UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

MÁSTER EN INGENIERÍA INDUSTRIAL

SISTEMA DE CONTROL ADAPTATIVO PARA UNA MICRO-ALMAZARA MEDIANTE UNA PLACA DE DESARROLLO NEXYS A7

TRABAJO FIN DE MÁSTER

Febrero - 2026

AUTOR: Antonio Aragón Ros

DIRECTOR/ES: Roberto Gutierrez Mazon

1. Resumen y palabras clave

El presente proyecto de investigación desarrolla un sistema de control inteligente mediante un control adaptativo de una almazara destinada a una finca particular ubicada en Hellín, Albacete (planos de localización/situación (sección 13)), con una capacidad que oscila entre 2500/2200 $\frac{Kg}{h}$.

Para modelarlo, utilizamos la FPGA NEXYS A7 [9] que integra la FPGA ARTIX-7 100T perteneciente a la serie 7 de AMD [2] donde cargamos el procesador *open source* RISC V [7] y sus interfaces (PLB (Processor Local Bus), OPB (On-chip Peripheral Bus), AXI (Advanced Extensible Interface), LMB (Local Memory Bus)y FSL (Fast Simplex Link)) [26].

Además, añadimos una comunicación ETHERNET para el puerto RJ-45 [14] y configuramos los puertos PMOD de la placa, donde establecemos diferentes comunicaciones (AXI-SPI, AXI GPIO...) utilizados para el muestreo de los diferentes sensores y el manejo de los actuadores con el objetivo de controlar la producción de aceite generando un sistema de calidad que asegure la calidad total.

Los actuadores a controlar (tolvas, ventiladores, lavadora, sinfines, variadores de frecuencia, molino, batidora, centrifuga horizontal, tamiz vibratorio, decanter, adicción de talco, centrífuga vertical, lipieza de filtros) son gestionados mediantes diferentes protocolos de comunicación cuya gestión vendrá monitorizada por los sensores de cada proceso (temperatura, humedad, temporizadores, caudalímetros, pHmetros...) y los parámetros seleccionados por el usuario.

Para terminar, utilizamos una estructura servidor-cliente para comunicarnos con la FPGA utilizando una aplicación en formato SCADA en configuración de controlador adaptativo [13].

Palabras clave

Control adaptativo, FPGA NEXYS A7, RISC-V, SCADA, PROFIBUS, Ethernet Industrial, Automatización de procesos, Almazara, Control de calidad, Protocolos de comunicación industrial, Sensores y actuadores, Industria 4.0, IoT.

Índice

2. Índices de tablas, gráficos y figuras.

Índice de figuras

3. Introducción

Una finca ubicada en el término municipal de Hellín desea instalar una microalmazara para la producción en frío y por lotes de AOVE (Aceite de Oliva Virgen Extra). Entre los requisitos del cliente la almazara debe permitir la catalogación, control y muestreo del proceso de extracción del AOVE separando la producción por lotes de calidad que son controlados durante el proceso de producción.

Figura 3.1: Localización de la finca en las inmediaciones de la zona.

Para abarcar todas las especificaciones del proyecto, seleccionaremos y dimensionaremos los elementos físicos de la instalación ubicándolos sobre un plano acotado. Para el mando de la planta utilizamos un control adaptativo que utiliza la placa de desarrollo **NEXYS A7** [9] como controladora ajustable con los siguientes componentes (figura 3.3):

- La FPGA Artix-7 (XC7A100T-1CSG324C) con:
  - 101.440 celdas lógicas.
  - *Configurable Logic Blocks* (CLBs)

        - 15.850 Slices
        - 1.188 Kb de memoria distribuida (RAM)
    - 240 DSP48E1 Slices
    - Memorias RAM embebida
        - 270 memorias de 16 Kb
        - 135 memorias de 36 Kb
        - Memoria total de 4.860 Kb
    - 6 *Clock Management Tiles* (CMTs)
    - 1 conexión PCI Express como bus de expansión
    - 8 *Gigabit Transceiver* (GTP)
    - 1 *Xilinx Analog to Digital Converter* (XADC)
    - 6 Bancos de I/O
    - 300 I/O máximas
- Memoria RAM DDR2 128 MiB
- Puerto *Ethernet Physical Layer* (PHY) 10/100 Mbps
- Conector para tarjeta microSD
- 4 conectores Pmod
- 1 conector XADC Pmod
- Conector de 12 bit VGA
- Salida PWM de audio
- Microfono PDM
- Un sensor de temperatura
- 2 pantallas con 4 display de 7 segmentos cada uno
- 16 switches

- 16 LEDs
- 5 botones
- 2 LEDs tri-color
- Conector USB
- Conector de alimentación

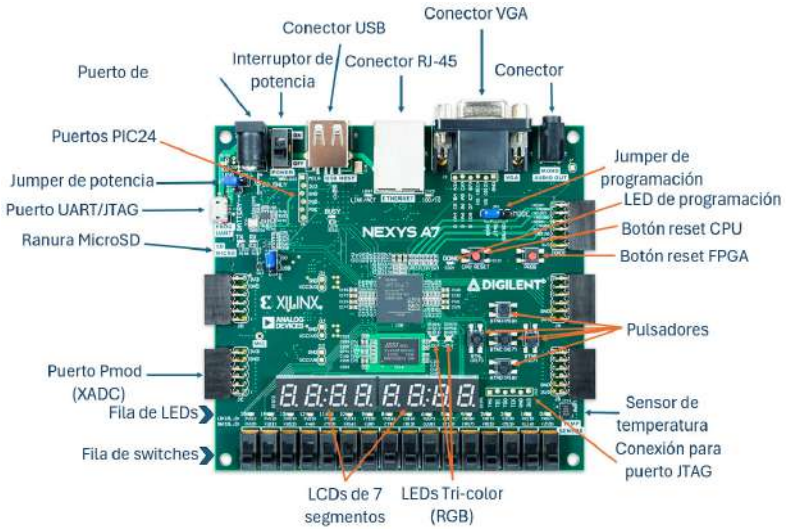


Figura 3.2: Placa de desarrollo NEXYS A7 con la Artix-7 100T.

Durante el control de los diferentes elementos, como son los captadores/sensores y actuadores y preactuadores de la almazara utilizamos un sistema de comunicación en bus con los equipos unidos entre sí a través de unas líneas comunes compartidas por todos los dispositivos donde se trabaja en una estructura maestro-esclavo, donde el maestro es la placa de desarrollo encargada de inicializar la red y recibir los datos de los módulos de E/S de los diferentes esclavos para procesarlos.

La almazara cuenta con dos zonas claramente diferenciadas.
Un **patio de recepción** donde se transporta, pesa y lava la aceituna para registrarla y garantizar la salubridad del proceso.
La **fábrica** productora encargada de la pasta de aceituna y extracción del aceite con un alto rendimiento y calidad.

Ambas partes cuentan con los siguientes elementos para la producción y control del proceso de elaboración del AOVE.

3.1. Proceso de producción del AOVE

Patio de recepción

Para la elaboración del aceite de oliva partimos desde el patio de recepción donde se reciben los contenedores con la materia prima sin tratar.
En el patio se encuentran la zona de pesado donde el camión con los contenedores son pesados y registrados. Posteriormente, se descarga la aceituna en un foso enrejado donde se encuentra la tolva de recepción y se realiza la primera separación de las ramas y la aceituna.
La aceituna es transportada sobre la tolva de castigo formada por una cinta transportadora funcionando a pocas revoluciones con un ventilador para impedir el antrojado (fermentación anaeróbica) por la acumulación de la aceituna [25]. En ella, se verifica y clasifica la calidad de la aceituna a partir de:

- Una analizadora con tecnología de espectroscopia del infrarrojo cercano
- Una sonda de humedad con microondas
- Una sonda de temperaturas

Figura 3.3: Sensores colocados en la cinta de castigo

A partir de los datos obtenidos en los sensores, definimos el tipo de proceso para la obtención del aceite, la calidad de la aceituna y su clasificación.

La cinta de castigo transporta la aceituna hacia la lavadora o despedregadora donde se realiza un desbaste de piedras y restos sólidos, durante el proceso controlamos la humedad mediante un sensor.
Por último, se termina la limpieza con un equipo de filtrado como paso previo al molino.

Fábrica

Para asegurar la calidad del aceite durante el proceso, una caldera de quemador de hueso (figura 3.4) es el encargado de mantener una temperatura en el agua entre los (20-27) ^{o}C [25], para ello la temperatura en la caldera es controlada con el quemador de huesos y el ventilador de aire.

Figura 3.4: Caldera de quemador de aceituna

Para transportar la aceituna por el foso hasta el molino, un tornillo sin fin de velocidad constante durante el lote, dirige la aceituna hasta la molienda donde se configura:

- el grado de molienda (dependiente del diámetro de la malla)
- La velocidad de giro de las palas del molino
- El caudal de alimentación

En la salida, el producto con textura de pasta es propulsada por una bomba de pistones hasta la batidora.
El objetivo de esta, es romper la emulsión de agua y aceite formado en el molino. Para ello, de forma lenta y continua se remueve y calienta la pasta mientras se aplica talco durante el proceso [25].

El proceso tendrá una duración entre los $20 < t < 30$ minutos con temperaturas que pueden variar entre los $25 < T < 27^{o}$C o $20 < T < 25^{o}$C para asegurar y controlar la calidad del producto final [25].

La disolución con ambos compuestos son separados por fuerza centrípeta mediante un **decanter** (centrífuga horizontal) que puede ser de bol o tornillo sinfin con un tamiz vibrador y una bomba de alperujo para separar el agua del aceite y una **centrífuga vertical** encargada de añadir agua tratada a la beta de aceite [25].

Finaliza el proceso con el llenado los depósitos decantadores donde se controla la purga de impurezas y la calidad final.
Por último, se almacena el aceite en grandes depósitos donde se cataloga como paso previo al envasado.

4. Sensores, actuadores y preactuadores

En la almazara debemos controlar cada una de las etapas de producción siendo necesario un sistema de realimentación negativa formada por [13]:

- Señales de referencia
- Unidades de realimentación
- Unidades comparadoras
- Señales de error
- Unidad de control
- Señales de salida

En la placa de desarrollo NEXYS A7 se aloja la unidad de control que recibe la señal de referencia, definida en relación al índice de funcionamiento, el cual es comparado con el índice deseado y actualizado corrigiendo el error en un mecanismo de adaptación que ajusta los parámetros del regulador o directamente la señal de salida (figura 4.1). Esta configuración es propia de un controlador adaptativo.

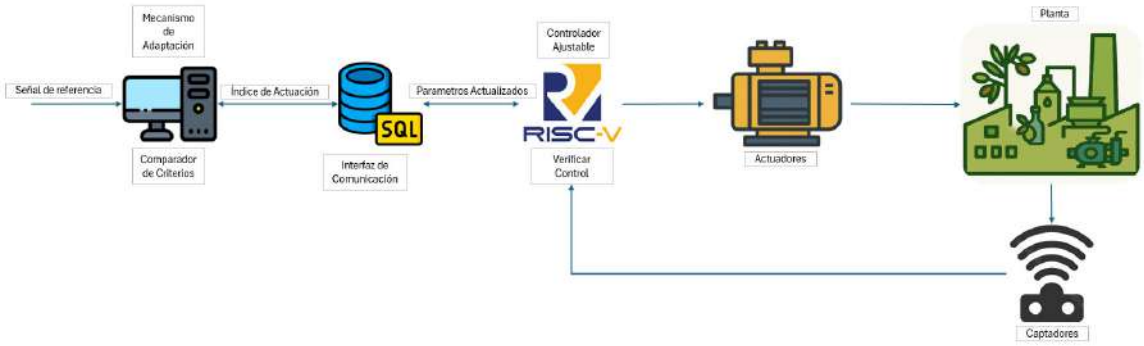


Figura 4.1: Control adaptativo.

4.1. Sensores, transductores y transmisores

Para obtener el estado del proceso de producción se identifican las variables de estado del sistema. Para ello, los sensores deben ser seleccionados para medir las magnitudes adecuadas.
Los detectores a implantar son:

- Patio
    1. Un sensor de temperatura colocado en el patio
    2. Una sonda de humedad en continuo (sonda de microondas) en la cinta de castigo
    3. Sensor de humedad en la lavadora
- Sala interior
    1. Sensores de temperatura en la batidora, decanter y centrífuga vertical
    2. Caudalímetro en la centrífuga vertical
    3. Sondas de llenado y purga en los depósitos
- Sensor de pH en el agua de la almazara
- Sensor de temperatura en el agua a la salida de la caldera

A partir del tipo y la forma de muestrear estas variables tenemos en cuenta los transductores a implementar para convertir estas señales en relación con magnitudes eléctricas y la forma de transmisión a la FPGA.

Los sensores escogidos de cada proceso son los aconsejados por los proveedores de los aparatos donde se han diseñado las ubicaciones e instalación en el dispositivo. Para seleccionar cada aparato se tendrá en cuenta estas especificaciones siendo esencial la potencia, **muestreo** y control de cada dispositivo, principalmente se establece una comunicación por protocolo TCP/IP e IO-Link.

4.2. Actuadores y preactuadores

Los equipos de control son principalmente **relés octoacoplados** y **variadores de frecuencia** como preactuadores de los motores y bombas que controlan las cintas transportadoras, tornillos sinfin, ventiladores, moto-vibrador, molino, batidora, centrífuga vertical y caldera.
Los equipos son:

- Un variador de frecuencia para motor para la cintas transportadora (patio de recepción).
- Un relé/contactor para la aventadora/deshojadora.
- Lavadora.
    - un relé/contactor para el motor de los 2 ventiladores centrífugos de doble oído.
    - un relé/contactor para el moto-vibrador de la bandeja vibratoria.
    - un relé/contactor para la válvula estranguladora y antirretorno del agua.
- Un variador de frecuencia para el motor del tornillo sin fin (entrada a la fábrica).
- Molino
    - Un variador de frecuencia para el motor de los martillos rotantes.
    - Un relé/contactor para el motor de la criba rotante.
    - Un autómata que controla el dosificador de talco
- Batidora
    - Un variador de frecuencia para el termoacelerador (preparador de masa de aceituna).
    - Un relé/contactor para la válvula de paso en la entrada.

    - Un relé/contactor para la alimentación del calentador resistivo.
    - Un relé/contactor para accionar el sistema de limpieza.

- Centrífuga horizontal
    - Un relé/contactor para el tornillo sinfín
    - Un variador de frecuencia para el motor de giro del tambor.
    - Un relé/contactor para la válvula estranguladora y antirretorno del agua a la centrífuga.

- Un variador de frecuencia para la bomba helicoidal de tornillo salomónico (transporte de la masa)
- Un relé/contactor para el tamiz vibrador
- Centrífuga vertical
    - 2 relés/contactores para las válvulas de paso
    - Un relé/contactor para la válvula estranguladora y antirretorno de agua.
    - Un variador de frecuencia para el motor del giro centrífugo

- Un variador de frecuencia para la bomba salomónica (salida de alperujo)
- Un variador de frecuencia para la bomba de pistones (transporte del aceite)
- Un variador de frecuencia para la bomba centrífuga de agua.
- Relés optoacoplados para las válvulas de los depósitos
- Caldera de biomasa policombustible
    - Variador de frecuencia para el quemador de hueso
    - Variador de frecuencia para el ventilador

- Depósitos de almacenamiento
    - Un relé optoacoplado para la bomba de desplazamiento positivo del llenado de los depósitos

- Un relé optoacoplado para la válvula de distribución del llenado de cada depósito
- relés optoacoplados para las válvulas de descarga de los depósitos

Para el control de los actuadores y preactuadores, así como del muestreo de cada uno de los elementos utilizamos una comunicación IP partiendo de una estructura maestro-esclavo con protocolo API REST, donde a partir de un switch de la marca Cisco (figura 4.2), con 24 puertos Gigabit PoE+ y 4 puertos SFP Uplinks para módulos SFP, controlamos y muestreamos toda la planta.

Product number	Total ports	SFP Uplinks	SFP fiber ports (S)	Copper PoE/PoE+ Ports[2] (P)	Default software
IE-4010-4S24P	28	4 FE/GE		24 FE/GE	LAN Base[1]

Figura 4.2: Switch Cisco.

4.2.1. Control de relés

En el control de las líneas con señal binaria (0/1), utilizamos una placa de relés optoacoplados de 8 canales y control por un puerto RJ-45. En ella, cada salida de los reles están aislados del puerto *ethernet* mediante comunicación óptica.

El circuito de cada relé controla la conmutación de un contactor trifásico schneider colocado en la línea del motor (figura 4.3). El circuito de control en la línea del motor es de señal continua de 24 V que es suministrada por un conversor AC/DC y alimentado de la red.

El control de la placa proviene de una de las bocas del *switch* controladas por protocolo API REST.

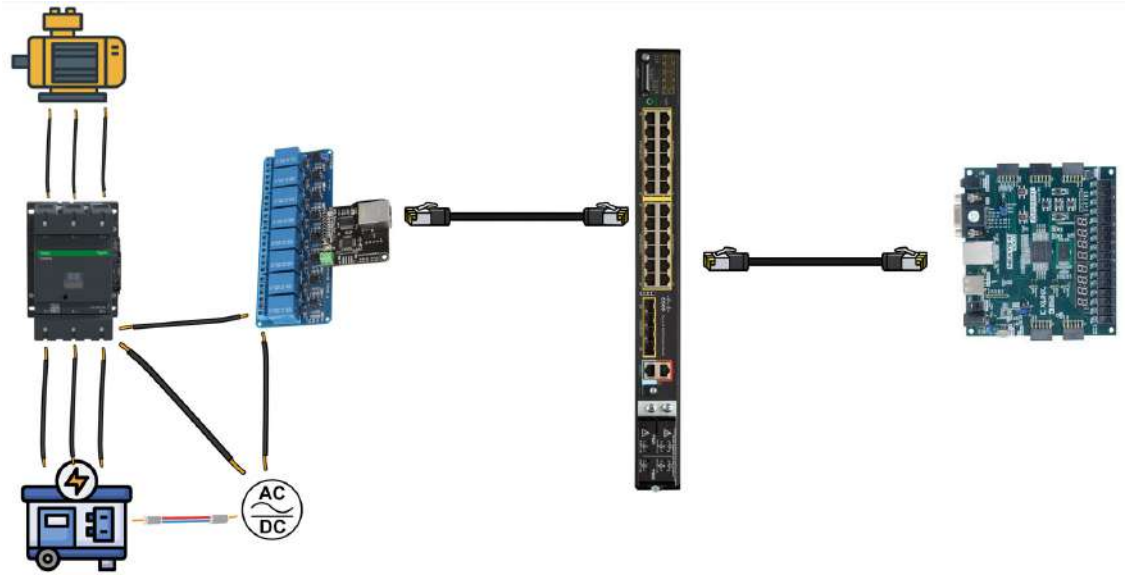


Figura 4.3: Control de activación y paro en los motores eléctricos.

4.2.2. Control de los variadores de frecuencia

Los variadores de frecuencia de marca **Schneider** (figura 5.1) están provistos de un puerto RJ-45 para comunicación ethernet TCP/IP. Desde ese puerto controlamos los parámetros del variador a través de su puerto IP.

Figura 4.4: Variador de frecuencia Schneider con interfaz TCP/IP.

5. Diseño del controlador

Partiendo de la placa de desarrollo NEXYS A7 100T [9] configuramos su FPGA, diseñando con la herramienta **Vivado** la estructura *hardware*.
En la aplicación, configuramos los distintos bloques IPs que albergan los diseños *hardware* con sus funcionalidades desde donde realizaremos el esquema de conexiones de comunicación entre ellos y con los puertos externos que suministra la placa (reloj y puertos físicos).
Posteriormente exportaremos el diseño para añadirlo a un proyecto en VITIS, plataforma que nos permite programar los bloques con archivos en lenguaje C.
Otra opción más sofisticada es utilizar la plataforma Petalinux que nos permite cargar un sistema operativo Linux en el bloque del procesador RISCV.

5.1. Vivado

Partiendo del programa de diseño Vivado desarrollamos el archivo *xca* con la estructura física a implantar.

Para ello, añadimos en su ventana gráfica los diferentes IPs (Archivos de Propiedad Intelectual) consistentes en módulos funcionales predefinidos y reutilizables (IP cores) que se pueden integrar y conectar en un diseño gráfico donde está esquematizado el diseño [6].

Configuramos el proyecto en Vivado seleccionando un proyecto RTL extensible a la plataforma VITIS.
En la configuración de la placa en el proyecto, nos descargamos del catálogo la placa **Nexys 100T** del vendedor *digilentinc.com*. Con ello, en la pestaña de *boards* nos aparecen todos los puertos y bloques que vienen preconfigurados en la placa, además la plataforma nos ayuda a interconectar todos los elementos de forma automática seleccionando su configuración pulsado en *auto conexion*.

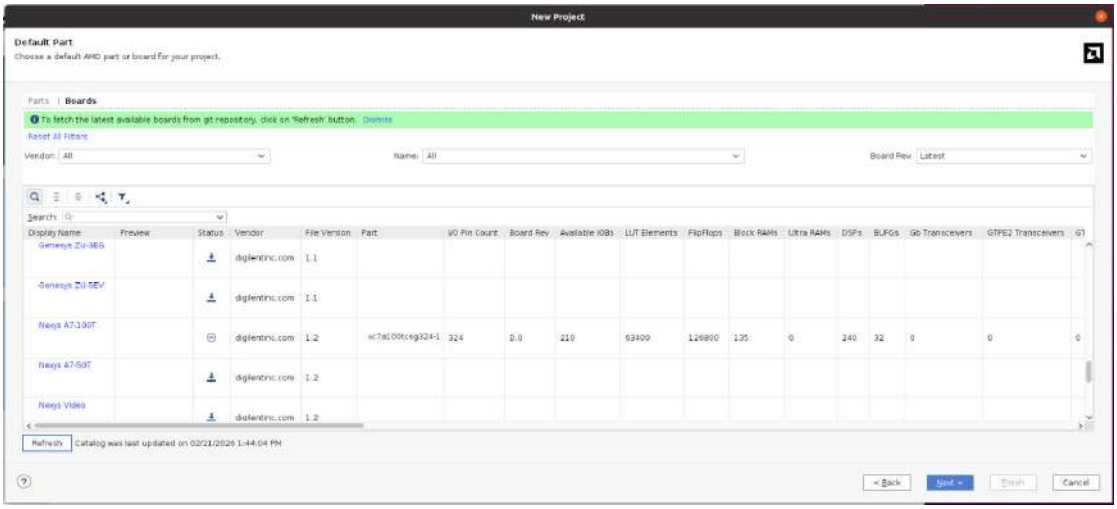


Figura 5.1: Selección de la placa en Vivado.

Para las interconexiones entre los diferentes elementos, tendremos en cuenta.

- La compatibilidad de la conexión. Misma protocolo de comunicación entre la salida y entrada de conexión entre bloques.
- Verificar que todos los bloques que contemplen una entrada de reloj, esté conectada a su reloj correspondiente.
- Asignar el botón de *reset* de la placa al bloque controlador de la acción y a los bloques que tengan configurado ese puerto
- Para los puertos externos de la placa asegurar que tenga el mismo nombre que en el archivo *xdc* (figura 15.2) y su correcta conexión.

5.1.1. IP MicroBlaze V

Comenzamos con el módulo Microblaze V (figura 5.6) que integra todo el diseño del procesador RISC V.
Vivado nos otorga dos posibles IPs con estructura RISC V. La IP Microblaze V y su IP compacta (MCS) [7]. Debido a la complejidad en las conexiones de los puertos partimos del diseño general, donde configuramos sus parámetros.

1. Procesador de 32 bits

2. Configuración Real-time Preset, con un preajuste en tiempo real orientado al control en tiempo real. Rendimiento optimizado, cachés pequeñas y depuración habilitada.

3. Optimización del tipo *Performance*

4. Habilitamos la cache de las instrucciones y los datos

5. En los recursos habilitamos a los contadores y temporizadores

6. Habilitamos las interrupciones de los datos e instrucciones AXI

7. Para la memoria local disponemos de interfaz de instrucciones y datos

8. Creamos una interfaz para los datos de los periféricos AXI

Figura 5.2: Bloque RISC-V en el esquema.

Al no tener que gestionar una gran cantidad de información en memoria y no tener que realizar cálculos complejos obtamos por un procesador de 32 bits que utiliza un diseño más compacto y gestiona la memorias de manera más ligera.
En la forma de compilar el diseño físico del procesador, indicamos una gestión que priorice el rendimiento frente al área de impresión con una mejor gestión de la cache.
Habilitamos la cache activando sus puertos para mejorar el flujo de datos e instrucciones en el procesador al tener la cache una mayor velocidad que la memoria principal.
Habilitan los puertos de comunicación con el controlador de memoria y la interfaz *AXI* en los periféricos esenciales para la comunicación con los sensores, control de los actuadores y comunicación con la base de datos.

En la pestaña principal del bloque habilitamos la memoria cache interna del componente con una estructura de 32 *bits*.

Figura 5.3: Parametros principales de configuración del IP Microblaze V.

En las características generales solo habilitamos el uso de contadores y temporizadores para poder capturar por *software* el rendimiento, utilizar librerías de control de tiempo y usar interrupciones.

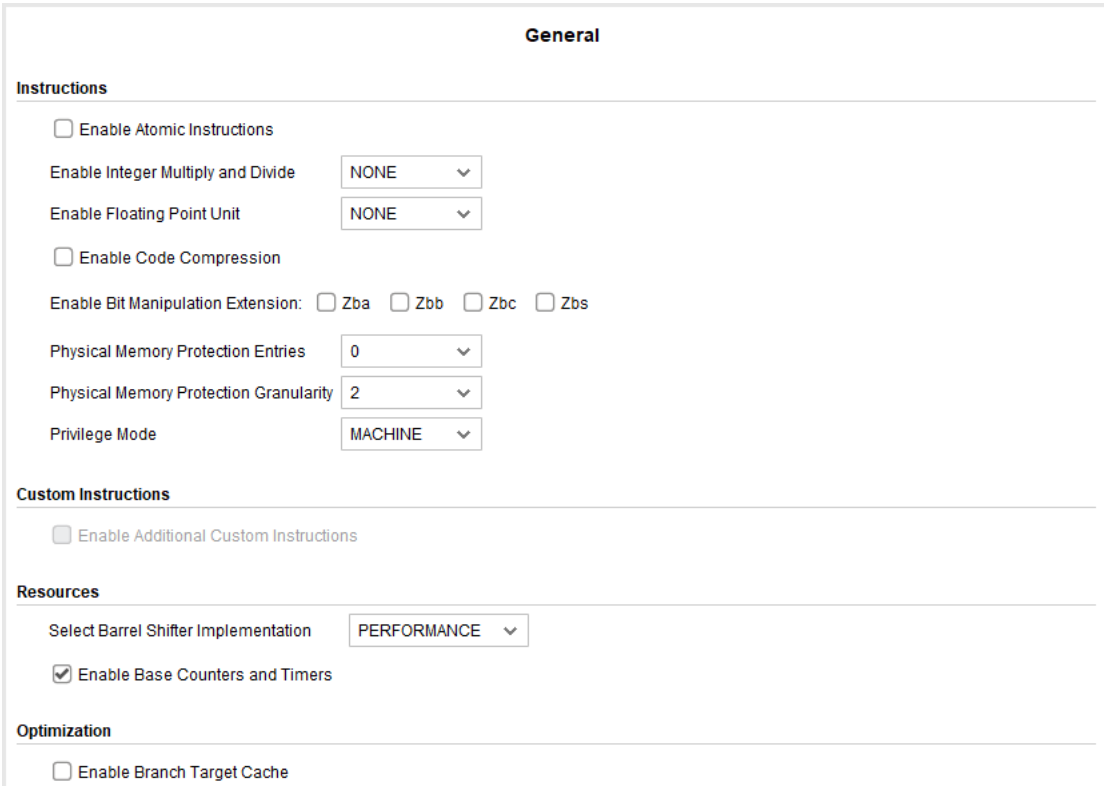


Figura 5.4: Parametros generales de configuración del IP Microblaze V.

Para controlar y gestionar los posibles errores del procesador habilitamos los buses de las excepciones en los datos e instrucciones, así como en el control de otros tipos de excepciones.

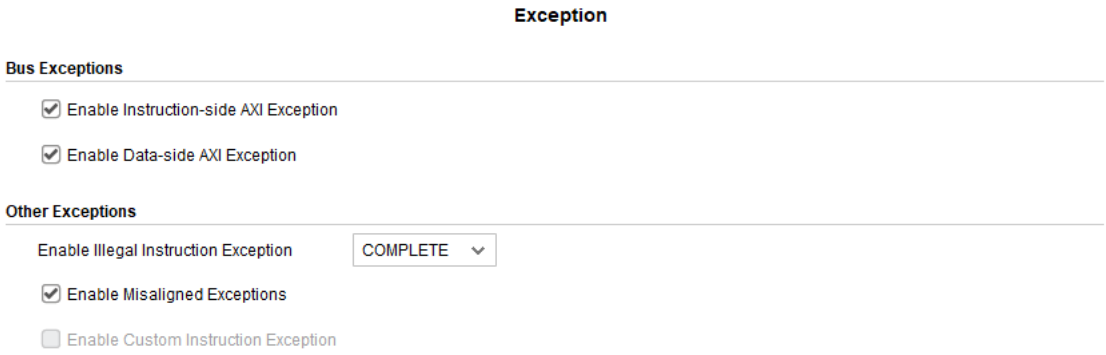


Figura 5.5: Parametros de las excepciones del IP Microblaze V.

En las conexiones exteriores habilitamos los buses con la memoria DDR2 y de los datos con la interfaz *AXI*.

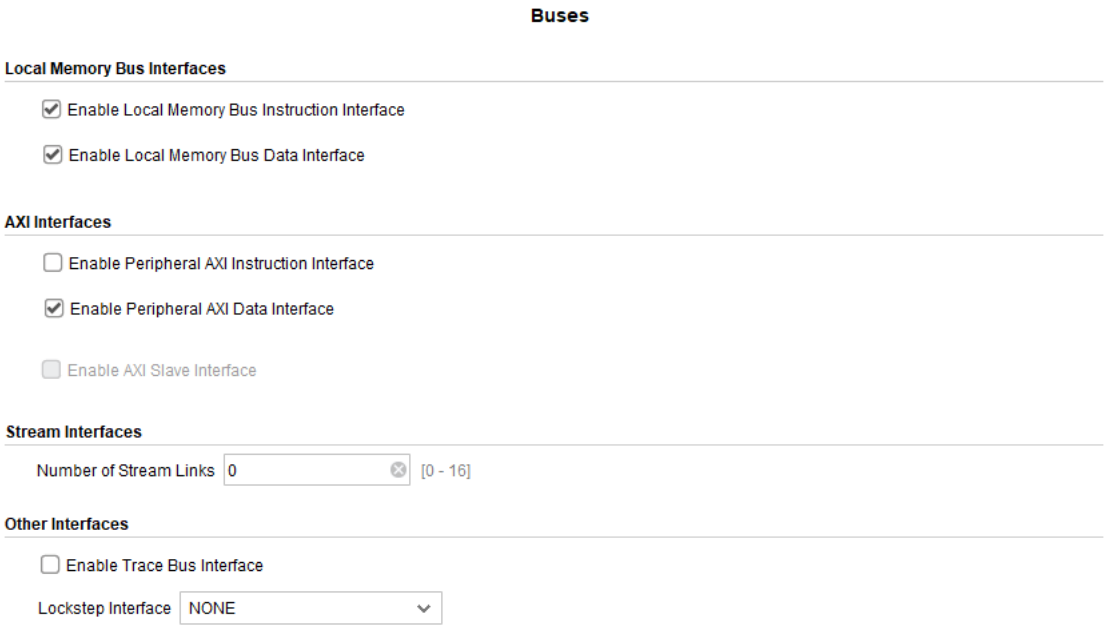


Figura 5.6: Parametros de los buses externos del IP Microblaze V.

Con esos parámetros, Vivado nos genera automáticamente la memoria local, conectando los puertos de instrucciones *ILMB* y datos *DLMB* a la arquitectura *hardware*.

En las conexiones de entrada al procesador establecemos el IP de depuración *Microblaze Debug Module (MDM) V* que configura un puente de comunicación entre el procesador y la plataforma de depuración (Vivado o Vitis) a partir del puerto JTAG [15] donde:

1. Cargar programas en la memoria del controlador.
2. Establecer *breakpoints*
3. Hacer depuraciones paso a paso
4. Modificar y leer registros en la memoria en tiempo real.
5. Resetear el procesador.

Para la gestión de las interrupciones de los periféricos o señales externas RISCV implementamos el IP AXI Interrupt Controller (figura 5.7) que agrupa las señales de interrupción de los periféricos [1], otorgándoles un orden de prioridad, enmascara y habilita interrupciones concretas accesibles por software, registran los estados del controlador y genera una interfaz AXI4-Lite para gestionar interrupciones desde software embebido como PetaLinux [22].

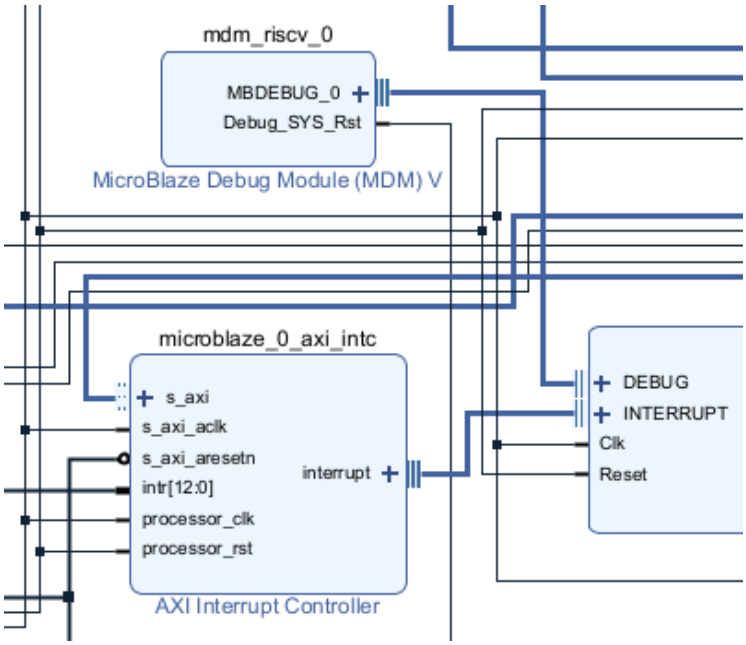


Figura 5.7: Señales de entrada del IP Microblaze V.

5.1.2. Memoria RAM

La placa de desarrollo NEXYS A7 lleva embebida una memoria RAM del tipo SDRAM con la tecnología DDR2 [9].
Para acceder a ella, implementamos el bloque IP **Memory Interface Generator (MIG 7 Series)** (figura 5.8) que actua como interfaz de comunicación entre el procesador y los pines de la memoria DDR2 [5].

El bloque IP recibe de entrada la señal de reloj principal de 100MHz que proviene del oscilador externo de la placa conectado a un bloque Clocking Wizard encargado de resetearlo. El bloque establece otro señal de reloj limpia y sincronizada para acceder la memoria SDRAM DDR2 a 200MHz que utilizamos como reloj principal para todo el diseño. Con ello, sincronizamos el procesador, periféricos y memoria a un mismo reloj [3].

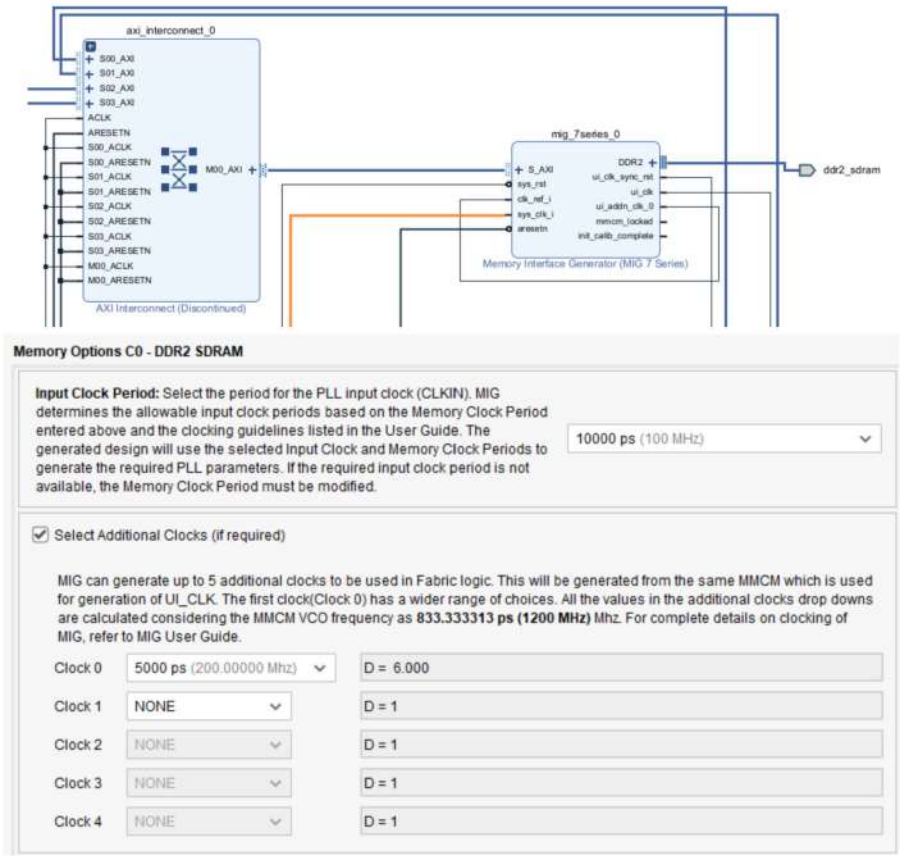


Figura 5.8: Configuración de la memoria SDRAM DDR2.

La configuración de los pines externos a la memoria por parte del bloque IP se realiza de forma automática (figura 15.1), para ello establecemos en el proyecto la placa de desarrollo NEXYS A7 100T e implementamos su configuración (figura 5.9).

Para reducir la carga de trabajo en el procesador implementamos el módulo IP **AXI Direct Memory Access** encargado de mover los datos entre la memoria principal y los periféricos AXI4-Stream sin intervención con el procesador [26].

En el bloque IP de Vivado habilitamos los canales de lectura y escritura donde conectamos el puerto *S_AXI_LITE* como un maestro del *AXI Interconnection* que tiene como esclavo al procesador y los puertos *M_AXI_MM2S* y *M_AXI_S2MM* como esclavos del *AXI Interconnection* de la memoria SDRAM DDR2 [21].

Además, añadimos un bloque propio *tone_generator_v1_0*, capaz de generar una señal periodica, y que nos sirve para verificar la cadena de transmisión y que el DMA funcionan correctamente a partir de un archivo *testbench* [1].

Conectándolo al puerto *S_AXIS_S2MM* (*Stream to Memory-Mapped*), la FPGA toma la señal generada por el bloque y la guarda en la memoria automáticamente.

Esta configuración nos permite hacer simulaciones en nuestro diseño y validar la comunicación entre la memoria y el procesador.

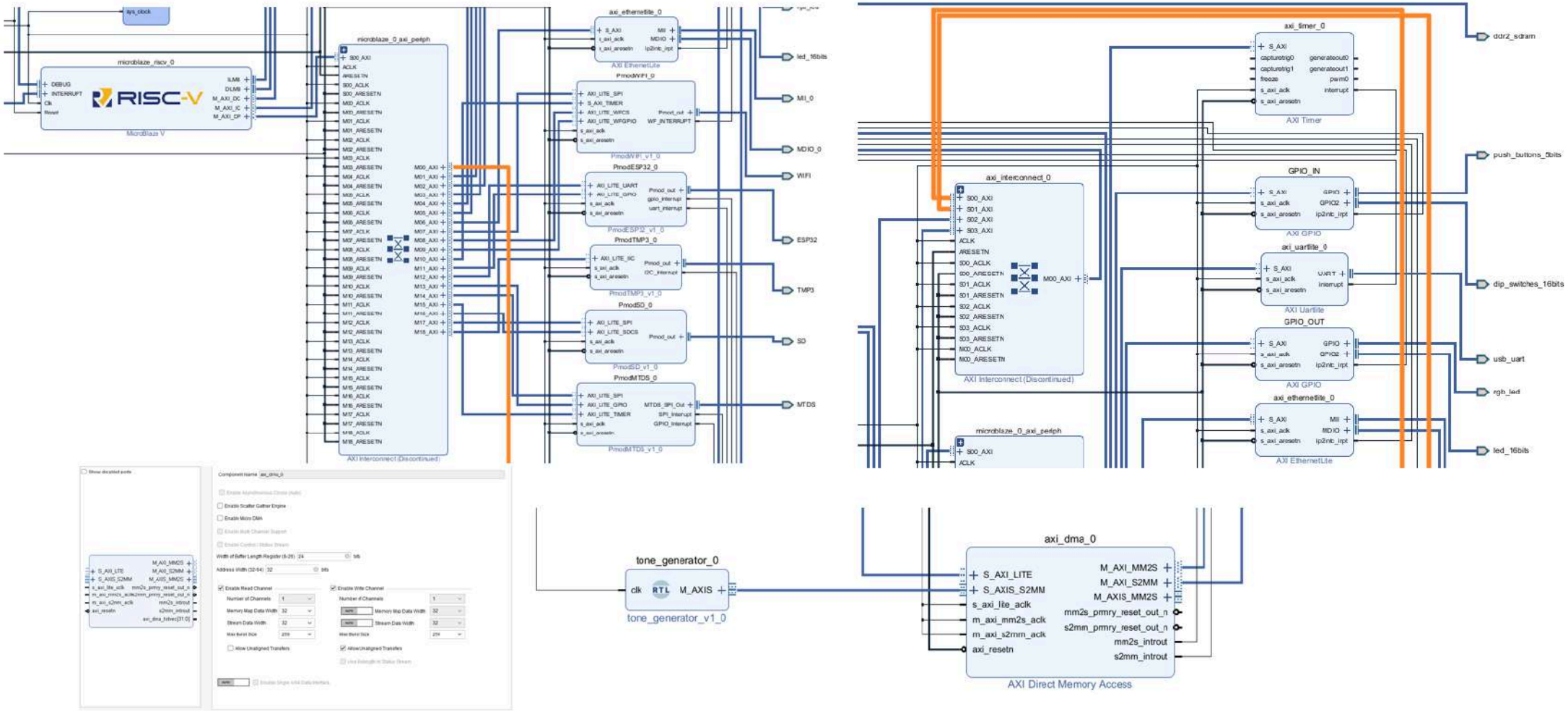


Figura 5.8: Composición del acceso a la memoria por los periféricos.

5.2. Acceso a memoria

Durante la ejecución de los programas en el procesador RISC V, la búsqueda y guardado de datos e instrucciones en la memoria RAM es muy frecuente. Por ello, para reducir la carga de trabajo del procesador y mejorar la transferencia de datos a los periféricos añadimos en nuestro esquema el bloque **Direct Memory Access** con la configuración (figura 5.10).

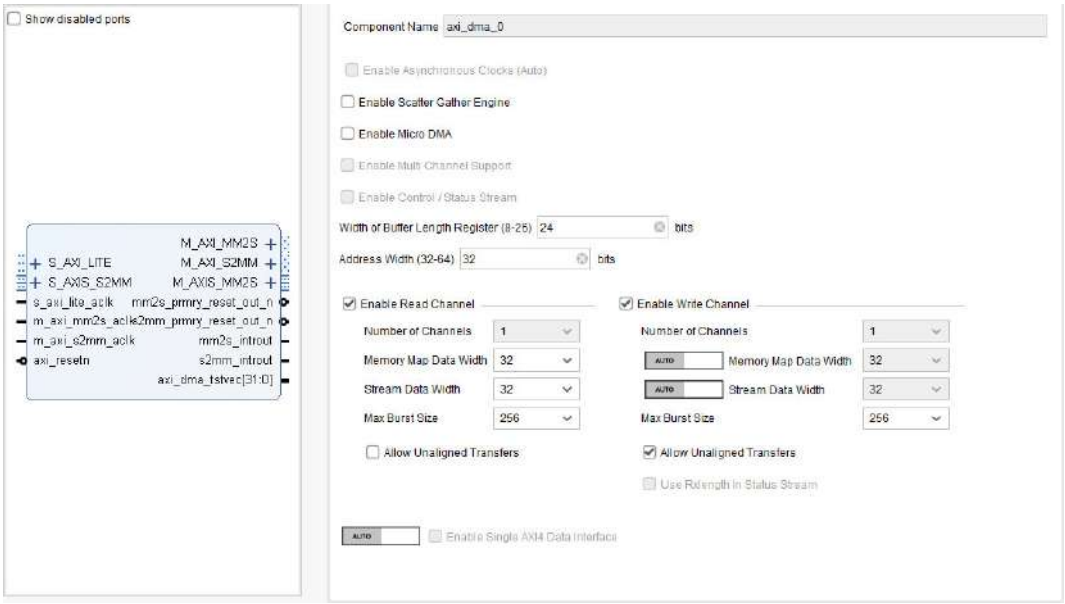


Figura 5.10: Configuración y conexión del DMA.

Este bloque mejora la comunicación con la memoria en:

1. Reduce el número de ciclos de acceso a memoria al pasar de copiar byte a byte con bucles recurrentes ejecutados por el procesador a emitir ráfajas *AXI burst* con gran ancho de banda, en nuestro esquema serían rafajas *AXI burst* de 32 bits definido en la configuración del procesador.

2. Mejora la latencia con el uso de las señales *MM2S* (bus de salida de la memoria hacia el periférico) y *S2MM* (bus desde la memoria al periférico) que genera un acceso directo en la memoria sin posibilidad de interrupciones y eliminando paradas de *software*.

3. Al liberar al procesador de esta tarea, ambos elementos trabajan en paralelo trabajando en colaboración de forma eficiente generando *pipelines* más eficientes y reduciendo el tiempo *end to end*.

5.2.1. Control de reinicio

Para gestionar y sincronizar el sistema de arranque y mantener un estado estable con el reloj. El bloque IP **Processor System Reset** (figura 5.11) añadido al diseño es conectado a la memoria, procesador y periféricos [3].

Este componente recibe las señales externas de reset y las sincroniza con el reloj del sistema para evitar *glitches*. Además genera resets limpios para los distintos bloques y sincroniza los relojes del sistema (100 MHz y 200 MHz).
También nos permite reiniciar el sistema desde el puerto JTAG con la plataforma Vitis usando el bloque *Microblaze Debug Module (MDM) V*.

La gestión de la señal *reset* en el diseño de nuestra placa garantiza un arranque predeterminado y seguro, garantizando el inicio correcto de todos los elementos secuencialmente donde.

1. La señal de reloj necesita tiempo para ajustar el oscilador y alinear su frecuencia con el reloj de entrada en el *clocking wizard*.

2. Los componentes que gestionan las memorias, tanto el *Memory Interface Generator* como el *AXI Direct Memory Access* necesitan inicializarse.

3. Los periféricos también deben sincronizarse en el *reset* con el procesador y el resto de elementos.

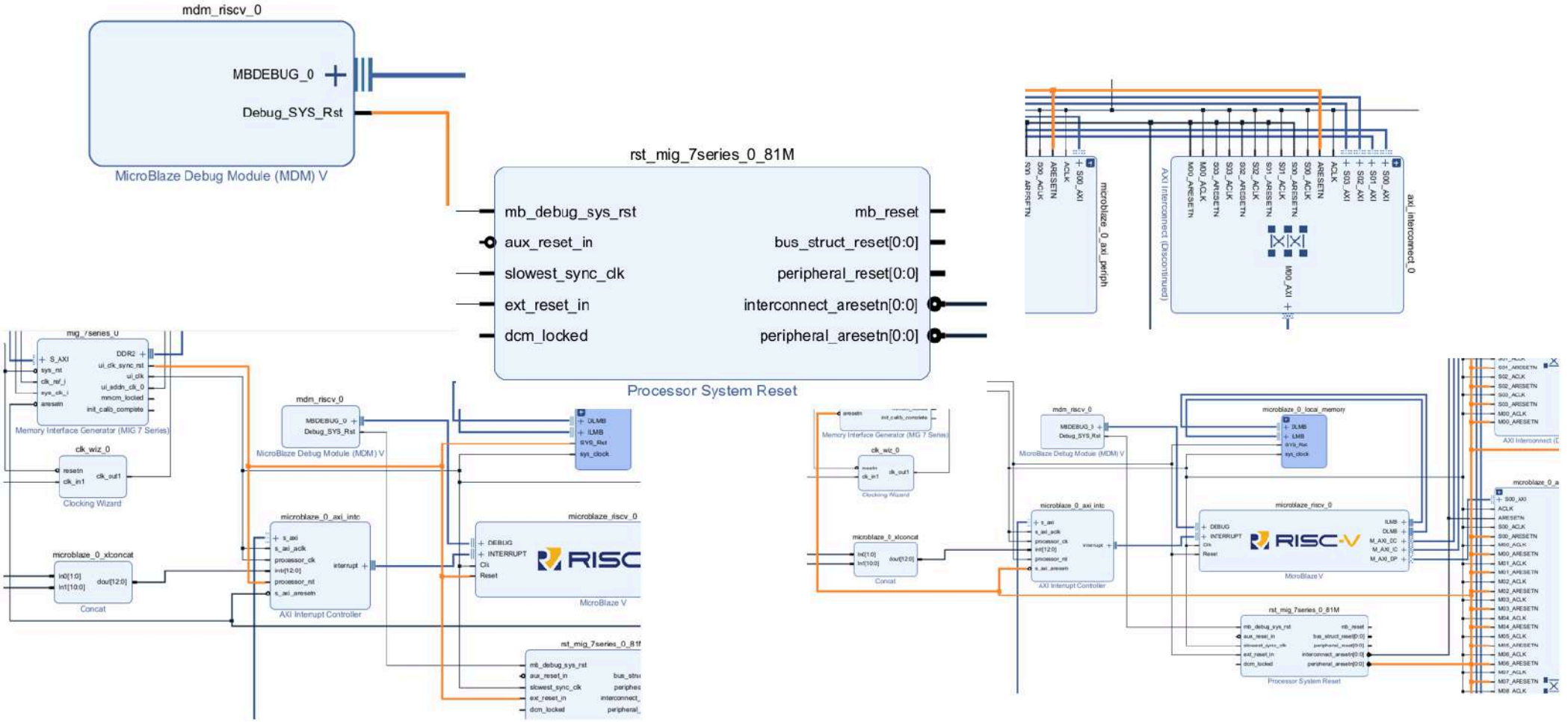


Figura 5.9: Sistema de RESET en el procesador.

5.2.2. Relojes

La placa de desarrollo NEXYS A7 integra una señal de reloj de 100MHz generada a partir de un oscilador de cristal soldado en la placa y conectado directamente a un puerto MRCC (*Multi-Region Clock Capable*) de la FPGA, el pin E3 [9].

El reloj externo se implementa como una señal externa del controlador a la memoria (MIG 7 Series) de donde un divisor de frecuencia genera la señal de reloj principal más estable de 200 MHz, señal de reloj que se conecta a los periféricos, procesador y memoria del sistema.

Otro reloj que genera internamente el sistema es utilizado en la comunicación Ethernet PHY del puerto RJ45 desde el cuál es necesario generar un reloj de 50 MHz desfasado $45^{\underline{o}}$ del reloj principal de 200MHz [12]. Este reloj lo genera internamente el bloque IP **AXI EthernetLite** (figura 5.12).

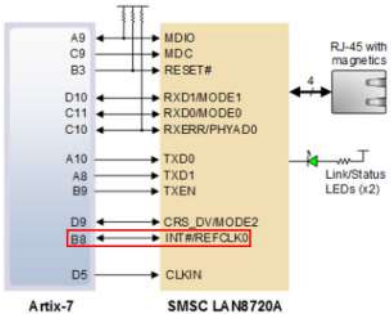


Figura 5.12: Esquema de conexión en la placa del puerto RJ-45.

5.2.3. Arquitectura de comunicaciones con periféricos

La placa de desarrollo NEXYS A7 implementa diferentes puertos de comunicación. En nuestro diseño implementamos aquellos más útiles para la comunicación remota, visualización y modificación de parámetros y control de preactuadores [23].
Los desarrolados son:

1. El puerto RJ-45: clave en el envío de los datos al sistema de visualización y

configuración del proceso [14].

También funciona como puerto principal para el control de los actuadores y preactuadores, así como de la comunicación con los sensores de la almazara.

2. Interfaz con los 5 pulsadores y 16 switches de la placa mediante comunicación GPIO.

   Primordial para modificar y resetear parametros del controlador de forma física y rápida.

3. Comunicación GPIO para los 16 LEDs físicos y 2 LEDs RGB

4. Puerto USB-UART. Para comunicarse físicamente con una comunicación UART a través de un terminal.

5. Temporizador de 32/64 bits interno que nos proporciona la posibilidad de generar:

   - Dos temporizadores programables con la posibilidad de generar interrupciones, eventos y la opción captura de eventos.
   - Un contador configurable.
   - La creación de una salida PWM (*Pulse Width Modulation*).
   - Posibilidad de detener los contadores durante la depuración software.

6. Puertos Pmod [8]. Configuramos cada uno de los 5 puertos con diferentes comunicaciones y utilidades (figura 5.18).

   Estos puertos desarrollados por *Digilent* nos permiten conectarnos con más de 80 módulos periféricos. Ofreciendo una gran diversidad de conexión con otros protocolos.

   6.1 Pmod WI-FI. Configuración para antena WI-FI Pmod.

   Otorga de comunicación inalámbrica al controlador.

   6.2 Pmod ESP32. Para la conexión a un módulo con el microprocesador Tensilica Xtensa en modo esclavo.

   Ideal para reducir la carga de trabajo del procesador RISC-V de la FPGA.

6.3 Pmod TMP3. Comunicación con un módulo con sensor de temperatura.
Usado para captar la temperatura ambiental de la almazara.

6.4 Pmod SD. Puerto para tarjeta SD.
Para añadir otra memoria externa a la FPGA.

6.5 Pmod MTDS. Confiere la comunicación con una pantalla táctil.
Ideal para mostrar información y modificación de parámetros en físico.

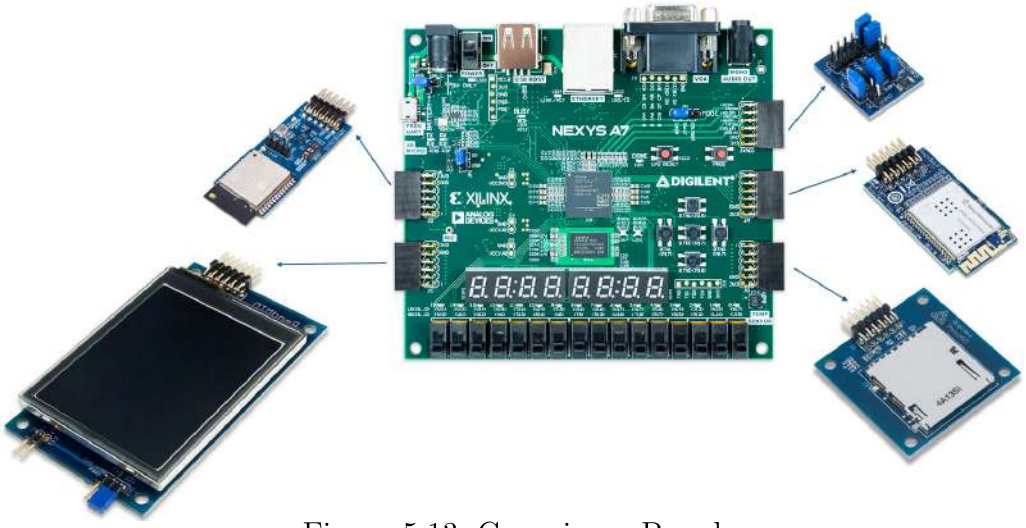


Figura 5.13: Conexiones Pmod.

El procesador MicroBlaze V se basa en la arquitectura de conjunto de instrucciones (ISA RISCV) compatible con el hardware del procesador clásico MicroBlaze.

Esta arquitectura nos permite desarrollar una comunicación AXI (*Advanced eXtensible Interface Protocol*) con un alto ancho de banda y velocidad, estableciendo una comunicación punto a punto [26].
El protocolo de esta interfaz es desarrollado como parte de la arquitectura AMBA (*Advanced Microcontroller Bus Architecture)*) [15].

En nuestro diseño, desarrollamos la interfaz (AMBA 4.0 AXI4) con capacidad para un alto rendimiento en el mapeo de la memoria (figura 5.14).
Con el uso de bloque IP **AXI Interconnect** [1], mapeamos las conexiones AXI de

[1]El bloque AXI Interconnect está abandonado por Xilinxs pero en nuestro diseño no es reemplazable al ser necesario colocar dos señales reset a los periféricos

entrada de los bloques de cada periférico con los puertos maestros del bloque de conexión, y un puerto esclavo AXI conectado al puerto *M_AXI_DP* (*Peripheral data interface*) del procesador [21].

Para mantener el sincronismo en nuestro diseño, asociamos cada periférico con el reloj interno de la memoria (200MHz) y su señal *RESET*.
Este esquema queda más detallado en el apartado 5.3 del documento.

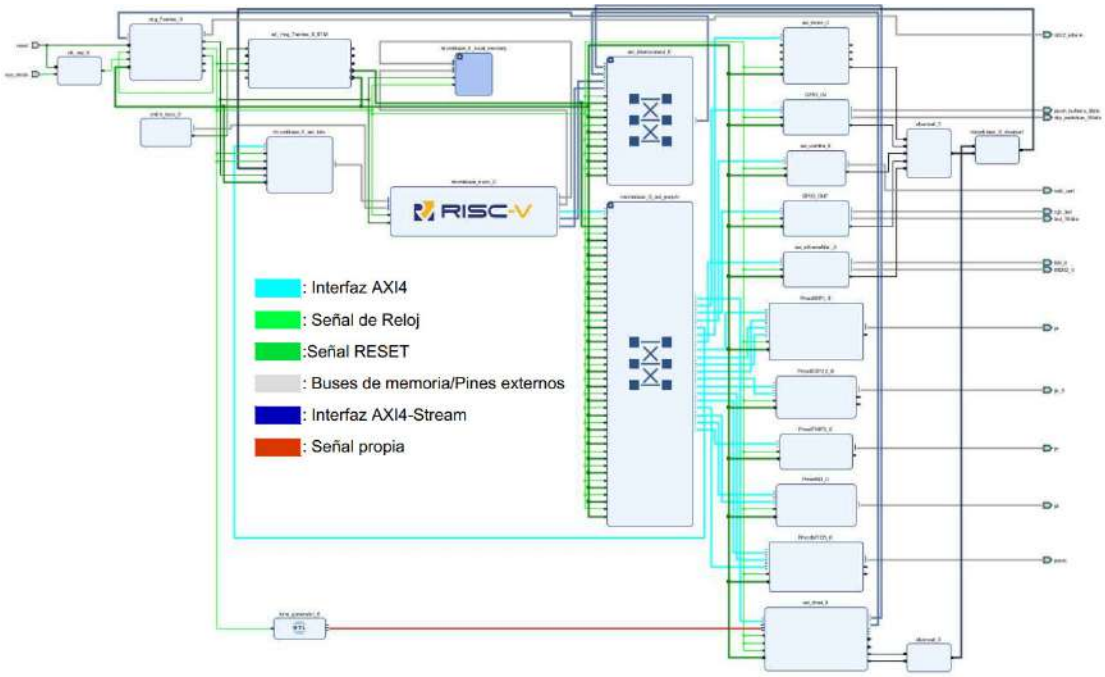


Figura 5.14: Esquema de conexiones de reloj y reset.

Los bloques implementados en el diseño de cada puerto son:

AXI EthernetLite

Este bloque define la comunicación por protocolo IP con la interfaz AXI4 (figura 5.15) y la frecuencia de reloj interna (200MHz) [19].

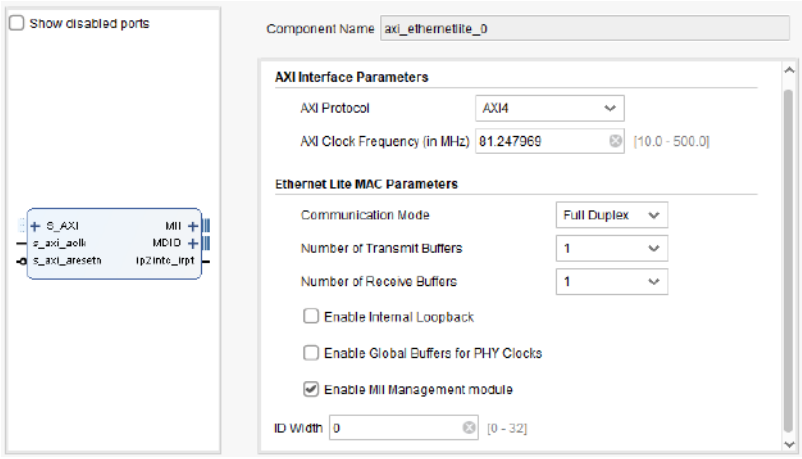


Figura 5.15: Configuración del bloque *Ethernet Lite*.

El controlador admite las siguientes funcionalidades [14].

1. El modo de bucle interno.
2. Soporta comunicación *Ping pong* tanto para RX como TX, siendo capaz de leer y escribir en los *buffers* al mismo tiempo.
3. Admite comunicación MII 10/100 Mbps PHY.
4. Admite el acceso a PHY a través de la interfaz MDIO.

En nuestro diseño utilizamos el modo de comunicación **Full Duplex**, que no utiliza la señal de reloj PHY, al no compartir el bus y solo necesitar la supervisión de sus propias transmisiones, con un solo *buffer* de comunicación y habilitando el módulo MII [17].

AXI Uartlite

Para configurar un protocolo UART (figura 5.16) en el puerto USB de la placa.

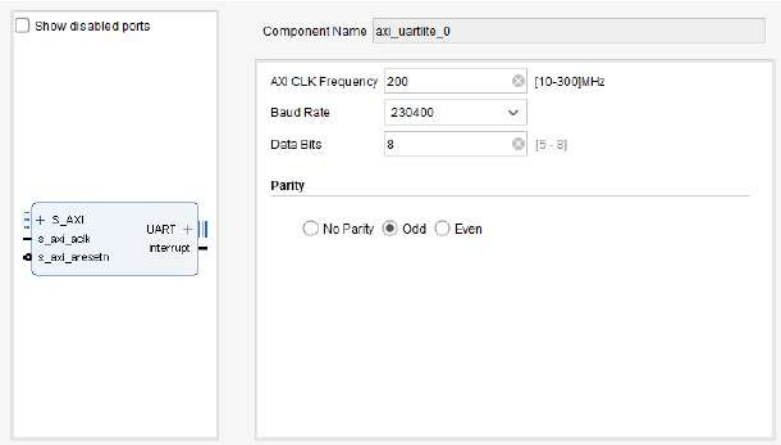


Figura 5.16: Configuración del bloque *AXI UARTlite*.

Parte de la implementación por protocolo AXI4-Lite en modo esclavo para el acceso a los registros y la transferencia de datos.

En nuestro diseño establecemos la frecuencia del reloj (200 MHz), un *Baud Rate* de 230.400$\frac{baudios}{s}$, un tamaño de datos de 8 bits y un bit de paridad par.

AXI GPIO

Utilizamos los bloques IP, **AXI GPIO** uno en configuración de entrada (*GPIO IN*), para los botones y switches. Otro en configuración de salida (*GPIO OUT*), para los leds RGB y los 16 leds colocados en cada switch de la placa (figura 5.17).

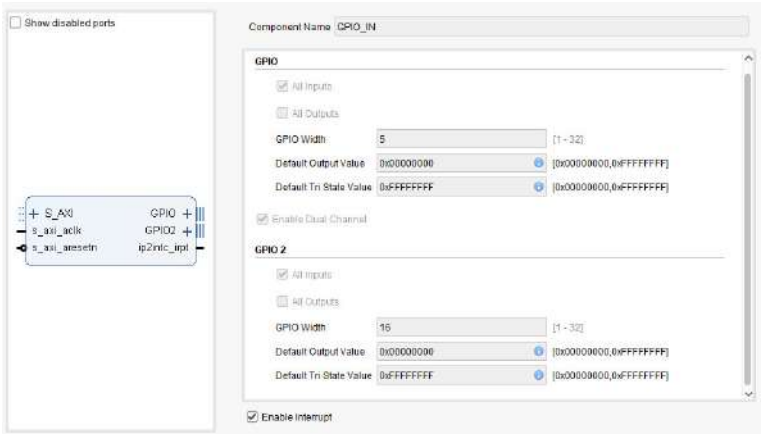


Figura 5.17: Configuración del bloque *AXI GPIO* en configuración de entrada (botones y switches).

Pmod

La interfaz con estos puertos se configura a través de bloques específicos (IP) desarrollados por *Digilent* [8] que configuran la interfaz con cada dispositivo *Pmod* (figura 5.18).
La librería donde se albergan estos bloques es accesible a través de *GitHub* y anexada a nuestro proyecto en Vivado a través de la configuración principal del proyecto.

Estos bloques no son configurables e integran toda la interfaz con cada dispositivo.

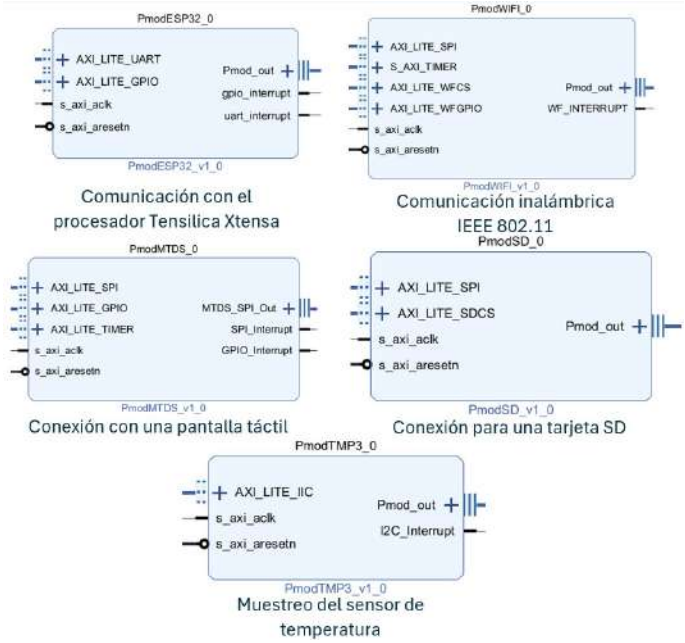


Figura 5.18: Configuración del bloque *AXI GPIO* en configuración de entrada (botones y switches).

5.3. Esquemas finales del diseño

Como análisis final, representamos el diseño de la FPGA con diferentes tipos de esquemas que configura Vivado. Estas formas de visualizar el diseño, nos permiten entender y verificar la estructura AXI en la configuración de la FPGA donde disgregamos el diseño en sus diferentes formas de visualización.

Diagrama de señales

En el esquema (figura 5.19), aparece representado el diseño *hardware* identificando por colores los diferentes tipos de conexiones/señales dependiendo de su comunicación.
En él, verificamos que los componentes que definen el puerto ethernet, GPIO y UART, así como el AXI timer y la interfaz de la memoria (DDA) se conectan al controlador de los eventos (*AXI Interrupt Controller*) que gestiona las interrupciones del flujo de instrucciones en el procesador.

Diagrama de comunicación

En este diagrama (figura 5.20) se muestra una vista simplificada del diseño donde solo mostramos los canales de transmisión de datos y la interfaz de direcciones entre los bloques del diseño. Esta vista nos ayuda a comprender y verificar las conexiones más críticas del diseño y que comunicación entre bloques se configuran.
Las señales del reloj y reset desaparecen para dar importancia y claridad a la interfaz AXI4 de datos e instrucciones

Diagrama por grupos

En esta vista (figura 5.21) asociamos en grupos, cada bloque IP.
Con ello, comprobamos la conexión y unión entre los diferentes bloques del diseño.

Diagrama Completo

Este último esquema representa el diseño detallado completo en Vivado.

Para asignar las salidas y señales de la placa a nuestro esquema en Vivado descargamos el archivo *.xdc* (figura 15.2) de nuestra placa donde comprobamos la tecnología y tipos de puertos que implementa. Esta información queda recogida en el anexo [15.2].

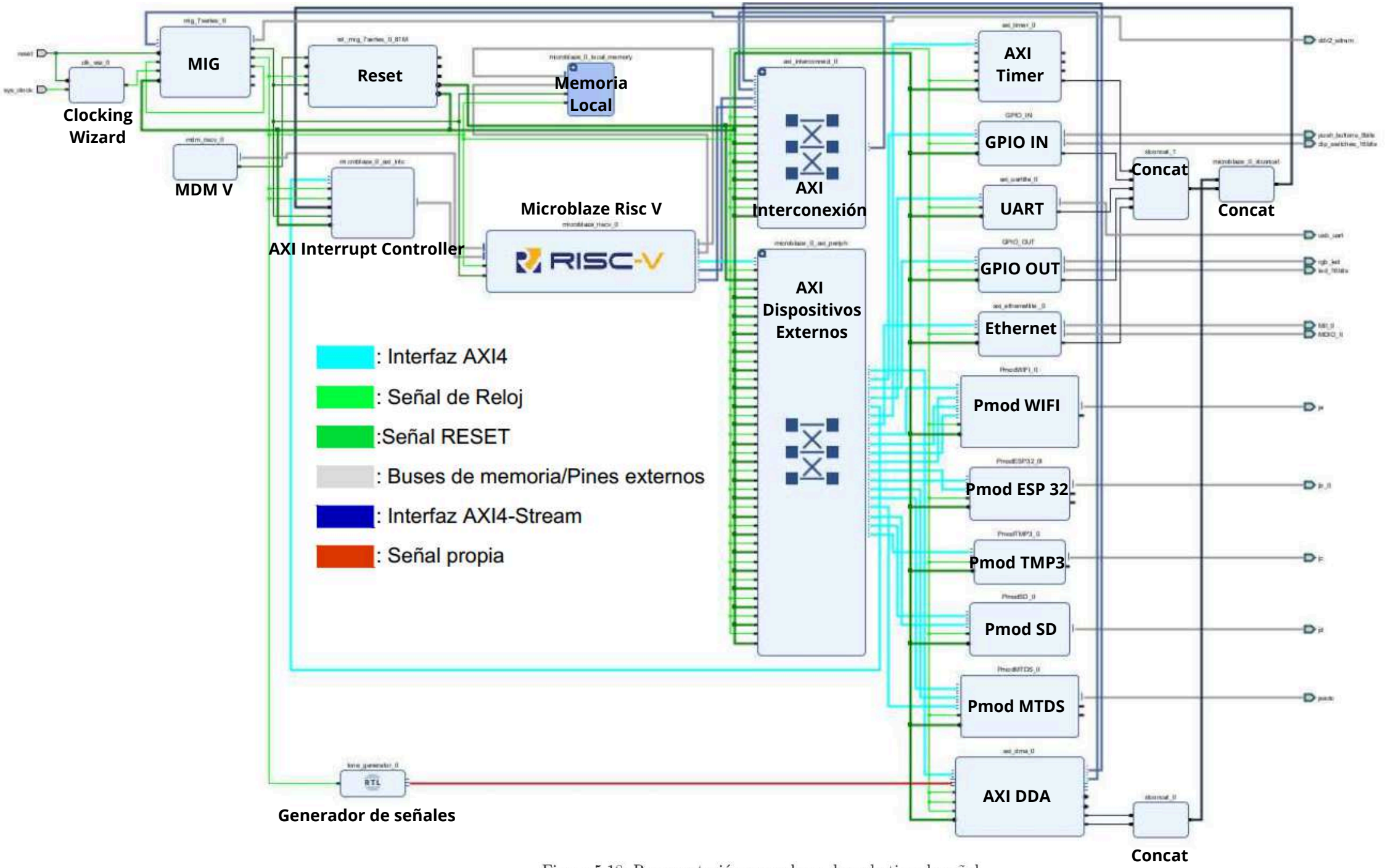


Figura 5.18: Representación por colores de cada tipo de señal.

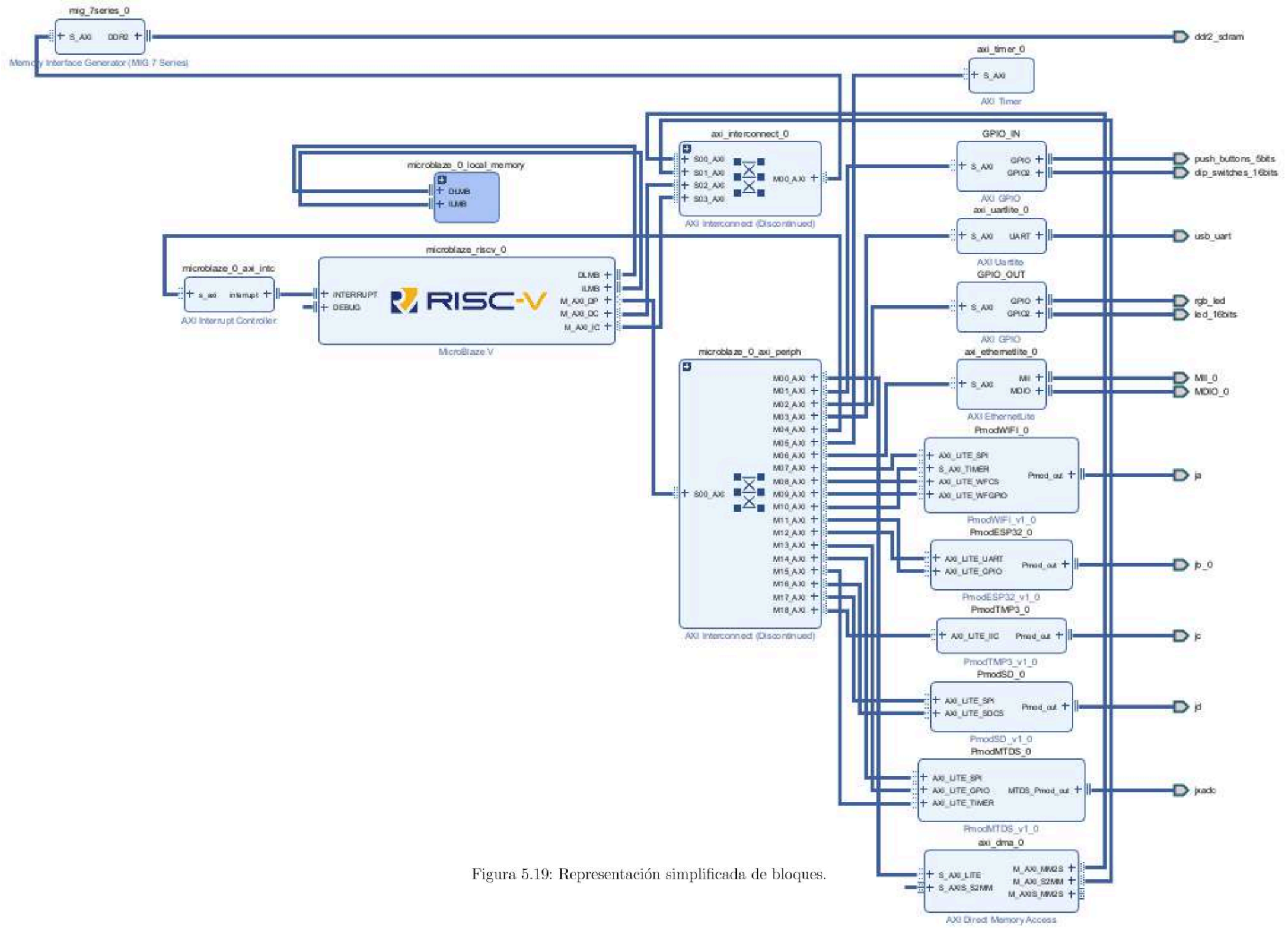


Figura 5.19: Representación simplificada de bloques.

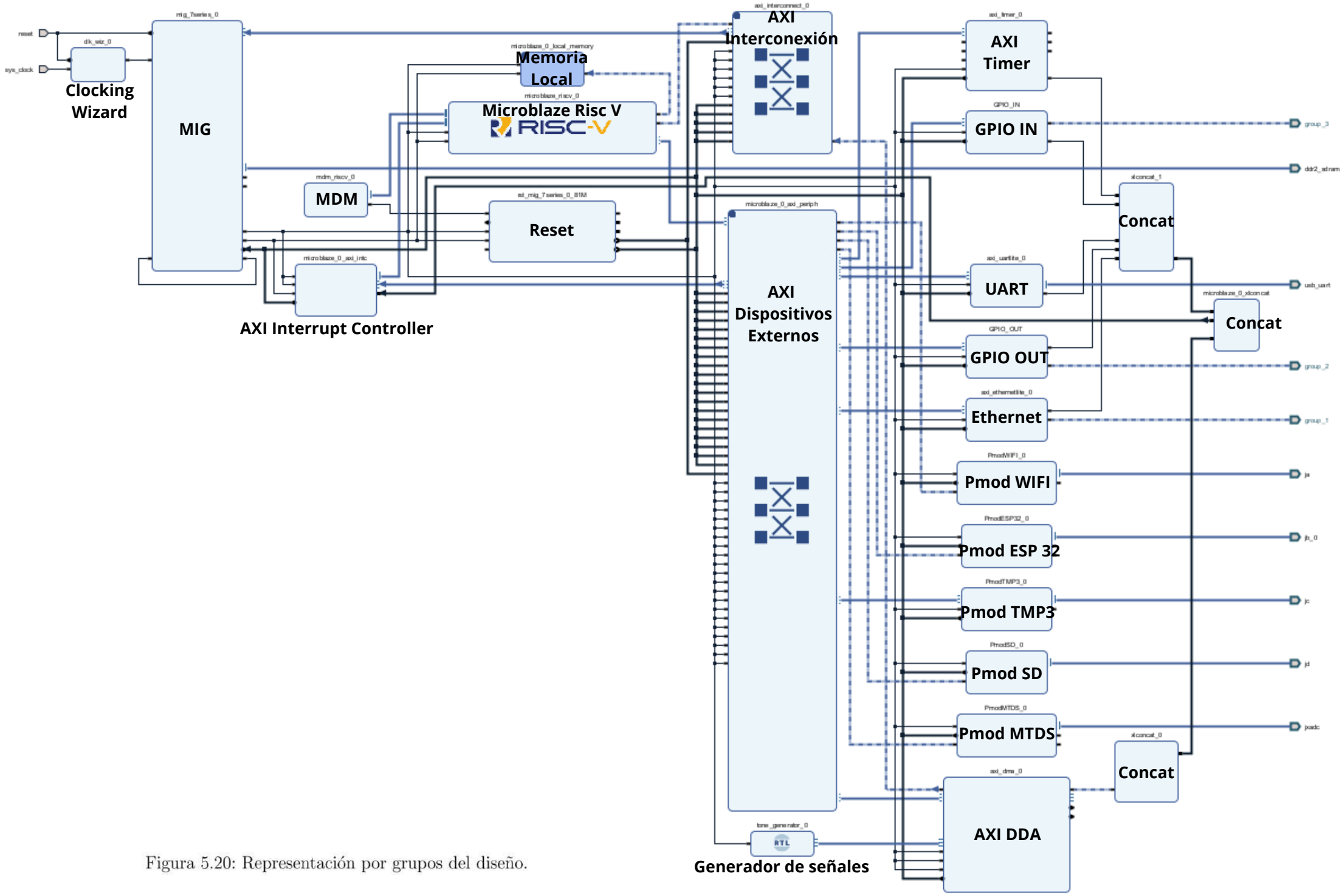


Figura 5.20: Representación por grupos del diseño.

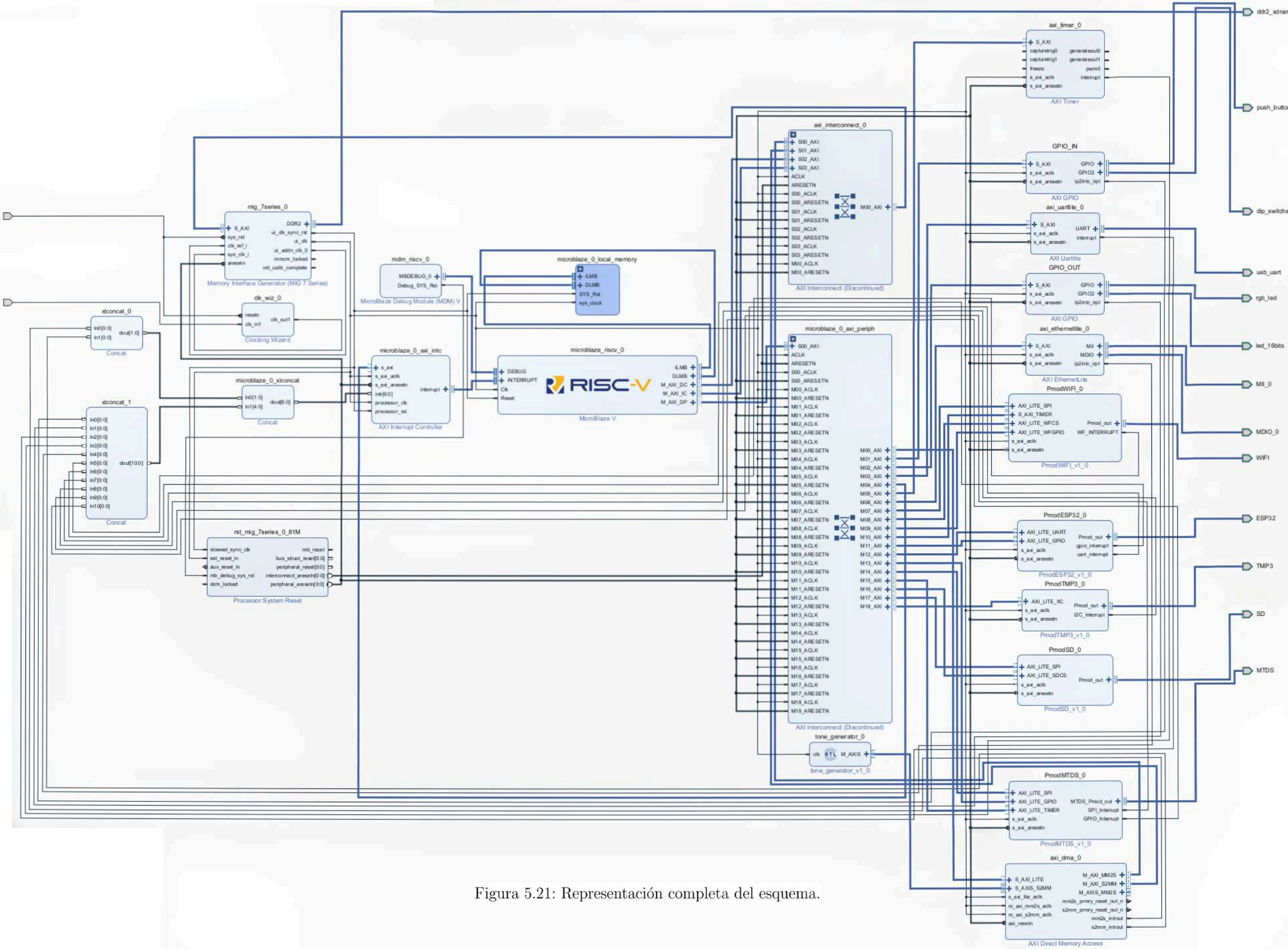


Figura 5.21: Representación completa del esquema.

6. Configuración del soporte lógico en la placa

Para ejecutar el programa de gestión y control de la almazara en el procesador RISC V que hemos modelado, nos encontramos con dos vías de ejecución principales.

La primera opción que nos ofrece Xilinxs es usar su plataforma de alto nivel **Vitis** donde programar el archivo de control en lenguaje C mediantes archivos *.c*.

La segunda opción es utilizar la herramienta **Petalinux** desde donde cargar un sistema operativo tipo Linux en la FPGA.
El programa de control, en lenguaje python *.py*, es cargado en el procesador mediante una tarjeta SD y ejecutado a través de la línea de comandos del sistema operativo (SO).
Este formato nos permite utilizar las librerías de *python* que nos proveen de más funcionalidades que las librerías que contiene Vitis en lenguaje C y de archivos *shell* que nos permite automatizar tareas y simplificar procesos.

6.1. Vitis

Partiendo del diseño por bloques en Vivado, somos capaces de exportarlo a la plataforma Vitis, que nos provee de un entorno de desarrollo con un nivel más alto de abstracción, mediante un archivo **Xilinx Support Archive** *.xsa* que registra la configuración *hardware* diseñada y previamente exportado desde Vivado (figura 6.1).

En la plataforma, con el diseño hardware cargado, el programa nos genera las librerías C y C++ (.h) con la implementación de nuestro diseño desarrollado en Vivado, desde la que podemos acceder a sus componentes, generando archivos C con extensión (.c) y guardados en en el proyecto [10].

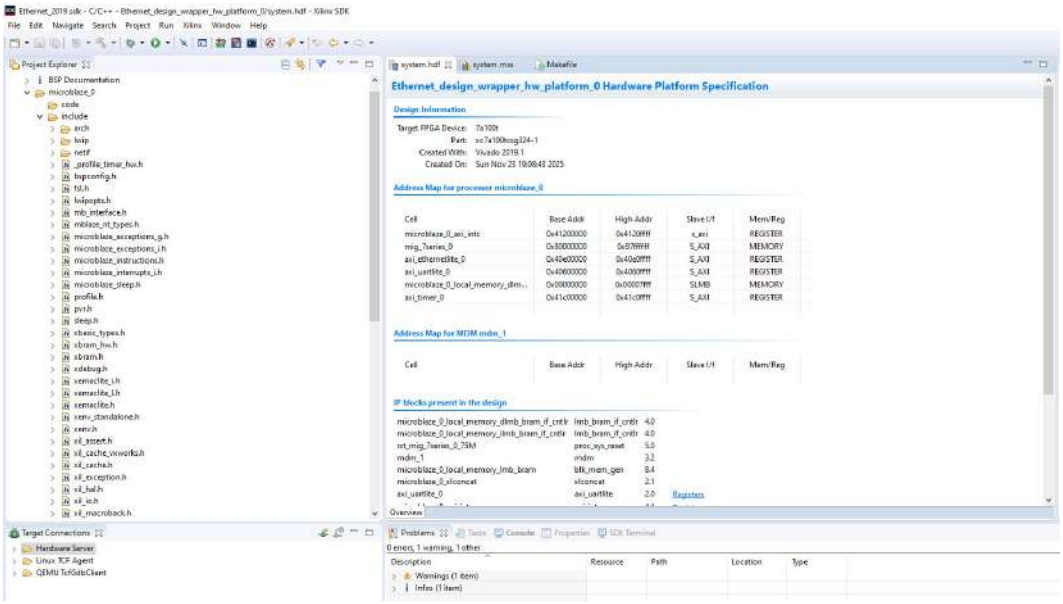


Figura 6.1: Ventana del proyecto SDK.

Una vez exportado el hardware, el programa trae predefinidos una serie de proyectos entre los que podemos encontrar una comunicación por protocolo IP y puerto Ethernet que otorga a la placa la dirección 192.168.1 en el puerto 10 desde la que se recibe la información (figura 6.2).

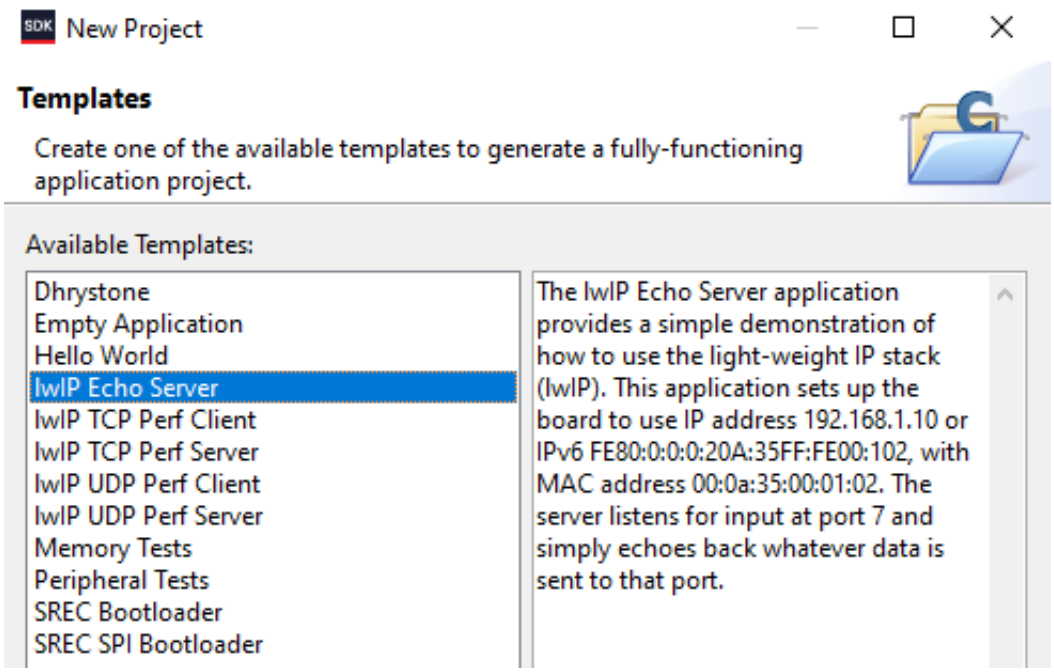


Figura 6.2: Proyectos predefinidos en SDK.

Para poder utilizar la placa como un ordenador unipersonal provisto de un sistema operativo Linux, utilizamos esta plataforma desde la que cargar el archivo petalinux a través del puerto JTAG.

Introduciendo en una tarjeta SD los programas y aplicaciones que queremos que se ejecuten en la FPGA y mediante protocolo IP iniciamos sesión de usuario con el servidor petalinux donde ejecutaremos nuestro controlador adaptativo mediante la interfaz de texto (terminal) que nos proporciona Linux.

Los archivos *hardware* del diseño (figura 6.3) en la plataforma VITIS son:

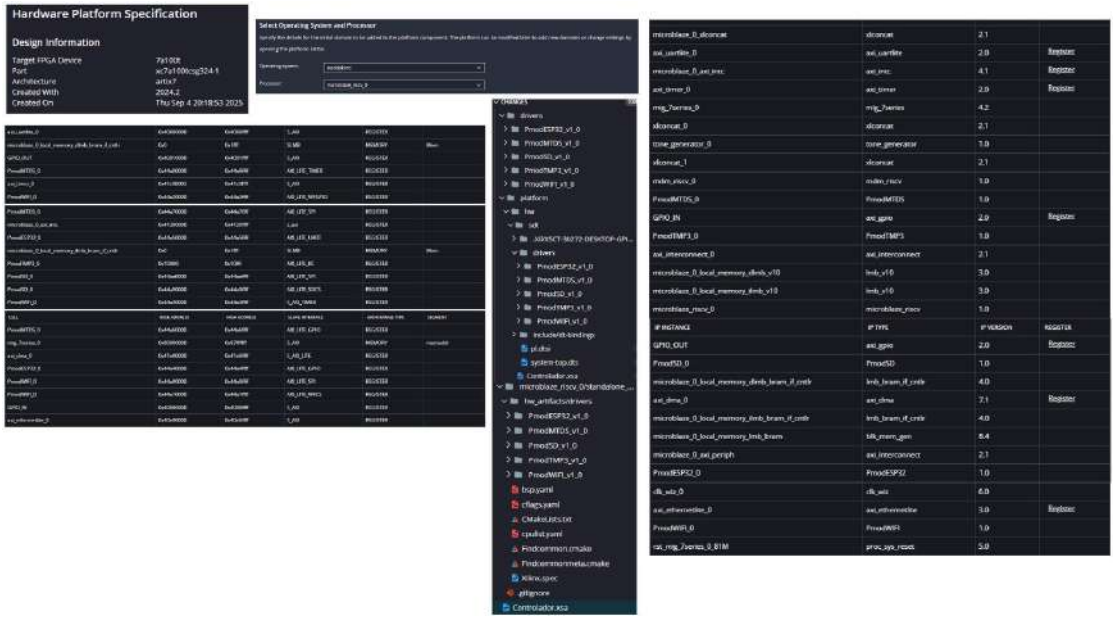


Figura 6.3: Archivos hardware en la plataforma VITIS.

6.2. Petalinux

Con el procesador RISC V (*hardware*) implementado en la FPGA hacemos uso de la herramienta **Petalinux** de *Xilinxs* para cargar un sistema operativo *linux* [4].

Este sistema operativo nos permite gestionar los recursos del *hardware* mediante una interfaz por **comandos** donde poder instalar las librerias y herramientas necesarias para ejecutar archivos python (*.py*) o script de shell (*.sh*).

Nuestro controlador es programado en *python* y cargado en el controlador mediante la línea de comandos de *petalinux*. Todos estos archivos están almacenados en una memoria SD (*flash*) configurada previamente con todos los archivos necesarios.

Para instalar la plataforma Petalinux [4] utilizamos una máquina con el sistema operativo Ubuntu 20.04.6 LTS (Focal Fossa) y los programas de Vivado, Vitis y Petalinux en su versión 2023.1.
Además, añadimos las siguientes librerías necesarias para su instalación: *iproute2, gawk, python3, python, build-essential, gcc, git, make, net-tools, libncurses5-dev, tftpd, zlib1g-dev, libssl-dev, flex, bison, libselinux1, gnupg, wget, git-core, diffstat, chrpath, socat, xterm, autoconf, libtool, tar, unzip, texinfo, zlib1g-dev, gcc-multilib, automake, zlib1g:i386, screen, pax, gzip, cpio, python3-pip, python3-pexpect, xz-utils, debianutils, iputils-ping, python3-git, python3-jinja2, libegl1-mesa, libsdl1.2-dev, pylint.*

Una vez instalados todos archivos necesarios procedemos a ejecutar el archivo *.run* de Petalinux (figura 6.4). Este ejecutable nos genera todo el entorno Petalinux y una serie de comandos que podemos ejecutar en el terminal para crear el proyecto.

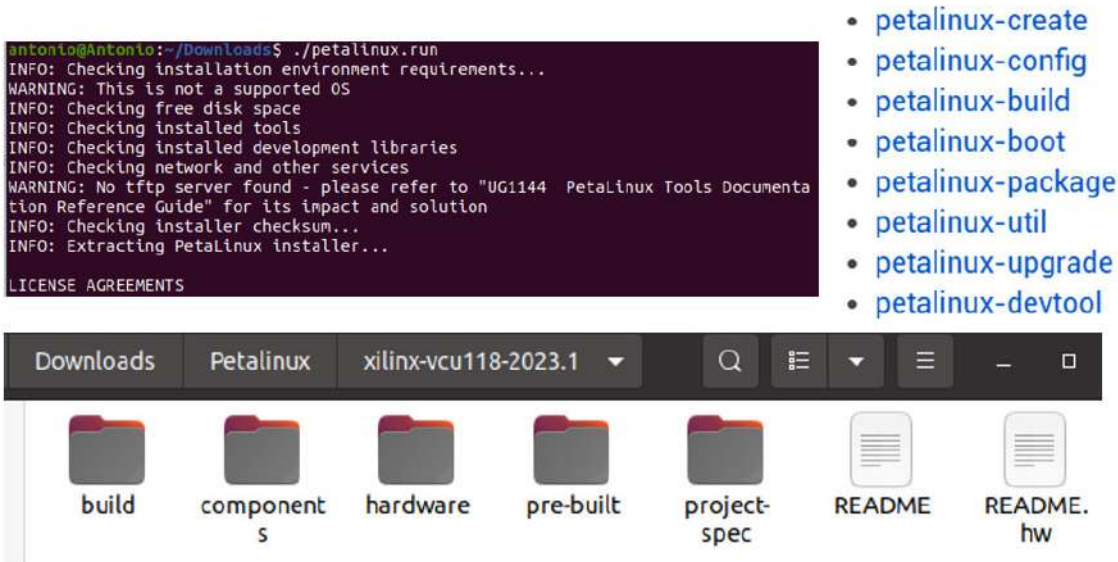


Figura 6.4: Archivos y comandos de Petalinux.

Una vez implementado el entorno Petalinux, creamos un proyecto (figura 6.5) donde conociendo el tipo de entorno *hardware*, procesador con una tipología compatible con Microblaze, que nos permite implementar esta funcionalidad, añadimos su extensión *.bsp* que nos incluyen imágenes del sistema, secuencia binaria y cargadores de inicio prediseñados [18]. Estas herramientas integradas permiten que un solo comando implemente e inicie estos elementos, ya sea en el hardware físico o en el emulador de sistema completo QEMU incluido, que nos sirve como base de nuestro entorno.

```
antonio@Antonio:~/Downloads/Petalinux$ petalinux-create -t project -s ./Microblaze.bsp
INFO: Create project:
INFO: Projects:
INFO:   * xilinx-vcu118-2023.1
INFO: Has been successfully installed to /home/antonio/Downloads/Petalinux/
INFO: New project successfully created in /home/antonio/Downloads/Petalinux/
```

Figura 6.5: Proyecto Petalinux.

Una vez ejecutado el proyecto se abre la siguiente consola (figura 6.6).

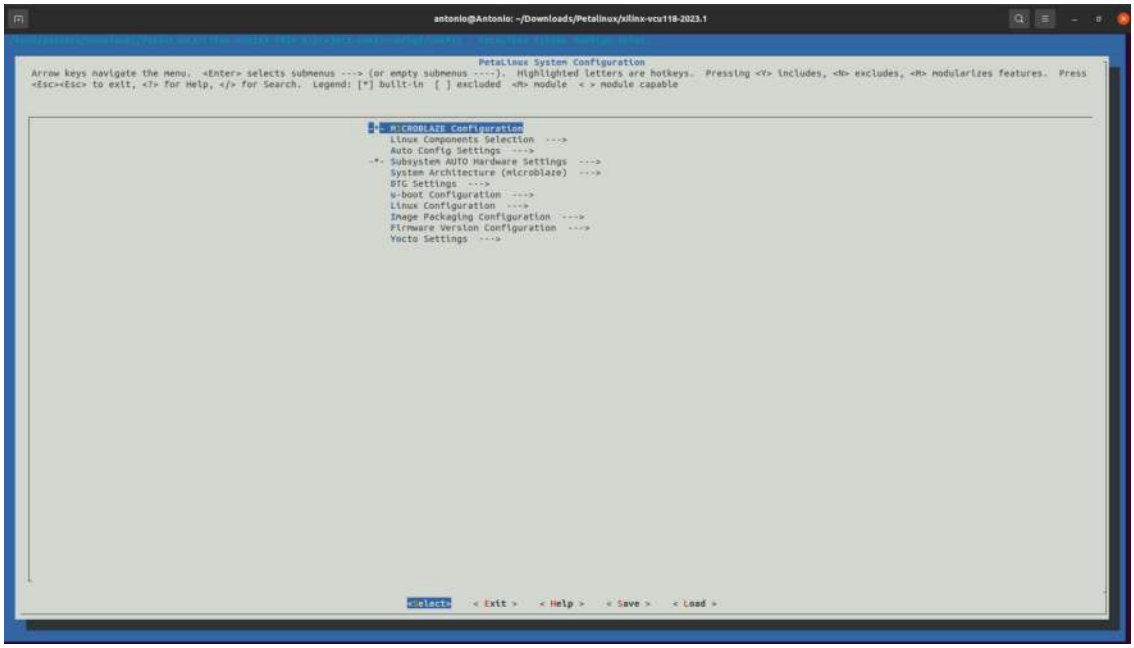


Figura 6.6: Ventana Petalinux.

Para particularizar la plataforma a nuestro diseño en Vivado, sustituimos el archivo *system.xsa* con nuestro diseño exportado de Vivado (figura 6.7). Con ello, la plataforma se adapta a la configuración presentada en nuestro diseño.

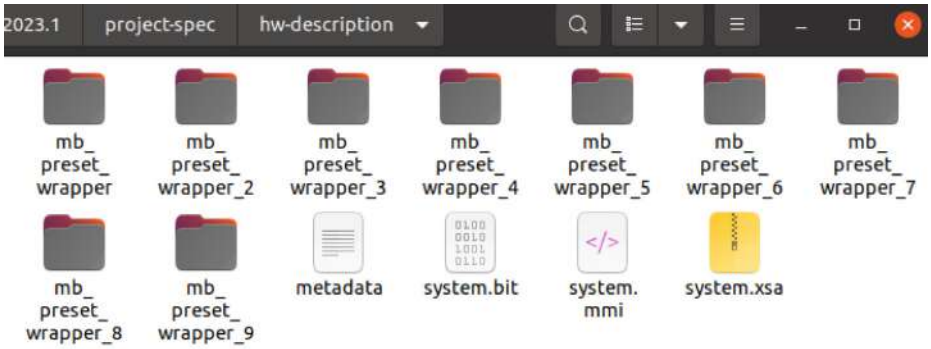


Figura 6.7: Archivos de hardware en el proyecto Petalinux.

Para configurar la interfaz de Petalinux, podemos modificar los archivos almacenados en la librería *meta-user* (figura 6.8). En ella, se cargan los componentes de la interfaz del sistema operativo.

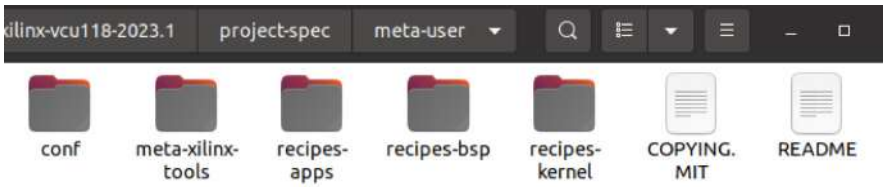


Figura 6.8: Carpetas de configuración de la interfaz Linux.

Para la configuración del sistema operativo, el propio Petalinux nos provee de diferentes librerías linux que son registradas en la carpeta **Images** del proyecto (figura 6.9). Aquí podemos añadir las librerías utilizadas en el archivo python *.py* donde definimos la lógica del controlador.

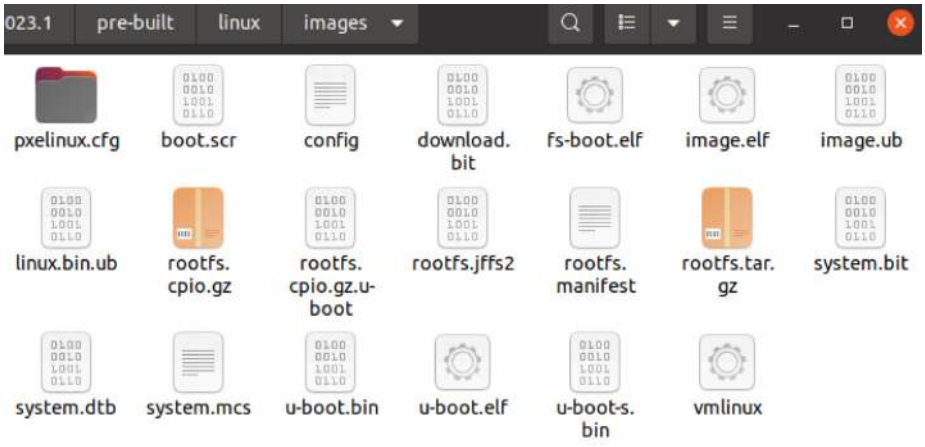


Figura 6.9: Archivos de hardware en el proyecto Petalinux.

Por último, podemos simular el petalinux (figura 6.10) en el simulador de QEMU [24].

Figura 6.10: Archivos de hardware en el proyecto Petalinux.

7. I/O link

Para el control de los sensores. Un maestro IO-Link con interfaz IIoT con protocolo API REST, permite una comunicación eficiente y estandarizada con los sensores comerciales de temperatura, caudalímetros, sonda de microondas, sensor NIR y pH-metros [16].

Para ello, los sensores envían sus lecturas a un maestro IO-Link, que envía los datos por protocolo TCP/IP al controlador, lo que permite el control en tiempo real de la planta.

IO-Link también nos ofrece ventajas para el IIoT. Permite la integración perfecta de los sensores en la infraestructura ethernet de la planta, lo que aumenta la eficiencia y la productividad. Gracias a la comunicación estandarizada por protocolo TCP/IP, nos ayuda a reducir el tiempo de inactividad en la almazara al permitir la detección temprana de problemas y una respuesta rápida en reportar cualquier error [16].
En los anexo (apartado 15.3) se detallan las ventajas, aplicaciones y esquemas del uso de esta configuración.

8. Diseño de la almazara

Partiendo de la capacidad de diseño a instalar 2500/2200 $\frac{Kg}{h}$ y las etapas del proceso (figura 8.1), seleccionamos los componentes básicos de la almazara.
Para garantizar la adaptación total de todas las máquinas en el proceso, seleccionamos los equipos provenientes de un único fabricante, es por ello que debido a la gran reputación en construcción de almazaras cercanas en la zona, nos decantamos por la empresa **Pieralisi** que cuenta con un historial de más de 41.500 plantas instaladas, siendo muchas de ellas automatizadas con autómatas industriales.
Los elementos instalados en el orden de producción son:

1. Elevador de cinta con deshojador
2. Lavadora Óptima L-10
3. Tolvín y Sinfin elevador
4. Molino de martillos modelo factoria
5. Batidora Simplex 3C 600L
6. Monobomba P50 del aceite y P60 para el alperujo
7. Decanter EFFE-3
8. VibroFiltro Fattoria
9. Separador vertical PLUTON
10. Depósitos

El esquema final con la disposición de cada elemento con los planos de situación y localización se encuentra en el apartado de planos (sección 13) con un presupuesto orientativo desarrollado en la sección (14) .

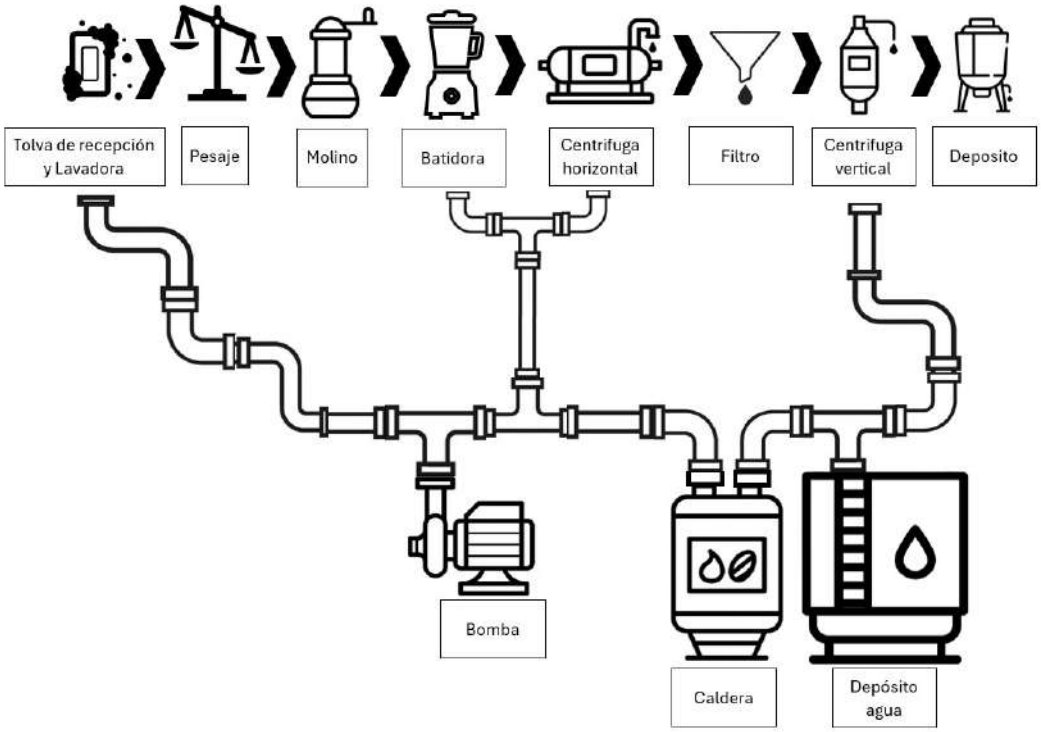


Figura 8.1: Flujo del proceso de producción del AOVE.

8.1. Tolva de recepción y tolvín con sinfin

La tolva de recepción cuenta con una cinta transportadora y una aventadora/deshojadora como primera limpieza de la aceituna (figura 8.2). El tolvín eleva la pasta de la aceituna a través de un tornillo sinfin.

Figura 8.2: Tolva y tolvín.

8.2. Lavadora

Para una limpieza exhaustiva, la lavadora Optima L10 (figura 8.3) se encarga de la salubridad del producto.

OPTIMA L10

Potencia instalada (kW)	1.9
Longitud total (mm)	3250
Anchura total (mm)	1150
Altura total (mm)	1550
Peso máquina (Kg)	500

Figura 8.3: Lavadora Optimal L10.

8.3. Molino

Como primera moliendo de la aceituna, el molino (figura 8.4) se encarga de cambiar el estado del producto a una pasa húmeda como paso previo a la centrífuga.

Este aparato necesita una gran demanda de energía debido a su potencia 22 kW, siendo necesario para la instalación eléctrica una manguera trifásica de cable de cobre de 16 mm, un magnetotérmico y un enchufe *cetac* trifásico de alto amperaje 40 A.

Figura 8.4: Molino M-25.

8.4. Centrífuga horizontal/Decanter

Para hacer la primera extracción líquido-líquido y poder romper la emulsión (separación del agua y huseo de la pulpa aceite) se realiza una extracción centrífuga (figura 8.5) con una decanter de 15 kW conectada a un cetac y un magnetotérmico de 25 A. El cable es de cobre de 16 mm.

Figura 8.5: Decanter/centrífuga horizontal.

8.5. Centrífuga vertical

Por último, una centrífuga vertical (figura 8.6) separa las impurezas. Cuenta con una potencia de $7,5\,kW$ y una protección trifásica de $16\,A$ con cable de $6\,mm$.

Modelo PLUTON

Potencia instalada (kW)	7.5
Longitud total (mm)	1270
Anchura total (mm)	1185
Altura total (mm)	1385
Número_polos	4
Voltaje (V)	220/380 - 480/600
Frecuencia (Hz)	50 / 60
Puesta en marcha	diretto
Velocidad (rpm)	6400
Diámetro tambor (mm)	316
Peso máquina (Kg)	1120
Peso tambor (Kg)	127

Características

De hecho, **permite lavar el interior del tambor y sus placas**

- con la máquina en funcionamiento;
- sin tener que desmontar el tambor.

Garantiza un **aceite siempre limpio y sin sedimentos** para todo el periodo de la campaña, gracias a la facilidad de limpieza del tambor en cualquier momento, permitiendo realizar la separación con la máquina a pleno rendimiento.

La separación se realiza **sin adición de agua** y, por lo tanto, sin producción de agua contaminada.

Figura 8.6: Centrífuga vertical.

8.6. Bombas

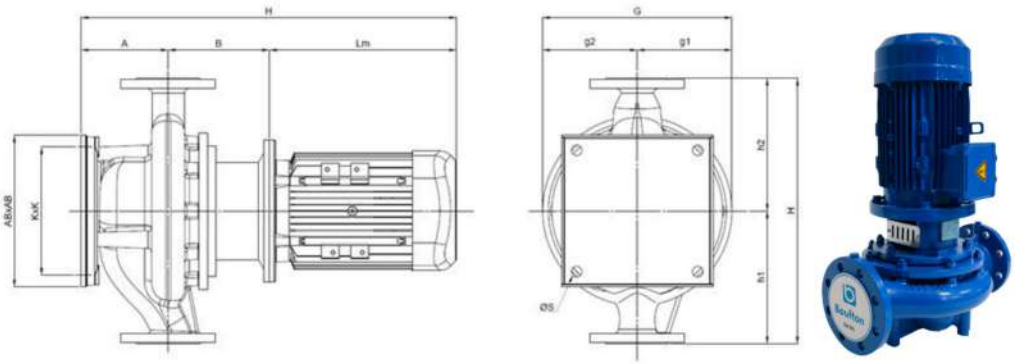
Para gestionar el movimiento de la pasta de aceituna y el alperujo se provee de una bomba de pistón (figura 8.7) de $5,6\,kW$ para el movimiento del aceite y un bomba salomónica (figura 8.8) de $1,5\,kW$ para la extracción del alperujo en el decanter.

Figura 8.7: Bomba de pistón para el transporte del aceite.

Figura 8.8: Bomba salomonica para la extracción del alperujo.

Bomba para el abastecimiento de agua

En el suministro de agua en el patio para la lavadora como en la fábrica con la línea de la batidora y decanter (centrífuga horizontal) como en la centrífuga vertical, seleccionamos una bomba de poca potencia y poco caudal (figura 8.9) al necesitar el sistema poco volumen de agua y estar ubicado en una zona de poca pendiente.
La bomba es de la marca **Boulton Pumps** con serie **IL-50/125** de potencia 550 W y 18 $\frac{m^3}{h}$.



Dimensions (mm)																			
Pump type	Motor			Outside dimensions														Base Plate	Weight (kg)
	n (RPM)	P (kW)	Size	Dne	DNb	A	B	Lm*	L*	H	h1	h2	G	g1	g2	AB	K	Ø5	
50-125	1.500	0,37	71M	50	50	113	165	223	501	300	160	140	212	110	102	170	130	14	T0	34
		0,55	80M				165	244	522							170	130		T0	38,5
		0,75	80M				165	244	522							170	130		T0	40
	3.000	1,5	90S				165	246	524							170	130		T0	43,5
		2,2	90L				165	266	544							170	130		T0	46,5
		3	100L				165	292	570							200	160		T1	52
		4	112M				165	336	614							200	160		T1	60
		5,5	132S				195	361	669							200	160		T1	75
		7,5	132S				195	361	669							200	160		T1	81

Field Charts / Campos de Trabajo / Courbes de débits / Curvas de Funcionamento

IL - 1450 rpm

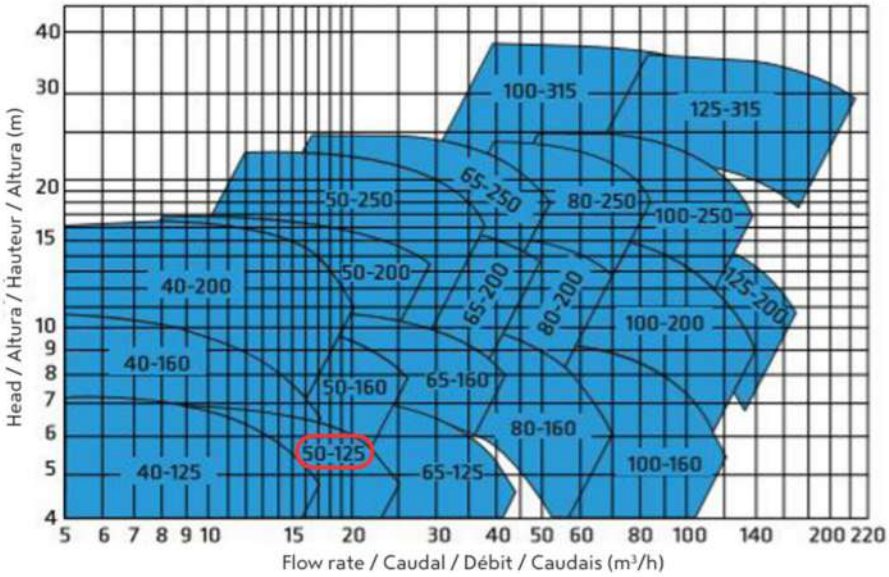


Figura 8.9: Bomba centrífuga de agua.

8.7. Caldera

Para un correcto control de la temperatura del agua se instala una caldera de hueso de aceituna (figura 8.10) de potencia máxima de 60 kW.

	60	80	100
Tarifa	Cód.: 1D3100607 Consultar	Cód.: 1D3100807 Consultar	Cód.: 1D3101007 Consultar
Clase	Clase 5 (EN303-5:2012)	Clase 5 (EN303-5:2012)	Clase 5 (EN303-5:2012)
Potencia nominal útil	60,97 kW	80 kW	98,03 kW
Potencia quemada	67,71 kW	89,03 kW	109,20 kW
Rendimiento	90,04%	90,02%	90%
Presión máxima de trabajo	3 bar	3 bar	3 bar
Presión de prueba hidráulica	4,5 bar	4,5 bar	4,5 bar
Temperatura máxima de trabajo	90 °C	90 °C	90 °C
Tensión	230 V - 50 Hz	230 V - 50 Hz	230 V - 50 Hz
Potencia eléctrica consumida	1,01 kWh	1,04 kWh	1,04 kWh
Consumo combustible a régimen	13,8 kg/h	18,4 kg/h	21,93 kg/h
Consumo medio al día	Aproximadamente el 30% del consumo a régimen		
Combustibles utilizables	Astilla, viruta, pellet, hueso de frutas, cáscara de frutos secos y cualquier combustible sólido triturado EN 14961 - 1		
Volumen tolva	480 dm³	480 dm³	480 dm³
Autonomía tolva	23 h/min	17 h/min	14 h/min
Pérdida de carga en agua 10 °C	72 mbar	87 mbar	109 mbar
Pérdida de carga en agua 20 °C	32 mbar	44 mbar	54 mbar
Temperatura mínima activación bomba	40 °C	40 °C	40 °C
Contenido de agua en caldera	170 l	215 l	260 l
Temperatura media humos (caldera limpia)	155 °C (±20%)	160 °C (±20%)	164,8 °C (±20%)
Depresión chimenea requerida	-20 Pa (±30%)	-20 Pa (±30%)	-20 Pa (±30%)
Diámetro chimenea	200 mm	200 mm	200 mm
Caudal de humos medio	107 Nm³/h	142 Nm³/h	173 Nm³/h
Volumen cámara de combustión	135 dm³	175 dm³	215 dm³
Dimensión apertura cámara de combustión	490x395 mm	490x395 mm	490x395 mm
Caudal válvula de descarga térmica	645 l/h	860 l/h	1.075 l/h
Peso caldera (tolerancia ±10%)	830 kg	910 kg	990 kg
Dimensiones	1.500/2.600/725 mm	1.500/2.600/925 mm	1.500/2.600/1.125 mm

Figura 8.10: Caldera de ACS en la almazara.

8.8. Elementos adicionales

Figura 8.11: Caldera de ACS en la almazara.

9. Estructura de la base de datos

Para gestionar la interfaz de comunicación entre el mecanismo de adaptación (programa SCADA) y el controlador ajustable (FPGA), diseñamos una base de datos **firebird** con tablas de solo lectura y escritura donde se insertan los **datos de consigna** del usuario utilizando el programa SCADA y se leen los datos de los sensores ubicados en la planta que son insertados por la FPGA tras la lectura de estos sensores cada un periodo de muestreo definido en los parámetros.

Comenzamos el diseño con las tablas de los datos nominales y máximos de los elementos ubicados en la almazara (figura 9.1), en ellas se definen su potencia nominal, flujo másico o caudal máximo, sus dimensiones, materiales y algunos datos singulares del aparato. Estas tablas pueden albergar varios registros, por si se tiene redundancia o repuesto en la planta.

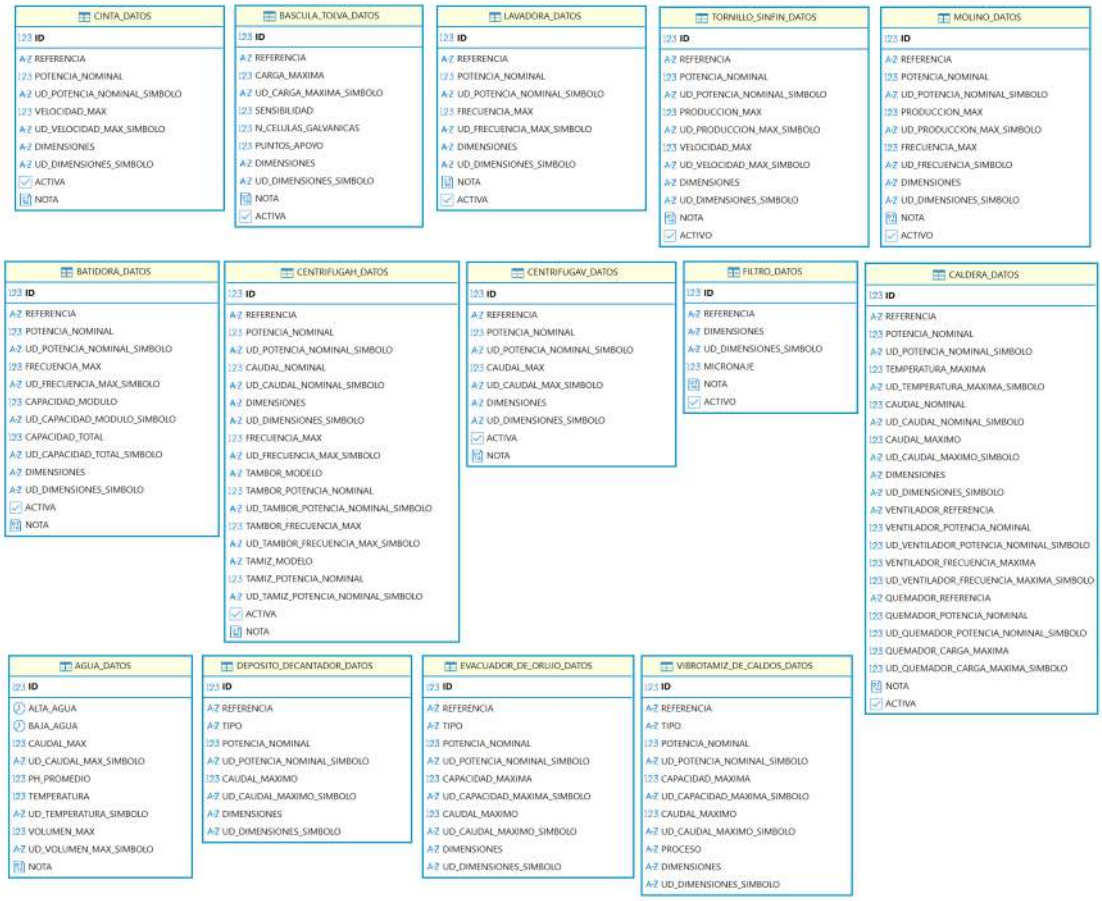


Figura 9.1: Tablas de los datos de los aparatos.

9.1. Estructura de los datos de consigna, válvulas y dosificadores

Para la estructura de las tablas de los **parámetros de control**, actualizada en base a la señales de referencia del usuario, definimos una tabla con los parámetros actualizados de referencia (figura 9.2), que leerá la FPGA como datos de consigna. Esta tabla se relaciona a su vez con las **válvulas** y los **dosificadores** en una relación 1:1 para identificar el dispositivo a controlar.

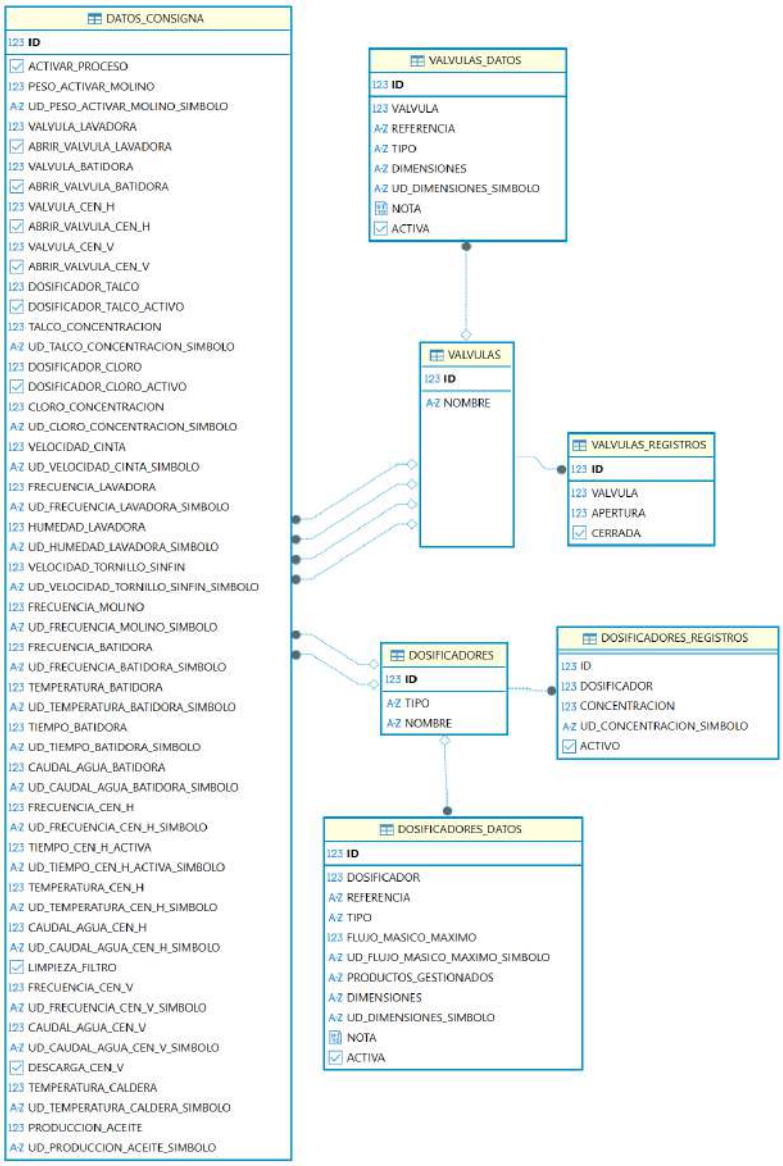


Figura 9.2: Tablas con los datos de consigna, gestión de las válvulas y dosificadores.

9.2. Estructura de las bombas, los depósitos y las entradas

Debido a la configuración de la almazara, definimos una estructura particular para la gestión del control de las bombas, depósitos y entradas de la almazara, donde realizamos una estructura de tablas 1:N con unas tablas principales, *BOMBAS* y *DEPOSITOS* y sus respectivas tablas de registros y datos (figura 9.3).

Para gestionar la propiedad del aceite procesado en la almazara, desarrollamos una tabla de propietarios donde se definen sus principales valores fiscales (NIF, DIRECCION, CODIGO_POSTAL, IBAN, BIC...) y su relación con el registro de entradas *ENTRADAS* y registro de los depósitos *DEPOSITOS_REGISTROS*.

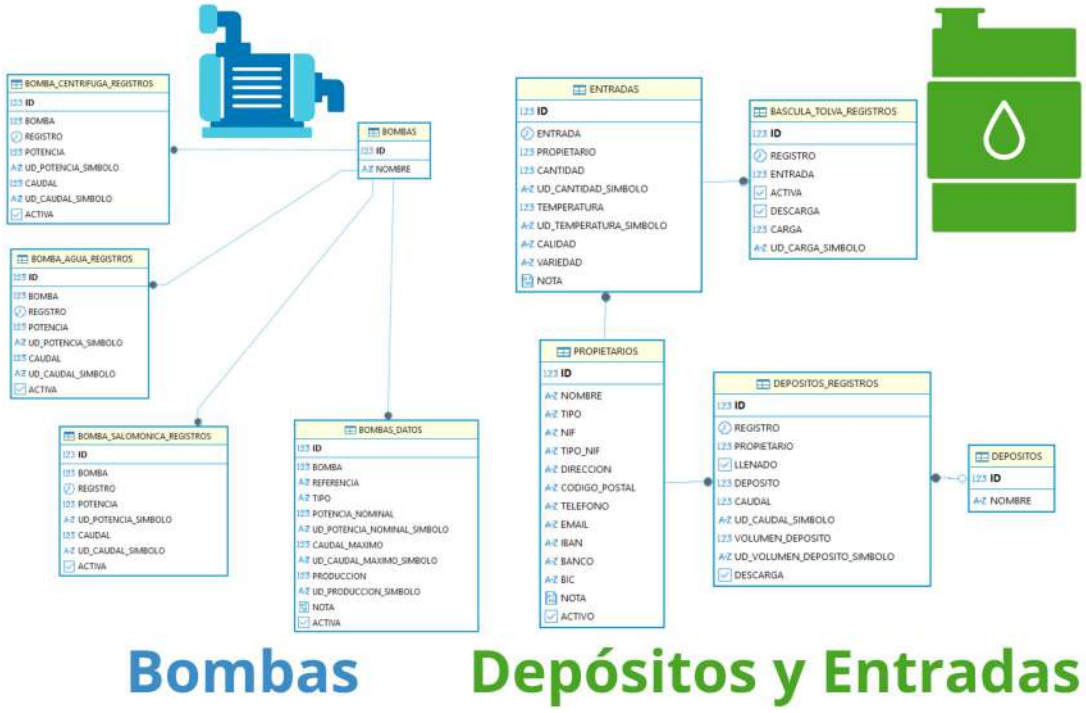


Figura 9.3: Tablas de las bombas y los depósitos.

9.3. Tablas de registros de cada aparato

Por último, definimos las tablas de escritura de los valores de funcionamiento de los aparatos y datos de los sensores ubicados en la instalación (figura 9.4), que son

escritos y conservados tras un periodo de muestreo válido.

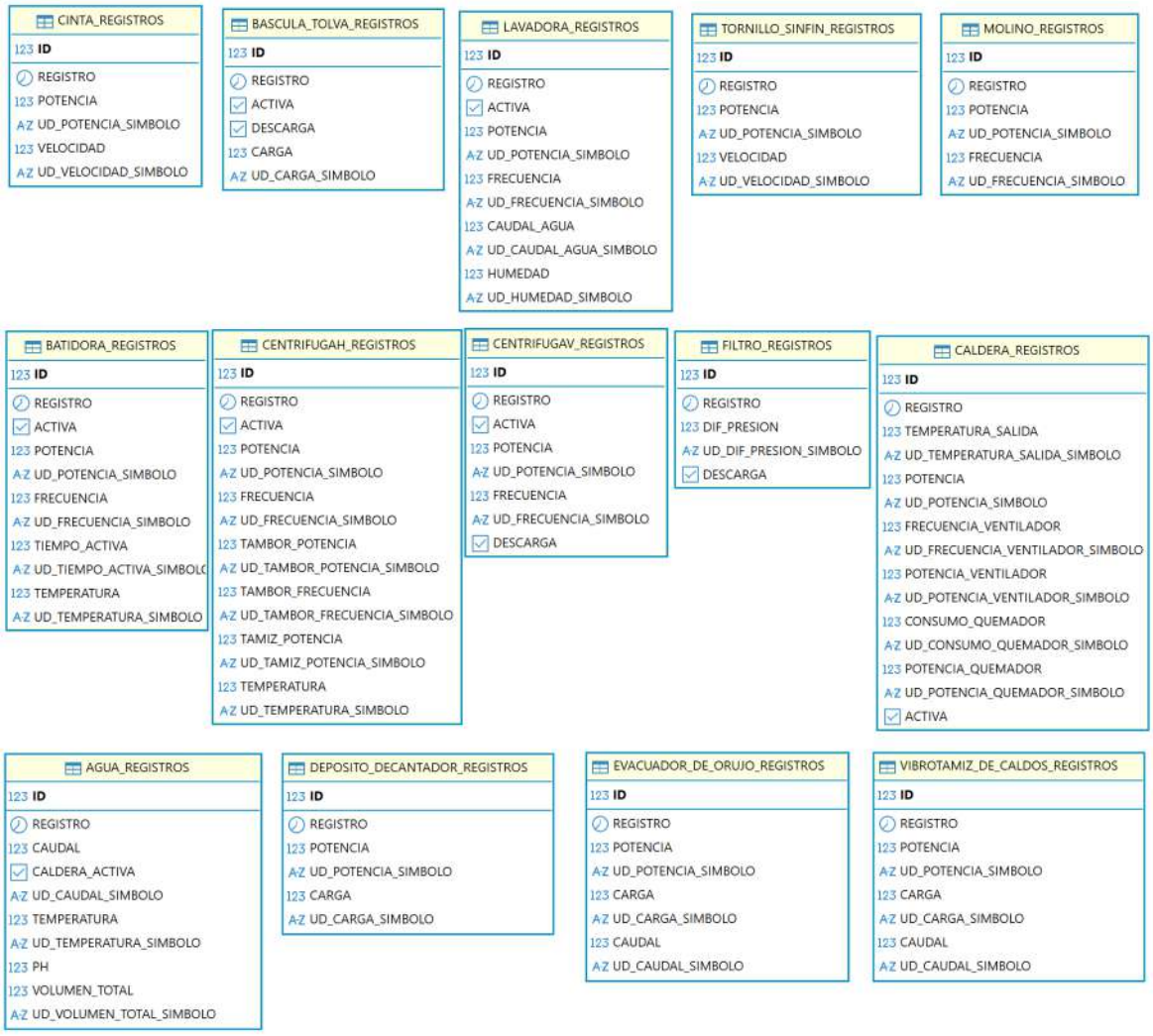


Figura 9.4: Tablas de registros de los elementos a controlar y sensores.

10. Sistema SCADA de control de la almazara

En la etapa de supervisión y mecanismo de adaptación ubicado en un servidor Windows externo y comunicado mediante una base de datos **Firebird** con el controlador ajustable, desarrollamos una pequeña aplicación windows en formato SCADA con la plataforma Delphi 12 FMX en un entorno de desarrollo con el lenguaje de programación Pascal.

La plataforma de la empresa Embarcadero nos proporciona una aplicación donde gestionar el *back end* y *front end* del servicio, donde desarrollamos una clase con los eventos y valores que puede modificar el usuario durante la producción del AOVE. Esta aplicación contempla la gestión del patio y fábrica de la Almazara, el control de los líquidos de la planta (aceite y agua) y el almacenamiento del producto final.

10.1. Conexión a la base de datos

Para realizar la conexión a la base de datos ubicada en un **docker** del propio servidor con la puerta de enlace *3055:3050*, es decir, en el puerto *3055* de forma externa y *3050* internamente, utilizamos el componente de la clase (figura **TFDConection**) que nos proporciona Delphi y diferentes componentes que configuran las units y parámetros necesarios para su gestión (TFDGUIxWaitCursor, TFDTransaction (que crea una transacción con la base de datos) y TFDPhysFBDriverLink) (figura 10.1).

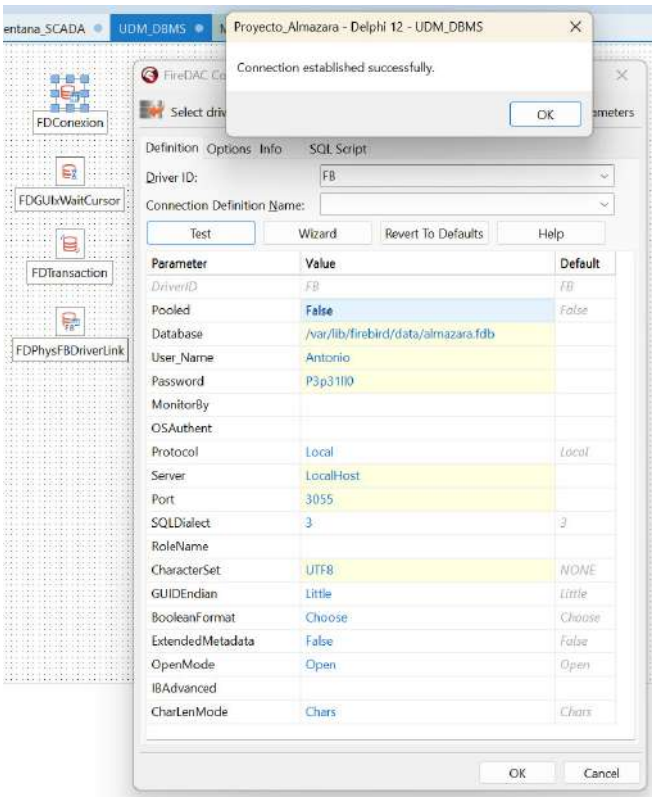


Figura 10.1: Componentes y parámetros de configuración de la conexión con la base de datos.

10.2. Front end de la aplicación

En el diseño de la intefaz del programa SCADA utilizamos la imagen (figura 8.1) que cargamos en la aplicación utilizando el objeto *TImageControl*, en donde se integran los objetos de la clase *TLabel* que albergan el título del parámetro a mostrar y que contiene a otro objeto de la clase *TEdit* (recuadro blanco) que contiene el valor y la unidad. Para seleccionar la concentración de cloro en la lavadora y talco en la centrífuga horizontal, intercambiamos el objeto *TEdit* por otro del tipo *TComboBox* que representa una lista desplegable donde seleccionar los valores de consigna. En la gestión del caudal de agua en cada máquina, colocamos los datos obtenidos por los caudalímetros, donde al pulsar podemos cambiar el valor de consigna, que se coloca arriba con un color verdoso, hasta que llegue a ese valor (figura 10.2).

Para posicionarlos en el dibujo y teniendo en cuenta que la ventana es redimensionada por el usuario, al crear el objeto principal *TForm* recopilamos los datos iniciales que vamos modificando cada vez que se redimensiona la ventana.

En la parte inferior de la imagen, representamos los valores en tiempo real de los elementos colocados en el patio de recepción, fábrica y de la gestión de los líquidos (cáldera, dosificadores y válvulas). En la representación de cada elemento colocamos un objeto del tipo *TRadioButton* que indica el estado de la máquina (activo o inactivo), el nombre del elemento y sus valores de control (frecuencia/velocidad, temperatura o carga), que pueden ser editables, y la potencia absorbida por la máquina de solo lectura (figura 10.2). La concentración en los dosificadores son las obtenidas por los sensores electroquímicos colocado en las entradas a la lavadora y centrífuga vertical.

Para mejorar la presentación de la aplicación, en el título incluimos un GIF animado con el movimiento de diferentes franjas verdes y de color crema que se mueven asíncronamente, esta animación también nos sirve como indicativo de que la aplicación se ha bloqueado o no funciona correctamente (figura 10.2).

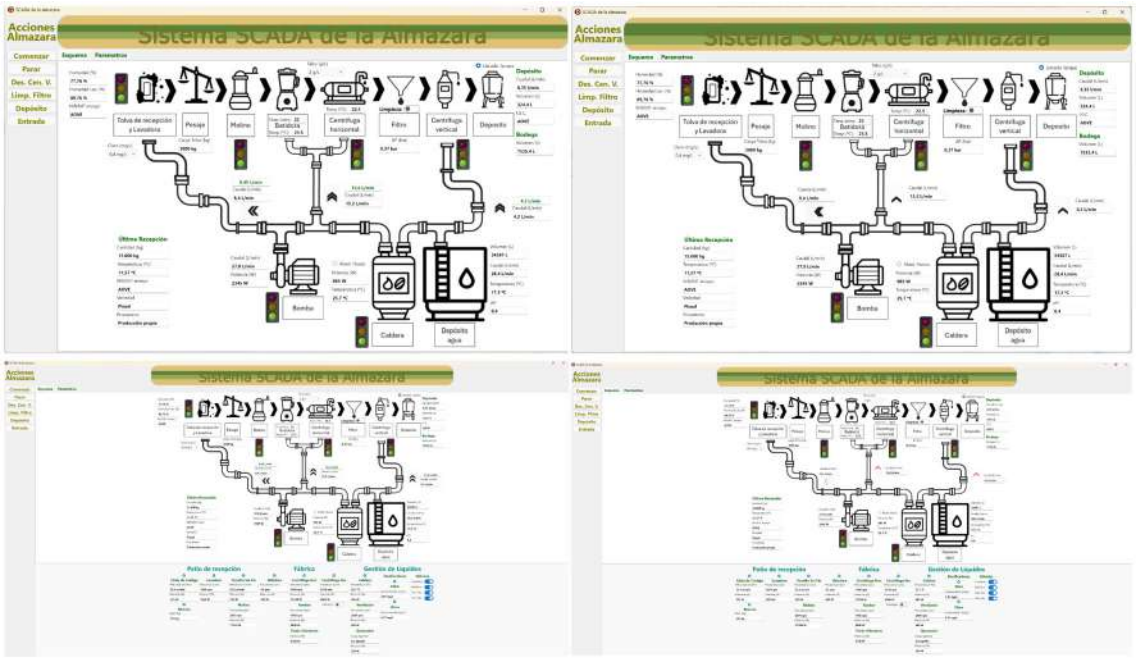


Figura 10.2: Aplicación SCADA en producción en regimen transitorio y permanente.

Utilizando un objeto de la clase *TTabControl* añadimos una ventana adicional donde se representan todos los datos de cada dispositivo de la almazara. Esta ventana sirve de detalle de la ventana principal (figura 10.3).

En ella, solo son editables los valores en color verde con los valores de consigna, el resto de valores son dependientes del estado de la almazara y nos sirven para conocer su estado y posibles ineficiencias, problemas o fallos de control. Cuando el valor coincide con el de consigna, el texto se oculta y al pulsar sobre el parámetro, aparece un texto encima de color verde donde cambiar el valor de consigna.

En la zona de **historiales** de la ventana, accedemos a los registros de eventos de cada componente, así como la gestión de las entradas a la almazara.

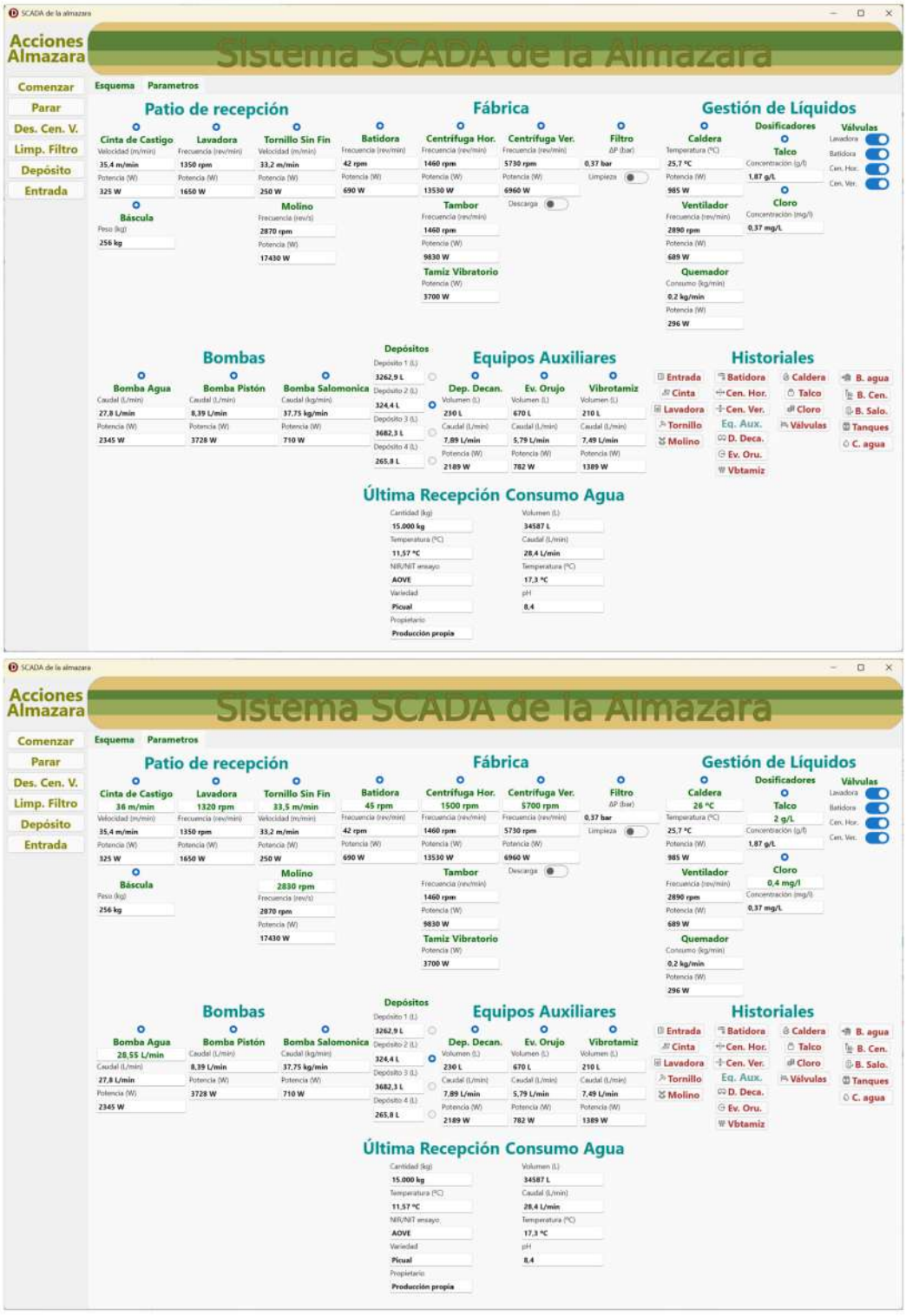


Figura 10.3: Parámetros de control y muestreo de la almazara.

Para indicar el estado del proceso de producción en cada sistema utilizamos diferentes GIFs de un semáforo (verde/activo, rojo/parado y ambar/regimen transitorio)

y otros GIFs de flechas y cruz roja para indicar el paso de agua por las tuberías de agua (figura 10.4).

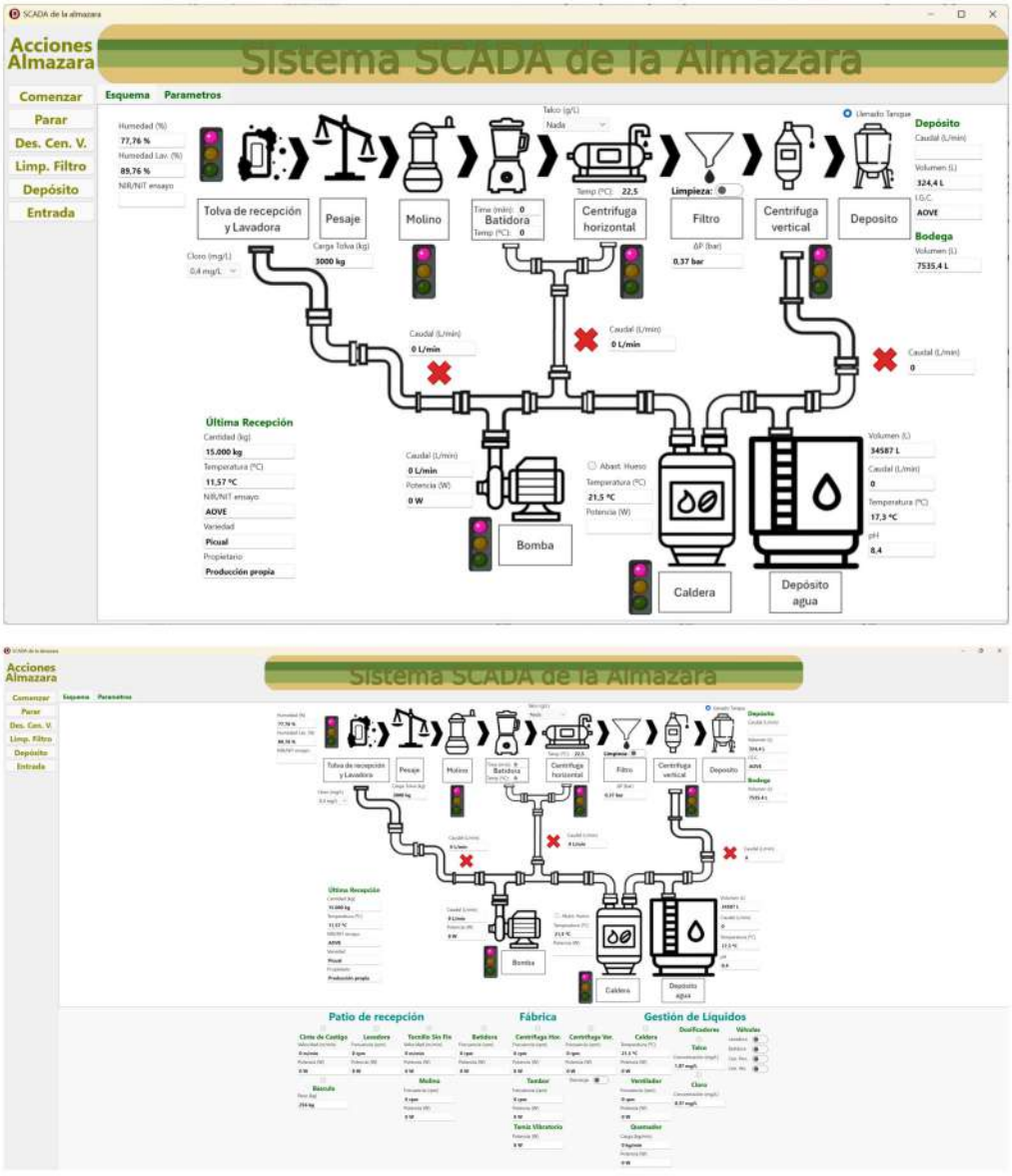


Figura 10.4: Aplicación SCADA parada.

Además, definimos 4 accciones principales (más utilizadas) que mostramos en un menú vertical colocado a la izquierda del formulario. Las **acciones** son: el **comienzo** y **paro** de la producción, la **descarga** de la centrífuga vertical y **limpieza** de los filtros, que solo se habilitan en momentos específicos del proceso de producción (parada de la centrífuga vertical para la descarga y el corte de caudal a los depósitos

para la limpieza del filtro) y por último, la indicación del **cambio** del depósito y **registro** de una nueva entrada (figura 10.5).

Figura 10.5: Registro de una nueva entrada y cambio del depósito.

Por último, para tener un registro de cada dato recopilado en cada equipo, en la zona de **historiales** de la pestaña de datos, visualizamos un histórico con el registro del tiempo y sus valores muestreados.

Como ejemplo, este sería el formulario de registros de la lavadora con un periodo de muestreo de 30 segundos (figura 10.6).

Figura 10.6: Histórico de registros de la lavadora.

11. Conclusiones y líneas futuras

Durante el desarrollo de este trabajo de investigación he llegado a diferentes conclusiones a cerca del uso de una FPGA como unidad de control para un proceso industrial, en nuestro caso la gestión de una almazara.

1. La escasa y difícil corroboración de la información expuesta en internet. Durante la recogida de información a partir de una revisión sistemática de todo el conocimiento de mi investigación, solo aparecen autores y proyectos hechos por la comunidad que en muchas ocasiones no he sido capaz de replicar. Fallos en el modelo en Vivado como errores en la comunicación entre los diferentes IPs, fallos abstractos, saltos entre los pasos con incongruencias entre las etapas...

2. Poca información otorgada por *Xilinxs*. Siguiendo al anterior punto, la empresa que desarrolla las FPGAs nos provee de una escasa información de su *software* con pocos ejemplos y casos reales, además sus comentarios en los foros de la plataforma son muy etérios y difíciles de comprender.

3. Uso de una placa de desarrollo de un proveedor externo *Digilent, a National Instruments Company*. La empresa que ha desarrollado la placa NEXYS A7 100T que contiene una FPGA Artix-7 es un comercializador externo que no aparece de base en el software de *Xilinxs* y que debes añadir modificando ciertas carpetas de los programas. Además, en muchos casos se perdía la configuración de la placa al abrir el programa dando fallos en la síntesis y teniendo que volver a cargar el programa.

4. Uso de una versión específica y anterior de los programas en un sistema operativo específico con la instalación de unas librerías concretas. En el desarrollo del apartado de **Petalinux** tuve que utilizar un sistema operativo específico *Ubuntu 20.04.6 LTS* y la versión 2023.1 del software de *Xilinxs* que descubrí a base de prueba y error con la lectura de varios proyectos hechos por la comunidad de *Petalinux*, además para la instalación de las diversas librerías había que instalarlas en un orden concreto al tener dependencias entre ellas.

5. Modificación de los archivos internos de los programas. No suele ser frecuente en los programas, la modificación de los archivos internos guardados en carpetas ocultas pero en algunos casos como en la configuración correcta de Vivado me he visto obligado a modificar estos archivos con riesgo a la pérdida del programa.

6. La pérdida de la ventaja del uso de una FPGA frente a otros dispositivos. Al cargar un procesador en la FPGA, se pierde la principal ventaja de estos dispositivos. La propiedad de poder tener un diseño específico del *hardware* realizado para la ejecución de un proceso concreto que se pierde si modelamos un procesador RISC V en esta. En vez de realizar este diseño, sería mejor el uso de un autómata, arduino o raspberry Pi que tienen más puertos específicos de control y muestreo de la planta.

7. El gran desarrollo de los procesadores actuales. El principal competidor de esta tecnología son los procesadores específicos desarrollados por Intel, AMD o Apple que con la nueva tecnología de fabricación GAA *Gate all around* pueden aumentar sus capacidades y rendimientos compitiendo solo las FPGAs en consumo de energía.

8. Futuro prometedor a esta tecnología. Aunque actualmente este campo de aplicación está empezando a desarrollarse y no hay mucha información de como gestionarlo, he visto que estos dispositivos pueden tener un nicho de mercado prometedor para el uso de aplicaciones específicas como puede ser la domótica, el control de máquinas eléctricas de forma más precisa como pueden ser máquinas CNC o tornos automatizados, así como en el campo de la pulvimetalurgia que necesitan de un control muy dinámico y específico del proceso.

9. En el ámbito más concreto del proyecto de la almazara, el uso de un controlador adaptativo es la opción más coherente para la gestión del proceso de producción ya que la materia prima (la aceituna) es muy diversa dependiendo del cliente/proveedor y es necesario cambiar constantemente los parámetros de cada elemento.

10. Es necesario en este proyecto, el desarrollo de una aplicación para windows con la que gestionar la interfaz con el almazarero y el uso de una base de datos desde donde gestionar todos los parámetros y llevar un histórico de cada proceso de producción, así como del almacenamiento y gestión del aceite.

Como líneas futuras de investigación y desarrollo del proyecto sería desarrollar la implementación de petalinux en la FPGA Artix-7 y ejecutar archivos de comunicación por vía ethernet para comprobar el correcto diseño del puerto. Además, desarrollar el controlador ajustable utilizando hilos en *python* con múltiples procesos, para ver si el procesador es capaz de soportarlo.

Por parte del diseño de la almazara, contactar con los proveedores de los dispositivos y buscar los captadores y controladores específicos con tecnología I/O link para ver su colocación y correcto muestreo de las variables de estado.

En el apartado del sistema SCADA, realizaría la comunicación con la placa mediante la interfaz de la base de datos y comprobaría la gestión con los retardos y asincronismos reales del sistema, además simularía fallos en el corte de suministro, fugas y otros eventos de la planta para ver como afrontarlos y detectarlos e igual que hacemos con un archivo *testBench* simularlo en el sistema.

Referencias

[1] Z. A., E. R., A. M., B. M. y H. L., «Design and Implementation of Pulse Width Modulation Using Hardware/Software MicroBlaze Soft-Core,» *International Journal of Power Electronics and Drive Systems*, 2017. DOI: 10.11591/ijpeds.v8.i1.pp167-175.

[2] AMD, «7 Series FPGAs Configuration,» *Technical Information Portal de AMD*, 2023. dirección: https://goo.su/NamHd.

[3] AMD, *MicroBlaze Processor Embedded Design User Guide (UG1579)*. dirección: https://n9.cl/0vanr.

[4] AMD, *PetaLinux Tools Documentation: Reference Guide (UG1144)*. dirección: https://n9.cl/hnv94.

[5] AMD, *Programming an Embedded MicroBlaze Processor*. dirección: https://n9.cl/7sp17.

[6] S. Churiwala, «Designing with Xilinx FPGAs : using vivado (S. Churiwala, Ed.; 1st ed. 2017.),» *Springer*, 2017. DOI: 10.1007/978-3-319-42438-5.

[7] Comunidad, *RISC-V*. dirección: https://github.com/riscv.

[8] Digilent, *Digilent reference Pmods*. dirección: https://goo.su/Xg7aCQ.

[9] Digilent, *Digilent References Nexys A7*. dirección: https://goo.su/mbUmbc.

[10] Digilent, *Nexys 4 DDR Getting Strarted with Microblaze Servers*. dirección: https://n9.cl/bqx4o.

[11] Digilent, «Nexys-A7-XADC/src/constraints/Nexys-A7-100T-Master.xdc,» *GitHub*, dirección: https://n9.cl/m1lxe.

[12] FPGAPS, *Implementing Gigabit Ethernet on FPGA with MicroBlaze and MIG - Part 1: Vivado Design*. dirección: https://n9.cl/apebx.

[13] C. Godoy Ortega y Janeth, «Control adaptativo en tiempo real,» *QUITO/EPN/2011*, 2011. dirección: https://goo.su/WsIQsU.

[14] HDLForBeginners, *Ethernet Toolbox*. dirección: https://n9.cl/yrmwe.

[15] M. I., L. N., O. S. B. y S. S. B., «Impact of Hardware/Software Partitioning and MicroBlaze FPGA Configurations on the Embedded Systems Performances. In Q. Zhu and A. T. Azar (Eds.), Complex System Modelling and Control Through Intelligent Soft Computations,» *Springer International Publishing AG*, 2014. DOI: 10.1007/978-3-319-12883-2_25.

[16] *IO-Link*. dirección: https://io-link.com/.

[17] U. C. de Madrid, *Demos Digilent*. dirección: https://n9.cl/6zwdr9.

[18] T. McDowell, *PetaLinux 101 - Getting Started Quickly*. dirección: https://www.youtube.com/watch?v=k03r2Ud42jY.

[19] I. Nanda y N. Adhikari, «Accelerator Design for Ethernet and HDMI IP Systems for IoT using Xilinx Vivado 18.X. International,» *Journal of Innovative Technology and Exploring Engineering*, págs. 652-656, 2019. DOI: 10.35940/ijitee.J8786.0881019.

[20] K. Pranitha y G. Kavya, «A Systematic Method for Hardware Software Co-design using Vivado HLS,» *International Journal of Recent Technology and Engineering*, págs. 467-472, 2019. DOI: 10.35940/ijrte.D7008.118419.

[21] E. Tarassov, *Vivado Risc V*. dirección: https://n9.cl/szx44p.

[22] A. Taylor, *MicroZed Chronicles: Building PetaLinux for MicroBlaze — Part 2*. dirección: https://n9.cl/i5kxm.

[23] A. Taylor, *MicroZed Chronicles: Building PetaLinux for MicroBlaze — Part One*. dirección: https://n9.cl/xncmc.

[24] A. Taylor, *Perfecting PetaLinux Workshop*. dirección: https://www.youtube.com/watch?v=Tloz2tJsJow&t=4802s.

[25] J. Vico Lizana, *Cómo gestionar una almazara paso a paso*, J. Vico Lizana, ed. 2022.

[26] N. Y., M. Z. y J. L., «Research on the Design of Multi-Core Embedded System Based on Microblaze,» *International Journal of Control and Automation*, págs. 425-434, 2015. DOI: 10.14257/ijca.2015.8.12.39.

12. Índice de términos y glosarios

FPGA (*Field-Programmable Gate Array*). Dispositivo desarrollado por la empresa *Xilinx* con la característica principal de ser reconfigurable físicamente y que permite implementar hardware digital a medida.

RISC V. Tipo de arquitectura del procesador basado en un conjunto de instrucciones (ISA) de tipo RISC, abierta y *open source*.

ISA (*Instruction Set Architecture*). Un tipo de arquitectura de instrucciones donde se definen el formato de las instrucciones, los registros, los tipos de datos, los modos de direccionamiento, el modelo de memoria y el manejo de interrupciones/excepciones.

CLB (*Configurable Logic Blocks*). Unidad básica de lógica en una FPGA que agrupa recursos reconfigurables para implementar circuitos digitales.

Slices. Subunidades físicas dentro de un CLB que agrupa un conjunto fijo de recursos lógicos básicos.

CMT. Bloque dedicado dentro de las FPGAs encargados de la generación, acondicionamiento y distribución de señales de reloj.

Puerto PCI Express. Interfaz de comunicación serie de alta velocidad, basada en enlaces punto a punto *point to point).*

GTP (*Gigabit Transceiver Port*). Bloque de transceptor serie de alta velocidad para enviar y recibir datos a varia velocidades donde se pueden implementar enlaces y protocolos rápidos como PCIe, Ethernet o SATA.

XADC (*Xilinx Analog to Digital Converter*). Puerto que proporciona conversión analógica-digital donde se puede muestrear señales analógicas externas.

Banco de I/O. Conjunto de pines de entrada/salida agrupados físicamente que comparten recursos eléctricos comunes como el estándar de señal admitido (LVCMOS, LVTTL...) y una o varias tensiones de alimentación.

Espectroscopia del infrarrojo cercano NIR *Near Infrared Spectroscopy.* Técnica analítica no destructiva que estudia la interacción de la radiación del infrarrojo cercano con la materia midiendo la absorción y la reflectancia o transmitancia del compuesto.

Comunicación TCP/IP. Conjunto de protocolos de transmisión de datos en red mediante el encadenamiento de paquetes y servicios. Proporciona una comunicación fiable y flexible que proporciona un control de errores, retransmisiones y control de flujo de la red.

Relé octoacoplado. Un dispositivo de conmutación en el que la señal de control y el circuito de potencia quedan aislados mediante un optoacoplador formado por un diodo LED y un fotodetector.

Protocolo API REST *Representational State Transfer.* Arquitectura de servicios web donde se accede a los dispositivos mendiante URLs y el estándar HTTP.

Puerto Gigabit PoE+. Entrada con interfaz *ethernet* que suministra por uno de sus pines alimentación eléctrica mediante el estandar IEEE 802.3at capaz de suministrar hasta 30 W.

Puerto SFP *Small Form factor Pluggable.* Puertos donde insertar módulos SFP para coble de fibra óptica.

Bus ILMB (*Instruction Local Memory Bus*). Bus interno de la memoria local para las instrucciones. Soporta el control de accesos síncronos y normalmente de un ciclo de reloj para datos almacenados localmente. Proporciona una comunicación de baja latencia destina a la lectura de la siguiente instrucción.

Bus DLMB (*Data Local Memory Bus*). Bus interno de la memoria local para el control de los datos, este bus permite los servicios de lectura y escritura de los datos con un control síncrono y simple.

Puerto JTAG *Joint Test Action Group.* Interfaz estándar de simulación, programación y depuración definida en el IEEE 1149.1 por la IEEE Standards

Association y que permite acceder a los circuitos integrados mediante una comunicación serie

Comunicación AXI (*Advanced eXtensible Interface*). Mecanismo de conexión con Interfaz AMBA mediante transacciones maestro-esclavo con comunicación síncrona basada en un mecanismo de transferencia de información donde los receptores no siempre están activos (*handshake valid/ready* que separa la lactura y escritura de datos en canales independientes.

Interfaz AXI4-Lite (*Advanced eXtensible Interface 4 Lite*). Mecanismo de conexión con Interfaz AMBA simplificada desarrollada por ARM y orientado a una comunicación de control entre un maestro y uno o varios esclavos. Utiliza la interfaz de bus AXI4 *Advanced eXtensible Interface 4* que proporciona una comunicación con la memoria mediante ráfagas *bursts*, una alta tasa de datos y baja latencia.

DMA (*Direct Memory Access*). Mecanismo que permite transferir datos entre la memoria y los periféricos sin necesidad de intervención continua del procesador. El procesador solo configura el protocolo de comunicación y el DMA ejecuta las transferencia de datos.

Testbench. Archivo de simulación programado en HDL *Hardware Description Language* y diseñado para la verificación del diseño mediante una simulación que puede ser funcional o temporal y manual o automatizada.

Pipeline. Es una estructura física consistente en introducir registros que dividan el proceso en tramos cuya ejecución se solapa en el tiempo.

tiempo *end to end*. En una *pipeline*, es el tiempo que tarda una etapa desde que entra en esta hasta que sale. Esta propiedad define la latencia que tendrá nuestro sistema.

end to end. Tiempo de latencia en la *pipeline* desde que entra el dato hasta que sale.

Puerto MRCC (*Multi Region Clock Capable*). Puerto de entrada y salida con conexiones dedicadas a la red de distribución de la señal de reloj que garantiza el enrutamiento de una señal de reloj externa con un bajo desfase *clock skew* a lo largo del circuito.

Arquitectura AMBA. Conjunto de protocolos e interfaces de interconexión especificada por ARM.

Comunicación MII 10/100 Mbps PHY (*Media Independent Interface*).Interfaz estándar para conectar el controlador *ethernet* con el diseño físico del dispositivo, formado por un bus paralelo de datos de 4 bits, señales de control y temporización separadas con un canal de gestión con el *hardware*.

Interfaz MDIO (*Management Data Input/Output*). Canal de gestión y configuración para comunicación MAC en serie.

IIoT (*Industrial Internet of Things*). Sistema industrial donde máquinas, sensores, actuadores y sistemas de control se conectan mediante redes para capturar datos, monitorizar procesos, automatizar decisiones y optimizar la producción asegurando la fiabilidad con un control predictivo de la planta y con baja latencia. Aspecto clave en la industria 4.0.

Sensor NIR (*Near Infrared Sensor*. Captador con tecnología de detección de la radiación del infrarrojo cercano, qye mide las magnitudes de absorción, reflactancia y transmitancia del compuesto.

Base de datos Firebird. Un gestor de bases de datos relacional, *open source*, multiplataforma, que utiliza comandos SQL. Destaca por su ligereza y facilidad de despliegue al poder funcionar tanto en modo servidor como embebido en un dispositivo.

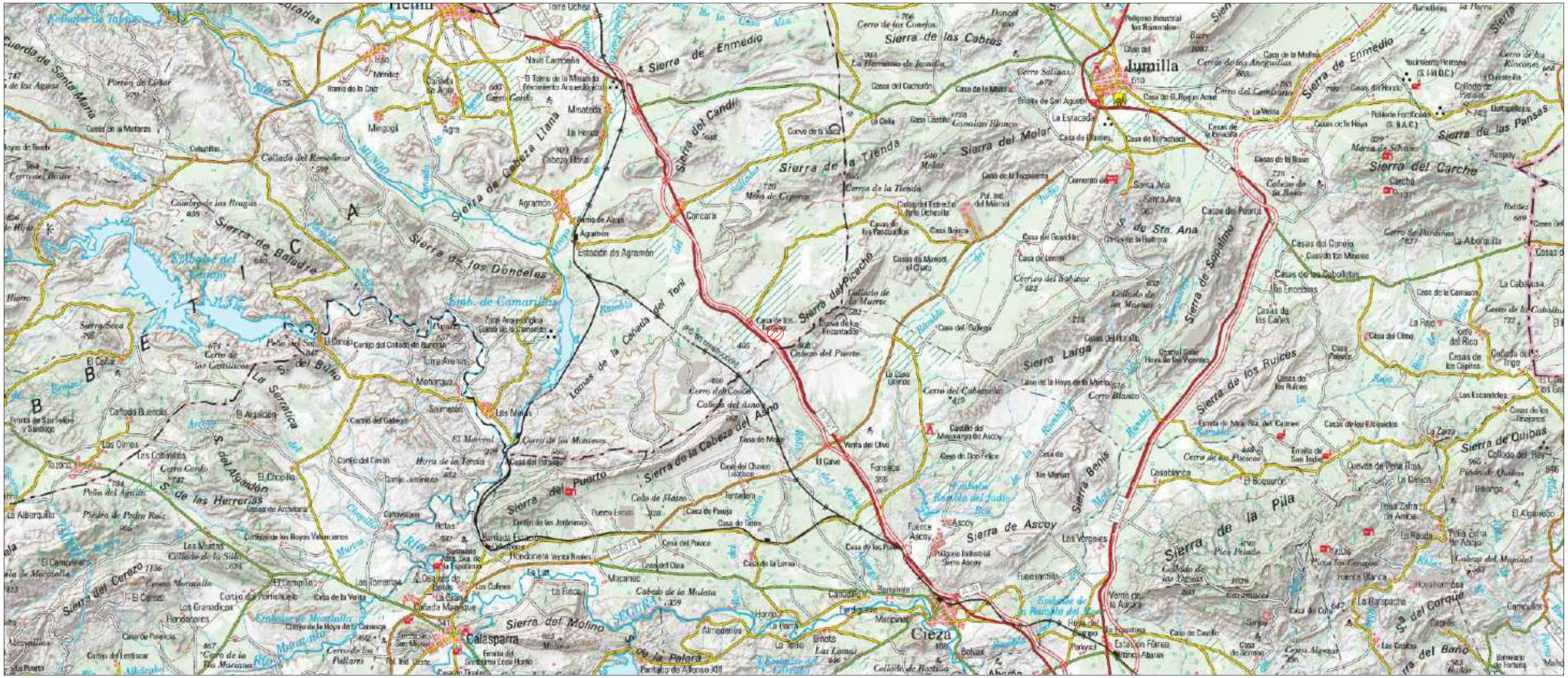
Back end. Parte del *software* que controla la lógica de negocio, el procesamiento de datos, la seguridad e identificación de los usuarios, la integración con otros servicios y el acceso a la base de datos.

Front end. Genera la interfaz del usuario responsable de la representación de los datos y la gestión de las acciones del usuario con el control de los valores modificables.

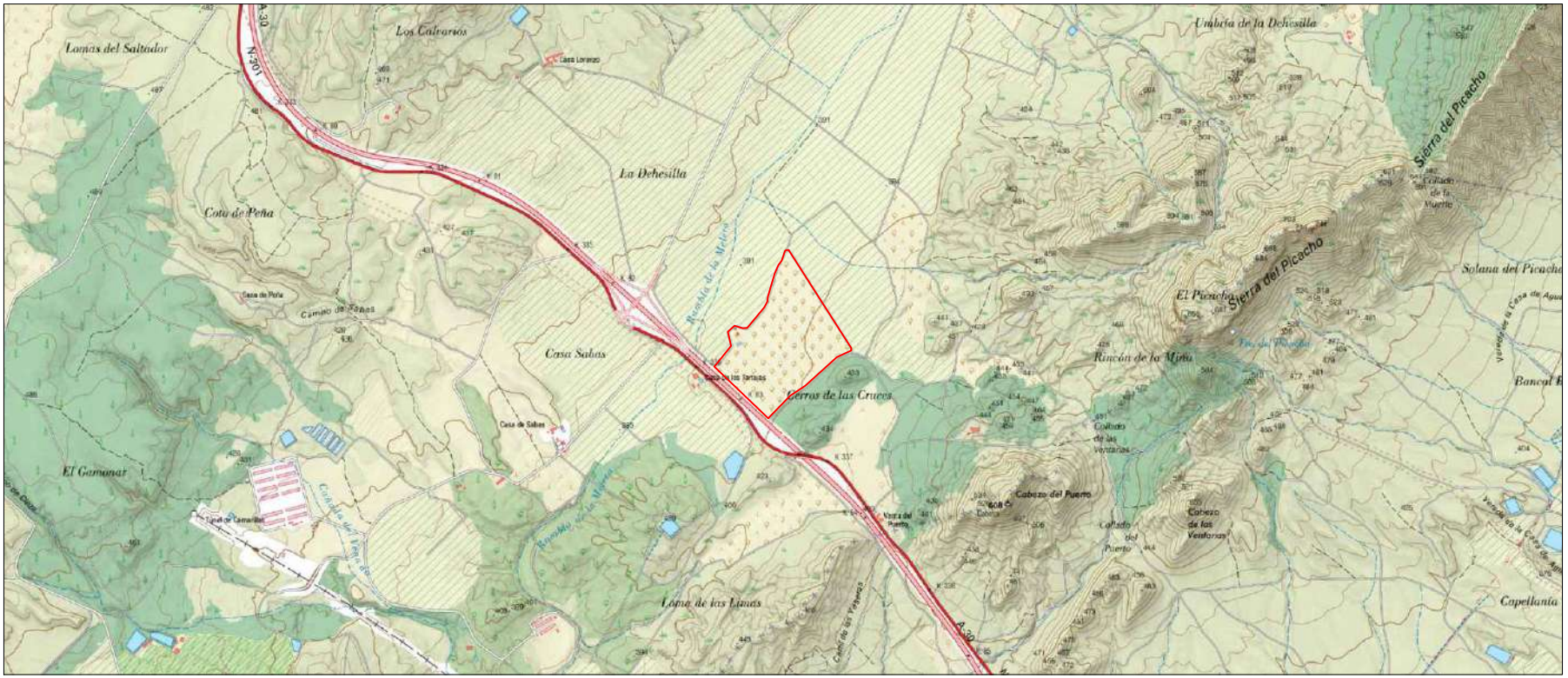
Docker. Plataforma contenedora de servicios que permite empaquetar aplicaciones junto con sus dependencias.

Units. Archivos con el código en lenguaje Pascal donde se definen las clases con sus atributos y funciones o procedimientos.

13. Planos

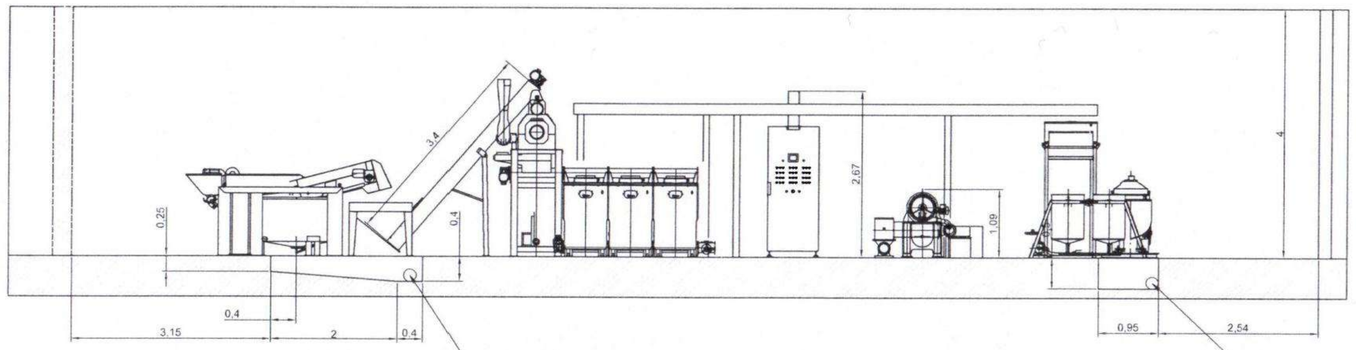


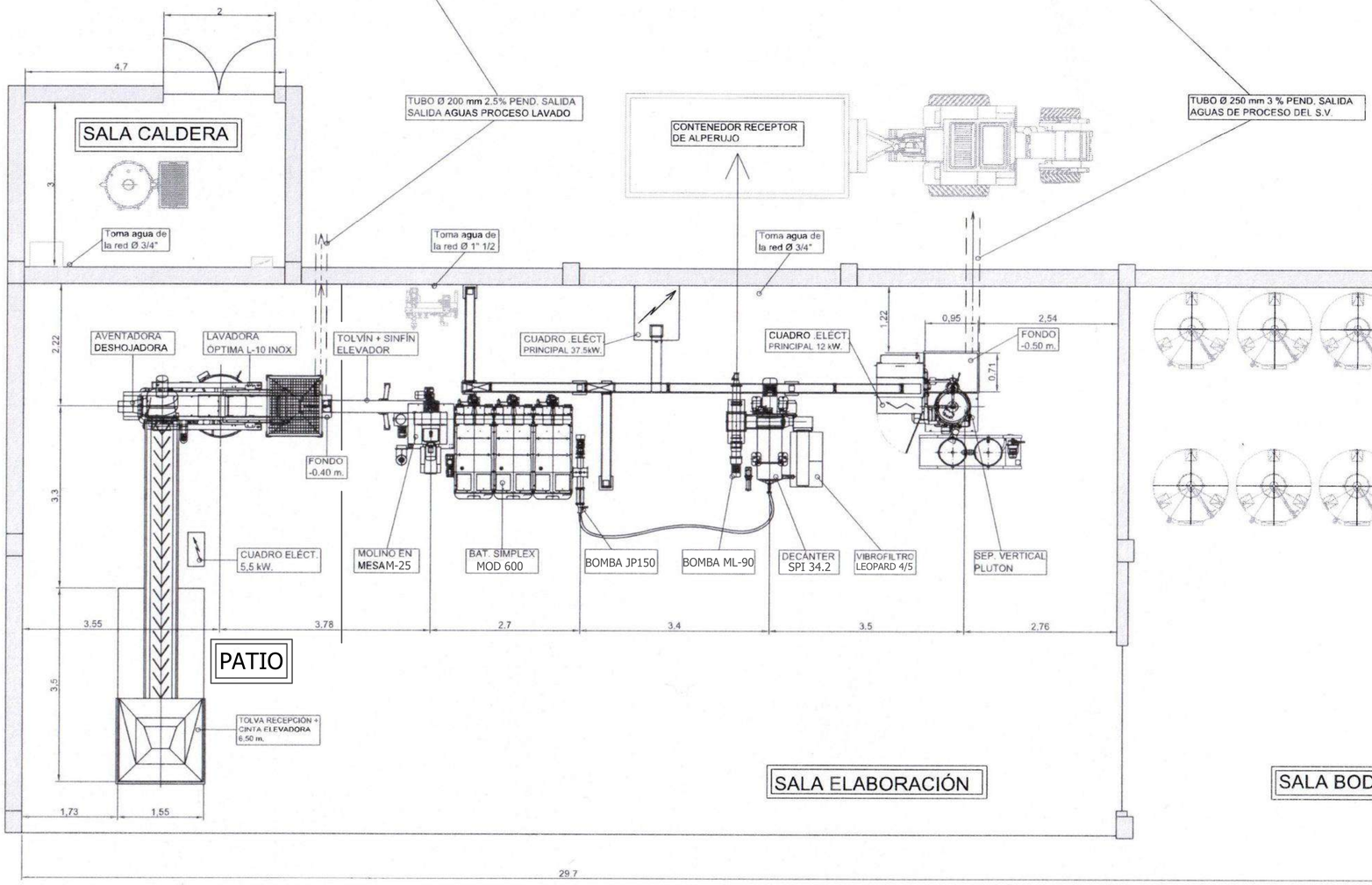
	ESCUELA POLITÉNICA SUPERIOR DE ELCHE UNIVERSIDAD MIGUEL HERNÁNDEZ	Plano Nº	1
		Febrero 2026	
Mapa de la localización de la finca.		Escala: ------	
Plano de Localización		Antonio Aragón Ros TFM	

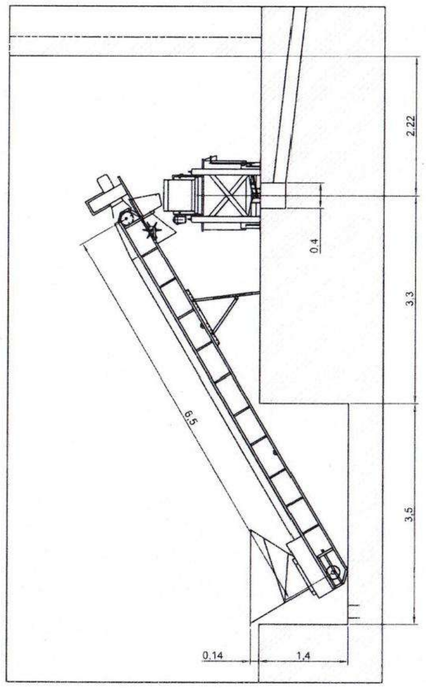


UNIVERSITAS Miguel Hernández	ESCUELA POLITÉNICA SUPERIOR DE ELCHE UNIVERSIDAD MIGUEL HERNÁNDEZ	Plano Nº	2
		Febrero 2026	
Mapa de la situación de la finca.		Escala: ------	
# Plano de Situación		Antonio Aragón Ros TFM	

UNIVERSITAS Miguel Hernández	ESCUELA POLITÉNICA SUPERIOR DE ELCHE UNIVERSIDAD MIGUEL HERNÁNDEZ	Plano Nº 3 Febrero 2026
Mapa de la zona edificable.		Escala: 1:500
Mapa 1		Antonio Aragón Ros TFM







UNIVERSITAS Miguel Hernández	ESCUELA POLITÉNICA SUPERIOR DE ELCHE UNIVERSIDAD MIGUEL HERNÁNDEZ	Escala: 1:5000
		Plano Nº 4
Plano completo Almazara		T.F.M

14. Presupuesto

A partir de los precios aproximados de cada elemento, con sus horas de instalación y el nivel profesional de la mano de obra, sumando los costes indirectos y directos de la obra y una partida de seguridad y salud final, supone un coste de ejecución material aproximado de **501.055,5 €**, que sumado a los gastos generales y beneficio industrial de la empresa adjudicataria y sus impuestos asociados nos da un total de **666.904,9 €**.

La justificación de precios y el resumen del presupuesto quedaría definido de la siguiente forma.

Anejo de justificación de precios

Núm.	Código	Ud	Descripción		Total
1	**AMB**	**1**	**Alimentación, molienda y batido**		
	TRP	1,0 Ud	Tolva de recepción de Pieralisis de 1,5 CV y longitud de 3,6 maccionado por un motor-reductor de 1,5 CV	5.800,0	5.800,0
	LVOPTIMA	1,0 Ud	Lavadora Optima L10 de 1,9 kW	18.750,0	18.750,0
	MM25	1,0 Ud	Molino M-25 con motores eléctricos de 25 CV para los martillos, de 3 CV para la criba y de 1 CV en la alimentación	21.230,0	21.230,0
	BDCH	1,0 Ud	Batidora/Decanter/Centrífuga horinzontal de 15 kW con tecnología en 2 fases ycon una sola salida líquida y otra sólida, que contiene orujo y agua.	35.800,0	35.800,0
	O1E	8,0 h	Oficial 1ª electricista.	17,5	140,0
	AE	8,0 h	Ayudante electricista.	11,4	91,2
	O1M	8,0 h	Oficial 1ª de mecánica.	17,5	140,0
	AM	8,0 h	Ayundate mecáncio	11,4	91,2
	%	2,0 %	Costes directos complementarios	82.042,4	1.640,8
		3,0 %	Costes indirectos	82.042,4	2.461,3
					86.144,5
2	**CVBSP**	**1**	**Centrífuga, bomba salomónica y de pistón**		
	CVPLUTON	1,0 Ud	Centrífuga vertical modelo PLUTON de 7,5 kW	31.200,0	31.200,0
	BSML90	1,0 Ud	Bomba salomónica ML-90 con variador de frecuencia y motor eléctrico de 2 CV	3.450,0	3.450,0
	BPJP150	1,0 Ud	Bomba de pistón para pasta de aceituna de capacidad 7.000 Kg/h y una potencia de 7,5 CV	10.200,0	10.200,0
	TBPVC	14,0 m	Tubo de PVC flexible con certificado especial europeo para el transporte de aceite de 63 mm de diámetro	37,1	519,3
	O1E	6,5 h	Oficial 1ª electricista.	17,5	113,8
	AE	6,5 h	Ayudante electricista.	11,4	74,1
	O1M	6,5 h	Oficial 1ª de mecánica.	17,5	113,8
	AM	6,5 h	Ayundate mecáncio	11,4	74,1
	%	2,0 %	Costes directos complementarios	45.745,0	914,9
		3,0 %	Costes indirectos	45.745,0	1.372,3
					48.032,2

Anejo de justificación de precios

Núm.	Código	Ud	Descripción		Total
3	**CDRBA**	**1**	**Caldera, bomba y depósito de agua**		
	CAC5H	1,0 Ud	Caldera de hueso de aceituna de 60 kW	30.000,0	30.000,0
	BC18	1,0 Ud	Bomba centrífuga de agua de 18 m3/h	455,0	455,0
	DAACS	1,0 Ud	Depósito de acumulación de ACS	8.300,0	8.300,0
	TBPVC	24,0 m	Tubería de ACS rígido de 50 mm	7,5	180,0
	O1F	19,0 h	Oficial 1ª fontanero	17,5	332,5
	AF	19,0 h	Ayudante de fontanería	11,5	218,5
	%	2,0 %	Costes directos complementarios	39.486,0	789,7
		3,0 %	Costes indirectos	39.486,0	1.184,6
					41.460,3
4	**EADA**	**1**	**Elementos auxiliares y depósitos de almacenamiento**		
	CAC5H	1,0 Ud	Depósito decantador DAC275-35	4.000,0	4.000,0
	BC18	1,0 Ud	Vibrotamiz de caldos de 2 fases VT2502F-50	6.200,0	6.200,0
	DAACS	1,0 Ud	Evacuador de orujo EV-800 de 1,1 kW	6.300,0	6.300,0
	TBPVC	24,0 m	Depósitos de 20000 l para el almacenamiento de aceite	12.000,0	288.000,0
	O1F	15,0 h	Oficial 1ª fontanero	17,5	262,5
	AF	15,0 h	Ayudante de fontanería	11,5	172,5
	O1E	3,0 h	Oficial 1ª electricista.	17,5	52,5
	AE	3,0 h	Ayudante electricista.	11,4	34,2
	%	2,0 %	Costes directos complementarios	304.935,0	6.098,7
		3,0 %	Costes indirectos	304.935,0	9.148,1
					320.268,5

Capítulos	Importe (€)
1 Alimentación, molienda y batido	86.144,5
2 Centrífuga, bomba salomónica y de pistón	48.032,2
3 Caldera, bomba y depósito de agua	41.460,3
4 Elementos auxiliares y depósitos de almacenamiento	320.268,5
5 Seguridad y salud en obra	5.150,0
Presupuesto de ejecución material (PEM)	501.055,5
5% de gastos generales	25.052,8
5% de beneficio industrial	25.052,8
Presupuesto de ejecución por contrata (PEC = PEM + GG + BI)	551.161,1
21% IVA	115.743,8
Presupuesto de ejecución por contrata con IVA (PEC = PEM + GG + BI + IVA)	666.904,9

Figura 14.1: Justificación de precios y resumen del presupuesto

15. Anexos

15.1. Puertos de la memoria DDR2 de la placa de desarrollo

Al configurar el bloque **MIG 7 Series**, el programa gestionas la conexión a sus puertos configurados en la placa.

Estos puertos son:

Pin Selection For Controller 0 - DDR2 SDRAM

	Signal Name	Bank Number	Byte Number	Pin Number	IO Standard
1	ddr2_dq[0]	34	T3	R7	SSTL18_II
2	ddr2_dq[1]	34	T3	V6	SSTL18_II
3	ddr2_dq[2]	34	T3	R8	SSTL18_II
4	ddr2_dq[3]	34	T3	U7	SSTL18_II
5	ddr2_dq[4]	34	T3	V7	SSTL18_II
6	ddr2_dq[5]	34	T3	R6	SSTL18_II
7	ddr2_dq[6]	34	T3	U6	SSTL18_II
8	ddr2_dq[7]	34	T3	R5	SSTL18_II
9	ddr2_dq[8]	34	T1	T5	SSTL18_II
10	ddr2_dq[9]	34	T1	U3	SSTL18_II
11	ddr2_dq[10]	34	T1	V5	SSTL18_II
12	ddr2_dq[11]	34	T1	U4	SSTL18_II
13	ddr2_dq[12]	34	T1	V4	SSTL18_II
14	ddr2_dq[13]	34	T1	T4	SSTL18_II
15	ddr2_dq[14]	34	T1	V1	SSTL18_II
16	ddr2_dq[15]	34	T1	T3	SSTL18_II
17	ddr2_dm[0]	34	T3	T6	SSTL18_II
18	ddr2_dm[1]	34	T1	U1	SSTL18_II
19	ddr2_dqs_p[0]	34	T3	U9	DIFF_SSTL18_II
20	ddr2_dqs_n[0]	34	T3	V9	DIFF_SSTL18_II
21	ddr2_dqs_p[1]	34	T1	U2	DIFF_SSTL18_II
22	ddr2_dqs_n[1]	34	T1	V2	DIFF_SSTL18_II
23	ddr2_addr[12]	34	T2	N6	SSTL18_II
24	ddr2_addr[11]	34	T0	K5	SSTL18_II
25	ddr2_addr[10]	34	T2	R2	SSTL18_II
26	ddr2_addr[9]	34	T2	N5	SSTL18_II
27	ddr2_addr[8]	34	T0	L4	SSTL18_II
28	ddr2_addr[7]	34	T0	N1	SSTL18_II
29	ddr2_addr[6]	34	T0	M2	SSTL18_II
30	ddr2_addr[5]	34	T2	P5	SSTL18_II
31	ddr2_addr[4]	34	T0	L3	SSTL18_II
32	ddr2_addr[3]	34	T2	T1	SSTL18_II
33	ddr2_addr[2]	34	T2	M6	SSTL18_II
34	ddr2_addr[1]	34	T2	P4	SSTL18_II
35	ddr2_addr[0]	34	T2	M4	SSTL18_II
36	ddr2_ba[2]	34	T2	R1	SSTL18_II
37	ddr2_ba[1]	34	T2	P3	SSTL18_II
38	ddr2_ba[0]	34	T2	P2	SSTL18_II
39	ddr2_ck_p[0]	34	T0	L6	DIFF_SSTL18_II
40	ddr2_ck_n[0]	34	T0	L5	DIFF_SSTL18_II
41	ddr2_ras_n	34	T2	N4	SSTL18_II
42	ddr2_cas_n	34	T0	L1	SSTL18_II
43	ddr2_we_n	34	T0	N2	SSTL18_II
44	ddr2_cke[0]	34	T0	M1	SSTL18_II
45	ddr2_odt[0]	34	T0	M3	SSTL18_II
46	ddr2_cs_n[0]	34	ByteLessPin	K6	SSTL18_II

Figura 15.1: Puertos en la memoria DDR2

15.2. Puertos de la placa de desarrollo

Para asignar los puertos externos del diseño como son la señal de reloj externa, de reset y los puertos de los periféricos, descargamos el archivo *.xdc* (figura 15.2) de nuestra placa de desarrollo NEXYS A7 [11] donde asignamos el nombre de cada señal externa con los pines externos de la placa.

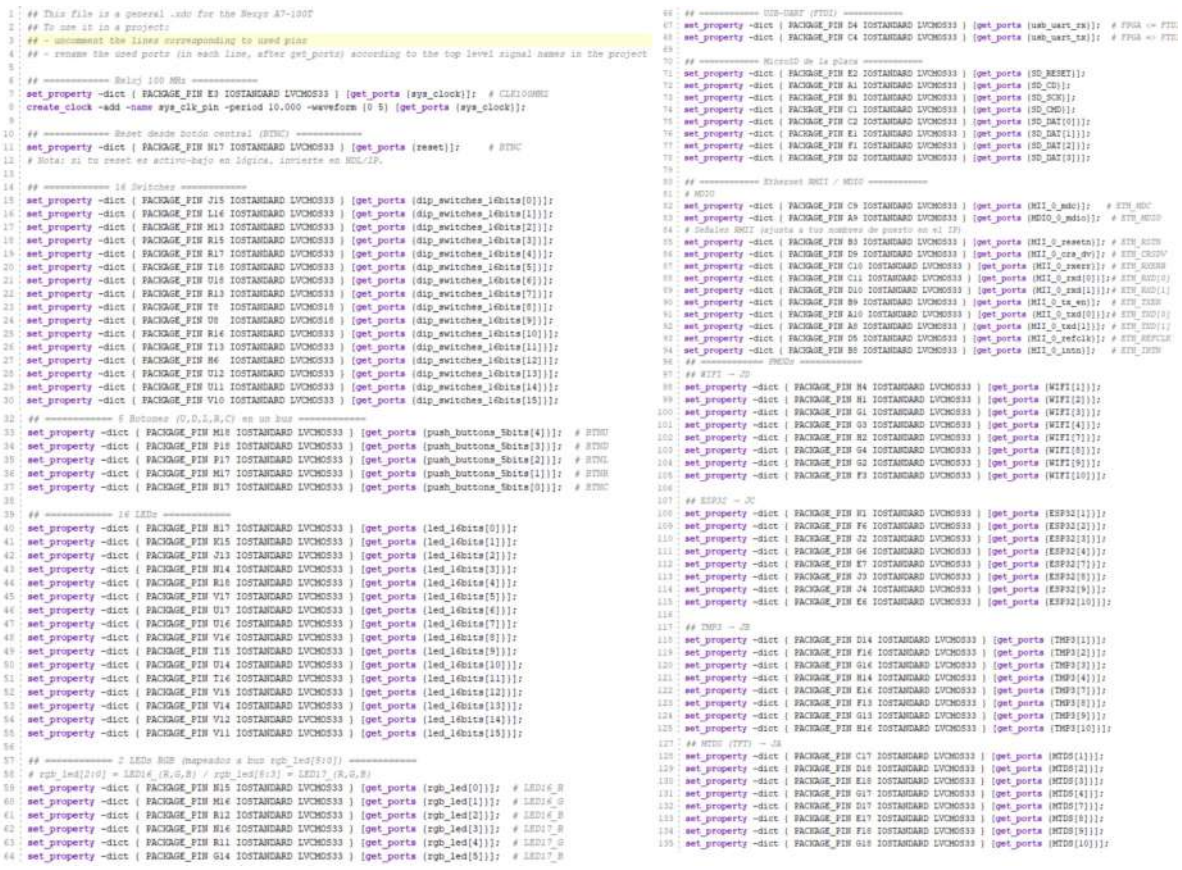


Figura 15.2: Esquema de conexiones en la placa de desarrollo.

15.3. Comunicación I/O link

I/O Link es la primera interfaz de E/S estandarizada del mundo. La tecnología se ha consolidado como estándar de comunicación para la fabricación de maquinaria e instalaciones. El número de productos utilizados con la interfaz IO-Link está creciendo rápidamente. El estándar de comunicación le ofrece muchas ventajas, consigue nuevas posibilidades de aplicación y desarrolla el Internet de las cosas industrial

IIoT. IO-Link es un protocolo de comunicación punto a punto de serie abierto para la conexión de sensores y actuadores a un sistema de automatización.

IO-Link Safety implementa una comunicación segura hasta el último metro. Con ello, los usuarios también se benefician de las ventajas típicas de IO-Link en el ámbito de la seguridad funcional, como la sencilla parametrización mediante IODD o la rápida sustitución de equipos.
Las **ventajas** del I/O Link son:

- Aumento de la productividad.
- Simplificación de la instalación, el mantenimiento y los procesos de sustitución.
- Mayor disponibilidad y precisión de los datos.
- Solución de fallos mediante diagnóstico remoto.

15.3.1. Facilidades, precisión de datos y configuración remota

En el sistema I/O Link, la instalación es sencilla, porque puede utilizar cables y conectores estándar.
El mantenimiento es sencillo, ya que puede monitorizar sus sistemas y llevar a cabo modificaciones en los ajustes. Mediante la detección de fallos de los dispositivos I/O Link podrá determinar fácil y rápidamente la causa de un problema.
El proceso de sustitución también es fácil gracias a los datos de parámetros guardados en el sistema de control. Si debe sustituirse el dispositivo IO-Link, los parámetros guardados se transfieren automáticamente al dispositivo IO-Link sin utilizar un equipo de programación. De este modo, su instalación vuelve a estar rápidamente lista para el servicio.

Los sensores analógicos sin I/O Link se someten a varias transformaciones A/D antes de que los datos lleguen al sistema de control, lo que puede conducir a fallos.

En dispositivos I/O Link solo hay una transformación A/D. Esto aumenta la precisión de medición y con ello la precisión de todo el proceso.

Desde una configuración remota, como puede ser el servidor Windows instalado en la Almazara se puede cambiar los parámetros de los dispositivos I/O Link respondiendo a los mensajes de estado que deja el propio aparato. Por tanto, para solucionar fallos ya no es necesario que se desconecte la línea de producción y se tenga que acceder físicamente al equipo.

15.3.2. Arquitectura del sistema I/O Link

La arquitectura de un sistema I/O Link (figura 15.3) consta de un sistema de control, un maestro I/O Link y uno o varios dispositivos I/O Link. Se conectan mediante un sencillo cable de tres hilos sin apantallar, lo que simplifica la instalación y la hace más rentable.

Los dispositivos IO-Link ofrecen tres tipos de datos:

1. **Datos de proceso**: informaciones de estado que el dispositivo I/O Link lee y envía al maestro I/O Link o informaciones que el maestro I/O Link envía al dispositivo I/O Link. En este proceso también se transfieren informaciones de estado. De este modo, puede comprobar si los datos de proceso son válidos. Un ejemplo de datos de proceso: la distancia medida por un sensor de distancia.

2. **Datos de servicio**: informaciones que se escriben en el dispositivo I/O Link o que pueden ser leídas por el dispositivo I/O Link. Ejemplo: el número de modelo o de fabricante del sensor ayuda a identificar un equipo o a configurar uno nuevo.

3. **Datos de evento**: notificaciones como mensajes de error que son enviadas por el dispositivo I/O Link al maestro I/O Link en cuanto se produce el evento, por ejemplo un sobrecalentamiento.

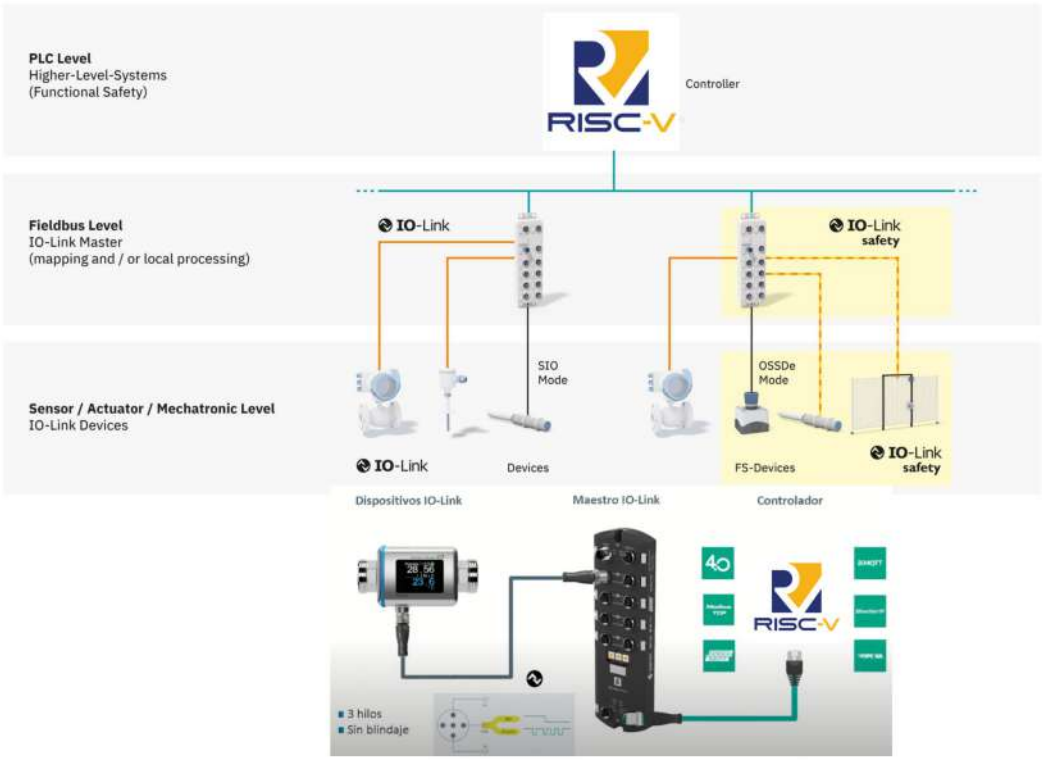


Figura 15.3: Configuración del sistema I/O Link.