FACULTAD DE INGENIERÍA Y CIENCIAS

UNIVERSIDAD ADOLFO IBÁÑEZ

Informe de Avance

TID: Sistema de Adquisición y Control de Datos Vía Ethernet para Instrumentación Científica

Bryan Alburquenque

Ingeniería Civil Informática

Profesor Guía: Cristian Fuentes

13 de diciembre de 2025

Resumen

Este documento presenta el avance en el desarrollo de una aplicación de escritorio para la comunicación bidireccional con FPGA mediante tramas Ethernet crudas. Se describe el proceso de investigación, validación de comunicación, especificación del protocolo, implementación de interfaz gráfica y optimización del envío concurrente de comandos. El hito de este proyecto es la aplicación en Python para configurar la comunicación con los dispositivos, pilar para desarrollos posteriores con documentación robusta y arquitectura que facilita el desarrollo.

Índice

1. Introducción y Contexto

El objetivo de este taller de investigación dirigida es desarrollar un sistema de adquisición y control de datos (DAQ) en Python capaz de establecer comunicación bidireccional a bajo nivel mediante tramas Ethernet crudas con hardware externo, específicamente FPGA que conforman detectores de rayos gamma basados en SiPMs.

La aplicación debe permitir:

- Lectura y decodificación de tramas Ethernet enviadas por las FPGA
- Visualización de datos en interfaz gráfica
- Envío de comandos de configuración para calibración de dispositivos
- Sincronización de comandos cuando múltiples FPGA están presentes

El alcance de este avance comprende la comunicación unidireccional (PC $\rightarrow$ FPGA) y la validación del protocolo definido.

2. Desarrollo por Hitos

2.1. Etapa 1: Investigación Inicial (Agosto)

La primera semana se dedicó a explorar las herramientas necesarias para trabajar con tramas Ethernet y monitoreo de comunicación.

2.1.1. Actividades

Se investigaron las librerías y software:

- **Scapy**: librería Python para manipulación de protocolos de red
- **Wireshark**: analizador de tramas de red
- **Ostinato**: generador y simulador de tramas Ethernet

2.1.2. Descubrimiento Clave

Se identificó que la configuración de interfaces de red en Linux Debian depende del adaptador utilizado. Para esta configuración específica, los nombres de interfaz permanecen consistentes si se mantiene el mismo adaptador USB-C a Ethernet en la misma posición.

2.1.3. Decisión Tomada

Se seleccionó **Scapy** como librería principal por su compatibilidad multiplataforma (Windows y Linux), esencial para que la aplicación funcione en ambos sistemas operativos.

2.2. Etapa 2: Validación de Comunicación (Agosto 24–25)

Se realizó la primera prueba práctica de envío y recepción de tramas Ethernet.

2.2.1. Experimento

- Interfaces utilizadas:
  - Emisora: `enp2s0f1` (MAC: `98:28:a6:0f:c8:df`)
  - Receptora: `enx00e04c6806d2` (MAC: `00:e0:4c:68:06:d2`)
- Se creó script Python con Scapy para enviar trama con payload mockado
- Se verificó recepción con Wireshark en interfaz receptora

2.2.2. Resultado

La trama fue enviada y recibida correctamente. Las direcciones MAC de origen y destino coincidieron exactamente con las esperadas. El payload se mantuvo intacto en el proceso de transmisión.

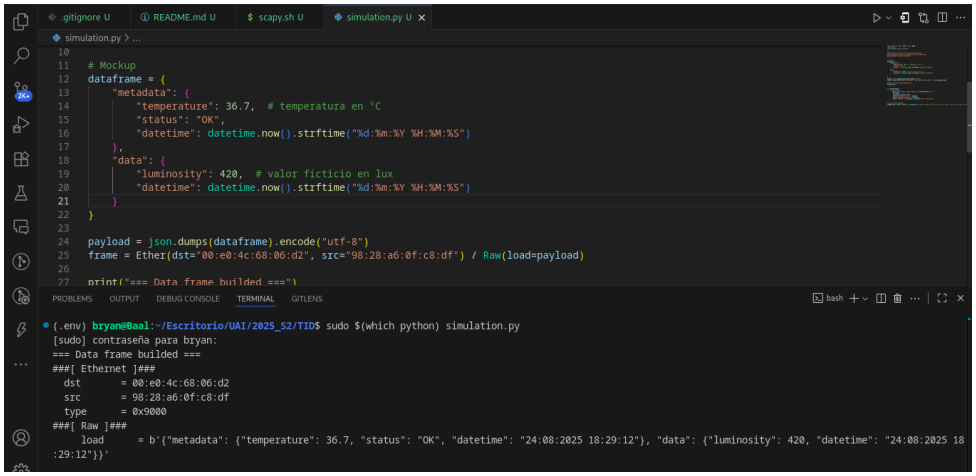


Figura 1: Script Python de prueba para envío de trama Ethernet.

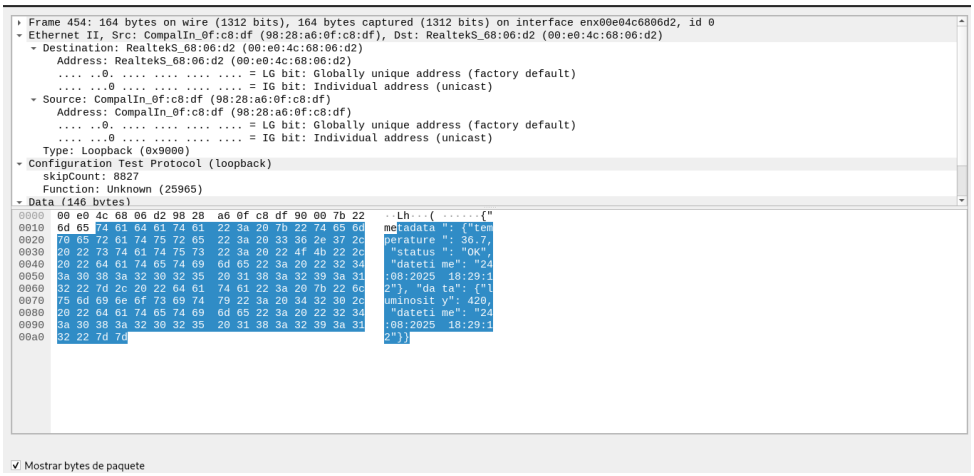


Figura 2: Captura de logs en Wireshark mostrando la recepción de la trama.

2.2.3. Estructura de Payload Validada

```
{
  "metadata": {
    "temperature": 36.7,
    "status": "OK",
    "datetime": "dd:mm:yyyy HH:MM:SS"
  },
  "data": {
    "luminosity": 420,
    "datetime": "dd:mm:yyyy HH:MM:SS"
  }
}
```

2.2.4. Decisión

Se confirmó que la aproximación técnica es viable. Se solicitó al profesor la documentación del formato real de tramas enviadas por los microcontroladores.

2.3. Etapa 3: Especificación del Protocolo (Septiembre)

Se recibió documentación técnica que define el formato exacto de las tramas.

2.3.1. Estructura de Trama Ethernet Definida

Campo	Bytes	Descripción	Valores
MAC Destino	6	Dirección del dispositivo destino	Único
MAC Origen	6	Dirección del dispositivo origen	Único
Length Package	2	Tamaño total del paquete	Variable
Padding Bytes	4	Relleno para alineación	`0x00`
Constantes	2	Identificador del protocolo	`0x02:0x03`
Appendix	1	Instrucción/Comando	Variable

Nota: los comandos abordados en esta primera etapa, correspondientes a los de la FPGA, no varian en cuanto al largo del paquete.

2.3.2. Descubrimiento

La estructura de trama es simple pero requiere precisión en el cálculo del campo `Length Package`, que debe incluir todos los bytes posteriores a las direcciones MAC.

2.3.3. Decisión

Se adoptó esta estructura como estándar para todas las comunicaciones con FPGA. Se iniciaron pruebas para implementar esta especificación en el código.

2.4. Etapa 4: Desarrollo de Interfaz Gráfica (Septiembre – Noviembre)

Se construyó una aplicación de escritorio para facilitar la interacción con las FPGA sin necesidad de scripts de línea de comandos.

2.4.1. Funcionalidades Implementadas (Iteración 1 – Septiembre 28)

- **Selector de comandos**: permite elegir qué instrucción enviar
- **Periodicidad**: configura el intervalo entre envíos repetidos
- **Repeticiones**: define cuántas veces se ejecuta un comando
- **Cálculo de tamaño**: incluye automáticamente el tamaño del frame

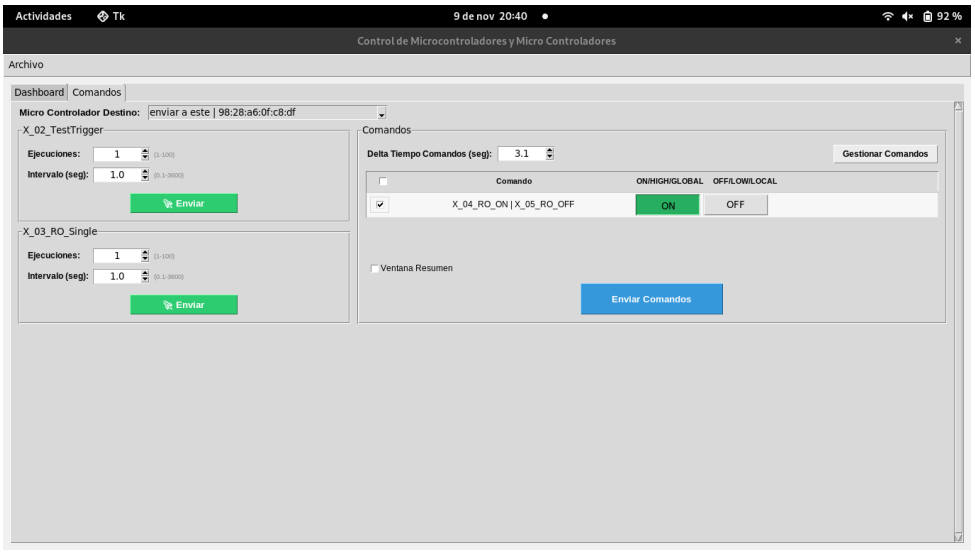


Figura 3: Primera aproximación de la aplicación: dashboard con información de estado y acceso a comandos.

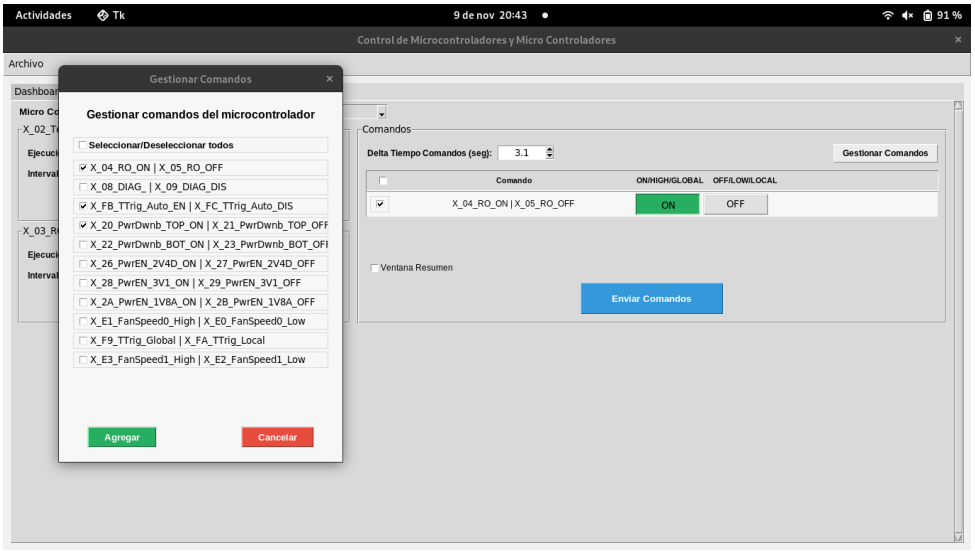


Figura 4: Primera aproximación de la aplicación: Gestor de comandos en la interfaz gráfica.

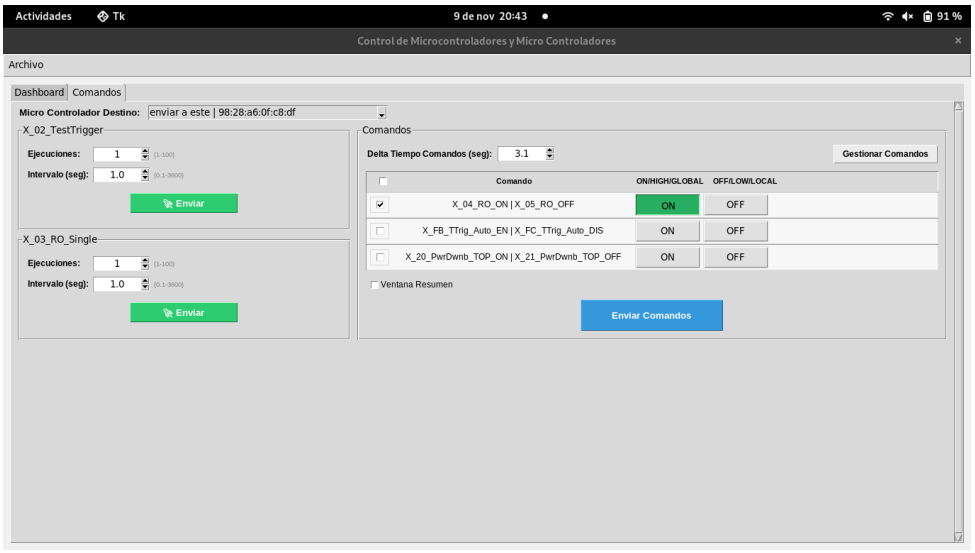


Figura 5: Primera aproximación de la aplicación: Gestor de comandos luego de modificar listado de comandos

2.4.2. Funcionalidades Mejoradas (Iteración 2 – Noviembre 9)

Se realizaron mejoras significativas basadas en retroalimentación:

1. **Gestor dinámico de comandos**: permite agregar y remover comandos según necesidad
2. **Persistencia de estado**: guarda la última instrucción enviada a cada FPGA
3. **Nombres de comandos actualizados**: se alinean con documentación oficial (`X_04_RO_ON`, `X_05_RO_OFF`, etc.)
4. **Resumen pre-envío**: ventana emergente opcional que muestra los comandos a ejecutar
5. **Preservación de interfaz**: los cambios en el gestor no afectan el estado actual

2.4.3. Decisión Arquitectónica

Se decidió mantener la estructura modular para permitir futuras expansiones como sistema de macros y soporte para comandos de CPU.

2.5. Etapa 5: Sincronización Concurrente de Comandos (Noviembre)

Se identificó que cuando hay múltiples FPGA, es crítico que reciban los comandos de forma sincronizada.

2.5.1. Problema Identificado

En un escenario multi-FPGA, el envío secuencial de comandos introduce desviaciones temporales importantes que podrían afectar la calibración de detectores.

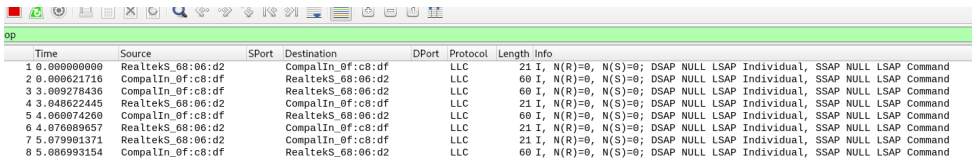
2.5.2. Solución Implementada

Se implementó un sistema de **envío concurrente mediante Threads**:

- Se genera una lista de comandos basada en la macro seleccionada
- Se crea un hilo para cada FPGA
- Cada hilo consume los comandos en orden, con repeticiones y delays configurados
- Los hilos ejecutan en paralelo, minimizando diferencias temporales

2.5.3. Resultado Experimental

Las pruebas con dos FPGA mostraron diferencias temporales en el rango de **centésimas de segundo** (aproximadamente 10 ms máximo). Esto se verificó mediante logs capturados en Wireshark, donde se puede observar que los comandos llegaron a ambos dispositivos con mínima desviación.


op

	Time	Source	SPort	Destination	DPort	Protocol	Length	Info
1	0.000000000	RealtekS_68:06:d2		CompalIn_0f:c8:df		LLC	21	I, N(R)=0, N(S)=0; DSAP NULL LSAP Individual, SSAP NULL LSAP Command
2	0.000621716	CompalIn_0f:c8:df		RealtekS_68:06:d2		LLC	60	I, N(R)=0, N(S)=0; DSAP NULL LSAP Individual, SSAP NULL LSAP Command
3	3.009278436	CompalIn_0f:c8:df		RealtekS_68:06:d2		LLC	60	I, N(R)=0, N(S)=0; DSAP NULL LSAP Individual, SSAP NULL LSAP Command
4	3.048622445	RealtekS_68:06:d2		CompalIn_0f:c8:df		LLC	21	I, N(R)=0, N(S)=0; DSAP NULL LSAP Individual, SSAP NULL LSAP Command
5	4.060074260	CompalIn_0f:c8:df		RealtekS_68:06:d2		LLC	60	I, N(R)=0, N(S)=0; DSAP NULL LSAP Individual, SSAP NULL LSAP Command
6	4.076089657	RealtekS_68:06:d2		CompalIn_0f:c8:df		LLC	21	I, N(R)=0, N(S)=0; DSAP NULL LSAP Individual, SSAP NULL LSAP Command
7	5.079901371	RealtekS_68:06:d2		CompalIn_0f:c8:df		LLC	21	I, N(R)=0, N(S)=0; DSAP NULL LSAP Individual, SSAP NULL LSAP Command
8	5.086993154	CompalIn_0f:c8:df		RealtekS_68:06:d2		LLC	60	I, N(R)=0, N(S)=0; DSAP NULL LSAP Individual, SSAP NULL LSAP Command

Figura 6: Logs de Wireshark mostrando envío de comandos hacia dos interfaces (FPGAs) a la vez.

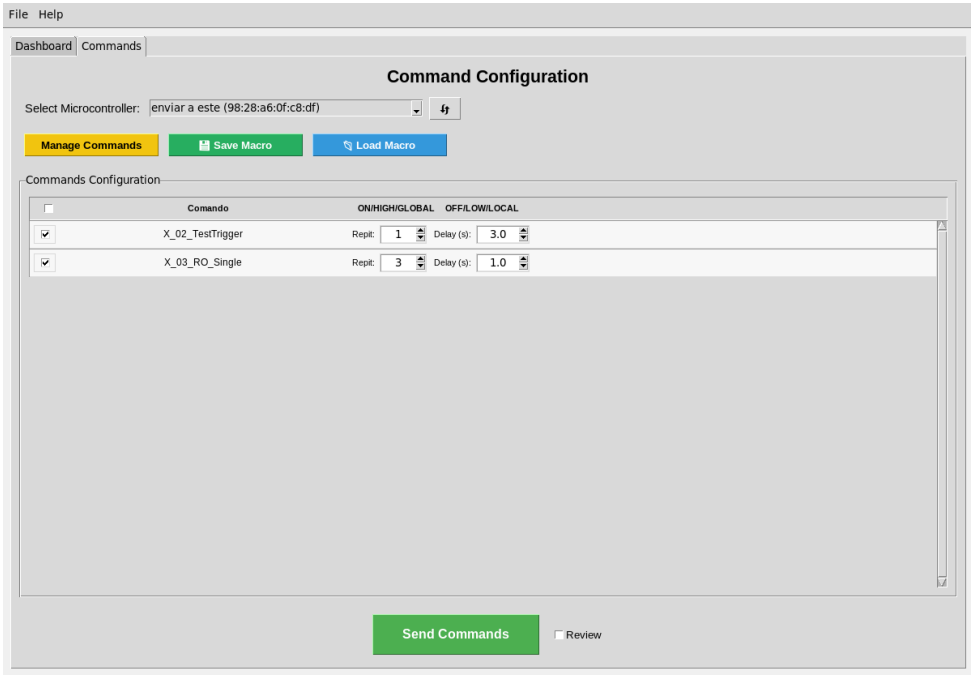


Figura 7: Listado de comandos enviados a las dos interfaces (FPGAs).

2.5.4. Decisión

Se validó que el enfoque de Threads es adecuado. Se deberá evaluar en pruebas futuras si estas diferencias afectan la reconstrucción de imágenes del detector de rayos gamma.

3. Decisiones Técnicas Clave

Decisión	Alternativa	Razón
Scapy para envío	Socket puro	Compatibilidad multiplataforma (Windows + Linux)
Threads para concurrencia	Asyncio/Multiprocessing	Simplicidad de implementación y sincronización temporal precisa
Estructura append-based	Protocolo estándar	Compatibilidad con FPGA existente y documentación disponible
GUI con PyQt/Tkinter	Línea de comandos	Usabilidad para operadores sin conocimiento técnico
Persistencia local	Base de datos	Simplificar configuración y recuperación de estado

4. Estado Actual (Noviembre 2025)

4.1. Componentes Implementados

- Captura y envío de tramas Ethernet crudas
- Interfaz gráfica funcional con gestor de comandos
- Envío concurrente a múltiples FPGA con sincronización
- Persistencia de estados

4.2. Funcionalidades Pendientes

- Calibración y validación en hardware real
- Implementación de comandos para CPU (pendiente)
- Captura y decodificación de datos entrantes (FPGA → PC)
- Visualización de datos en tiempo real
- Interfaz de calibración interactiva

4.3. Ultimos pasos

Con el objetivo de que la aplicación sea fácil de escalar, se decidió junto con el profesor lo siguiente:

- Refactorizar la aplicación monolítica de un solo archivo utilizando el patrón Modelo Vista Controlador.
- Documentar las funcionalidades de aplicación y estructura del proyecto.

Estos puntos esperan ser implementados al final del TID junto al informe final.

5. Próximos Pasos

1. **Pruebas con FPGA real**: validación del protocolo con hardware físico en lugar de simulación
2. **Captura de datos**: implementar lectura y decodificación de tramas enviadas por FPGA
3. **Visualización**: desarrollar gráficas de datos en tiempo real

4. **Refinamiento temporal**: evaluar si desviaciones de envío concurrente afectan experimentos
5. **Soporte CPU**: extender protocolo para incluir comandos a procesador central

6. Artefactos y Referencias

6.1. Repositorio de Código

El código está disponible en:

`https://github.com/alburquenqueletelier/TID_IO_Ethernet_Simulation`

Ramas principales:

- `main`: version actualizada y estable de la aplicación
- `demo-full`: mockup de características esperadas (guía inicial para desarrollar)

6.2. Documentación Técnica

- Firmware.doc: especificación de firmware
- Diagnosis.doc: protocolo de diagnóstico
- Frames_PETITION_Ethernet.xlsx: definición de estructura de tramas

7. Conclusiones

El desarrollo ha avanzado desde la investigación inicial hasta una plataforma funcional de comunicación. Los hitos alcanzados incluyen:

- **Validación técnica**: demostrado que la comunicación Ethernet cruda es viable con Scapy
- **Especificación clara**: protocolo completamente definido y documentado
- **Interfaz usable**: aplicación gráfica que simplifica la interacción con FPGA
- **Sincronización**: implementación de envío concurrente con desviaciones mínimas

Las decisiones tomadas priorizan **compatibilidad multiplataforma**, **sincronización temporal precisa** y **facilidad de uso**. El sistema está listo para transicionar a pruebas con hardware real en lo que respecta a comandos de FPGA, donde se validarán los supuestos técnicos y se medirá el impacto de la precisión temporal en la reconstrucción de imágenes.