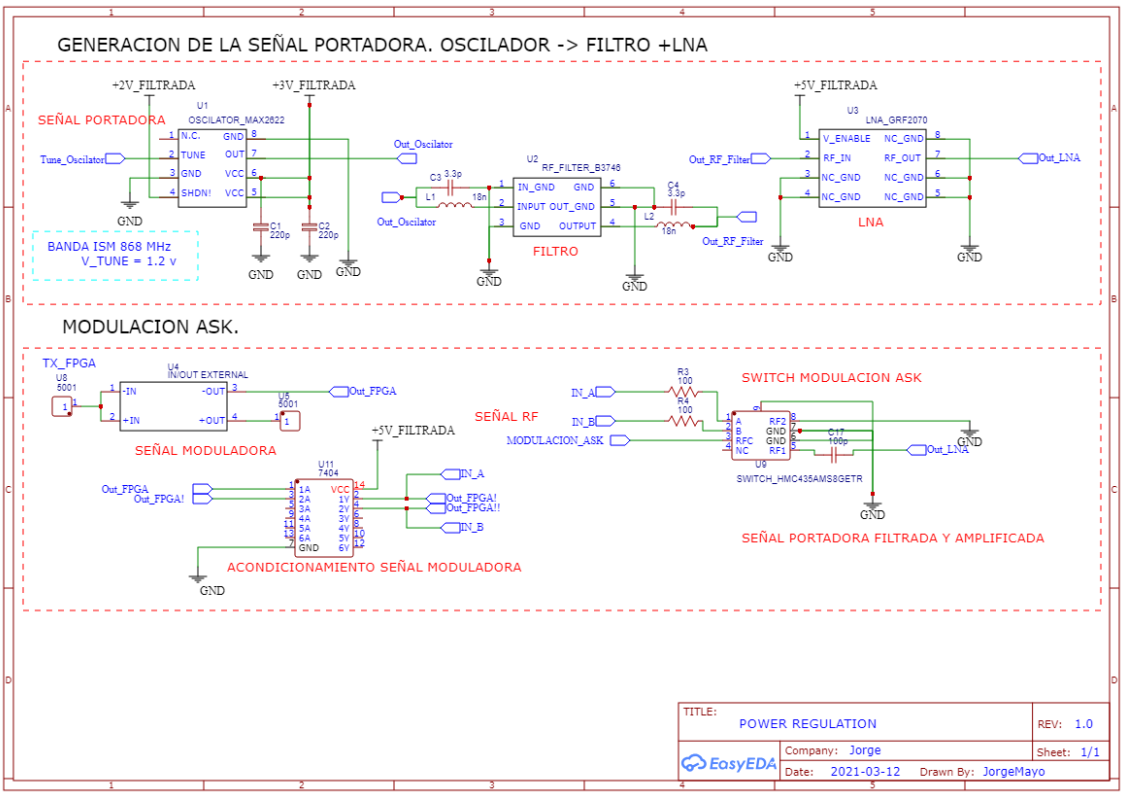
Debido a la falta de instrumentación por parte del alumno, se realizan dos diseños de la PCB de transmisión. En el primero, se introducen conmutadores con el objetivo de cortocircuitar las diferentes etapas de la transmisión, con el objetivo de poder testear el diseño realizado identificando así posibles fallos en las etapas desarrolladas, un posible problema asociado a esta forma de testeo que hay que tener presente, son las posibles capacidades parasitas introducidas en el circuito por los conmutadores añadidos. En el segundo, se diseña la etapa de transmisión sin ningún tipo de conmutador, evitando la posibilidad de testeo y las posibles capacidades parasitas introducidos por estos.

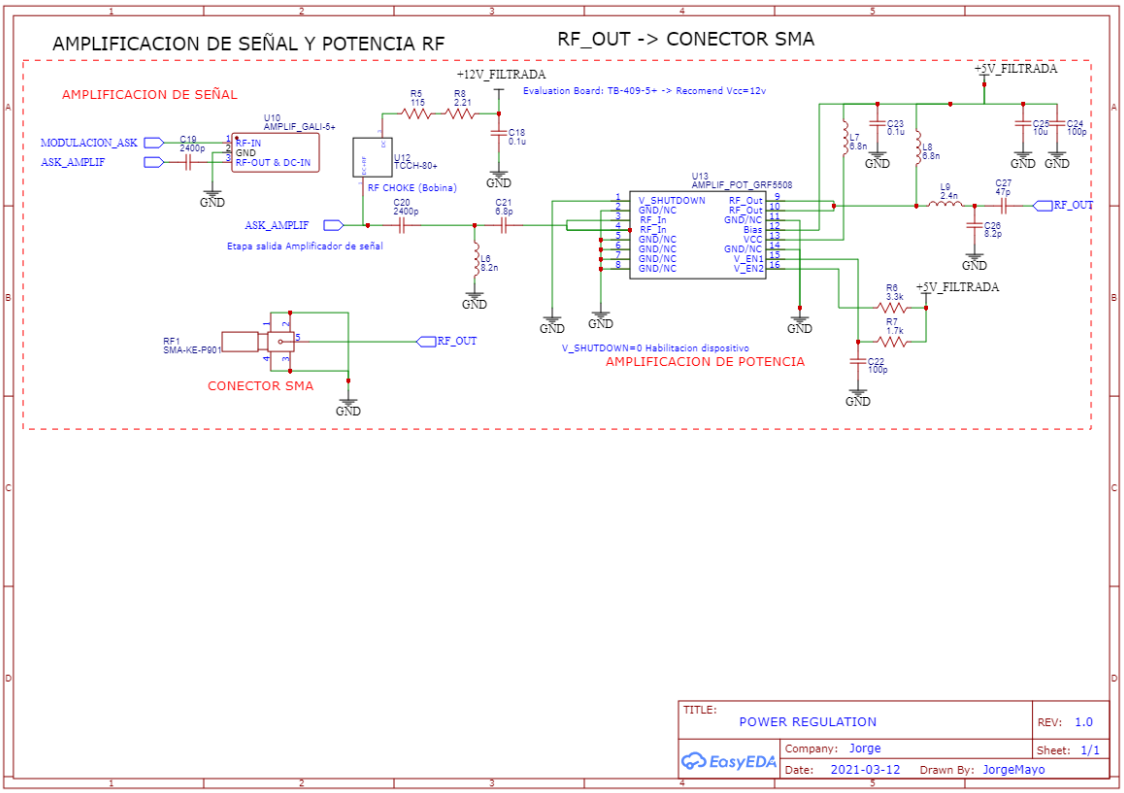
- **Parte común a ambos diseños.** La parte del diseño hardware referente a las regulaciones de tensión es idéntico a ambos diseños, por lo que se ofrece la solución del diseño una única vez.



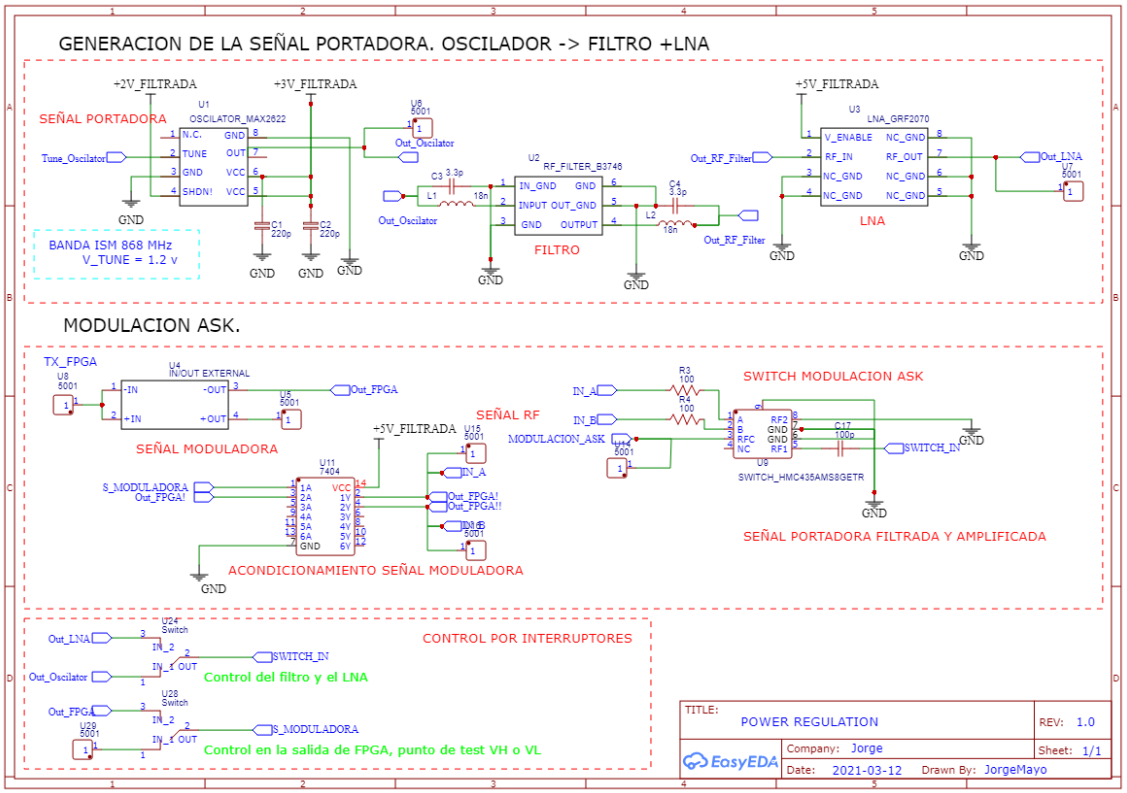
Se pueden apreciar un total de cuatro reguladores de tensión. De manera conjunta a los cuatro reguladores aparece una etapa de filtrado de alimentación para cada una de las tensiones generadas.

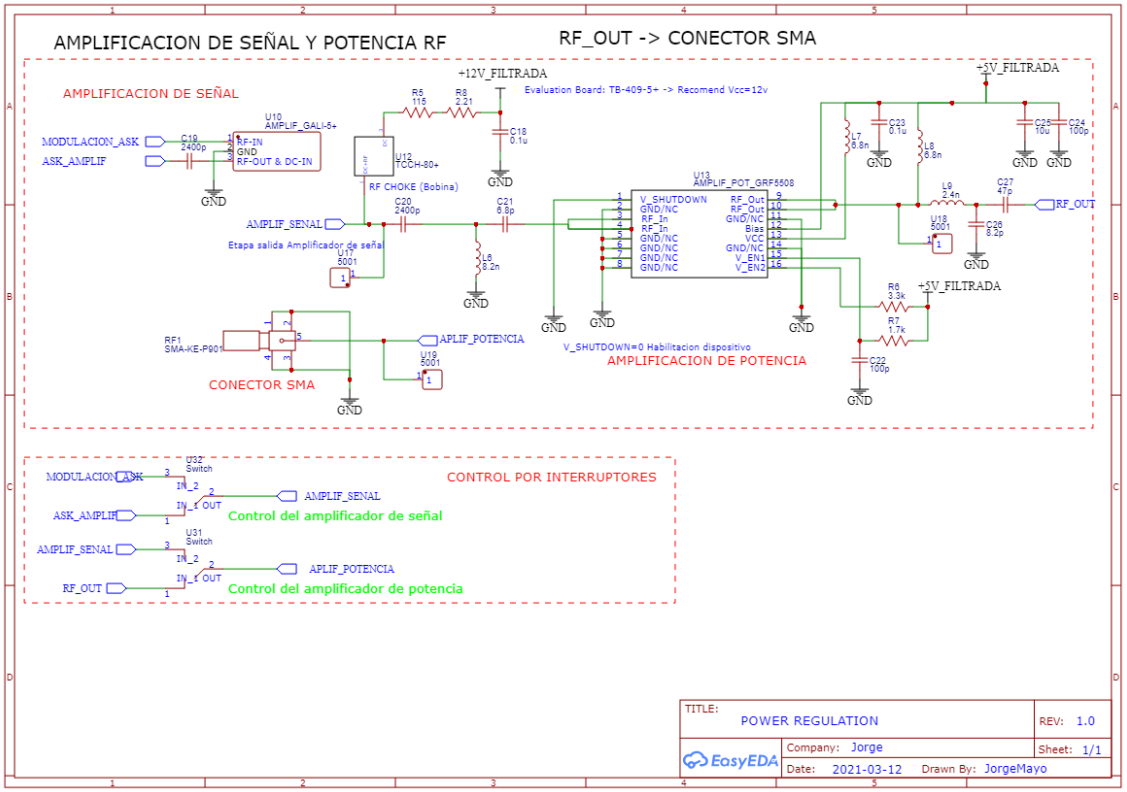
- **Diseño hardware de la PCB sin conmutadores.** Este diseño hardware ofrecería la estructura que se implementaría directamente como solución de diseño. Evitando los efectos de las posibles capacidades parasitas introducidas por los interruptores, pero eliminando también la posibilidad de testear las diferentes etapas diseñadas.





- **Diseño hardware de la PCB con conmutadores.** Para este diseño se introducen cuatro conmutadores con el objetivo de poder cortocircuitar las diferentes etapas diseñadas en la transmisión, para poder testear el diseño realizado identificando así posibles fallos en las etapas desarrolladas. Es importante recordar el problema asociado a esta forma de testeo relacionado con las posibles capacidades parasitas introducidas en el circuito por los conmutadores añadidos

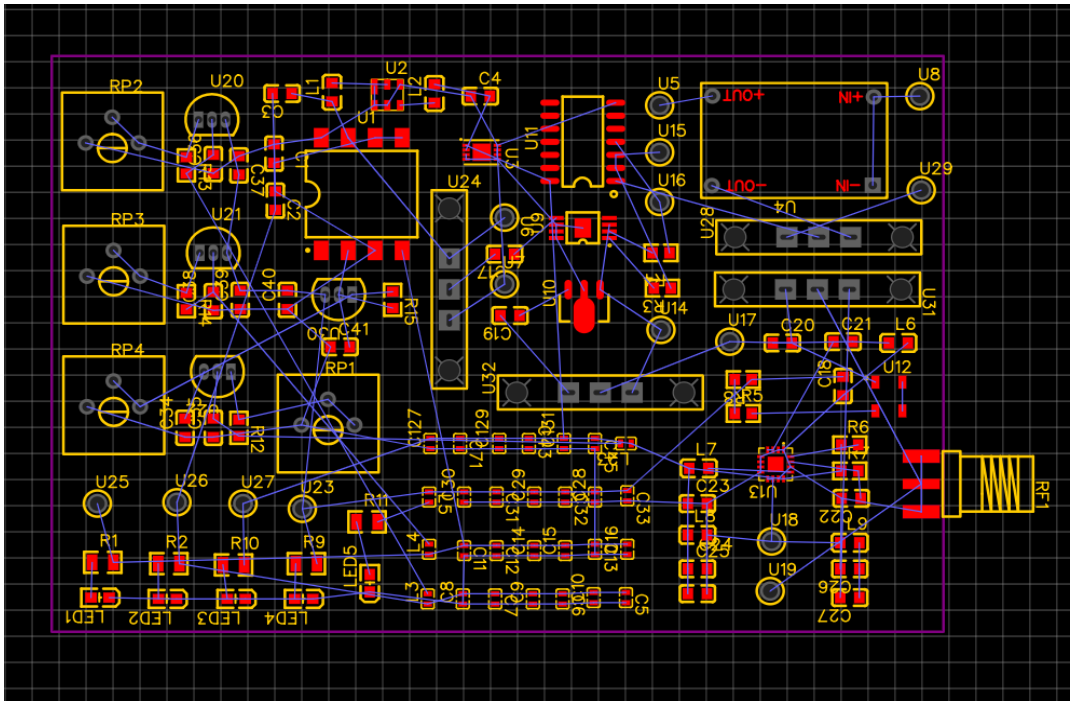




ANEXO VII. CAPTURA DE LA SITUACION ACTUAL DEL DISEÑO DE LA PCB DE TRANSMISION.

Como se ha comentado en el anexo anterior, debido a la falta de instrumentación por parte del alumno, se realizan dos diseños de la PCB de transmisión.

- **Diseño de la PCB con conmutadores.**



- **Diseño de la PCB sin conmutadores.**