UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

****

TRABAJO FIN DE MÁSTER

ViCON - Sistema de Visión Configurable aplicado a Reconocimiento Facial

MÁSTER EN SISTEMAS ELECTRÓNICOS PARA ENTORNOS INTELIGENTES

CHRISTIAN DIEGO COBOS MARCOS

MÁLAGA, 2025

E.T.S. DE INGENIERÍA DE TELECOMUNICACIÓN, UNIVERSIDAD DE MÁLAGA

**ViCON – Sistema de Visión Configurable aplicado a Reconocimiento Facial**

Autor: Christian Diego Cobos Marcos

Tutor: Martin González García

Departamento: Departamento de Tecnología Electrónica

Titulación: Máster en Sistemas Electrónicos para Entornos Inteligentes

Palabras clave: FPGA, VHDL, FT245, MT9V111, Basys3, Visión por Computador, Qt, OpenCV, Procesado de Imágenes en Tiempo Real.

**Resumen**

Este proyecto presenta el diseño e implementación de un sistema de adquisición y transmisión de imágenes en tiempo real, configurable y basado en FPGA. En el apartado hardware, se utiliza una FPGA Xilinx Artyx-7 implementada en la placa Digilent Basys3, junto con un sensor de imagen MT9V111 y una interfaz USB FT245 asíncrona para la comunicación con un PC. La gestión y control del sensor de imagen y de la interfaz USB, así como la lógica de procesamiento en la FPGA, se realizan mediante módulos diseñados en VHDL. La arquitectura resultante coordina la solicitud, captura, sincronización y transmisión de los datos de vídeo en tiempo real hacia el sistema de destino.

En el lado software, se ha desarrollado una aplicación en C++/Qt, que realiza solicitudes de imagen, recibe los fotogramas transmitidos por la FPGA y los procesa mediante OpenCV, habilitando tareas de visualización, detección y reconocimiento facial. El diseño se validó mediante simulación y pruebas en placa, logrando la transmisión y reconstrucción fiable de imágenes en escala de grises a través de la interfaz USB.

Los resultados muestran la viabilidad de la solución como un sistema embebido de bajo coste para aplicaciones de visión por computador en tiempo real, destacando la modularidad de la arquitectura y la posibilidad de extenderla hacia entornos de Edge AI y sistemas inteligentes.

E.T.S. DE INGENIERÍA DE TELECOMUNICACIÓN, UNIVERSIDAD DE MÁLAGA

**ViCON – Configurable Vision System Applied to Face Recognition**

Author: Christian Diego Cobos Marcos

Supervisor: Martin González García

Department: Departamento de Tecnología Electrónica

Degree: Máster en Sistemas Electrónicos para Entornos Inteligentes

Keywords: FPGA, VHDL, FT245, MT9V111, Basys3, Computer Vision, Qt, OpenCV, Real-Time Image Processing.

**Abstract**

This project presents the design and implementation of a configurable, FPGA-based system for real-time image acquisition and image transmission. On the hardware side, an FPGA Xilinx Artix-7 implemented on the Digilent Basys3 board is used, together with an MT9V111 image sensor and an asynchronous FT245 USB interface for communication with a PC. The management and control of both the image sensor and the USB interface, as well as the processing logic within the FPGA, are carried out through modules designed in VHDL. The resulting architecture coordinates the request, capture, synchronisation and real-time transmission of video data to the target system.

On the software side, a C++/Qt application has been developed, which request and image, receives the frames transmitted by the FPGA and processes them using OpenCV, enabling visualisation, detection and facial recognition tasks. The design was validated through simulation and board testing, achieving reliable transmission and reconstruction of grayscale images through the USB interface.

The results show the viability of the solution as a low-cost embedded system for real-time computer vision applications, highlighting the modularity of the architecture and the possibility of extending it to Edge AI environments and intelligent systems.

ÍNDICE

[1. Introducción 8](#_Toc207967899)

[1.1 Objetivos 8](#_Toc207967900)

[1.2 Entorno de usuarios 9](#_Toc207967901)

[1.3 Directivas del proyecto 10](#_Toc207967902)

[1.3.1 Oportunidad de aplicación 10](#_Toc207967903)

[1.3.2 Descripción del problema 11](#_Toc207967904)

[1.3.3 Descripción de la solución 11](#_Toc207967905)

[1.4 Alternativas y competencia 12](#_Toc207967906)

[1.4.1 Alternativas 12](#_Toc207967907)

[1.4.2 Posicionamiento de ViCON 13](#_Toc207967908)

[1.5 Estructura de la memoria 13](#_Toc207967909)

[2. Requisitos y casos de uso 15](#_Toc207967910)

[2.1. Requisitos del proyecto 15](#_Toc207967911)

[2.1.1. Descripción de los Requisitos Funcionales 17](#_Toc207967912)

[2.1.2. Descripción de los Requisitos No Funcionales 19](#_Toc207967913)

[2.2. Casos de uso 20](#_Toc207967914)

[2.2.1. C1 – Visualizar imágenes 22](#_Toc207967915)

[2.2.2. C2 – Detectar rostros 23](#_Toc207967916)

[2.2.3. C3 – Reconocer rostros 23](#_Toc207967917)

[3. Diseño y desarrollo del sistema 25](#_Toc207967918)

[3.1. Arquitectura Física 25](#_Toc207967919)

[3.1.1. Sensor de imagen: MT9V111 25](#_Toc207967920)

[3.1.2. FPGA: Digilent Basys 3 27](#_Toc207967921)

[3.1.3. Interfaz USB: FTDI FT232H-B 29](#_Toc207967922)

[3.1.4. Equipo host: PC. 31](#_Toc207967923)

[3.1.5. Esquema hardware 32](#_Toc207967924)

[3.2. Arquitectura Lógica 36](#_Toc207967925)

[3.2.1. Desarrollo hardware en FPGA (VHDL) 38](#_Toc207967926)

[3.2.2. Desarrollo software en C++ 51](#_Toc207967927)

[4. Resultados 57](#_Toc207967928)

[4.1. Verificación de requisitos 57](#_Toc207967929)

[4.1.1. Entorno 1: Simulación TCL en Vivado 57](#_Toc207967930)

[4.1.2. Entorno de pruebas de comunicación preliminar en HOST. 64](#_Toc207967931)

[4.1.3. Pruebas sobre la aplicación software final. 66](#_Toc207967932)

[4.2. Resultados de las pruebas del sistema 68](#_Toc207967933)

[4.2.1. Temporización en el test de *throughput* del FT245. 68](#_Toc207967934)

[4.2.2. Imposibilidad de transmitir imágenes en color a 10 FPS 69](#_Toc207967935)

[4.3. Plan de validación 70](#_Toc207967936)

[4.3.1. Pruebas del plan de validación 70](#_Toc207967937)

[5. Conclusiones y trabajo futuro 79](#_Toc207967938)

[5.1. Trabajo futuro 80](#_Toc207967939)

[Referencias 81](#_Toc207967940)

[Anexo A: Diagrama relacional de requisitos 82](#_Toc207967941)

[Anexo B: Estructura del repositorio de código del proyecto 84](#_Toc207967942)

[Anexo C: Guía de usuario de ViCON 86](#_Toc207967943)

Lista de figuras

[Figura 2.1: Diagrama de casos de uso 22](#_Toc207967944)

[Figura 3.1: Arquitectura física del sistema. 25](#_Toc207967945)

[Figura 3.2: Sensor de imagen Micron MT9V111. 26](#_Toc207967946)

[Figura 3.3: Digilent Basys 3. 28](#_Toc207967947)

[Figura 3.4: Módulo USB FTDI UM232H-B. 30](#_Toc207967948)

[Figura 3.5: Esquema hardware. 33](#_Toc207967949)

[Figura 3.6: Descripción de pines de entrada/salida de la Digilent Basys 3. 34](#_Toc207967950)

[Figura 3.7: Montaje hardware. 35](#_Toc207967951)

[Figura 3.8: Vista general de la arquitectura lógica. 37](#_Toc207967952)

[Figura 3.9: Vista principal de Vivado, con el proyecto ViCON abierto. 39](#_Toc207967953)

[Figura 3.10: Diagrama de bloques del módulo FT245. 40](#_Toc207967954)

[Figura 3.11: Diagrama de estados del módulo FT245\_RX. 41](#_Toc207967955)

[Figura 3.12: Cronograma de lectura del interfaz FT245. Fuente: (FTDI, 2024) 42](#_Toc207967956)

[Figura 3.13: Diagrama de estados del módulo FT245\_TX. 44](#_Toc207967957)

[Figura 3.14: Cronograma de escritura del interfaz FT245. Fuente: (FTDI, 2024) 44](#_Toc207967958)

[Figura 3.15: Diagrama de estados del módulo de captura de imagen. 47](#_Toc207967959)

[Figura 3.16: Diagrama temporal del sensor MT9V111. Fuente: (Micron, 2004) 48](#_Toc207967960)

[Figura 3.17: Diagrama de estados del módulo de control. 49](#_Toc207967961)

[Figura 3.18: Gráfica de utilización de recursos en la FPGA Xilinx Artix-7. 50](#_Toc207967962)

[Figura 3.19: Interfaz de QT Creator con el proyecto ViCON abierto. 51](#_Toc207967963)

[Figura 3.20: Diagrama de flujo de la aplicación software. 53](#_Toc207967964)

[Figura 3.21: Captura de pantalla de la interfaz gráfica de la aplicación. 55](#_Toc207967965)

[Figura 4.1: Captura de la simulación del módulo FT245\_TX. 58](#_Toc207967966)

[Figura 4.2: Captura de la simulación del módulo FIFO. 59](#_Toc207967967)

[Figura 4.3: Captura de la simulación del módulo FT245\_Rx 60](#_Toc207967968)

[Figura 4.4: Captura de simulación de integración FIFO-FT245\_Tx. 61](#_Toc207967969)

[Figura 4.5: Captura de simulación de integración FIFO-FT245\_Rx. 62](#_Toc207967970)

[Figura 4.6: Captura de simulación del módulo de captura de datos del sensor. 63](#_Toc207967971)

[Figura 4.7: Captura de simulación del módulo de control del sistema FPGA. 64](#_Toc207967972)

[Figura 4.8: Error encontrado en el test de transmisión en FPGA. 66](#_Toc207967973)

[Figura 4.9: Muestra de imágenes activas en Vis1. 71](#_Toc207967974)

[Figura 4.10: Muestra de imágenes no activas en Vis1. 71](#_Toc207967975)

[Figura 4.11: Muestra de imágenes activas en Det1. 72](#_Toc207967976)

[Figura 4.12: Muestra de imágenes no activas en Det1. 73](#_Toc207967977)

[Figura 4.13: Muestra de rostro localizado en Det2. 74](#_Toc207967978)

[Figura 4.14: Muestra de rostro no localizado en Det2. 74](#_Toc207967979)

[Figura 4.15: Muestra de imágenes activas en Rec1. 75](#_Toc207967980)

[Figura 4.16: Muestra de imágenes permaneciendo activas en Rec1. 76](#_Toc207967981)

[Figura 4.17: Muestra de rostro conocido en Rec2. 77](#_Toc207967982)

[Figura 4.18: Muestra de rostro desconocido en Rec2. 77](#_Toc207967983)

Lista de tablas

[Tabla 1.1: Descripción del problema. 11](#_Toc207967984)

[Tabla 1.2: Descripción de la solución. 11](#_Toc207967985)

[Tabla 2.1:Requisitos del proyecto. 17](#_Toc207967986)

[Tabla 2.2: Actores del sistema. 21](#_Toc207967987)

[Tabla 3.1: Asignación de pines a señales. 35](#_Toc207967988)

[Tabla 3.2: Temporizaciones del modo FIFO asíncrono del módulo FT232H. Fuente: (FTDI, 2024) 42](#_Toc207967989)

[Tabla 3.3: Utilización de recursos de la FPGA Xilinx Artix-7. 50](#_Toc207967990)

# Introducción

En la última década, los sistemas de visión artificial han experimentado un notable crecimiento, consolidándose como una de las tecnologías clave en ámbitos como la automoción, la seguridad y el control de accesos biométrico, donde aplicaciones como el reconocimiento facial se han establecido como algunas de las más representativas. No se puede obviar que, hoy en día, la capacidad de adquirir y procesar información visual en tiempo real constituye un pilar fundamental para el desarrollo de aplicaciones inteligentes en sectores industriales y académicos.

No obstante, el desarrollo de estos sistemas plantea importantes retos cuando se trabaja con plataformas de bajo coste y recursos limitados, como es el caso de las FPGAs del entorno educativo. La transmisión continua de vídeo supone un elevado consumo de memoria intermedia y un ancho de banda difícil de sostener con interfaces sencillas, lo que limita la viabilidad de una cadena completa de visión en tiempo real.

En este contexto surge el presente Trabajo Fin de Máster, cuyo enfoque diferencial radica en la adquisición de imágenes bajo demanda. En lugar de transmitir un flujo continuo de vídeo, el sistema desarrollado permite que un equipo anfitrión solicite a la FPGA la captura de un *frame* cuando lo requiera. De esta forma se optimiza el uso de la memoria disponible, se evita la saturación del enlace de comunicación y se proporciona al usuario un control preciso sobre el momento de adquisición.

La plataforma desarrollada se compone de un sensor de imagen digital (MT9V111), una FPGA de bajo coste (Digilent Basys3) encargada de gestionar la recepción y envío de datos, y un módulo de comunicación USB (FT232H-B en modo FT245 asíncrono) que facilita la transferencia de imágenes en un PC. Es en éste último dispositivo donde se implementa la parte software de este trabajo, responsable de la visualización y del procesado de las imágenes, detección y, opcionalmente, reconocimiento facial.

De este modo, ViCON se plantea como una solución académica y experimental que no solo permite consolidar los conocimientos adquiridos en desarrollo en FPGA, visión artificial e interfaces de usuario, sino que también ofrece un marco práctico para validar algoritmos de visión artificial en condiciones reales. Su carácter configurable y extensible lo convierte en un punto intermedio entre las soluciones comerciales de alto coste y las implementaciones puramente software, aportando una alternativa accesible para la docencia, la investigación y el prototipado de aplicaciones de visión embebida.

## Objetivos

El presente Trabajo Fin de Máster tiene como propósito el diseño e implementación de un sistema de visión configurable que permita la adquisición de imágenes bajo demanda, integrando hardware y software en una plataforma accesible orientada a la docencia, la investigación y validación de algoritmos de visión artificial, y más concretamente de detección y reconocimiento facial.

* **Objetivo general**
  + Desarrollar un sistema de visión basado en FPGA que permita la captura de imágenes bajo demanda de un sensor de imagen CMOS y su transmisión a un PC. Será en este dispositivo donde se implemente una aplicación, con su correspondiente interfaz de usuario, que ejecute algoritmos de detección y reconocimiento facial en tiempo real..
* **Objetivos específicos:**
  + Diseñar en VHDL los módulos necesarios para la comunicación con el sensor de imagen (MT9V111) y el módulo USB (FTDI FT232H-B en modo FT245 asíncrono), asegurando la correcta transmisión de datos bajo demanda hacia el equipo anfitrión.
  + Implementar una aplicación software que permita solicitar imágenes a la FPGA de acuerdo con sus necesidades, visualizarlas en tiempo real y aplicar sobre ellas algoritmos de detección y reconocimiento facial.
  + Mantener un repositorio de versionado del código, tanto VHDL como de la aplicación, que incluya histórico de cambios y permita trazabilidad del desarrollo.
  + Validar el correcto funcionamiento del sistema, mediante bancos de pruebas en simulación y pruebas en placa con los correspondientes módulos integrados.

## Entorno de usuarios

El sistema desarrollado en este Trabajo Fin de Máster está concebido para ofrecer una plataforma flexible y versátil en el ámbito de la visión artificial, con un enfoque particular en el reconocimiento facial. Por consiguiente, es aplicable a diferentes perfiles de usuario y contextos de utilización.

En primer lugar, **el entorno académico y de investigación** constituye un área de aplicación de relevancia. La plataforma ViCON puede emplearse como herramienta didáctica en asignaturas relacionadas con sistemas digitales, diseño hardware y visión artificial, facilitando la comprensión práctica de conceptos teóricos a través de una arquitectura real de adquisición y procesamiento de imágenes. Del mismo modo, grupos de investigación en electrónica, inteligencia artificial y visión por computador pueden utilizarla como banco de pruebas para validar algoritmos y realizar experimentos en tiempo real, entre otras aplicaciones.

En segundo lugar, **el ámbito industrial** representa otro conjunto relevante de usuarios. Empresas dedicadas al desarrollo de sistemas de visión artificial pueden usar esta plataforma como prototipo para explorar soluciones de control de accesos biométricos, vigilancia inteligente o supervisión de procesos industriales , entre otros. La posibilidad de capturar imágenes bajo demanda hace que el sistema sea especialmente adecuado en aplicaciones donde no se requiere un flujo continuo de vídeo, optimizando el uso de memoria y ancho de banda sin comprometer la eficacia de los algoritmos.

Finalmente, también resulta de interés en **entornos de aplicación cotidiana**, donde la integración de cámaras y algoritmos de reconocimiento facial ya forma parte de la vida cotidiana. Entre ellos se encuentran:

* **Seguridad doméstica:** sistemas de cámaras que permitan la identificación de personas autorizadas en el acceso a viviendas o espacios privados.
* **Dispositivos móviles y electrónicos personales:** reconocimiento facial como método de autenticación rápido y seguro.
* **Entornos públicos y de transporte:** validación biométrica para control de accesos en aeropuertos, estaciones o edificios oficiales.
* **Sistemas de asistencia y salud:** aplicaciones que faciliten la monitorización de pacientes o el reconocimiento de usuarios en entornos hospitalarios y de cuidados.

En definitiva, este TFM se dirige tanto a la comunidad académica y científica, como al sector profesional e incluso a aquellas potenciales aplicaciones de uso personal. Esto demuestra la transversalidad y relevancia de los sistemas de visión artificial, y su posible uso en un amplio abanico de escenarios.

## Directivas del proyecto

### Oportunidad de aplicación

El **desarrollo de sistemas de visión artificial en tiempo real**, y más concretamente su aplicación al reconocimiento facial, requiere de plataformas capaces de adquirir y procesar imágenes de manera eficiente. Sin embargo, la mayoría de las soluciones disponibles en el mercado se basan bien en equipos de alto coste, que ofrecen gran potencia a cambio de menor accesibilidad; o bien en dispositivos comerciales como cámaras USB, que no permiten un control preciso sobre el flujo de adquisición de datos.

En este contexto, el sistema propuesto ofrece **una plataforma accesible y configurable para la adquisición y transmisión de imágenes bajo demanda**, optimizando recursos en entornos de hardware limitado. De este modo, constituye una oportunidad tanto en los ya mencionados ámbitos académico e investigador, como en aplicaciones prácticas, al facilitar el funcionamiento de los algoritmos desarrollados en escenarios reales.

### 1.3.2 Descripción del problema

A continuación, la tabla 1.1 describe el problema a resolver en este Trabajo Fin de Máster:

|  |  |
| --- | --- |
| El problema de | La ausencia de plataformas de bajo coste que permitan capturar, procesar y transmitir imágenes bajo demanda y en tiempo real para experimentación y validación de algoritmos de visión artificial. |
| Afecta a | Estudiantes, investigadores y profesionales que necesitan un entorno práctico para el desarrollo de aplicaciones de visión artificial. |
| Lo cual tiene como impacto | Dificultad para probar e implementar algoritmos de reconocimiento facial en condiciones reales, dependencia de equipos de alto coste o simulaciones poco representativas. |
| Una solución satisfactoria sería | Desplegar un sistema configurable que integre sensor de imagen, FPGA y comunicación USB, capaz de capturar y transmitir imágenes bajo demanda hacia un entorno de usuario final. |

Tabla 1.1: Descripción del problema.

### 1.3.3 Descripción de la solución

A continuación, la tabla 1.2 describe la solución propuesta en este Trabajo Fin de Máster:

|  |  |
| --- | --- |
| Para | Estudiantes, investigadores y profesionales en visión artificial y sistemas embebidos. |
| Los cuales | Requieren una plataforma práctica, accesible y reconfigurable para validar algoritmos de visión artificial y/o implementar prototipos. |
| ViCON | Es un sistema hardware-software basado en FPGA que integra un sensor de imagen digital CMOS y un módulo FT245 a USB, diseñado para capturar imágenes bajo demanda y transmitirlas hacia un PC para su visualización y procesado. |
| Que | Permite la captura, envío y visualización de imágenes según sean solicitadas, sirviendo como base para aplicaciones de detección y reconocimiento facial. |
| Frente a | Cámaras comerciales que transmiten vídeo sin control del usuario y soluciones propietarias de alto coste que limitan la accesibilidad y la adaptabilidad. |
| Este desarrollo | Ofrece una solución flexible, extensible y de bajo coste, orientada tanto a entornos académico como a aplicaciones prácticas de visión artificial, y más específicamente al reconocimiento facial. |

Tabla 1.2: Descripción de la solución.

## Alternativas y competencia

## 1.4.1 Alternativas

En el ámbito de la visión artificial y el reconocimiento facial existen diversas soluciones consolidadas que permiten la adquisición y procesamiento de imágenes en tiempo real. Sin embargo, estas alternativas presentan limitaciones que justifican el desarrollo de un sistema como ViCON.

* **Cámaras USB y dispositivos comerciales de propósito general.**
  + Descripción: Cámaras conectadas a un PC mediante USB, que transmiten un flujo continuo de vídeo.
  + Ventajas:
    - Bajo coste.
    - Facilidad de uso.
    - Amplia compatibilidad.
  + Limitaciones:
    - Ausencia de control sobre la adquisición de *frames*.
    - Imposibilidad de optimizar el uso de memoria o el ancho de banda.
    - Dependencia de un PC de propósito general
  + Comparación con ViCON: Este desarrollo ofrece control de la adquisición bajo demanda y permite experimentar con hardware digital, aportado valor didáctico, cercanía al hardware y configurabilidad.
* **Plataformas embebidas de alto nivel.**
  + Descripción: Sistemas con procesadores y aceleradores integrados para visión por computador y redes neuronales.
  + Ejemplos: Raspberry Pi, NVIDIA Jetson, Intel Movidius.
  + Ventajas:
    - Gran potencia de cálculo.
    - Ecosistema de software optimizado.
    - Orientación a IA.
  + Limitaciones:
    - Mayor consumo energético.
    - Coste superior.
    - Menor visibilidad y control de bajo nivel sobre la cadena de adquisición de imágenes.
  + Comparación con ViCON: Mientras estas plataformas están orientadas a ejecución eficiente de algoritmos, ViCON permite controlar cómo se adquieren y transmiten los datos, en un entorno con coste inferior y compatible con dispositivos de propósito general y específico.
* **Kits FPGA comerciales especializados en visión.**
  + Descripción: Sistemas basados en FPGA diseñados específicamente para aplicaciones de visión por computador y su integración con cámaras y sensores de imagen.
  + Ejemplos: Xilinx Kria KV260 Vision AI Kit, Avnet Embedded Vision Kit, Terasic DE10-Nano + MIPI Camera Kit, Lattice Embedded Vision Kit.
  + Ventajas:
    - Alta flexibilidad.
    - Integración con cámaras nativa.
    - Idóneos para aplicaciones industriales.
  + Limitaciones:
    - Coste elevado.
    - Complejidad de uso.
    - Orientación a desarrolladores profesionales.
  + Comparación con ViCON: ViCON se sitúa como una alternativa de bajo coste y mayor simplicidad a estas plataformas. Su enfoque está orientado hacia la docencia e investigación básica en reconocimiento facial, más que al entorno industrial y/o profesional.

### 1.4.2 Posicionamiento de ViCON

ViCON se diferencia de las soluciones anteriormente mencionadas en los siguientes aspectos:

* La **captura** de imágenes se realiza **bajo demanda**.
* Propone una solución que integra **hardware y software** en un flujo completo de adquisición, transmisión y procesado de imágenes.
* Está **orientado al reconocimiento facial**, como un caso de uso representativo y de alta relevancia actual.
* Ofrece un **coste y complejidad reducidos**, haciéndolo accesible a estudiantes, docentes e investigadores que no disponen de acceso a plataformas profesionales de visión embebida.

## Estructura de la memoria

La presente memoria se organiza en cinco capítulos principales. En el presente capítulo, el **Capítulo 1**, se introducen los objetivos del proyecto, el entorno de usuarios y las directivas que guían el desarrollo, así como un análisis de alternativas y competencia. El **Capítulo 2** recoge los requisitos del sistema y los casos de uso definidos, estableciendo la base técnica sobre la que se estructura el trabajo. El **Capítulo 3** describe el diseño y el desarrollo del sistema, diferenciando la arquitectura física (componentes hardware y conexionado) y la arquitectura lógica (desarrollo WHDL y software). En el **Capítulo 4** se presentan los resultados obtenidos, incluyendo la verificación de requisitos, pruebas del sistema y plan de validación. Finalmente, el **Capítulo 5** expone las conclusiones del trabajo realizado y plantea posibles líneas de mejora y ampliación futura.

La memoria se complementa con las **referencias bibliográficas** y un **anexo** en el que se recoge el **diagrama relacional de requisitos**.

# Requisitos y casos de uso

El diseño de un sistema de visión artificial requiere establecer de manera clara los requisitos que guiarán su desarrollo, garantizando que las funcionalidades implementadas respondan a los objetivos definidos. En este apartado se describen tanto los **requisitos hardware y software del sistema**, como los **casos de uso** para los que se pretende proporcionar una solución.

Los **requisitos** constituyen la **base técnica sobre la que se estructura el proyecto**, abarcando desde las especificaciones de adquisición y transmisión de imágenes hasta las condiciones de procesado necesarias para el reconocimiento facial. Su definición asegura la trazabilidad del trabajo y permite verificar, en fases posteriores, que la implementación satisface las necesidades planteadas.

Por otra parte, los **casos de uso** **ilustran cómo interaccionan los distintos usuarios con el sistema** y qué funcionalidades esperan obtener de él. Estos casos permiten contextualizar el proyecto en escenarios reales (académicos, de investigación o de aplicación práctica) y demuestran la utilidad de captura bajo demanda aplicada a la validación de algoritmos de detección y reconocimiento facial.

En conjunto, este capítulo establece el marco de referencia para la implementación del sistema, alineando los aspectos técnicos con los escenarios de aplicación previstos.

## Requisitos del proyecto

El presente apartado recoge el **conjunto de requisitos identificados** para el desarrollo del proyecto. Estos se presentan de forma unificada en la Tabla 2.1, incluyendo tanto especificaciones de software como de hardware y de gestión del proyecto.

En la columna de prioridad, el valor *“F”* hace referencia a un requisito fundamental del proyecto, el valor *“D”* a un requisito deseable y el valor *“O”* a un requisito opcional. La columna de precedencia indica la relación de dependencia entre requisitos, que se muestra en detalle en el Anexo A.

En secciones posteriores, los requisitos se organizarán en funcionales y no funcionales, añadiendo una breve descripción para facilitar su trazabilidad y validación.

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Nombre | Prioridad | Precedencia |
| SW1 | Aplicación | F |  |
| SW2 | Interfaz de usuario | F | SW1 |
| SW3 | Lenguaje de programación | F | SW1 |
| SW4 | Entrada de información | F | SW1 |
| SW5 | Recepción de la información | F | SW4 |
| SW6 | Visualización de imágenes | F | SW2 |
| SW7 | Solicitud de envío de imágenes | F | SW4, SW5 |
| SW8 | Habilitación de representación de imágenes | F | SW6 |
| SW9 | FPS | F | SW7 |
| SW10 | Imágenes en escala de grises | F | SW5 |
| SW11 | Resolución de imagen | F | SW5 |
| SW12 | Imágenes en color | D | SW5 |
| SW13 | Biblioteca de visión | F | SW1 |
| SW14 | Detección de rostros | F | SW13 |
| SW15 | Habilitación de detección de rostros | F | SW14 |
| SW16 | Rango de detección | F | SW14 |
| SW17 | Indicación de detección | F | SW14 |
| SW18 | Reconocimiento de rostros | O | SW14 |
| SW19 | Habilitación de reconocimiento | O | SW18 |
| SW20 | Sistema Operativo | F | SW1 |
| SW21 | Ejecutable | F | SW1 |
| SW22 | Dependencias | F | SW21 |
| SW23 | Instalación de dependencias | F | SW22 |
| SW24 | Versionado aplicación | F | SW22 |
| SW25 | Histórico de cambios | F | SW24 |
| HW1 | FPGA | F |  |
| HW2 | VHDL | F | HW1 |
| HW3 | Versionado VHDL | F | HW2 |
| HW4 | Histórico de cambios VHDL | F | HW3 |
| HW5 | Validación | F | HW1 |
| HW6 | Entorno de desarrollo | F | HW3, HW4 |
| HW7 | Interfaz host | F | HW1 |
| HW8 | Interfaz sensor | F | HW1 |
| HW9 | Solicitud de información al sensor | F | HW7, HW8 |
| HW10 | Captura de la información del sensor | F | HW7, HW9 |
| HW11 | Comunicación con el host | F | HW7 |
| HW12 | Gestión de memoria llena | D | HW10 |
| PR1 | Fecha de entrega | F |  |

Tabla 2.1:Requisitos del proyecto.

### Descripción de los Requisitos Funcionales

Los **requisitos funcionales** describen las **capacidades esenciales que debe proporcionar el sistema**, definiendo la interacción esperada entre los componentes hardware y software para garantizar la adquisición, transmisión, procesamiento y visualización de imágenes en tiempo real. Se hace especial énfasis en las funcionalidades de detección y reconocimiento facial.

De acuerdo con el análisis realizado, los requisitos funcionales se han organizado en **tres bloques temáticos** (adquisición y transmisión de imágenes, visualización y control, y procesamiento y reconocimiento facial). Así, quedan claramente reflejados los distintos niveles de interacción entre hardware y software.

**Adquisición y transmisión de imágenes.**

Este primer bloque agrupa los requisitos relacionados con la **captura de datos por parte del sensor de imagen, la gestión de la comunicación hardware y la transmisión de la información hacia el host**. Estos requisitos garantizan que el sistema pueda solicitar, recibir y transferir imágenes bajo demanda de manera eficiente.

* **SW4:** La aplicación deberá recibir como datos de entrada las imágenes generadas por el sensor de imagen conectada al sistema.
* **SW5:** La recepción de la información procedente de la FPGA se realizará mediante un enlace USB, garantizando la correcta transmisión hacia el dispositivo anfitrión.
* **HW7:** El hardware debe ser capaz de transmitir datos a demanda del dispositivo host, permitiendo la visualización en tiempo real.
* **HW8:** La FPGA se comunicará con el sensor de imagen mediante una implementación hardware desarrollada a medida, con el fin de gestionar de manera eficiente la transferencia de información entre ambos dispositivos.
* **HW9:** La FPGA solicitará información de imagen a el sensor de imagen en respuesta a una petición del host, asegurando la coherencia en la comunicación.
* **HW10:** La FPGA capturará la información recibida desde el sensor y la transmitirá al host para su posterior procesado.
* **HW11:** La FPGA se comunicará con el host utilizando una interfaz FTDI FTD245 asíncrona para la transmisión de los datos de imagen.
* **HW12:** El sistema será capaz de gestionar adecuadamente situaciones de memoria llena, manteniendo activa la detección en tiempo real.

**Visualización y control de imágenes.**

El segundo bloque se centra en la **representación de las imágenes en la aplicación software y en las funcionalidades de control asociadas**. Aquí se definen tanto los parámetros mínimos de visualización (resolución, tasa de refresco…) como las opciones de habilitar o deshabilitar la adquisición de imágenes.

* **SW6:** La aplicación deberá mostraren tiempo real las imágenes recibidas del sensor.
* **SW7:** Las imágenes a visualizar deberán ser solicitadas explícitamente a la FPGA bajo demanda, en función de la capacidad de representación del sistema software.
* **SW8:** Las representación de imágenes, y por tanto su recepción, deberá ser habilitable y deshabilitable desde la aplicación.
* **SW9:** La aplicación garantizará una tasa mínima de refresco de 10 fotogramas por segundo cuando la visualización esté habilitada.
* **SW10:** El sistema mostrará como mínimo imágenes en escala de grises.
* **SW11:** La resolución de las imágenes será VGA (640 x 480 píxeles).
* **SW12:** La aplicación ofrecerá además la posibilidad de mostrar imágenes en color.

**Procesamiento y reconocimiento facial.**

Finalmente, este bloque engloba los requisitos asociados al **procesado de las imágenes**, destacando las funcionalidades de detección y reconocimiento facial.

* **SW13:** La aplicación hará uso de la biblioteca OpenCV para implementar los algoritmos de detección y reconocimiento de rostros
* **SW14:** La aplicación deberá implementar un algoritmo de detección de rostros sobre las imágenes recibidas.
* **SW15:** La aplicación permitirá habilitar y deshabilitar la detección de rostros sobre las imágenes.
* **SW16:** El reconocimiento facial deberá ofrecer un funcionamiento preciso y fiable en un rango de distancias comprendido entre 1 y 5 metros entre el sensor de imagen y los rostros de las personas.
* **SW17:** Los rostros detectados deberán señalarse sobre la propia imagen mostrada, garantizando una interpretación clara por parte del usuario.
* **SW18:** La aplicación deberá ser capaz de determinar si los rostros detectados se corresponden con aquellos previamente almacenados en una base de datos.
* **SW19:** La aplicación deberá permitir habilitar y deshabilitar la funcionalidad de reconocimiento de rostros de manera independiente.

En resumen, los requisitos aquí definidos garantizan que el sistema ViCON pueda llevar a cabo de forma integrada el ciclo completo de adquisición, transmisión, visualización y procesamiento de imágenes en tiempo real. Además, se definen requisitos para las funcionalidades de detección y reconocimiento de rostros, destinados a validar el sistema en escenarios de aplicación práctica y real.

Estos requisitos constituyen la base para el diseño y la implementación del sistema, y se usarán como referencia para verificar que el desarrollo cumple con los objetivos planteados.

### Descripción de los Requisitos No Funcionales

Los **requisitos no funcionales** establecen las **condiciones bajo las cuales debe operar el sistema**, definiendo aspectos de compatibilidad, usabilidad, mantenibilidad y gestión del proyecto. A diferencia de los funcionales, que han descrito qué hará el sistema, estos requisitos se centran en cómo debe comportarse y bajo qué restricciones técnicas y de gestión debe desarrollarse.

Se han agrupado en tres bloques temáticos: software, hardware y de proyecto.

**Requisitos no funcionales de software.**

Este bloque recoge las **especificaciones relacionadas con la portabilidad, la gestión de dependencias y la organización del código** de la aplicación software.

* **SW1:** Se proporcionará una aplicación software para ser ejecutada en un dispositivo host.
* **SW2:** La aplicación deberá ofrecer una interfaz de usuario para interactuar con el usuario.
* **SW3:** El software será desarrollado en C o C++.
* **SW20:** La aplicación será compatible, al menos, con el sistema operativo Windows 10.
* **SW21:** El ejecutable entregado será autocontenido.
* **SW22:** El ejecutable incluirá todas las dependencias necesarias para su correcta instalación.
* **SW23:** En caso de que alguna dependencia no pueda ser incluida, el ejecutable deberá detectar la situación y proceder a su instalación automática sin intervención manual.
* **SW24:** El código fuente de la aplicación será versionado y almacenado en un repositorio bajo un sistema de control de versiones.
* **SW25:** El repositorio deberá permitir el acceso tanto al desarrollador como al cliente, con trazabilidad de cambios y posibilidad de recuperar versiones anteriores del código.

**Requisitos no funcionales de hardware.**

Estos requisitos establecen las **condiciones del entorno hardware, así como las herramientas de desarrollo y validación** necesarias para garantizar la correcta implementación del sistema hardware.

* **HW1:** Se hará uso de un chip FPGA de bajo coste perteneciente a una gama actual.
* **HW2:** El diseño hardware será implementado en lenguaje VHDL.
* **HW3:** El código fuente VHDL desarrollado será versionado y almacenado en un repositorio bajo un sistema de control de versiones adecuado.
* **HW4:** El repositorio debe permitir el acceso tanto al desarrollador como al cliente, tal posibilitando el seguimiento de cambios y la recuperación de versiones anteriores.
* **HW5:** La validación del desarrollo en FPGA se llevará a cabo mediante scripts TCL, garantizando que se cumpla la funcionalidad especificada.
* **HW6:** El entorno de desarrollo hardware será Vivado Design Suite.

**Requisitos de proyecto.**

Este bloque hace referencia a la **planificación temporal y de gestión del desarrollo**.

* **PR1:** El proyecto debe terminar el 05/09/2025.

En conjunto, estos requisitos garantizan que el sistema no solo cumpla con las funcionalidades previstas, sino que también lo hagan dentro de un marco de calidad, mantenibilidad y viabilidad técnica acorde con los objetivos del TFM.

## Casos de uso

En este apartado se describe la **interacción del sistema con su entorno definiendo los actores que participan y los casos de usos pertinente**. El objetivo es capturar, de forma concisa y verificable, cómo el usuario desencadena la adquisición bajo demanda y la visualización de imágenes; además de activar, de manera opcional, la detección y el reconocimiento facial sobre las imágenes recibidas.

Se definen los actores del sistema como aquellas entidades externas a la frontera de éste y que interactúan activamente con él. En este caso, **la frontera del sistema incluye el sensor de imagen, la FPGA, el interfaz de comunicación y la aplicación de PC**. Por tanto, todos estos elementos forman parte del sistema y no se modelan como actores. Del mismo modo, la persona o personas que puedan estar frente al sensor de imagen tampoco serán actores, sino objeto de procesamiento. Esto es porque no tienen una acción directa sobre el sistema (no interactúan con el mismo), sino que tienen una relación pasiva con él.

Por ende, solamente encontraremos un actor en el sistema de ViCON: el operador de la aplicación.

|  |  |
| --- | --- |
| Nombre | Descripción |
| Usuario | Operador de la aplicación. Será el actor que solicite y detenga tanto la adquisición de imágenes como la detección de rostros y su reconocimiento. Además, interpreta en pantalla los resultados. |

Tabla 2.2: Actores del sistema.

Por su parte, los casos de uso del sistema derivan del flujo de operación de la aplicación, y se relacionan entre sí de forma jerárquica. Estos casos de uso quedan evidenciados en la figura 2.1, mientras que en los siguientes apartados se detallará la especificación de cada uno.

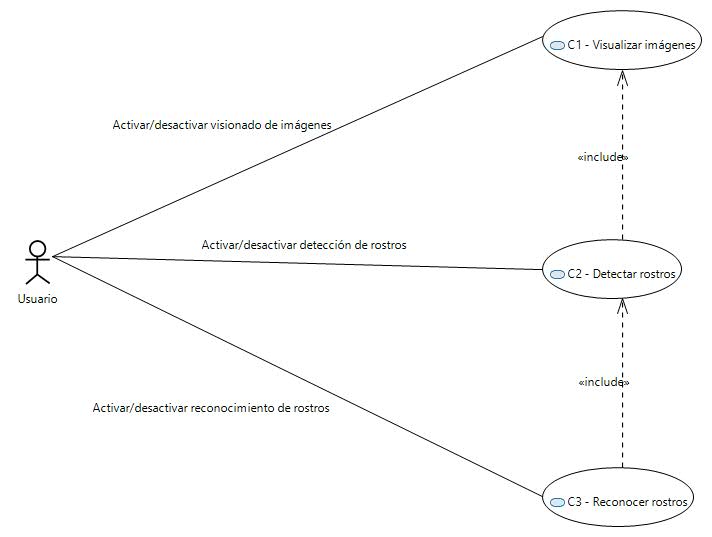


Figura 2.1: Diagrama de casos de uso

### C1 – Visualizar imágenes

* **Actor principal:** Usuario.
* **Objetivo:** Solicitar y visualizar en la aplicación imágenes procedentes de la FPGA en tiempo real
* **Precondiciones:**
  + El sensor de imagen está conectado a la FPGA.
  + La FPGA tiene el interfaz USB conectado, y éste a su vez está conectado al PC.
  + La FPGA tiene su correspondiente archivo de programación cargado.
  + La aplicación está en ejecución
* **Postcondiciones:** El usuario recibe en pantalla un flujo de imágenes bajo demanda de la propia aplicación.
* **Flujo principal:**

1. El usuario solicita la visualización de imágenes.
2. La aplicación envía una petición a la FPGA.
3. La FPGA captura un *frame* del sensor de imagen.
4. La FPGA transmite las imágenes capturadas al host mediante el interfaz USB.
5. La aplicación muestra las imágenes en tiempo real al usuario.

Este flujo se repite constantemente mientras se mantenga la solicitud de imágenes activada.

* **Flujos alternativos:**
  + 2a – La FPGA no responde: La aplicación vuelve a solicitar un *frame* tras un tiempo de *timeout*.
  + 4a – El enlace USB se interrumpe: La aplicación notifica pérdida de conexión.

### C2 – Detectar rostros

* **Actor principal:** Usuario.
* **Objetivo:** Habilitar la detección de rostros en las imágenes mostradas.
* **Precondiciones:**
  + C1 (Visualizar imágenes) está en ejecución.
* **Postcondiciones:** El sistema indica visualmente en pantalla los rostros detectados.
* **Flujo principal:**

1. El usuario habilita la opción de detección de rostros en la aplicación.
2. La aplicación procesa las imágenes recibidas con la librería OpenCV.
3. El sistema identifica regiones de interés que corresponden a rostros.
4. Los rostros detectados se marcan en la imagen mostrada.

* **Flujos alternativos:**
  + 2a – El algoritmo no encuentra rostros: la aplicación no muestra ninguna región de interés sobre la imagen.

### C3 – Reconocer rostros

* **Actor principal:** Usuario.
* **Objetivo:** Identificar a personas en base a los rostros previamente aprendidos.
* **Precondiciones:**
  + C2 (Detectar rostros) está en ejecución.
  + La base de datos de rostros está disponible
* **Postcondiciones:** Cada rostro detectado aparece etiquetado con la identidad reconocida (si existe).
* **Flujo principal:**

1. El usuario habilita la opción de reconocimiento facial en la aplicación.
2. La aplicación envía los rostros detectados al módulo de reconocimiento.
3. El sistema clasifica los rostros en función de la información previamente aprendida.
4. La aplicación muestra en pantalla la identidad correspondiente junto a la región de interés del rostro reconocido.

* **Flujos alternativos:**
  + 2a – La base de datos no está disponible: la aplicación informa al usuario y mantiene la detección activa sin reconocimiento.
  + 3a – El rostro no ha sido aprendido/reconocido: se muestra un rostro “desconocido”

# Diseño y desarrollo del sistema

El diseño del sistema ViCON se ha estructurado siguiendo una **aproximación multinivel**, diferenciando entre **arquitectura física y arquitectura lógica**. Esta separación facilita describir, por un lado, los componentes tangibles que conforman el sistema; y , por otro, la organización funcional de dichos elementos y el desarrollo software y VHDL asociado.

## Arquitectura Física

En este apartado se presentan los componentes hardware empleados en la implementación (sensor de imagen, FPGA, módulo de comunicación USB y equipo anfitrión), así como las conexiones establecidas entre ellos. Así, se ofrece una visión global de la infraestructura sobre la que se construye el ViCON; proporcionando, además, una justificación de la elección de cada dispositivo de acuerdo con los requisitos establecidos.

A continuación, la figura 3.1 muestra un esquema de los elementos hardware presentes en el sistema, así como de la interrelación entre ellos.



Figura 3.1: Arquitectura física del sistema.

### Sensor de imagen: MT9V111

El sensor de imagen usado en el sistema es la **MT9V111 de Micron**, un sensor digital CMOS de ¼” en **formato de salida VGA** (640 x 480 píxeles) con arquitectura *System-on-Chip* (SoC). Este componente integra tanto el núcleo del sensor como un procesador de flujo de imagen (*Image Flow Processor*, IFP), lo cual lo convierte en una solución compacta y eficiente para aplicaciones embebidas.

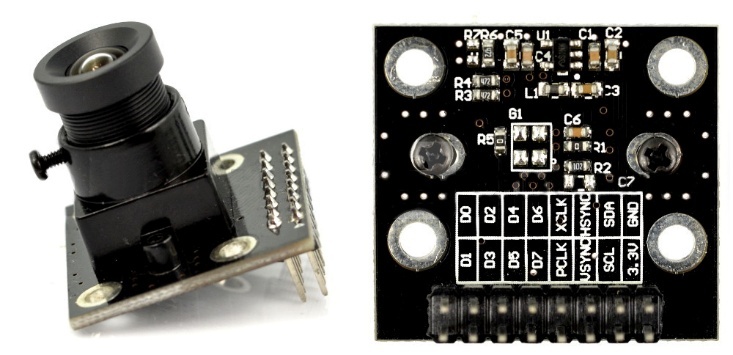


Figura 3.2: Sensor de imagen Micron MT9V111.

Entre sus características principales destacan:

* **Resolución activa:** Configurable, establecida en 640 x 480 píxeles (VGA).
* **Tamaño de píxel:** 5.6 µm x 5.6 µm.
* **Formato óptico:** ¼” (diagonal 4.48 mm).
* **Tasa de *frames*:** Hasta 30 fps a 27 MHz; 15 fps en la configuración por defecto a 12 MHz.
* **Conversión A/D:** 10 bits, integrada en el chip.
* **Consumo típico:** < 80 mW a 15 fps.
* **Rango dinámico:** 60 dB.
* **Sensibilidad:** 1.9V/lux·s a 550 nm.

Como ya se ha mencionado anteriormente, el sensor incluye un procesador de flujo de imagen que realiza operaciones de preprocesado de la imagen: corrección de color, recuperación cromática, balance de blancos… Estas funcionalidades permiten que el sensor entregue imágenes preprocesadas de alta calidad, aliviando la carga de la FPGA y del PC en tareas de acondicionamiento básico.

Con respecto a su interfaz, el dispositivo posee las siguientes entradas y salidas:

* **Entradas:**
  + **XCLK:** Señal de reloj de entrada que determina la tasa de captura de *frames*. Puede configurarse en un rango de 12 MHz a 27 MHz.
  + **RST#:** Señal de reset asíncrono, activa a nivel bajo.
  + **SDA/SCL:** Líneas de datos y reloj de la interfaz de configuración serie. Aunque el sensor puede configurarse mediante ellas, en este proyecto no se hace uso de dicha funcionalidad.
* **Salidas:**
  + **PIXCLK:** Señal de reloj de salida asociada al flujo de datos. Mantiene la misma frecuencia que XCLK, con un pequeño desfase inducido por la lógica interna del sensor. El sensor de imagen presenta un nuevo dato válido en cada flanco de subida de PIXCLK.
  + **VSYNC:** Señal de sincronización vertical. Cuando se encuentra en nivel alto, indica que se está transmitiendo una imagen completa a través de la interfaz de datos; en caso contrario, no hay información válida disponible.
  + **HREF:** Señal de sincronización horizontal. Se activa en nivel alto durante la transmisión de cada línea de la imagen, siempre dentro del intervalo de VSYNC. En este periodo, los datos de la imagen se pueden capturar en los flancos de subida de PIXCLK.
  + **CAMERA[7:0]:** Bus de datos paralelo de 8 bits que transmite la información de la imagen del sensor de imagen a la FPGA, byte a byte.

**Estas señales constituyen la interfaz fundamental entre el sensor de imagen y la FPGA**, permitiendo el control del flujo de imágenes y su correcta sincronización. El uso de un bus de datos paralelo acompañado de señales de sincronismo (PIXCLK, VSYNC y HREF) simplifica la lógica de adquisición y garantiza un flujo de información adecuado para su transmisión posterior hacia el PC anfitrión.

La elección de este sensor responde a dos criterios principales:

* **Flexibilidad y bajo consumo:** La posibilidad de trabajar entre 12 y 27 MHz de frecuencia de reloj (15 fps y 30 fps respectivamente) permite ajustar el equilibrio entre tasa de *frames*, consumo y complejidad del diseño.
* **Procesamiento de imagen integrado:** Al incluir un sistema de preprocesado de la imagen, reduce la necesidad de desarrollar algoritmos adicionales en la FPGA o en la aplicación para estos efectos. Esto nos permite reservar los recursos disponibles para las funcionalidades clave del proyecto: la adquisición y transmisión de los *frames*, y los algoritmos de detección y reconocimiento de rostros.

### FPGA: Digilent Basys 3

La **Basys 3** es una **placa de desarrollo de bajo coste** diseñada por Digilent, basada en una FPGA Artyx-7 XC7A35T-1CPG236C de Xilinx. Se trata de un recurso ampliamente utilizado en entornos docentes y de investigación, pues combina un precio accesible con un conjunto de prestaciones que permiten implementar desde circuitos combinacionales simples hasta procesadores embebidos y sistemas complejos.

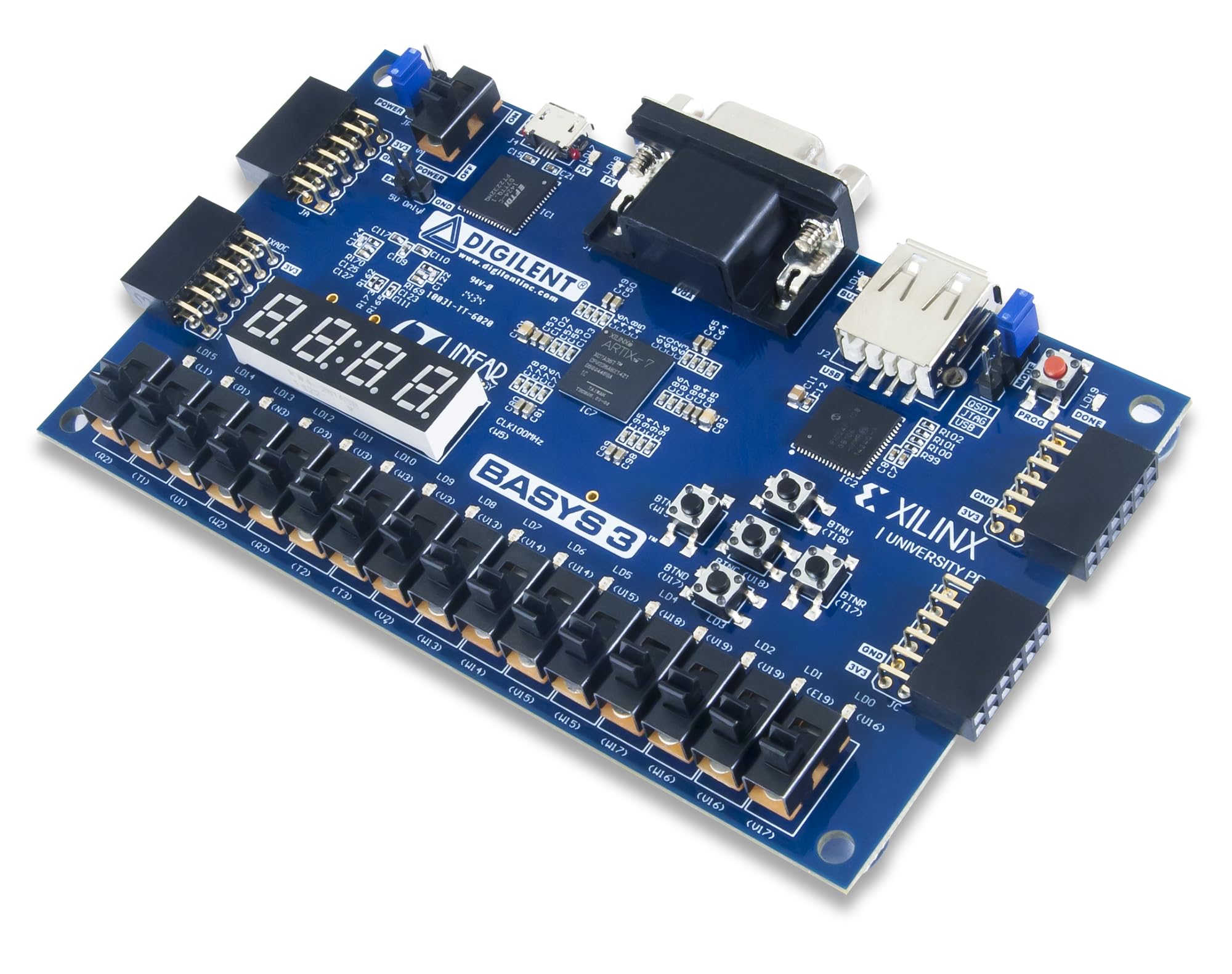


Figura 3.3: Digilent Basys 3.

Entre sus características principales destacan:

* **FPGA Artix-7 XC7A35T** con 33.280 celdas lógicas, 90 bloques DSP y 1.800 Kb de memoria BRAM.
* **Velocidades de reloj** internas de hasta 450 MHz.
* **Memoria de configuración SPI Flash** de 32 Mbit, que permite el almacenamiento de *bitstreams* y reconfiguración autónoma al encender.
* Entradas y salidas integradas:
  + 16 switches, 16 LEDs de usuario y 5 botones pulsadores-
  + *Display* de 7 segmentos de 4 dígitos.
  + Conector VGA (12 bits de color).
  + **Tres puertos Pmod para expansión** (Además de un puerto Pmod dedicado a entradas analógicas del XADC).
* Conectividad USB: **puerto micro-USB para programación JTAG/UART** y puerto USB Host para dispositivos HID (ratones, teclados, memorias…).
* **Oscilador de 100 MHz integrado** como reloj base para los diseños.

La **Basys 3 es compatible con el entorno Vivado Design Suite de Xilinx**. Este software facilita la síntesis, implementación y depuración de diseños en VHDL, así como la generación de *bitstreams* para configuración de la FPGA. La placa soporta varios modos de programación, pero en este proyecto **se hará uso del modo JTAG a través del puerto micro-USB**.

En el marco de este TFM, la Basys 3 actúa como **núcleo de procesamiento hardware**, interconectando el sensor de imagen MT9V111 con el módulo de comunicación USB y gestionando la transmisión bajo demanda hacia el PC anfitrión. La FPGA alberga la lógica de:

* **Controlador de sensor de imagen:** Captura de *frames* y sincronización.
* **Máquina de estados de control:** Gestión de la adquisición bajo demanda.
* **Interfaz FIFO-FT245 asíncrono:** Para recepción de solicitudes y transmisión de datos al host.

La elección de la plataforma Basys 3 responde a varios criterios:

1. **Disponibilidad en el entorno académico:** Es la plataforma FPGA estándar del máster, lo que facilita soporte y reutilización de recursos.
2. **Bajo coste y orientación docente:** Su precio y características la sitúan como una plataforma ideal para proyectos de formación e investigación inicial.
3. **Recursos suficientes para el proyecto:** El Artix-7 XC7A35T proporciona capacidad lógica y de memoria más que suficiente para implementar los módulos necesarios del sistema.
4. **Facilidad de integración de los módulos:** Las interfaces de expansión Pmod permiten conectar periféricos externos y comunicarse con el PC sin elementos adicionales.

### Interfaz USB: FTDI FT232H-B

El sistema hace uso del **módulo UM232H-B**, un *break-out board* desarrollado por FTDI que implementa el integrado **FT232H**. Este dispositivo actúa como convertidor USB 2.0 Hi-Speed a interfaces serie o paralelo configurables, resultando en una solución sencilla para conectar hardware digital a un PC mediante USB. Posee un conector de 18 pines a 2 mm que **permite emplearlo en múltiples modos** (UART, FIFO 245…)

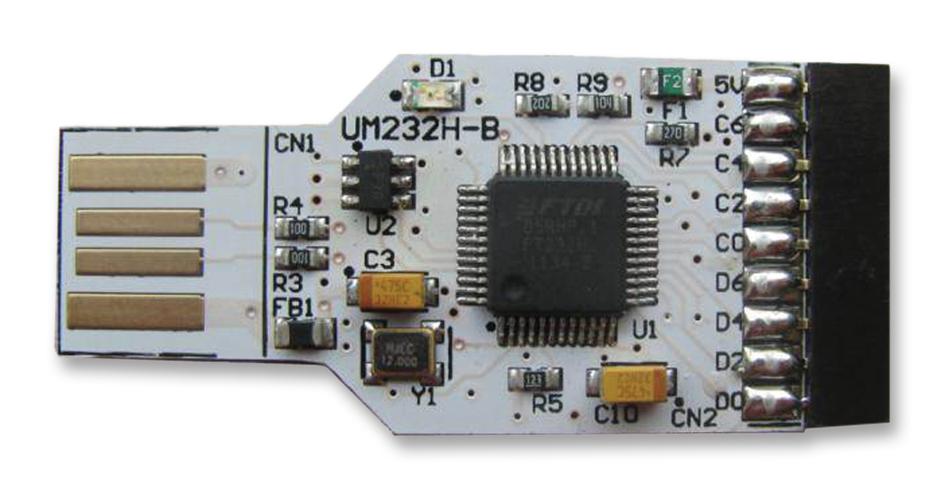


Figura 3.4: Módulo USB FTDI UM232H-B.

En este TFM se hace uso de la **configuración FIFO asíncrono FT245**, lo que permite intercambiar datos mediante un bus paralelo de 8 bits con control por señales de lectura y escritura, funcionando como un **puente eficiente entre la FPGA y el PC.**

Entre sus características principales destacan:

* **EEPROM interna** configurable para ajustar pines y parámetros USB.
* **Conectividad USB 2.0** de alta velocidad (hasta 480 Mbps teóricos).
* **Drivers multiplataforma**: soporta tanto controladores VCP (*Virtual COM Port)*  como D2XX (FTDI Chip, 2023) (API directa), compatibles con Windows, Linux, macOS y Android.
* Tensión de E/S de 3.3 V, tolerante a 5 V.

Y con respecto a su interfaz, disponemos de las siguientes señales:

* **Bus de datos.**
  + **DATA[7:0] :** Bus de datos bidireccional de 8 bits. Se colocan o reciben datos en este bus en función de las señales de control activas
* **Señales de control principales:**
  + **TXE#:** Indica a la FPGA que existen datos disponibles en la FIFO del FT245 cuando está a nivel bajo.
  + **RXF#:** Indica a la FPGA que la FIFO tiene espacio libre para recibir nuevos datos si está a nivel bajo.
  + **WR#:** Señal activada por la FPGA para escribir un byte en la FIFO cuando TXE# está activo
  + **RD#:** Señal activada por la FPGA para leer un byte de la FIFO cuando RXF# está activo.
* **Señales auxiliares**
  + **OE#:** No se usa en modo asíncrono, por lo que se fuerza a nivel alto.
  + **SIWU#:** No se emplea en este proyecto, por lo que se fuerza a nivel alto.
  + **CLKOUT:** Salida de reloj generada por el FT232H, no empleada por el modo asíncrono.
  + **PWRSAV#:** Señal de ahorro energético, mantenida en nivel alto en este diseño.

Esta interfaz **permite que la FPGA reciba y transmita datos desde y hacia el PC de manera sencilla**, sin necesidad de implementar una pila USB completa. La comunicación se reduce a operaciones de lectura y escritura sobre el bus paralelo de 8 bits, reguladas por señales de estado que garantizan que el intercambio de información es fiable.

En la arquitectura del proyecto, **el FT232H es el interfaz de comunicación** entre la FPGA y el PC. Su **modo FT245 asíncrono** habilita la transmisión bajo demanda de *frames* capturados por el sensor de imagen, en este caso haciendo uso del driver D2XX pertinente.

1. El PC solicita un *frame* a través del driver D2XX.
2. El módulo FT232H gestiona la petición USB y la convierte en señales del interfaz FT245.
3. La FPGA recibe la petición, realiza la captura y entrega los datos de imagen a través de la interfaz paralelo.
4. El FT232H encapsula la información y la transmite por USB al PC.

La elección de este módulo se fundamenta en los siguientes motivos:

* **Compatibilidad con la FPGA:** Su modo FT245 asíncrono simplifica la lógica de comunicación frente a protocolos más complejos (por ejemplo, Ethernet).
* **Simplicidad de integración software:** La librería D2XX permite implementar en C++ la transmisión de datos sin necesidad de controladores personalizados.
* **Flexibilidad y coste reducido:** Al ser un módulo de propósito general de bajo coste, resulta accesible en entornos académicos.
* **Fiabilidad probada:** FTDI es un estándar de facto en la industria para soluciones de puente USB, lo que garantiza robustez y soporte multiplataforma.

### Equipo host: PC.

El último elemento de la arquitectura física es el equipo anfitrión, un **ordenador personal de propósito general** que actúa como destino final de los datos capturados por el sistema. Su función principal es solicitar imágenes a la FPGA a través del enlace USB, recibir los *frames* transmitidos y ofrecer un entorno de ejecución para el software de procesado y visualización.

Las características más relevantes del equipo host son:

* **Sistema operativo:** Windows 10/11, que garantiza compatibilidad con los controladores oficiales de FTDI y la librería D2XX empleada en este proyecto.
* **Interfaz de comunicación:** Dispone de puertos USB 2.0/3.0 estándar, a través del cual se realiza la comunicación con la FPGA por medio del módulo FT232H.
* **Capacidad de procesado:** Es suficiente para ejecutar los algoritmos de visualización, detección y reconocimiento facial desarrollados en C++ sobre la biblioteca OpenCV.
* **Naturaleza flexible:** Al tratarse de un ordenador de propósito general, se facilita la depuración, el análisis de resultados y la integración futura de algoritmos más complejos.

En el contexto de este proyecto, el PC desarrolla tres funciones fundamentales:

1. **Generar solicitudes de adquisición hacia la FPGA**, activando el flujo de datos bajo demanda.
2. **Recibir y almacenar temporalmente los frames** transmitidos a través del enlace USB.
3. **Proporcionar capacidad de visualización y procesado software**¸ si bien la descripción detallada de estos módulos quedará reflejada en la arquitectura lógica del proyecto.

La inclusión de un PC en la arquitectura física responde a criterios de compatibilidad, ya que cualquier equipo, bien sea de sobremesa o portátil, con Windows 10/11 puede desempeñar este rol.

### Esquema hardware

Tras presentar la arquitectura física del sistema, se procede a detallar el **conexionado específico de los distintos elementos hardware**. El esquema hardware recoge las señales principales intercambiadas entre sensor de imagen, FPGA y módulo USB.

A diagram of a computer chip

AI-generated content may be incorrect.

Figura 3.5: Esquema hardware.

El esquema de la Figura 3.5 resume el conexionado entre los distintos dispositivos. El sensor de imagen MT9V111 transmite imágenes mediante un bus paralelo de 8 bits sincronizado por las señales PIXCLK, VSYNC y HREF. Por su parte, la comunicación entre FPGA y el módulo FT232H-B se realiza en modo FT245 asíncrono mediante un bus paralelo de 8 bits y las líneas de control RXF#, TXE#, RD# y WR#.

Una vez definido el esquema hardware general del sistema, resulta necesario detallar la asignación de pines de la FPGA. Esta permite materializar el conexionado entre esta y los dispositivos externos. Esta información se recoge en el fichero de restricciones del entorno Vivado (en formato .xdc), y establece la correspondencia entre las señales lógicas descritas en el diseño y los pines físicos del encapsulado de la FPGA en la placa Basys 3.

La asignación de pines definida en dicho fichero de restricciones no es arbitraria, sino que responde a la distribución física de los conectores Pmod de la Basys 3. En este proyecto, las señales de comunicación con el módulo FT232H-B en modo FT245 asíncrono se han enrutado a los conectores Pmod B y Pmod C, mientras que el sensor de imagen MT9V111 se conecta a los Pmod A y XDAC. Esta organización facilita el cableado físico y permite aprovechar la disposición de pines de la propia placa. Para mayor claridad, en la Figura 3.6 se incluye el diagrama de puertos de la placa asociado a cada pin de los Pmod, mostrando su correspondencia con los pines del encapsulado del chip Artix-7.

A blue electronic board with a digital display

AI-generated content may be incorrect.

Figura 3.6: Descripción de pines de entrada/salida de la Digilent Basys 3.

La Tabla 3.1 muestra la relación entre las señales y pines tal como queda descrita en el mencionado archivo de restricciones:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Señal | Pin FPGA | Dirección | Dispositivo | Notas |
| DATA[0] | A14 | INOUT | FT232H-B |  |
| DATA[1] | A15 | INOUT | FT232H-B |  |
| DATA[2] | A16 | INOUT | FT232H-B |  |
| DATA[3] | A17 | INOUT | FT232H-B |  |
| DATA[4] | B15 | INOUT | FT232H-B |  |
| DATA[5] | C15 | INOUT | FT232H-B |  |
| DATA[6] | B16 | INOUT | FT232H-B |  |
| DATA[7] | C16 | INOUT | FT232H-B |  |
| RXF# | K17 | IN | FT232H-B |  |
| TXE# | L17 | IN | FT232H-B |  |
| RD# | M18 | OUT | FT232H-B |  |
| WR# | M19 | OUT | FT232H-B |  |
| SIWU# | N17 | OUT | FT232H-B | Fijado a nivel alto |
| CLKOUT | P17 | IN | FT232H-B | No se usa en el proyecto |
| OE# | P18 | OUT | FT232H-B | Fijado a nivel alto |
| PWRSAV# | R18 | OUT | FT232H-B | Fijado a nivel alto |
| CAMERA[0] | L3 | IN | MT9V111 |  |
| CAMERA[1] | M3 | IN | MT9V111 |  |
| CAMERA[2] | M2 | IN | MT9V111 |  |
| CAMERA[3] | M1 | IN | MT9V111 |  |
| CAMERA[4] | N2 | IN | MT9V111 |  |
| CAMERA[5] | N1 | IN | MT9V111 |  |
| CAMERA[6] | J1 | IN | MT9V111 |  |
| CAMERA[7] | K2 | IN | MT9V111 |  |
| SDA | G2 | OUT | MT9V111 | No se usa en el proyecto |
| SCL | G3 | OUT | MT9V111 | No se usa en el proyecto |
| HREF | J2 | IN | MT9V111 |  |
| VSYNC | H2 | IN | MT9V111 |  |
| XCLK | L2 | OUT | MT9V111 |  |
| PIXCLK | H1 | OUT | MT9V111 |  |
| RST# | J3 | OUT | MT9V111 |  |

Tabla 3.1: Asignación de pines a señales.

Además de las señales externas asociadas al sensor de imagen y al módulo de comunicación, el fichero de restricciones incluye la asignación del reloj principal del sistema, proveniente del oscilador interno de la Basys 3 a 100 MHz. Esta señal, conectada al pin W5 de la FPGA, constituye la referencia temporal que gobierna todo el diseño en el dominio digita. A partir de este reloj base se generan, mediante los recursos internos de gestión del reloj, las señales necesarias para sincronizar los distintos módulos del sistema.

Por último, se muestra en la Figura 3.7 el montaje hardware definitivo, a falta de conectar al PC host.



Figura 3.7: Montaje hardware.

## Arquitectura Lógica

La arquitectura lógica del sistema ViCON describe la **organización funcional de los distintos módulos que conforman la solución,** tanto en la FPGA como en el PC anfitrión. A diferencia de la arquitectura física, que se centra en los dispositivos empleados y su interconexión, la arquitectura lógica se ocupa de cómo se estructuran las funciones de adquisición, transmisión, visualización y procesado de imágenes que permiten dar soporte a los casos de uso previamente definidos.

Para ello, se establecen dos niveles de desarrollo complementarios:

* **Desarrollo hardware en FPGA (VHDL):** Implementa la lógica de adquisición de imágenes desde el sensor de imagen MT9V111, la sincronización de datos, el almacenamiento temporal en FIFO y la transmisión bajo demanda hacia el PC a través del módulo FT232H-B en modo FT245 asíncrono.
* **Desarrollo software en PC:** Implementa la aplicación de usuario, responsable de solicitar los datos a la FPGA, gestionar la recepción y visualización de imágenes en tiempo real y ejecutar los algoritmos de detección y reconocimiento facial sobre los *frames* adquiridos.

De esa manera, la arquitectura lógica proporciona una visión integral del flujo de datos desde la captura de la imagen hasta la visualización y análisis en aplicación, asegurando la trazabilidad entre los requisitos funcionales definidos y la implementación final del sistema.

La Figura 3.8 muestra la arquitectura lógica del sistema ViCON, que integra módulos hardware en la FPGA y módulos software en el PC anfitrión. Cada bloque implementa una función específica en la cadena de adquisición, transmisión y procesado de imágenes, garantizando la trazabilidad respecto a los requisitos y casos de uso definidos previamente.

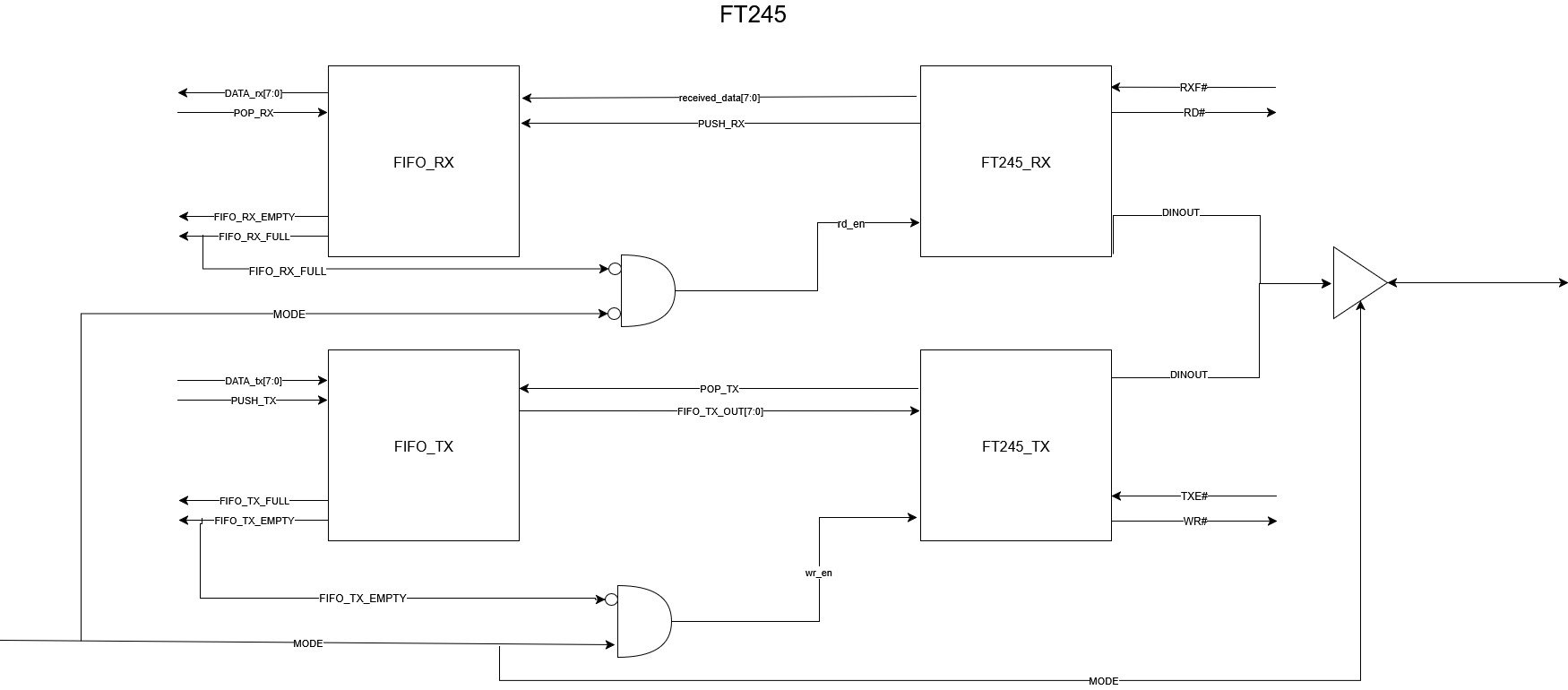


Figura 3.8: Vista general de la arquitectura lógica.

En conjunto, esta arquitectura lógica permite **capturar imágenes en tiempo real** bajo demanda, **transmitirlas** de forma eficiente hacia el PC y **aplicar algoritmos** de visión artificial en un entorno flexible. La descripción detallada de cada bloque se aborda en los apartados siguientes, en los que se analiza su diseño y funcionamiento específico.

### Desarrollo hardware en FPGA (VHDL)

#### Entorno de desarrollo.

El diseño hardware en la FPGA Basys 3 (Digilent, 2017) se desarrolló empleando el entorno **Vivado Design Suite** (AMD, 2025), herramienta oficial de Xilinx para el flujo completo de diseño digital. Vivado permite la **síntesis y simulación** de código, la **implementación física** del diseño en la FPGA y la **generación del *bitstream*** de configuración. Además, proporciona utilidades para la depuración en hardware y la gestión de restricciones, entre otras herramientas.

En la Figura 3.9 se muestra la vista principal del entorno, con el proyecto ViCON abierto y configurado para la **FPGA Artix-7 XC7A35T** (Xilinx, 2021), que constituye el núcleo de la placa Basys 3. En esta ventana se aprecian los distintos paneles que estructuran el flujo de trabajo: el navegador de fuentes, donde se organizan los módulos VHDL; el panel de ejecución, donde se resume el estado de síntesis e implementación; y la consola inferior, donde se pueden lanzar scripts TCL y ver notificaciones de advertencias y errores.

A screenshot of a computer

AI-generated content may be incorrect.

Figura 3.9: Vista principal de Vivado, con el proyecto ViCON abierto.

La elección de Vivado, más allá de ser un requisito obligatorio del proyecto, responde a su compatibilidad directa con la familia Artix-7 de FPGA, en la que se basa la Basys 3, además de ser el entorno recomendado por el fabricante.

#### Arquitectura del módulo FT245

El módulo FT245 constituye la **interfaz de comunicación** entre la FPGA y el módulo FT232H-B (FTDI Chip, 2022), configurado en modo FT245 asíncrono. Su función es permitir la transmisión y recepción de datos mediante un bus paralelo de 8 bits acompañado de las correspondientes señales de control (WR#, RD#, TXE#, RXF#).

Con el fin de garantizar un flujo de datos fiable y desacoplado de los dominios de reloj implicados, la arquitectura del módulo se divide en cuatro componentes principales:

* Submódulo de transmisión (FT245\_TX)
* FIFO de transmisión (FIFO\_TX)
* Submódulo de recepción (FT245\_RX)
* FIFO de recepción (FIFO\_RX)

Esta arquitectura incluye la **lógica de habilitación de los módulos de recepción y transmisión**, así como un buffer triestado que permite gestionar el uso compartido del bus de entrada y salida del módulo, en función de si se requiere recibir o transmitir datos.

La Figura 3.10 muestra la arquitectura general del módulo FT245m con la interconexión de sus diferentes bloques

A diagram of a diagram

AI-generated content may be incorrect.

Figura 3.10: Diagrama de bloques del módulo FT245.

* **Submódulo de recepción**

El bloque FT245\_RX es responsable de **gestionar la lectura de datos** enviados desde el host. Su operación depende de la señal RXF#, que indica si existen datos disponibles en la FIFO interna del FT245. Cuando RXF# se encuentra activa (nivel bajo), el submódulo habilita la señal RD# y coloca el byte recibido en la salida.

El comportamiento del módulo se controla mediante una máquina de estados finita, ilustrada en la Figura 3.11. La FSM comienza en el estado **IDLE**, permaneciendo a la espera de la señal de habilitación de la recepción, rd\_en. Una vez habilitado el módulo y detectada la disponibilidad de datos (RXF# en nivel bajo), el submódulo transita por los estados de lectura (de 10 ns cada uno por estar trabajando con un reloj de 100 MHz), al final de los cuales se captura el dato. Tras ello, se añade un estado propio de esta implementación, en el que se da tiempo a que la memoria FIFO asociada al módulo se actualice. Finalmente, se comprueba si la detección se ha mantenido habilitada, para volver al estado de espera de RXF# o al de IDLE si no lo está.

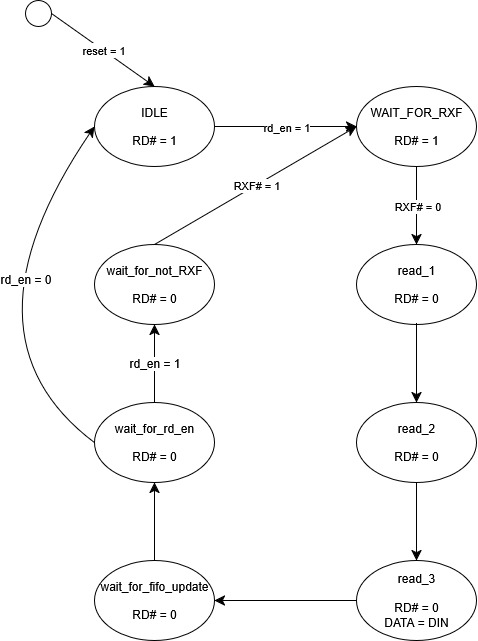


Figura 3.11: Diagrama de estados del módulo FT245\_RX.

De este modo, el submódulo asegura que cada byte recibido se lee de forma síncrona con la FIFO interna del dispositivo y se transfiere al dominio de la FPGA de manera estable.

Un detalle a destacar con respecto a la implementación de este módulo es la necesidad de sincronizar la señal RXF#, proveniente de un dominio de reloj ajeno al de la propia FPGA. Para ello, se ha implementado un sincronizador de dos etapas, que permitirá reducir drásticamente las probabilidades de metaestabilidad en el sistema.

Adicionalmente, la temporización de las señales asociadas a la lectura se corresponde con la definida en la hoja de datos del módulo FT232H (FTDI, 2024), mostrada en la Figura 3.12. En ella se observa cómo la activación de la señal RD# depende de la condición RXF#, garantizando la validez de los datos en la línea tras un retardo máximo especificado. Estos valores temporales están recogidos en la Tabla 3.2, donde se detalla la temporización mínima del pulso de RD#, los retardos de propagación de datos y las condiciones de *setup* y *hold* necesarias.

A diagram of a diagram of a diagram

AI-generated content may be incorrect.

Figura 3.12: Cronograma de lectura del interfaz FT245. Fuente: (FTDI, 2024)

A screenshot of a computer program

AI-generated content may be incorrect.

Tabla 3.2: Temporizaciones del modo FIFO asíncrono del módulo FT232H. Fuente: (FTDI, 2024)

* **FIFO de recepción**

Este módulo de memoria cumple como **buffer intermediario** para almacenar los datos que provienen del host antes de ser procesados por la lógica de la FPGA. Gracias a este mecanismo, es posible absorber posibles diferencias de velocidad y evitar pérdida de información en escenarios de carga variable.

Esto no es de tal relevancia en el caso de la FIFO de recepción, pues el sistema está diseñado de tal forma que solo recibirá un byte por solicitud de imagen, mas sí lo será para la FIFO de transmisión.

* **Submódulo de transmisión**

El bloque FT245\_TX **gestiona el envío de datos** desde la FPGA hacia la FIFO del FT232H.B. Este módulo comprueba el estado de la señal TXE#, que indica si el dispositivo puede aceptar nuevos datos. Cuando TXE# se encentra activa (nivel bajo), se habilita la señal WR# y se coloca un nuevo byte en el bus de datos.

El funcionamiento del submódulo sigue una máquina de estados finita, representada en la Figura 3.13. En ella se observa el **estado inicial IDLE**, en el que el módulo permanece hasta recibir la señal wr\_en (cuando se haya solicitado la escritura y la FIFO de transmisión no se encuentre vacía). Posteriormente, se transita al estado **WAIT\_FOR\_TXE**, en el que se monitoriza la disponibilidad de la FIFO externa. Cuando esta se encuentra preparada, la FSM avanza por una secuencia de estados (**OUTPUT\_DATA** y **WRITE\_1** hasta **WRITE\_4**) que permiten estabilizar los datos y completar completamente el ciclo de escritura. Estos estados tendrán una duración de 10 ns, ya que ese es el periodo del reloj maestro de la FPGA.

En este caso, hay dos detalles a destacar en esta implementación:

* Al igual que en el módulo de recepción, la señal de control TXE# ha sido sincronizada mediante un sincronizador de dos etapas para prevenir casos de metaestabilidad. Esto es porque dicha señal de control proviene de un dominio de reloj distinto al de la FPGA.
* Se ha añadido un **estado extra de espera** (WRITE\_4) para la escritura, con respecto a la implementación temporal mínima del interfaz de transmisión (40 ns respecto a 30 ns). Esto es porque el módulo FT232H-B no sigue perfectamente el estándar FT245 asíncrono en este aspecto temporal. Se hablará en mayor profundidad de esta desviación al tratar la validación del sistema.

El comportamiento temporal de la interfaz de escritura se ajusta a la especificación del protocolo FIFO asíncrono definido por el FT232H, cuyo diagrama se muestra en la Figura 3.14. En él se representan las relaciones de temporización entre las señales TXE#, WR# y los datos en el bus. Los valores mínimos y máximos de dichos parámetros ya fueron recogidos en la Tabla 3.2, presentada en el submódulo de recepción.

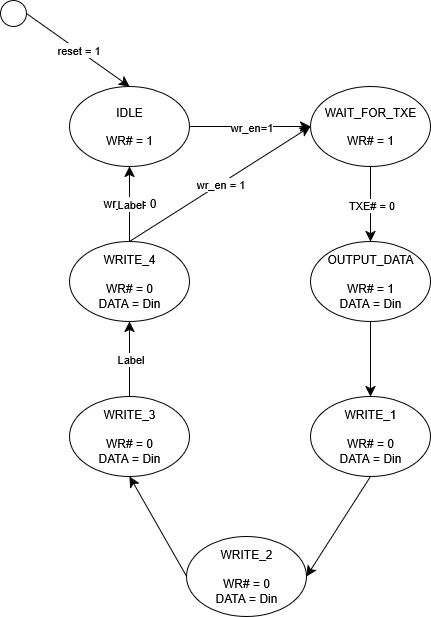


Figura 3.13: Diagrama de estados del módulo FT245\_TX.

A diagram of a diagram

AI-generated content may be incorrect.

Figura 3.14: Cronograma de escritura del interfaz FT245. Fuente: (FTDI, 2024)

De este modo, el submódulo de transmisión asegura que cada dato se escribe en el bus solo cuando el dispositivo lo admite, evitando condiciones de pérdida.

* **FIFO de transmisión.**

De forma **similar a la FIFO de recepción**, la FIFO\_TX actúa como buffer intermediario entre la lógica de control de la FPGA y el submódulo FT245\_TX. Su función es desacoplar la **alta frecuencia de recepción de datos del sensor de imagen** (12-27 MHz) respecto al ritmo más lento e irregular de aceptación de datos. De esta forma, los datos pueden almacenarse temporalmente hasta que TXE# permita su envío.

Para la implementación de la FIFO de transmisión se seleccionó un ancho de bus de 12 bits. Este valor ha sido seleccionado de forma empírica, comprobando que este era el menor valor que permitía una correcta transmisión de los *frames*, es decir, sin obtener una pérdida parcial de información que imposibilite su reconstrucción en el equipo host.

En conclusión, esta organización modular del bloque FT245 proporciona un sistema robusto y escalable. Esta estructura garantiza un intercambio de datos seguro con el host, tanto en la recepción como en la transmisión.

#### Módulo de captura de imágenes

El módulo del sensor de imagen implementa la lógica necesaria para **capturar imágenes procedentes del sensor MT9V111** y entregarlas al sistema en forma de flujo de datos de 8 bits. A diferencia del bloque FT245, este módulo se organiza en una única unidad funcional, gobernada por una máquina de estados finita que secuencia el proceso de adquisición en función de las señales de control VSYNC, HREF y PIXCLK, así como la petición externa de imagen (IMAGE\_REQUEST).

Uno de los aspectos de relevancia de este sensor es que permite enviar imágenes en distintos formatos, entre ellos BGR y YUV, en función de la configuración deseada. Como se ha comentado anteriormente, no se hará uso de la interfaz de comunicación del módulo para su configuración. Es por ello que se mantendrá el **formato por defecto, en este caso YUV** (concretamente en su **versión UYVY**). Este formato permite enviar información sobre la imagen por parejas de píxeles, tal que a cada píxel le corresponde un valor de luminancia (es decir, intensidad de luz) independiente, pero compartiendo los valores relativos al color cada dos píxeles.

Partiendo de esta información, el bloque en la FPGA recibe como entradas el bus de datos, junto con las señales de sincronización VSYNC y HREF, y el reloj de píxel PIXCLK. Desde la FPGA se suministran además las señales de reset y de petición de imagen. Como salidas, el módulo entrega el dato capturado en un bus y un *flag* de validez del dato para almacenar en la FIFO de transmisión. Además, se entrega una señal de final de captura (FRAME\_END), que indica al sistema que se ha terminado de capturar la imagen.

Para esta implementación, se ha determinado finalmente **no hacer uso de imágenes en color**, a pesar de que el sensor lo permite, ya que la transmisión de la información de color duplica la cantidad de información que se debe transmitir a través del módulo FT245. Como se verá más adelante, esto supone no cumplir con el mínimo de imágenes por segundo requerido por limitaciones en el *throughput* del módulo transmisor, con lo cual se ha sacrificado un requisito deseable por mantener una tasa de refresco que permita una visualización fluida.

El funcionamiento interno está regido por la ya mencionada máquina de estados finita, representada en la Figura 3.15, en la que se diferencian los siguientes estados principales:

* **IDLE:** Estado de reposo en el que el sistema permanece hasta que se recibe una petición de imagen.
* **WAIT\_FOR\_IMAGE:** Transición inicial que valida la sincronización con la señal FRAME\_VALID (que resulta de la sincronización de VSYNC).
* **WAIT\_FOR\_FRAME:** Espera al inicio de una nueva imagen, delimitada por la señal VSYNC.
* **WAIT\_FOR\_LINE:** El sistema se prepara para la llegada de una nueva línea válida de píxeles (HREF = 1, en su versión sincronizada).
* **WAIT\_FOR\_SKIP:** Estado de sincronización para descartar los valores de crominancia de los distintos píxeles
* **WAIT\_FOR\_DATA:** En cada flanco de subida, se habilita la captura de un byte de luminancia del bus de datos.
* **SKIP\_STOP y SEND\_STOP:** Estados terminales que cierran la transmisión o el descarte de datos de cada byte, para dar paso al siguiente si lo hubiera.

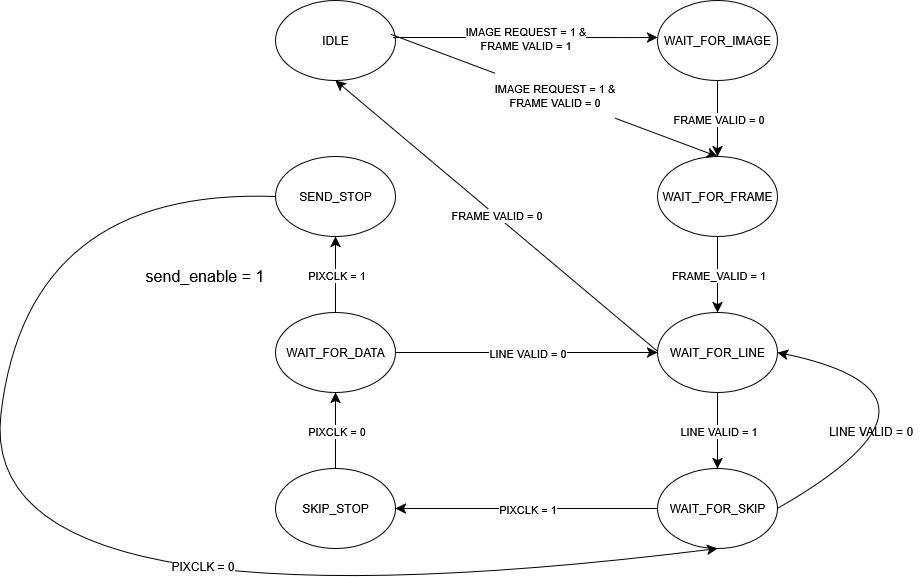


Figura 3.15: Diagrama de estados del módulo de captura de imagen.

Esta organización garantiza que los datos solo se capturen cuando existe una petición activa y las señales de control del sensor validen la información transmitida, evitando lecturas espurias fuera de los intervalos útiles.

Por otra parte, y dado que **el módulo trabaja en dos dominios de reloj** (el de PIXCLK y el del reloj principal del sistema a 100 MHz), se hace necesario **sincronizar las señales que trabajan en el primero de los dominios** para evitar metaestabilidad. Para ello, como se ha hecho anteriormente, se han incorporado sincronizadores de dos etapas en aquellas señales que cruzan de un dominio a otro (PIXCLK, VSYNC y HREF).

En cuanto a la temporización del sensor, sigue el diagrama definido en la hoja de datos del MT9V111, representado en la Figura 3.16, en el que se muestran las relaciones entre VSYNC, HREF, el bus de datos y el reloj de píxeles. Se puede observar que VSYNC/FRAME\_VALID delimita el inicio y el fin de la imagen, HREF/LINE\_VALID delimita el inicio y el fin de la línea, y PIXCLK marca el momento de captura de los datos de acuerdo con su flanco de subida.

A close-up of a diagram

AI-generated content may be incorrect.

Figura 3.16: Diagrama temporal del sensor MT9V111. Fuente: (Micron, 2004)

En conclusión, este diseño garantiza la correcta integración con la lógica de control de la FPGA y con la interfaz FT245, cumpliendo los requisitos de adquisición de imágenes en tiempo real

#### Módulo de control.

El bloque de la máquina de estados de control tiene como objetivo **coordinar la adquisición de imágenes desde el sensor de imagen y la transmisión hacia el host** a través de la interfaz FT245. Su funcionamiento se basa en una secuencia de estados bien definida, que asegura que cada petición recibida desde el host se traduzca en la captura y envío de un único *frame* completo.

La Figura 3.17 refleja estos estados:

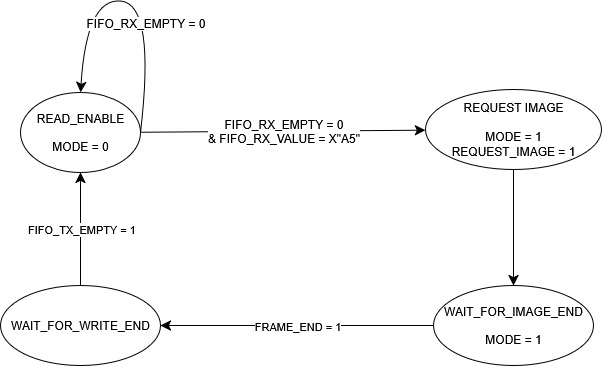


Figura 3.17: Diagrama de estados del módulo de control.

* **READ\_ENABLE:** Estado inicial en el que la FSM permanece a la espera de datos en la FIFO de recepción. En este estado se monitoriza el contenido de la FIFO para detectar comandos enviados por el host.
* **REQUEST\_IMAGE:** Cuando se detecta el comando de petición de imagen (valor 0xA5 en la FIFO de recepción), la FSM habilita la captura, indicando al módulo del sensor de imagen que debe iniciar la adquisición de una nueva imagen.
* **WAIT\_FOR\_IMAGE\_END:** En este estado, la FSM mantiene activo el modo de transmisión mientras se produce la captura y el envío de la imagen. El sistema permanece en este estado hasta que recibe la señal FRAME\_END proveniente del módulo del sensor, que marca la finalización de la imagen actual
* **WAIT\_FOR\_WRITE\_END:** Una vez concluido el *frame*, la FSM espera a que la FIFO de transmisión esté vacía, asegurando que todos los bytes de la imagen han sido enviados correctamente al host.

El funcionamiento de la FSM garantiza que el sistema opere bajo una política de captura bajo demanda, iniciada exclusivamente por comandos enviados desde el host. De este modo, se evita el envío continuo de datos y se optimiza el uso de la interfaz USB.

#### Utilización de recursos

Una vez sintetizado el diseño en la FPGA Xilinx Artix-7 integrada en la Basys3, se ha generado el informe de utilización de recursos. La Tabla 3.3 muestra el número de elementos empleados en relación con la capacidad total de la FPGA, mientras que la Figura 3.18 representa de forma gráfica el porcentaje de utilización de cada recurso principal.

A screenshot of a white sheet with numbers

AI-generated content may be incorrect.

Tabla 3.3: Utilización de recursos de la FPGA Xilinx Artix-7.

A graph with a green bar

AI-generated content may be incorrect.

Figura 3.18: Gráfica de utilización de recursos en la FPGA Xilinx Artix-7.

Del análisis de los resultados se desprende que el diseño ocupa únicamente un 5.34% de las LUTs y un 50% de los FF disponibles, lo que evidencia una baja demanda de lógica combinacional y secuencial respecto a la capacidad total del dispositivo. Asimismo, el consumo de memoria BRAM (2%) es reducido y se emplea únicamente un MMCM de los 5 disponibles, reflejando un uso moderado de los recursos de reloj.

El aspecto más significativo es la utilización de pines de entrada/salida (IOs), que alcanza un 56.60%. Este valor es coherente con la naturaleza del sistema, caracterizado por la interconexión intensiva con el sensor de imagen MT9V111 y el módulo de comunicaciones FT232H-B.

En conjunto, la síntesis confirma que el diseño es eficiente y escalable, dejando un amplio margen de recursos libres para la incorporación de módulos adicionales o futuras mejoras.

### Desarrollo software en C++

#### Entorno de desarrollo y librerías software.

El desarrollo de software para la aplicación del host se ha llevado a cabo usando el ***framework* QT** (QT Group, 2025), un entorno de programación en C++ ampliamente utilizado para la creación de interfaces gráficas de usuario y aplicaciones interactivas multiplataforma. QT proporciona un modelo de programación basado en señales y *slots*, que facilita la gestión de eventos y la comunicación entre componentes de la aplicación. Gracias a esta característica, se logra una interacción eficiente entre los elementos de control de la interfaz (botones, menús, áreas de visualización) y la lógica interna del programa.

Además de la gestión de la interfaz gráfica, QT ofrece soporte nativo para operaciones de entrada/salida, manejo de hilos y temporizadores, funcionalidades que resultan especialmente útiles en el desarrollo de aplicaciones en tiempo real como la presente. En se muestra el entorno de desarrollo QT Creator con el proyecto ViCON abierto.

A screen shot of a computer

AI-generated content may be incorrect.

Figura 3.19: Interfaz de QT Creator con el proyecto ViCON abierto.

Junto con QT, se empleó la **librería OpenCV**, estándar de facto en aplicaciones de visión por computador. OpenCV (OpenCV, 2025) proporciona un amplio conjunto de algoritmos optimizados para procesado de imágenes y vídeo, incluyendo detección y reconocimiento de rostros.

En el contexto del proyecto, OpenCV se utiliza para implementar de forma directa los algoritmos de detección de rostros mediante clasificadores en cascada y de reconocimiento facial a través de descriptores previamente entrenados. Su uso garantiza una implementación fiable, con un tiempo de desarrollo reducido, y permite validad experimentalmente la viabilidad del sistema sin desarrollar algoritmos desde cero.

Por otra parte, también se hace uso de la ya mencionada **librería D2XX de FTDI**. Esta se usa como interfaz para comunicarse con el dispositivo UM232H-B, tal que sea posible establecer una conexión con él, configurarlo y usarlo para comunicarse con la FPGA.

La combinación de QT como entorno de desarrollo para la interfaz y la lógica de control, junto con OpenCV para el procesado de imágenes y D2XX para la comunicación, proporcionan una plataforma flexible y fácilmente extensible que se ajusta a los requisitos de este TFM.

#### Arquitectura general de la aplicación software

La aplicación desarrollada en el host se concibe bajo un **enfoque modular**, de manera que las distintas funciones del sistema se encuentran claramente separadas y pueden evolucionar de forma independiente. La arquitectura distingue **tres componentes principales**: el módulo de comunicación, responsable del intercambio de datos con la FPGA mediante la librería D2XX; el módulo de procesado, que integra los algoritmos de visión artificial e implementados con OpenCV; y la interfaz gráfica de usuario, desarrollada con QT, que actúa como capa de interacción con el usuario final.

Esta organización asegura un bajo acoplamiento y alta cohesión entre bloques, lo que facilita tanto la depuración como la extensibilidad futura del sistema. Asimismo, refleja la lógica de los casos de uso definidos previamente: desde la solicitud de imagen al dispositivo, pasando por la ejecución de algoritmos de detección y reconocimiento facial, hasta la visualización de resultados en tiempo real.

En la Figura 3.20 se muestra el diagrama de bloques que resume el flujo de ejecución de la aplicación, estructurado en tres componentes bien diferenciados: interfaz gráfica (verde), comunicación con el dispositivo (azul) y algoritmia de visión (morado). El diagrama representa tanto el flujo de datos como el flujo de control y visualización, reflejando los casos de uso definidos: visualización, detección y reconocimiento.

La aplicación se apoya en un **esquema concurrente con dos hilos de ejecución en paralelo**. Esta separación resulta esencial para mantener la independencia entre las tareas de adquisición y procesado de datos y las funciones propias de la interfaz gráfica (visualización y gestión de eventos). De este modo, la captura y transmisión de imágenes se realizan sin bloquear la interacción con el usuario, garantizando un rendimiento estable en tiempo real y una experiencia de uso fluida.

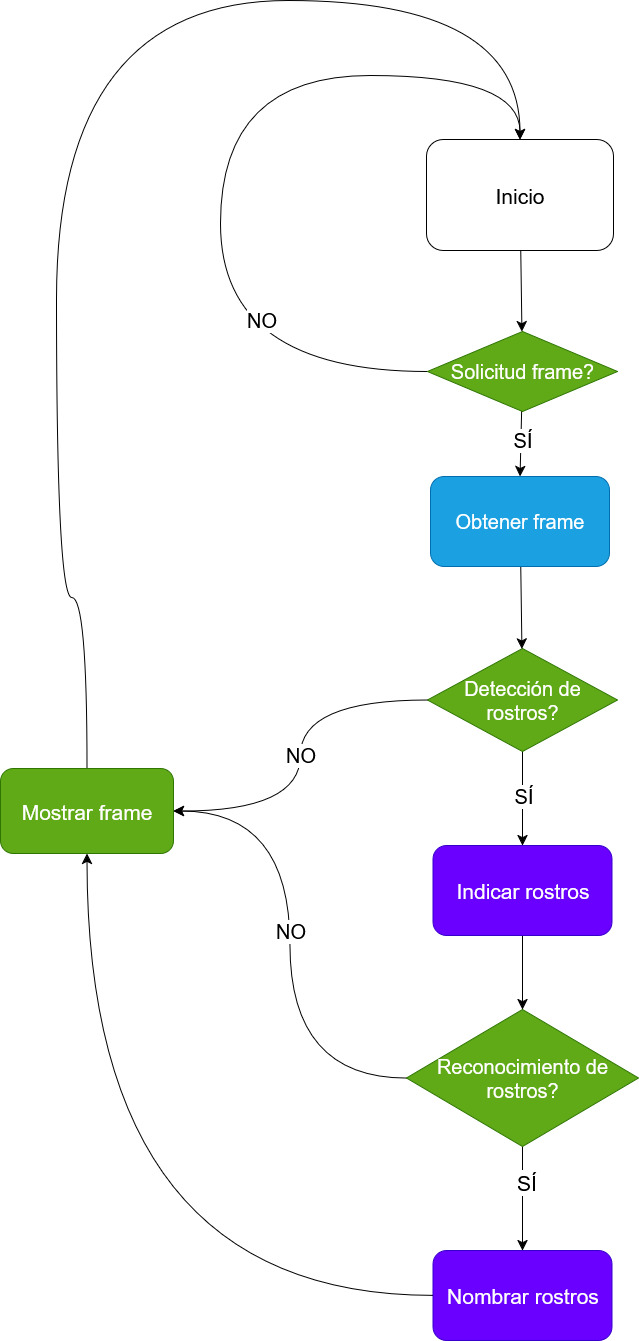


Figura 3.20: Diagrama de flujo de la aplicación software.

1. **Inicio y solicitud de *frames*.**

Tras iniciar la aplicación, el usuario puede activar la adquisición mediante el control de inicio. Esta acción habilita la decisión sobre la solicitud de *frame*, y en caso afirmativo se emite un comando de petición al dispositivo.

1. **Obtención del *frame***.

El módulo de comunicación, basado en la librería D2XX, lee el flujo de bytes devuelto por el FT245, comprobando su completitud y reconstruyendo una imagen VGA en escala de frises de 8 bits. En caso de error o *timeout*, el flujo regresa a la decisión inicial para permitir reintentos controlados

1. **Ejecución opcional de algoritmia**.

Una vez disponible el *frame*, el sistema evalúa si está activa la detección de rostros. Si lo está, se aplica el detector y los resultados se muestran solapados sobre la imagen. Si, además, está habilitado el reconocimiento facial, se clasifican las regiones detectadas y se les añaden etiquetas de identidad (indicando “desconocido” si no es posible reconocer un rostro).

1. **Visualización y lazo de tiempo real**.

El *frame*, con o sin anotaciones, se muestra en la interfaz. Desde este punto, se puede detener la adquisición o continuar con el bucle solicitando un nuevo *frame*, manteniendo la cadencia de tiempo real.

Esta organización garantiza una arquitectura modular, en la que cada componente mantiene una independencia modular clara respecto a los demás. Gracias a esta separación, el sistema resulta más sencillo de depurar, mantener y ampliar con nuevas funcionalidades en el futuro.

Por último, la Figura 3.21 muestra la interfaz gráfica desarrollada con QT para la aplicación. En la parte superior se sitúan los controles de usuario, que permiten iniciar o detener la captura de vídeo, así como habilitar de forma independiente la detección y el reconocimiento facial. Debajo se incluye un indicador de estado del sistema, que informa en todo momento de la operación activa y posibles errores. La zona central está destinada a la visualización de los *frames* recibidos desde la FPGA y sobre la que se superponen, cuando están activos, los resultados de los algoritmos de visión (rectángulos y etiquetas de rostros. Finalmente, en la parte inferior derecha, se muestra el indicador de frecuencia de refresco (FPS), utilizado para evaluar el rendimiento en tiempo real.

A screen shot of a video

AI-generated content may be incorrect.

Figura 3.21: Captura de pantalla de la interfaz gráfica de la aplicación.

#### Bloque de procesado: detección y reconocimiento facial

El bloque de procesado constituye el núcleo de la aplicación software, ya que es el encargado de aplicar los algoritmos de visión artificial sobre los *frames* recibidos. Su funcionamiento se organiza de forma secuencial y opcional:

1. **Recepción del *frame*:** Una vez que el módulo de comunicación reconstruye la imagen VGA en escala de grises, esta se entrega al bloque de procesado.
2. **Detección de rostros:** Se ejecuta un algoritmo basado en clasificadores de características locales (en este caso, Haar Cascades de OpenCV), encargado de localizar regiones candidatas a rostros. El resultado se transmite en forma de coordenadas rectangulares que delimitan las áreas detectadas.
3. **Reconocimiento facial (opcional):** Sobre las regiones detectadas se aplica un modelo de reconocimiento, en este caso un clasificador LBPH[[1]](#footnote-1) (*Local Binary Patterns Histograms*). Este módulo compara cada rostro con la base de datos previamente registrada, asignando una etiqueta de identidad o indicando el valor desconocido en caso de no existir coincidencias suficientes.

Conviene destacar que la funcionalidad de reconocimiento facial constituye una **extensión opcional de la funcionalidad básica** del sistema. De acuerdo con los requisitos definidos, el sistema debería garantizar únicamente la adquisición, transmisión y detección de rostros en tiempo real, mientras que el reconocimiento se ha implementado como una ampliación experimental que demuestra la flexibilidad y potencial de la plataforma.

El *frame* resultante, con anotaciones gráficas de detección y reconocimiento en caso de estar activados, es finalmente entregado a la interfaz gráfica para su visualización, tal y como se muestra en la Figura 3.21.

# Resultados

En este capítulo se presentan los **resultados obtenidos tras materializar el desarrollo del proyecto**, en su fase de verificación y validación. El objetivo es comprobar el grado de cumplimiento de los requisitos previamente definidos, y evaluar el comportamiento del sistema en condiciones reales de funcionamiento.

De este modo, el capítulo pretende otorgar una visión integral sobre la correctitud, robustez y adecuación del sistema desarrollado frente a los requisitos iniciales.

## Verificación de requisitos

En primer lugar, se expone la **verificación de requisitos**, donde se detallan las pruebas diseñadas para confirmar que cada especificación funcional y no funcional ha sido correctamente implementada.

### Entorno 1: Simulación TCL en Vivado

La primera fase de pruebas se ha llevado a cabo en el **entorno de simulación de Vivado**, utilizando los bancos de prueba (*testbenches*) desarrollados para cada uno de los módulos VHDL implementados. Este entorno permite validar la lógica de diseño de forma previa a su síntesis e implementación física e la FPGA, reduciendo significativamente el riesgo de errores y facilitando la depuración del sistema.

Durante las simulaciones **se analizaron tanto los módulos individuales** (por ejemplo, los submódulos de recepción y transmisión FT245) **como la interacción entre bloques integrados**. Para cada caso, se observó la respuesta del sistema a una serie de entradas a través de sus señales de salida e intermedias. De esta forma, ha sido posible verificar la temporalidad de las señales, el cumplimiento de los protocolos de comunicación y la correcta evolución de las máquinas de estados definidas.

En conjunto, las simulaciones en Vivado constituyen un paso fundamental para asegurar que la implementación VHDL cumple con los requisitos funcionales básicos antes de proceder a la validación en la FPGA real.

#### Simulación del módulo FT245\_TX

|  |  |  |
| --- | --- | --- |
| **Requisito/s** | | HW2, HW5, HW6, HW7, HW11 |
| **Prueba** | Simulación de operación del módulo de transmisión FT245 | |
| **Salida** | 1. El funcionamiento (puesta de datos en salida y secuencia de señales de control) es el esperado cuando se realiza una operación de transmisión aislada mediante el estímulo de las correspondientes señales de control. 2. El funcionamiento (puesta de datos en salida y secuencia de señales de control) es el esperado cuando se realizan operaciones de transmisión *back-to-back* mediante el estímulo de las correspondientes señales de control. | |
| **Errores** | 1. Ningún error encontrado | |

La Figura 4.1 muestra el comportamiento simulado del módulo FT245\_Tx, donde se puede ver que actúa correctamente ante transmisiones aisladas, transmisiones consecutivas y cuando el mismo se resetea.

A screenshot of a computer

AI-generated content may be incorrect.

Figura 4.1: Captura de la simulación del módulo FT245\_TX.

#### Simulación del módulo FIFO

|  |  |  |
| --- | --- | --- |
| **Requisito/s** | | HW2, HW5, HW6, HW12 |
| **Prueba** | Simulación de operación del módulo de memoria FIFO | |
| **Salida** | 1. El módulo opera adecuadamente al almacenar datos nuevos. 2. El módulo opera adecuadamente al extraer datos de la memoria. 3. El módulo se comporta adecuadamente cuando se solicitan datos y la memoria está vacía. 4. El módulo se comporta adecuadamente cuando se trata de almacenar datos y la memoria está llena. | |
| **Errores** | 1. Ningún error encontrado | |

La Figura 4.2 muestra el comportamiento en simulación de la memoria FIFO. Se observa que, para una memoria de ancho de bus de 2 bits, esta se llena con la llegada de 4 valores y no se pueden agregar más. La memoria, a continuación, se vacía completamente y puede volver a llenarse con datos nuevos sin problema. Además, el comportamiento al resetear es el esperado.

A screenshot of a computer

AI-generated content may be incorrect.

Figura 4.2: Captura de la simulación del módulo FIFO.

#### Simulación de módulo FT245\_RX

|  |  |  |
| --- | --- | --- |
| **Requisito/s** | | HW2, HW5, HW6, HW9, HW11 |
| **Prueba** | Simulación de operación del módulo de recepción FT245 | |
| **Salida** | 1. El funcionamiento (recepción de un único dato) es el esperado cuando se realiza la operación de recepción de un único byte aislado mediante el estímulo de las correspondientes señales de control criterios temporales adecuados. 2. El funcionamiento (recepción de varios datos seguidos) es el esperado cuando se realiza la operación de recepción de varios bytes consecutivos mediante el estímulo de las correspondientes señales de control con los criterios temporales adecuados. | |
| **Errores** | 1. Ningún error encontrado | |

La Figura 4.3 muestra el comportamiento del módulo FT245\_Rx en simulación, donde se puede observar que funciona adecuadamente en condiciones de recepción de datos aislada, de recepción de datos consecutiva y ante un reset.

A screenshot of a computer

AI-generated content may be incorrect.

Figura 4.3: Captura de la simulación del módulo FT245\_Rx

#### Simulación de integración módulo FT245\_TX y FIFO

|  |  |  |
| --- | --- | --- |
| **Requisito/s** | | HW2, HW5, HW6, HW7, HW11, HW12 |
| **Prueba** | Simulación de integración del módulo de transmisión FT245 con el módulo de memoria FIFO. | |
| **Salida** | 1. Se pueden introducir datos en la FIFO de recepción hasta su límite, y la presencia de los mismos (junto con la habilitación del módulo de escritura) inicia la transmisión de datos a través del interfaz FT245. | |
| **Errores** | 1. No se encuentran errores. | |

La Figura 4.4 muestra el comportamiento cuando se integra el módulo FT245\_Tx a su memoria FIFO. Se puede comprobar cómo, al insertar datos en memoria, esta de forma autónoma permite que se envíen los mismos hasta que la memoria se vacía.

A screenshot of a computer

AI-generated content may be incorrect.

Figura 4.4: Captura de simulación de integración FIFO-FT245\_Tx.

#### Simulación de integración Módulo FT245\_RX y FIFO

|  |  |  |
| --- | --- | --- |
| **Requisito/s** | | HW2, HW5, HW6, HW7, HW11, HW12 |
| **Prueba** | Simulación de integración del módulo de recepción FT245 con el módulo de memoria FIFO. | |
| **Salida** | 1. Los datos recibidos son almacenados adecuadamente en la FIFO de recepción, y estos pueden extraerse de la misma sin problema alguno. | |
| **Errores** | 1. No se encuentran errores. | |

La Figura 4.5 muestra el comportamiento análogo al apartado anterior, pero para el módulo FT245\_Rx. En este caso, es el módulo FT245\_Rx el que, al recibir un dato, lo introduce en la memoria FIFO de forma automática, tal que si esta se llena no se permitirá recibir más datos.

A screenshot of a computer

AI-generated content may be incorrect.

Figura 4.5: Captura de simulación de integración FIFO-FT245\_Rx.

#### Simulación del módulo de captura de datos del sensor.

|  |  |  |
| --- | --- | --- |
| **Requisito/s** | | HW2, HW5, HW6, HW8, HW9, HW10 |
| **Prueba** | Simulación de operación del módulo del sensor de imagen en la FPGA | |
| **Salida** | 1. El módulo de captura es capaz de capturar una imagen completa, incluyendo todas sus líneas, tomando únicamente los píxeles de luminancia enviados en el formato UYVY. | |
| **Errores** | 1. No se encuentran errores | |

La Figura 4.6 muestra la simulación del módulo de captura de datos del sensor de imagen. Se simula una imagen de dos filas y dos píxeles por fila, tal que se pueda comprobar el comportamiento cuando se solicita la imagen antes de que la envíe el sensor, justo cuando la envía el sensor, y cuando ya hay un envío en curso (en este caso, esperando a la siguiente imagen).

A screenshot of a computer

AI-generated content may be incorrect.

Figura 4.6: Captura de simulación del módulo de captura de datos del sensor.

#### Simulación del módulo de control del sistema de la FPGA.

|  |  |  |
| --- | --- | --- |
| **Requisito/s** | | HW2, HW5, HW6, HW8, HW9, HW10 |
| **Prueba** | Simulación de operación del módulo de control del sistema de la FPGA | |
| **Salida** | 1. El módulo de control del sistema es capaz de pasar por todas las etapas de solicitud, captura y envío de imágenes del sistema bajo las condiciones adecuadas. | |
| **Errores** | 1. Ningún error encontrado. | |

La Figura 4.7 muestra el comportamiento de la máquina de estados que conforma el módulo de control de la FPGA. Se puede observar cómo transita por los estados descritos en la Figura 3.17, generando las salidas que le permiten controlar recepción, transmisión e inicio de captura de imagen.

A screenshot of a computer

AI-generated content may be incorrect.

Figura 4.7: Captura de simulación del módulo de control del sistema FPGA.

### Entorno de pruebas de comunicación preliminar en HOST.

Antes de integrar la lógica de comunicación en la aplicación completa, se definió un **entorno preliminar de pruebas** con el objetivo de verificar de manera independiente la correcta operación de la librería D2XX en el equipo host y su interacción con el módulo FT245 en la FPGA.

Para ello, se desarrolló un conjunto de scripts en C++, diseñados específicamente para ejercitar las funciones básicas de la librería en un entorno controlado y con la mínima complejidad. Estos programas, ejecutados en un equipo con sistema operativo Windows 10/11, permitieron detectar y abrir el dispositivo FT245 y usarlo para realizar operaciones de escritura y lectura con la FPGA.

Esto proporcionó una base sólida para **verificar que la capa de comunicación era completamente funcional** antes de proceder a su integración con la aplicación, y aislar potenciales problemas relacionados con la misma y con la librería D2XX.

#### Test de recepción de datos en FPGA

|  |  |  |
| --- | --- | --- |
| **Requisito/s** | | SW3, SW5, SW20, SW24, SW25 |
| **Prueba** | Enviar datos incrementales desde el host hacia la FPGA con periodo 1 segundo a través del interfaz FT245 y mostrarlos en el *display* 7 segmentos. | |
| **Salida** | 1. El *display* 7 segmentos mostraba los datos adecuados con periodo de 1 segundo. | |
| **Errores** | 1. Ningún error encontrado | |

#### Test de throughput de transmisión en FPGA

|  |  |  |
| --- | --- | --- |
| **Requisito/s** | | SW3, SW5, SW20, SW24, SW25 |
| **Prueba** | Enviar datos incrementales desde la FPGA hacia el host a la máxima velocidad permitida por el enlace. | |
| **Salida** | 1. Los datos recibidos en llegada, en un desarrollo inicial, no eran los esperados (se repetía el mismo valor dos veces y luego incrementaba en dos unidades). Se puede observar el comportamiento en la Figura 4.8, donde se ve que solo la mitad de los datos recibidos coinciden con lo esperado. 2. El *throughput* mostrado por el dispositivo es de 10 MB/s | |
| **Errores** | 1. La temporización del chip FT232H-B no cumple con exactitud las especificaciones del protocolo FT245. | |

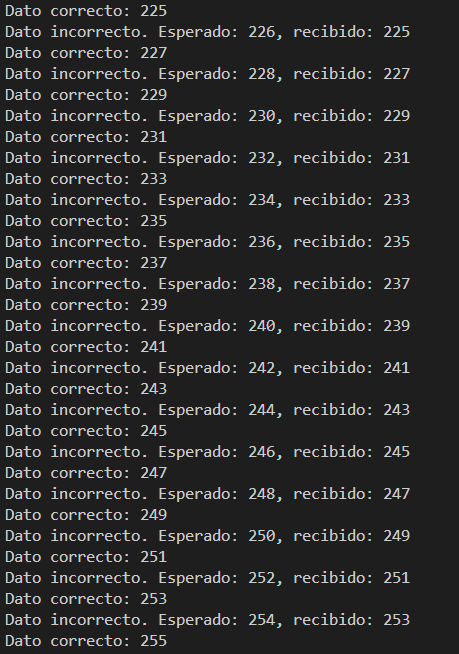


Figura 4.8: Error encontrado en el test de transmisión en FPGA.

#### Test de recepción y transmisión en bucle en FPGA.

|  |  |  |
| --- | --- | --- |
| **Requisito/s** | | SW3, SW5, SW20, SW24, SW25 |
| **Prueba** | Enviar una serie de datos a la FPGA y comprobar que al recibirlos en el host son los mismos. | |
| **Salida** | 1. Los datos se pueden enviar sin problema alguno 2. El dato recibido en el host se corresponde con el enviado. | |
| **Errores** | 1. Ningún error encontrado. | |

### Pruebas sobre la aplicación software final.

La fase final de validación se llevó a cabo **ejecutando la aplicación completa desarrollada en QT sobre un equipo host** con sistema operativo Windows 10, conectado al dispositivo hardware a través del módulo FT245. Este entorno reproduce las condiciones de funcionamiento previstas para el sistema, permitiendo verificar de forma conjunta tanto los módulos hardware en la FPGA como los componentes software en el host.

El procedimiento de prueba consistió en la **interacción directa del usuario con la interfaz gráfica de la aplicación**, empleando los controles disponibles para iniciar/detener la adquisición de vídeo, habilitar la detección y el reconocimiento facial. De este modo, se ha comprobado el cumplimiento de los requisitos software definidos previamente, comprobando aspectos como la correcta visualización de los *frames*, el funcionamiento de los botones de control, la correcta ejecución de los algoritmos de detección y reconocimiento facial y la actualización de los indicadores de estado.

Este entorno permitió además **observar el comportamiento global del sistema en condiciones de uso prolongado**, identificando posibles problemas de rendimiento, estabilidad o sincronización entre hardware y software. Los resultados obtenidos confirmaron la correcta integración de todos los componentes y validaron el sistema frente a los requisitos funcionales y no funcionales del software.

#### Test de visualización de imagen

|  |  |  |
| --- | --- | --- |
| **Requisito/s** | | SW1, SW2, SW3, SW4, SW5, SW6, SW7, SW8, SW9, SW10, SW11, SW20, SW21, SW22, SW23 |
| **Prueba** | Iniciar la aplicación software, iniciar la recepción de imágenes y pararla. | |
| **Salida** | 1. Las imágenes se pueden visualizar correctamente y a una frecuencia superior a 10 fps. 2. Al parar, se dejan de visualizar imágenes y queda un indicador de fondo que indica que no hay imagen. | |
| **Errores** | 1. No ha sido posible cumplir con el requisito SW12, ya que no se pueden cumplir los requisitos obligatorios y entregar imágenes en color al mismo tiempo. | |

#### Test de detección de rostros

|  |  |  |
| --- | --- | --- |
| **Requisito/s** | | SW1, SW2, SW3, SW4, SW5, SW6, SW7, SW8, SW9, SW10, SW11, SW13, SW14, SW15, SW16, SW17, SW20, SW21, SW22, SW23 |
| **Prueba** | Iniciar la aplicación software, iniciar la recepción de imágenes y la detección de rostros, y parar esta última. | |
| **Salida** | 1. Los rostros se recuadran en verde cada vez que son detectados. 2. Al parar, se dejan de visualizar recuadros verdes frente a los rostros. | |
| **Errores** | 1. Ningún error encontrado. | |

#### Test de reconocimiento de rostros

|  |  |  |
| --- | --- | --- |
| **Requisito/s** | | SW1, SW2, SW3, SW4, SW5, SW6, SW7, SW8, SW9, SW10, SW11, SW13, SW14, SW15, SW16, SW17, SW18, SW19, SW20, SW21, SW22, SW23 |
| **Prueba** | Iniciar la aplicación software, iniciar la recepción de imágenes, la detección de rostros y el reconocimiento de los mismos, y parar este último. | |
| **Salida** | 1. Los rostros recuadrados son etiquetados adecuadamente, en función de si el algoritmo los reconoce o no. 2. Al parar, se dejan de visualizar etiquetas en los rostros detectados. | |
| **Errores** | 1. Ningún error encontrado. | |

## Resultados de las pruebas del sistema

Durante la fase de validación se han identificado dos incidencias relevantes en el comportamiento del sistema, relacionados con las limitaciones del hardware empleado y la gestión de recursos de la FPGA.

### Temporización en el test de *throughput* del FT245.

En las pruebas de transferencia continua de datos se observó que **el módulo FT232H-B usado no cumple estrictamente con las especificaciones temporales definidas por el protocolo FT245 asíncrono**. En particular, la señal WR# requería un tiempo de mantenimiento superior al indicado en el diagrama de temporización. Para solventar esta limitación, **se modificó la lógica de control del módulo FT245\_TX** de la FPGA para garantizar **10 ns extra** para el mantenimiento a nivel bajo de esta señal (correspondiente al periodo t10 en el diagrama). Esta corrección permitió **estabilizar las transferencias sin pérdida de datos**, garantizando la robustez de la comunicación a costa de una ligera relajación en la temporización.

A table with text and numbers

AI-generated content may be incorrect.

A diagram of a diagram

AI-generated content may be incorrect.

### Imposibilidad de transmitir imágenes en color a 10 FPS

El segundo problema está relacionado con el requisito SW12 (transmisión de imágenes en color), definido como deseable pero obligatorio. Durante las pruebas de integración se constató que la FPGA no dispone de recursos suficientes para almacenar en la FIFO de transmisión la cantidad de datos necesaria cuando se trabaja en **formato de imagen en color** (2 bytes por píxel). Como consecuencia, **la imagen no se podía representar**, y había que reducir la frecuencia del reloj del sensor de imagen (al mínimo, 12 MHz) para poder obtener imágenes en color (a un máximo de 7.5 FPS)

Por este motivo, la aplicación se limitó a **operar en escala de grises**, en el cual sí es posible mantener el umbral de rendimiento establecido. En este caso, el sensor trabajará con una **señal de reloj de 24 MHz**, y las imágenes se reciben con una frecuencia que queda entre los 10 FPS y los 15 FPS.

En conjunto, estas pruebas demuestran que, pese a las limitaciones impuestas por el hardware disponible, **el sistema cumple con todos los requisitos fundamentales definidos en el proyecto.** Los ajustes aplicados en la temporización de la interfaz FT245 aseguran la fiabilidad de la comunicación, mientras que la renuncia al requisito deseable de imágenes en color garantiza la viabilidad práctica del sistema en tiempo real.

## Plan de validación

|  |  |
| --- | --- |
| **Nombre del entregable** | **Pruebas asociadas para su validación** |
| *Bitstream* para Basys 3 | * + - Vis1 |
| Ejecutable autocontenido de la aplicación software | * + - Vis1     - Det1     - Det2     - Rec1     - Rec2 |

### Pruebas del plan de validación

|  |  |
| --- | --- |
| **Visualización de imágenes** | **Vis1** |
| **Descripción:**  El objetivo de la prueba es comprobar que la aplicación puede solicitar imágenes a demanda a la FPGA, recibirlas y representarlas adecuadamente a la frecuencia mínima de 10 FPS. | |
| **Prerrequisitos**  1. Aplicación software compilada y ejecutable creado.  2. *Bitstream* de FPGA generado y cargado en la FPGA  3. Sensor de imagen conectado a la FPGA  4. FT232H-B conectado a la FPGA y al host. | |
| **Pasos:**  1. Ejecutar la aplicación en el host.  2. Iniciar la representación de imágenes.  3. Parar la representación de imágenes. | |
| **Resultado esperado:**  **Las imágenes se pueden representar cuando está activa dicha funcionalidad, y dejan de hacerlo cuando no está activa.** | |
| **Resultado obtenido:**  **Las imágenes se muestran cuando está activa la representación.** A screen shot of a video  AI-generated content may be incorrect.  Figura 4.9: Muestra de imágenes activas en Vis1.  **Las imágenes no están activas cuando desactiva.**    Figura 4.10: Muestra de imágenes no activas en Vis1. | |

|  |  |
| --- | --- |
| **Detección activa sin rostros** | **Det1** |
| **Descripción:**  La aplicación debe funcionar adecuadamente cuando se activa la detección de rostros sin que haya rostros en la imagen. La detección de rostros debe poder pararse sin problema. | |
| **Prerrequisitos**  1. Aplicación software compilada y ejecutada.  2. *Bitstream* de FPGA generado y cargado en la FPGA  3. Sensor de imagen conectado a la FPGA  4. FT232H-B conectado a la FPGA y al host. | |
| **Pasos:**  1. Iniciar la detección de rostros  2. Parar la detección de rostros | |
| **Resultado esperado:**  **Las imágenes se representan normalmente en ambos casos** | |
| **Resultado obtenido:**  **Las imágenes se muestran cuando está activa la detección.**    Figura 4.11: Muestra de imágenes activas en Det1.  **Las imágenes siguen activas cuando no está activa la detección.**    Figura 4.12: Muestra de imágenes no activas en Det1. | |

|  |  |
| --- | --- |
| **Detección activa con rostros** | **Det2** |
| **Descripción:**  La aplicación debe funcionar adecuadamente cuando se activa la detección de rostros cuando hay rostros en la imagen. La detección de rostros debe poder pararse sin problema. | |
| **Prerrequisitos**  1. Aplicación software compilada y ejecutada.  2. *Bitstream* de FPGA generado y cargado en la FPGA  3. Sensor de imagen conectada a la FPGA  4. FT232H-B conectado a la FPGA y al host.  5. Colocar un rostro frente al sensor. | |
| **Pasos:**  1. Iniciar la detección de rostros  2. Quitar el rostro de la imagen | |
| **Resultado esperado:**  **Las imágenes se representan normalmente en ambos casos, y el rostro queda indicado cuando aparece en la imagen.** | |
| **Resultado obtenido:**  **El rostro se recuadra en la imagen.**    Figura 4.13: Muestra de rostro localizado en Det2.  **El recuadro desaparece con el rostro**    Figura 4.14: Muestra de rostro no localizado en Det2. | |

|  |  |
| --- | --- |
| **Reconocimiento activo sin rostros** | **Rec1** |
| **Descripción:**  La aplicación debe funcionar adecuadamente cuando se activa la detección de rostros y su reconocimiento sin que haya rostros en la imagen. El reconocimiento de rostros debe poder pararse sin problema. | |
| **Prerrequisitos**  1. Aplicación software compilada y ejecutada.  2. *Bitstream* de FPGA generado y cargado en la FPGA  3. Sensor de imagen conectada a la FPGA  4. FT232H-B conectado a la FPGA y al host.  5. Iniciar la detección de rostros | |
| **Pasos:**  1. Iniciar el reconocimiento de rostros  2. Parar el reconocimiento de rostros | |
| **Resultado esperado:**  **Las imágenes se representan normalmente en ambos casos** | |
| **Resultado obtenido:**  **Las imágenes se muestran cuando está activo el reconocimiento.**    Figura 4.15: Muestra de imágenes activas en Rec1.  **Las imágenes siguen activas cuando no está activo el reconocimiento.**    Figura 4.16: Muestra de imágenes permaneciendo activas en Rec1. | |

|  |  |
| --- | --- |
| **Reconocimiento activo con rostros** | **Rec2** |
| **Descripción:**  La aplicación debe funcionar adecuadamente cuando se activa la detección de rostros y su reconocimiento con presencia de rostros en la imagen. El reconocimiento de rostros debe poder pararse sin problema. | |
| **Prerrequisitos**  1. Aplicación software compilada y ejecutada.  2. *Bitstream* de FPGA generado y cargado en la FPGA  3. Sensor de imagen conectado a la FPGA  4. FT232H-B conectado a la FPGA y al host.  5. Iniciar la detección de rostros  6. Iniciar el reconocimiento de rostros | |
| **Pasos:**  1. Mostrar un rostro conocido al sensor  2. Mostrar un rostro desconocido al sensor | |
| **Resultado esperado:**  **Las imágenes se representan normalmente en ambos casos** | |
| **Resultado obtenido:**  **El etiquetado del rostro conocido es correcto.**    Figura 4.17: Muestra de rostro conocido en Rec2.  **El rostro se etiqueta como desconocido.**    Figura 4.18: Muestra de rostro desconocido en Rec2. | |

# Conclusiones y trabajo futuro

El desarrollo del sistema ViCON ha permitido implementar una plataforma configurable de adquisición y procesado de imágenes basada en FPGA, orientada a la ejecución de algoritmos de reconocimiento facial en tiempo real. A lo largo del proyecto se han diseñado e integrado los distintos módulos hardware en VHDL, la interfaz de comunicación FT245 y la aplicación software en QT con soporte de OpenCV.

En términos generales, el sistema cumple con los requisitos fundamentales definidos en la fase inicial:

* Captura de imágenes desde el sensor de imagen MT9V11 en resolución VGA y escala de grises.
* Transmisión fiable de datos desde y hacia el host mediante el módulo FT232H-B.
* Visualización en tiempo real a, al menos, 10 FPS y con posibilidad de habilitar algoritmos de detección y reconocimiento facial.
* Interfaz gráfica intuitiva, que permite al usuario controlar de manera sencilla las distintas funcionalidades.

No obstante, durante el proceso de validación se identificaron ciertas limitaciones derivadas del hardware disponible. Tras los ajustes pertinentes, ha sido posible solventar estas limitaciones (relativas a las restricciones de temporización del chip FT232H-B y a las limitaciones de memoria de la Basys 3), y presentar una solución viable que cumple con todos los requisitos obligatorios establecidos para el proyecto.

En conjunto, los resultados obtenidos demuestran la viabilidad del sistema como solución para aplicaciones de visión artificial de bajo coste y como banco de pruebas experimental para algoritmos de visión embebida en tiempo real. La solución ofrecida es a su vez accesible y flexible, pudiéndose emplear en ámbitos académicos, de investigación y en productos comerciales.

Además, el desarrollo de este trabajo ha permitido poner en práctica de forma integrada los conocimientos adquiridos en distintas asignaturas del máster. En particular, se han aplicado los fundamentos de la asignatura **Diseño de Sistemas Empotrados Basados en FPGA**, mediante la implementación en VHDL de los distintos módulos de comunicación, control y captura de imagen los conceptos de **Visión Artificial**, a través del uso de algoritmos de detección y reconocimiento facial con OpenCV**;** y las competencias de **Interfaces de Usuario**, materializadas en el diseño e implementación de una aplicación con interfaz gráfica en QT que permite la interacción del usuario con el sistema. Esta relación directa con el plan formativo evidencia la coherencia entre los objetivos académicos del máster y los resultados alcanzados en el proyecto.

## Trabajo futuro

A partir del desarrollo realizado, existen diversas líneas de mejora y extensión que permitirían aumentar las capacidades del sistema y mejorar sus características de cara a entornos de uso práctico:

* **Soporte para imágenes en color:** Explorar el uso de FPGAs con mayor cantidad de recursos de memoria, o recurrir a módulos externos de memoria, para poder implementar módulos de memoria de mayor tamaño. Así, se podrán transmitir imágenes en color sin sacrificar la tasa de refresco.
* **Optimización de la comunicación:** Evaluar modos de transferencia alternativos al FT245 asíncrono, como su versión síncrona o interfaces de mayor ancho de banda.
* **Mejora de algoritmos de visión:** Integrar módulos de visión más avanzados, basados en redes neuronales para detección y reconocimiento facial. Explorar la posibilidad de realizar una ejecución parcial o total de los mismos en la FPGA o recurriendo a aceleradores hardware.
* **Escalabilidad del sistema:** Permitir la conexión de múltiples cámaras/sensores de imagen o el aumento de la resolución de captura, con el fin de ampliar el rango de aplicaciones posibles en entornos de vigilancia o control de accesos, entre otros.
* **Robustez y portabilidad del software:** Extender la compatibilidad de la aplicación a otros sistemas operativos (Linux, macOS), haciendo uso de las correspondientes librerías para cada uno de estos sistemas operativos.

En conclusión, ViCON constituye un punto de partida sólido para el desarrollo de sistemas embebidos de visión artificial, combinando hardware reconfigurable y software flexible. Las mejoras planteadas en el trabajo futuro permitirán incrementar su rendimiento y adaptabilidad, consolidando su utilidad como plataforma experimental y como base para aplicaciones de visión en tiempo real.

# Referencias

* AMD. (2025). *Vivado Design Suite User and Reference Guides.*
* Digilent. (2017). *Basys 3 FPGA Board Reference Manual.*
* FTDI. (2024). *FT232H Single Channel Hi-Speed USB to Multipurpose UART/FIFO IC.*
* FTDI Chip. (2022). *UM232H-B USB to Setial/Paralel breakout module.*
* FTDI Chip. (2023). *D2XX Programmer's Guide.*
* Micron. (2004). *MT9V111 1/4-Inch SOC VGA CMOS Active-Pixel Digital Image Sensor.*
* OpenCV. (1 de Junio de 2025). *OpenCV Docs*. Obtenido de https://docs.opencv.org/4.x/index.html
* QT Group. (2025). *QT Documentation.*
* Xilinx. (2021). *Xilinx Artyx-7 FPGAs.*

# Anexo A: Diagrama relacional de requisitos

En el presente anexo se presenta el diagrama relacional de requisitos, en el que se muestran las relaciones y dependencias entre los requisitos mostrados en el apartado 2 de la memoria. El diagrama en árbol muestra, mediante flechas, el requisito de partida como aquél del que parten las flechas, y el requisito que depende de éste como aquél al que apunta la flecha.

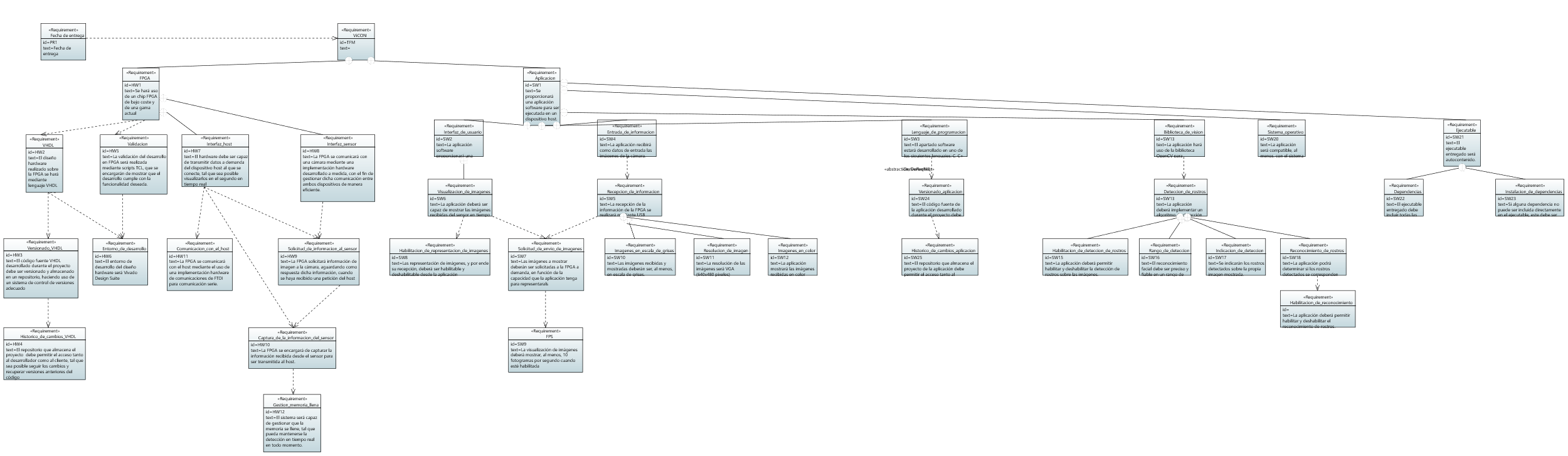


Figura A.1: Diagrama relacional de requisitos.

# Anexo B: Estructura del repositorio de código del proyecto

Para garantizar la trazabilidad del trabajo y disponer de un control riguroso sobre el código desarrollado, se ha creado un repositorio GIT, alojado en la plataforma GitHub. Este repositorio se organiza de manera jerárquica en diferentes carpetas, lo que permite separar claramente los ámbitos de documentación, desarrollo en FPGA y software de aplicación. De este modo, se facilita tanto la reproducción del flujo de diseño como el mantenimiento y evolución del sistema

La estructura general del repositorio se muestra en la figura B.1:

A screenshot of a computer program

AI-generated content may be incorrect.

Figura B.1: Diagrama de contenido del repositorio.

En primer lugar, la carpeta **1.Documentación** contiene los diagramas creados durante el diseño del sistema, los archivos asociados a la redacción de la memoria y un directorio de material complementario (miscelánea) para este proyecto, donde se incluyen recursos adicionales en forma de referencias.

La carpeta **2.FPGA** integra todos los archivos fuente en VHDL, así como un apartado para los *testbench* de pruebas y el script TCL de generación del proyecto.

Finalmente, en la carpeta **3.Host**, se encuentran los componentes software desarrollados sobre el equipo anfitrión. Este apartado incluye el proyecto de la interfaz gráfica (en la subcarpeta GUI), el código asociado al entrenamiento de los algoritmos de reconocimiento (subcarpeta *FaceRecognition*) y un entorno de pruebas para la comunicación con el módulo FT245 (subcarpeta FT245\_Test).

# Anexo C: Guía de usuario de ViCON

Este anexo describe el procedimiento de uso de la aplicación ViCON desde la perspectiva del usuario final. Se detallan los pasos necesarios para la instalación, ejecución y utilización de las principales funcionalidades: adquisición de imágenes, detección de rostro y reconocimiento facial.

**Requisitos previos**

* **Hardware necesario**
  + Sensor de imagen Mt9V111 conectado a la placa Basys3 (con adaptador para conectarse a los conectores Pmod A y XDAC)
  + Módulo FT232H-B conectado al PC mediante USB y a la Basys3 en los puertos Pmod B y C. Deberá estar configurado en modo FIFO asíncrono.
  + PC con sistema operativo Windows 10/11.
* **Software necesario**
  + Xilinx Vivado Design Suite para programar la FPGA.
  + *Bitstream* generado para el Artyx-7
  + Carpeta con el contenido de la aplicación software, incluyendo DLLs y ejecutable.

Para **ejecutar la aplicación**:

1. Conectar los dispositivos hardware si aún no están conectados.
2. Encender la FPGA y programar el *bitstream* desde Vivado.
3. Ejecutar la aplicación software (doble clic sobre el ejecutable). Ver Figura C.1.

A screenshot of a computer

AI-generated content may be incorrect.

Figura C.1: Interfaz de la aplicación software

El **funcionamiento básico** de la aplicación es el siguiente:

* **Iniciar/parar captura:** Para ello solo hace falta hacer clic en el botón “Inicio video” en la parte superior izquierda de la ventana. Cuando inicie, el botón cambiará a verde. Para parar la captura, solo hay que volver a hacer clic sobre el mismo botón.
* **Detección de rostros:** Activar/desactivar la casilla de reconocimiento de rostros. Aparecerá sobre la imagen de vídeo un recuadro verde en torno a las caras detectadas.
* **Reconocimiento facial:** Activar/desactivar la casilla “Reconocimiento facial”. Aparecerá junto a los recuadros de los rostros una etiqueta identificando las caras, si la detección de rostros está activa.
* **Estado:** En la parte central superior, mostrará el estado del sistema (pendiente de inicio, detección iniciada…
* **Indicador de FPS:** en la esquina inferior izquierda aparece un indicador de FPS, que se actualiza con el valor instantáneo de FPS cuando la captura está en curso.

Se puede observar la interfaz con todas sus funcionalidades activas en la Figura C.2:

A screenshot of a person's face

AI-generated content may be incorrect.

Figura C.2: Interfaz de usuario con todas las funcionalidades activas.

Finalmente, para **detener el sistema**:

1. Detener el vídeo.
2. Cerrar la aplicación desde la ventana principal.
3. Apagar la FPGA y desconectar los dispositivos.

**Mensajes de error y solución de problemas comunes.**

* **No se inicia la adquisición:** Verificar la conexión USB del FT232H, estará conectado cuando mantenga una luz amarilla fija encendida. De no ser así, es posible que el módulo esté defectuoso o no funcione adecuadamente.
* **No se muestran rostros detectados:** Confirmar que la casilla de detección está marcada y que la iluminación en escena es la adecuada.
* **Reconocimiento incorrecto:** Revisar que el rostro está incluido en la base de datos, y que esta dispone de suficiente información para reconocerlo.

1. En el repositorio se dispone de un script C++ de entrenamiento que permite entrenar este algoritmo para que reconozca las distintas caras que se requiera. [↑](#footnote-ref-1)