Taller de Automatización de equipos GW-Instek

Técnicas de automatización y adquisición de datos en sistemas de hardware

1. Objetivos

1.1. Objetivo General

Desarrollar habilidades para la automatización de equipos de instrumentación GW-Instek, específicamente osciloscopios y generadores de funciones, mediante comandos SCPI y programación en Python, utilizando comunicación por protocolo TCP/IP para optimizar la adquisición y análisis de datos en sistemas de prueba.

1.2. Objetivos Específicos

1. Analizar y comprender los protocolos de comunicación entre computadoras y equipos de instrumentación GW-Instek, identificando las características específicas de las interfaces de red utilizadas en laboratorios modernos.
2. Desarrollar e implementar scripts funcionales en Python utilizando las bibliotecas PyVISA y PyVISA-py para establecer comunicación bidireccional con osciloscopios y generadores de funciones GW-Instek.
3. Usar la sintaxis y estructura de comandos SCPI para controlar remotamente equipos de medición.
4. Desarrollar scripts que puedan automatizar el sacado de mediciones directo desde Python, incluyendo la captura, procesamiento y visualización de datos en tiempo real.

2. Marco Teórico

2.1. GW-Instek

GW Instek es una empresa fundada en 1975, especializada en la fabricación de equipos de medición y prueba para aplicaciones de ingeniería electrónica. La compañía se ha posicionado como un proveedor líder de instrumentos de alta calidad, incluyendo osciloscopios digitales, generadores de funciones, fuentes de alimentación y multímetros. En los últimos años, GW Instek ha incorporado capacidades avanzadas de programación y automatización en sus equipos, especialmente en la serie MPO-2000, que integra funcionalidad de scripting con Python, permitiendo a los usuarios controlar sistemas automatizados de prueba sin necesidad de una computadora adicional.

Los equipos GW-Instek modernos están diseñados con interfaces de comunicación estándar como USB, RS-232 y Ethernet, facilitando la integración con sistemas computarizados. Esta característica resulta particularmente útil en entornos educativos, laboratorios de investigación y líneas de producción, donde la automatización de mediciones reduce significativamente el tiempo de prueba y mejora la reproducibilidad de los resultados.

2.2. Comandos SCPI

SCPI (Standard Commands for Programmable Instruments, o Comandos Estándar para Instrumentos Programables) es un lenguaje de comandos estandarizado diseñado para controlar instrumentos de prueba y medición. Desarrollado en la década de 1990, SCPI proporciona una sintaxis común y coherente que permite a los usuarios controlar equipos de diferentes fabricantes utilizando comandos similares, lo que facilita enormemente la automatización de sistemas de instrumentación heterogéneos.

La estructura de los comandos SCPI sigue un formato jerárquico utilizando dos puntos (:) para separar niveles, similar a la navegación por directorios en un sistema de archivos. Los comandos típicamente reflejan la estructura de menús del panel frontal del instrumento, facilitando su comprensión. Existen dos formas principales de comandos SCPI: comandos de configuración para cambiar parámetros del instrumento y comandos de consulta (que terminan con signo de interrogación) para obtener información del dispositivo.

Por ejemplo, para consultar la identificación de un instrumento, el comando estándar es ”*IDN?”(Identification Query), mientras que para establecer un valor específico en un osciloscopio podría utilizarse un comando como ÇHANNEL1:SCALE 0.5”para ajustar la escala vertical del canal 1 a 0.5 voltios por división. Este enfoque estructurado permite desarrollar programas coherentes para el control de instrumentos, independientemente del fabricante.

2.3. Python para Automatización

Python se ha convertido en el lenguaje de programación preferido para aplicaciones científicas y de ingeniería debido a su sintaxis clara, facilidad de aprendizaje y amplia disponibilidad de bibliotecas especializadas. En el contexto de automatización de equipos de medición, Python ofrece un entorno ideal para desarrollar interfaces de comunicación con instrumentos, procesamiento de datos y visualización de resultados.

Para la comunicación con instrumentos, existen varias bibliotecas especializadas como PyVISA, una implementación en Python de la arquitectura VISA (Virtual Instrument Software Architecture). PyVISA proporciona una capa de abstracción que simplifica la comunicación con equipos de medición a través de diferentes interfaces físicas, incluyendo USB, GPIB, RS-232 y TCP/IP. Esta biblioteca permite enviar comandos SCPI a los equipos y recibir respuestas, facilitando la automatización de mediciones complejas.

2.4. PyVISA y PyVISA-py

PyVISA es una biblioteca de Python que proporciona una interfaz para comunicarse con instrumentos de medición utilizando el estándar VISA. Esta biblioteca permite a los desarrolladores controlar todo tipo de dispositivos de medición independientemente de la interfaz física (GPIB, RS-232, USB o Ethernet) mediante un conjunto coherente de funciones. PyVISA actúa como un envoltorio (wrapper) para bibliotecas VISA de bajo nivel, como las proporcionadas por National Instruments o Keysight.

PyVISA-py, por otro lado, es una implementación pura de Python del estándar VISA, eliminando la dependencia de bibliotecas VISA comerciales. Esta característica es particularmente útil en sistemas operativos donde las implementaciones tradicionales de VISA pueden no estar disponibles, como en ciertas distribuciones de Linux. PyVISA-py admite comunicación a través de TCP/IP, USB, GPIB y puertos serie, lo que lo hace adecuado para una amplia gama de aplicaciones de automatización de instrumentos.

Utilizando PyVISA o PyVISA-py, los desarrolladores pueden escribir scripts para enviar comandos SCPI a equipos GW-Instek, configurar parámetros, realizar mediciones, capturar formas de onda y procesar datos, todo desde un entorno Python unificado. Un ejemplo básico de código para comunicarse con un instrumento sería:

```
import pyvisa

# Crear administrador de recursos
rm = pyvisa.ResourceManager('@py')

# Abrir conexión con el instrumento (usando dirección TCP/IP)
instrumento_ip = '172.118.1.247' # IP asignada al equipo
instrumento_puerto = 3000 #El puerto puede cambiar de un equipo a otro
cadena_recurso = f'TCPIP0::{instrumento_ip}::{instrumento_puerto}::SOCKET'

# Abrir conexión con el osciloscopio
instrumento = rm.open_resource(cadena_recurso)
instrumento.timeout = 10000  # 10 segundos de timeout
instrumento.read_termination = '\n'
instrumento.write_termination = '\n'

# Enviar comando SCPI y leer respuesta
# Verificar conexión con comando de identificación
print("Verificando conexión con *IDN?...")
```

```
identificacion = instrumento.query('*IDN?').strip()
print(f"Osciloscopio identificado: {identificacion}")

# Cerrar conexión
instr.close()
print("Conexión cerrada correctamente")
```

2.5. DHCP y Configuración TCP/IP

DHCP (Dynamic Host Configuration Protocol) es un protocolo de red que permite la asignación automática de direcciones IP y otros parámetros de configuración de red a dispositivos en una red. En el contexto de la automatización de equipos de medición, DHCP facilita la integración de instrumentos como osciloscopios y generadores de funciones de GW-Instek en redes existentes, eliminando la necesidad de configurar manualmente cada dispositivo.

Al conectar un equipo GW-Instek a una red con servidor DHCP activo, el dispositivo puede obtener automáticamente una dirección IP, máscara de subred, puerta de enlace y servidores DNS, simplificando la configuración inicial. Sin embargo, para aplicaciones de automatización estables, generalmente se recomienda asignar direcciones IP estáticas a los instrumentos, garantizando que siempre sean accesibles en la misma dirección, lo que facilita el desarrollo y mantenimiento de scripts de automatización.

La configuración de red para equipos GW-Instek generalmente se realiza a través del panel frontal del instrumento, accediendo al menú de configuración de utilidades o sistema. Es importante considerar aspectos de seguridad al integrar equipos de instrumentación en redes corporativas, ya que muchos de estos dispositivos tienen capacidades de seguridad limitadas. Por ello, en entornos sensibles se recomienda utilizar redes dedicadas o segmentadas para los equipos de laboratorio.

2.6. Comunicación vía Puerto TCP

La comunicación TCP/IP (Transmission Control Protocol/Internet Protocol) es un conjunto de protocolos que permite la transmisión confiable de datos entre dispositivos en redes. Para la automatización de equipos GW-Instek, se utiliza TCP como protocolo de transporte debido a su confiabilidad y capacidad para garantizar la entrega ordenada de datos, características esenciales cuando se envían comandos de control a instrumentos de precisión.

Los equipos GW-Instek que soportan conectividad Ethernet generalmente utilizan puertos TCP específicos para la comunicación SCPI. Por ejemplo, muchos instrumentos utilizan el puerto 5025 para conexiones de socket VISA o el puerto 3500 para conexiones directas. La comunicación a través de TCP permite enviar comandos SCPI como cadenas de texto, seguidas típicamente por terminadores de línea específicos (como CR+LF, o \r\n en notación de Python) que indican el final de un comando.

Desde la perspectiva de la programación con Python, se puede establecer una comunicación TCP/IP con equipos GW-Instek de dos maneras principales: utilizando PyVISA con el formato de recurso ”TCPIP0::[dirección IP]::[puerto] ::SOCKET”, o utilizando directamente sockets TCP de Python para aplicaciones más simples o cuando se requiere un control más detallado sobre la comunicación.

3. Implementación Práctica

3.1. Configuración del Entorno

Para implementar un sistema de automatización de equipos GW-Instek, se requiere la preparación adecuada del entorno de trabajo, tanto en el aspecto de hardware como de software. A continuación se detallan los pasos necesarios para establecer este entorno:

1. **Creación de un entorno virtual con uv:**

   ```
   pip install uv
   ```

2. **Configuración de PowerShell para permitir la ejecución de scripts:**

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope Process
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser
```

3. **Activación del entorno virtual desde el editor de código:**

```
.\venv\Scripts\Activate.ps1
```

   Al activarse correctamente, aparecerá en la terminal:

```
(Automatizacion_Proyecto) PS D:\Proyectos Python\Automatizacion_Proyecto>
```

4. **Instalación de las bibliotecas necesarias:**

```
uv pip install pyvisa pyvisa-py numpy matplotlib streamlit
```

5. **Configuración de red para los equipos GW-Instek:**
   - Activar la interfaz Ethernet en el instrumento.
   - Configurar el modo de obtención de dirección IP (DHCP o manual).
   - Si se usa IP estática, configurar dirección IP, máscara de subred y puerta de enlace.
   - Verificar la conectividad mediante ping desde la computadora.

4. Comandos SCPI para Equipos GW-Instek

4.1. Comandos SCPI para Osciloscopios

Los osciloscopios GW-Instek responden a una variedad de comandos SCPI que permiten el control completo del instrumento. A continuación se presentan los comandos más utilizados con explicaciones detalladas de su función:

Comando	Descripción	Tipo
`*IDN?`	Solicita la identificación del instrumento. Devuelve información como marca, modelo, número de serie y versión de firmware.	Consulta
`:AUTOSet`	Ejecuta la función de configuración automática del osciloscopio, ajustando la escala vertical, horizontal y el trigger para visualizar correctamente la señal.	Configuración
`:CHANnel1:COUPling AC`	Configura el acoplamiento del Canal 1 a AC, bloqueando la componente DC de la señal. Útil para visualizar señales pequeñas superpuestas a voltajes DC.	Configuración
`:CHANnel1:COUPling?`	Consulta el tipo de acoplamiento actual del Canal 1. Puede devolver AC, DC o GND.	Consulta
`:CHANnel2:COUPling AC`	Configura el acoplamiento del Canal 2 a AC, bloqueando la componente DC de la señal.	Configuración
`:CHANnel2:COUPling?`	Consulta el tipo de acoplamiento actual del Canal 2.	Consulta
`:CHANnel1:DISPlay ON`	Activa la visualización del Canal 1 en la pantalla del osciloscopio.	Configuración
`:CHANnel2:DISPlay ON`	Activa la visualización del Canal 2 en la pantalla del osciloscopio.	Configuración
`:CHANnel1:POSition 0`	Ajusta la posición vertical del Canal 1 a 0 divisiones desde el centro. Permite centrar la forma de onda en la pantalla.	Configuración
`:CHANnel1:POSition?`	Consulta la posición vertical actual del Canal 1 en divisiones.	Consulta
`:CHANnel2:POSition 0`	Ajusta la posición vertical del Canal 2 a 0 divisiones desde el centro.	Configuración
`:CHANnel2:POSition?`	Consulta la posición vertical actual del Canal 2 en divisiones.	Consulta
`:STOP`	Detiene la adquisición actual del osciloscopio, congelando la visualización de la señal en pantalla.	Configuración
`:MEASure:SOURce CH1`	Establece el Canal 1 como fuente para las mediciones automáticas.	Configuración
`:MEASure:PK2pk?`	Mide y devuelve el valor pico a pico de la fuente seleccionada. Es la diferencia entre el valor máximo y mínimo de la señal.	Consulta
`:MEASure:AMPLitude?`	Mide y devuelve la amplitud de la fuente seleccionada. En señales periódicas, representa la diferencia entre el nivel alto y bajo estable.	Consulta

Estos comandos permiten un control básico del osciloscopio. Existen muchos más comandos específicos dependiendo del modelo exacto del osciloscopio GW-Instek que se esté utilizando.

4.2. Comandos SCPI para Generadores de Funciones

Los generadores de funciones GW-Instek pueden controlarse mediante comandos SCPI para configurar todos los aspectos de las señales generadas. Los siguientes comandos son fundamentales para su automatización:

Comando	Descripción	Tipo
`*RST`	Restablece el generador a su configuración predeterminada de fábrica. Útil para comenzar desde un estado conocido.	Configuración
`SOURce1:FUNCtion SINusoid`	Configura la forma de onda del canal 1 como sinusoidal. Otras opciones comunes incluyen SQUare (cuadrada), RAMP (rampa), PULSe (pulso) y NOISe (ruido).	Configuración
`SOURce1:FUNCtion?`	Consulta la forma de onda actualmente configurada en el canal 1.	Consulta
`SOURce1:FREQuency 1000`	Establece la frecuencia de la señal del canal 1 a 1000 Hz (1 kHz). El rango de frecuencias depende del modelo específico del generador.	Configuración
`SOURce1:FREQuency?`	Consulta la frecuencia actual configurada en el canal 1.	Consulta
`SOURce1:AMPLitude 1`	Configura la amplitud de la señal del canal 1 a 1 voltio. Dependiendo del modelo, puede ser interpretado como valor pico, pico a pico o RMS.	Configuración
`SOURce1:AMPLitude?`	Consulta la amplitud actual configurada en el canal 1.	Consulta
`SOURce1:DCOffset 0`	Establece el offset DC de la señal del canal 1 a 0 voltios. Permite desplazar verticalmente la señal generada.	Configuración
`SOURce1:DCOffset?`	Consulta el valor de offset DC actual configurado en el canal 1.	Consulta
`OUTPut1 ON`	Activa la salida del canal 1, permitiendo que la señal configurada esté presente en el conector de salida del instrumento.	Configuración
`OUTPut1 OFF`	Desactiva la salida del canal 1, sin cambiar la configuración de la señal.	Configuración
`OUTPut1?`	Consulta el estado actual de la salida del canal 1 (activada o desactivada).	Consulta
`SOURce2:FUNCtion SQUare`	Configura la forma de onda del canal 2 como cuadrada (en modelos con múltiples canales).	Configuración
`SOURce1:PHASe 90`	Establece la fase de la señal del canal 1 a 90 grados. Útil para generar señales con relaciones de fase específicas.	Configuración
`SOURce1:PHASe?`	Consulta la fase actual configurada en el canal 1.	Consulta

4.3. Ejemplo de Secuencia de Comandos

El siguiente ejemplo muestra una secuencia típica de comandos SCPI para configurar un generador de funciones y capturar la señal resultante con un osciloscopio:

```
# Configuración del generador de funciones
generador.write("*RST")                    # Reset
generador.write("SOURce1:FUNCtion SIN")    # Onda sinusoidal
generador.write("SOURce1:FREQuency 1000")  # 1 kHz
generador.write("SOURce1:AMPLitude 0.5")   # 0.5V
generador.write("SOURce1:DCOffset 0")      # Sin offset
generador.write("OUTPut1 ON")              # Activar salida

# Configuración del osciloscopio
osciloscopio.write("*RST")                 # Reset
osciloscopio.write(":AUTOSet")             # Autoconfiguración
osciloscopio.write(":CHANnel1:DISPlay ON") # Mostrar canal 1
osciloscopio.write(":CHANnel1:COUPling AC")# Acoplamiento AC
osciloscopio.write(":TIMebase:SCALe 0.0005")# 0.5ms/div
osciloscopio.write(":MEASure:SOURce CH1")  # Medición en CH1
```

```
osciloscopio.query(":MEASure:FREQuency?")  # Medir frecuencia
osciloscopio.query(":MEASure:PK2pk?")      # Medir pico a pico
```

Esta secuencia ilustra cómo los comandos SCPI permiten automatizar un flujo de trabajo completo de generación y medición de señales, estableciendo una configuración específica y obteniendo mediciones precisas sin intervención manual.

4.4. Consideraciones de Comunicación

Al trabajar con múltiples instrumentos GW-Instek en la misma red, como en este caso con un osciloscopio y un generador de funciones, es importante tener en cuenta las siguientes consideraciones:

1. **Direcciones IP únicas**: Cada instrumento debe tener una dirección IP única en la red para evitar conflictos. Si se utiliza DHCP, verificar que las direcciones asignadas son diferentes; si se usa configuración manual, asignar direcciones IP distintas pero en la misma subred.
2. **Control de sesiones**: No es recomendable mantener múltiples conexiones simultáneas al mismo instrumento, ya que esto puede causar comportamientos inesperados. Cerrar siempre una conexión antes de establecer otra.
3. **Gestión de excepciones**: Como se muestra en los scripts de ejemplo, implementar siempre un manejo adecuado de excepciones para detectar y solucionar problemas de comunicación. Los errores más comunes incluyen tiempos de espera agotados (timeout) debido a instrumentos apagados o problemas de red.
4. **Tiempos de espera**: Algunos comandos, especialmente aquellos que involucran adquisición de datos o cambios significativos en la configuración, pueden requerir tiempo para completarse. Es recomendable incluir pequeñas pausas (utilizando `time.sleep()`) después de comandos críticos.
5. **Compatibilidad de comandos**: Aunque los comandos SCPI siguen un estándar, pueden existir variaciones entre diferentes modelos y series de instrumentos. Consultar siempre la documentación específica del modelo que se está utilizando.

La verificación inicial de la comunicación con los instrumentos, como se muestra en los scripts anteriores, proporciona una base sólida para desarrollar aplicaciones más complejas que aprovechen todo el potencial de automatización ofrecido por los equipos GW-Instek.

4.5. Verificación de Conexión con el Osciloscopio

Una vez configurado el entorno de desarrollo y la red, el primer paso crucial es verificar la correcta comunicación con el instrumento. El siguiente script permite establecer conexión con un osciloscopio GW-Instek utilizando PyVISA y confirmar que la comunicación está funcionando correctamente.

```
import pyvisa
import time

# Crear administrador de recursos usando el backend PyVISA-py
rm = pyvisa.ResourceManager('@py')

# Configurar parámetros de conexión
instrumento_ip = '172.118.1.247'
instrumento_puerto = 3000
cadena_recurso = f'TCPIP0::{instrumento_ip}::{instrumento_puerto}::SOCKET'

try:
    # Abrir conexión con el osciloscopio
    instrumento = rm.open_resource(cadena_recurso)
    instrumento.timeout = 10000  # 10 segundos de timeout
    instrumento.read_termination = '\n'
```

```
    instrumento.write_termination = '\n'

    # Verificar conexión con comando de identificación
    print("Verificando conexión con *IDN?...")
    identificacion = instrumento.query('*IDN?').strip()
    print(f"Osciloscopio identificado: {identificacion}")

    # Cerrar conexión
    instrumento.close()
    print("Conexión cerrada correctamente")

except Exception as e:
    print(f"Error al conectar con el osciloscopio: {e}")
    print(f"Tipo de error: {type(e).__name__}")
    print("Verificar que la dirección IP y puerto sean correctos")
```

Este script realiza las siguientes operaciones:

1. Importa las bibliotecas PyVISA y time necesarias para la comunicación con el instrumento.
2. Crea un administrador de recursos (ResourceManager) utilizando el backend puro de Python ('@py').
3. Define los parámetros de conexión: dirección IP y puerto del osciloscopio.
4. Construye la cadena de recurso VISA para conexión mediante socket TCP/IP.
5. Establece la conexión con el instrumento configurando los parámetros de terminación de lectura/escritura.
6. Envía el comando universal "*IDN?"(Identification Query) para verificar la conexión.
7. Muestra la respuesta del instrumento, que debería incluir información como fabricante, modelo y número de serie.

Si la comunicación es exitosa, la respuesta típica de un osciloscopio GW-Instek tendrá un formato similar a: `GW Instek,GDS-xxxx,SN:xxxxxxxxxx,Vx.xx.xx`

Este script simple es fundamental, ya que confirma que el entorno de Python está correctamente configurado y que la conexión de red con el instrumento funciona adecuadamente, estableciendo así la base para operaciones más complejas de automatización.

4.6. Verificación de Conexión con el Generador de Funciones

De manera similar al osciloscopio, es fundamental verificar la correcta comunicación con el generador de funciones antes de proceder con operaciones más complejas. El siguiente script establece comunicación con un generador de funciones GW-Instek y confirma su identidad mediante el comando SCPI estándar.

```
import pyvisa
import time

# Crear administrador de recursos usando el backend PyVISA-py
rm = pyvisa.ResourceManager('@py')

# Configurar parámetros de conexión
instrumento_ip = '172.118.1.248'  # Dirección IP del generador de funciones
instrumento_puerto = 5025        # Puerto estándar para muchos generadores
cadena_recurso = f'TCPIP0::{instrumento_ip}::{instrumento_puerto}::SOCKET'

try:
```

```
    # Abrir conexión con el generador de funciones
    instrumento = rm.open_resource(cadena_recurso)
    instrumento.timeout = 10000  # 10 segundos de timeout
    instrumento.read_termination = '\n'
    instrumento.write_termination = '\n'

    # Verificar conexión con comando de identificación
    print("Verificando conexión con *IDN?...")
    identificacion = instrumento.query('*IDN?').strip()
    print(f"Generador de funciones identificado: {identificacion}")

    # Cerrar conexión
    instrumento.close()
    print("Conexión cerrada correctamente")

except Exception as e:
    print(f"Error al conectar con el generador de funciones: {e}")
    print(f"Tipo de error: {type(e).__name__}")
    print("Verificar que la dirección IP y puerto sean correctos")
```

Este script sigue una estructura similar a la utilizada para el osciloscopio, con los siguientes aspectos específicos para el generador de funciones:

1. La dirección IP corresponde al generador de funciones en la red.
2. El puerto utilizado es 5025, que es común para comunicación SCPI en muchos generadores de funciones, aunque algunos modelos de GW-Instek pueden utilizar otros puertos como 3000 o 5000.
3. Los parámetros de terminación de lectura/escritura están configurados con '\n' (carácter de nueva línea), pero algunos instrumentos pueden requerir diferentes configuraciones como '\r\n' (retorno de carro + nueva línea).

La respuesta típica a la consulta de identificación de un generador de funciones GW-Instek tendrá un formato como: `GW Instek,AFG-xxxx,SN:xxxxxxxxxx,Vx.xx`

Es importante destacar que para algunos modelos de generadores de funciones GW-Instek, especialmente los más antiguos, podría ser necesario ajustar parámetros adicionales de comunicación como la velocidad de transmisión (baud rate) o configurar explícitamente el uso de caracteres de terminación específicos para garantizar una comunicación estable.

5. Automatización Integrada de Instrumentos

5.1. Control Simultáneo de Generador de Funciones y Osciloscopio

Una de las aplicaciones más potentes de la automatización con Python es la capacidad de controlar múltiples instrumentos de manera coordinada, creando sistemas de prueba completos. El siguiente script demuestra cómo integrar el control de un generador de funciones y un osciloscopio GW-Instek para configurar una señal de prueba y medir sus características automáticamente.

```
import pyvisa
import time

def automatizacion_integrada():
    rm = pyvisa.ResourceManager('@py')

    # Configuración de conexión para el generador de funciones
    generador_ip = '172.118.1.246'
```

```
generador_puerto = 1026
generador_recurso = f'TCPIP0::{generador_ip}::{generador_puerto}::SOCKET'

# Configuración de conexión para el osciloscopio
osciloscopio_ip = '172.118.1.247'
osciloscopio_puerto = 3000
osciloscopio_recurso = f'TCPIP0::{osciloscopio_ip}::{osciloscopio_puerto}::SOCKET'

try:
    print("=== INICIANDO SISTEMA DE PRUEBA AUTOMATIZADO ===\n")

    # 1. CONECTAR Y CONFIGURAR EL GENERADOR DE FUNCIONES
    print("Conectando con el generador de funciones...")
    generador = rm.open_resource(generador_recurso)
    generador.timeout = 10000
    generador.read_termination = '\n'
    generador.write_termination = '\n'

    # Verificar conexión con el generador
    id_generador = generador.query('*IDN?').strip()
    print(f"Generador identificado: {id_generador}")

    # Resetear y configurar el generador
    print("Configurando el generador de funciones...")
    generador.write('*RST')
    time.sleep(1)

    # Configuración de la señal de prueba
    generador.write('SOURce1:FUNCtion SINusoid')  # Forma de onda senoidal
    generador.write('SOURce1:FREQuency 15000')    # Frecuencia: 15 kHz
    generador.write('SOURce1:AMPlitude 0.05')     # Amplitud: 50 mVpp
    generador.write('SOURce1:DCOffset 0')         # Offset: 0V

    # Verificar la configuración del generador
    forma = generador.query('SOURce1:FUNCtion?').strip()
    frecuencia = generador.query('SOURce1:FREQuency?').strip()
    amplitud = generador.query('SOURce1:AMPlitude?').strip()
    offset = generador.query('SOURce1:DCOffset?').strip()

    print("Configuración del generador completada:")
    print(f"- Forma de onda: {forma}")
    print(f"- Frecuencia: {frecuencia} Hz")
    print(f"- Amplitud: {amplitud} Vpp")
    print(f"- Offset: {offset} V")

    # 2. CONECTAR Y CONFIGURAR EL OSCILOSCOPIO
    print("\nConectando con el osciloscopio...")
    osciloscopio = rm.open_resource(osciloscopio_recurso)
    osciloscopio.timeout = 10000
    osciloscopio.read_termination = '\n'
    osciloscopio.write_termination = '\n'

    # Verificar conexión con el osciloscopio
    id_osciloscopio = osciloscopio.query('*IDN?').strip()
```

```
print(f"Osciloscopio identificado: {id_osciloscopio}")

# Configurar el osciloscopio para la adquisición
print("Preparando el osciloscopio para la adquisición...")
osciloscopio.write(':AUTOSet')  # Configuración automática
time.sleep(2)  # Dar tiempo para que AutoSet complete

# 3. ACTIVAR LA SALIDA DEL GENERADOR
print("\nActivando salida del generador...")
generador.write('OUTPut1 ON')
estado_salida = generador.query('OUTPut1?').strip()
print(f"Estado de salida: {'ACTIVA' if estado_salida == '1' else 'INACTIVA'}")

# 4. REALIZAR MEDICIONES CON EL OSCILOSCOPIO
print("\nTomando mediciones del Canal 1...")
# Limpiar mediciones previas
osciloscopio.write(':MEASure:CLEar ALL')
time.sleep(1)

# Configurar mediciones para el Canal 1
osciloscopio.write(':MEASure:SOURce1 CH1')
osciloscopio.write(':MEASure:SOURce2 CH1')
time.sleep(1)

# Activar y obtener medición pico a pico
osciloscopio.write(':MEASure:PK2PK ON')
time.sleep(1)
pk2pk_ch1 = osciloscopio.query(':MEASure:PK2PK?').strip()

# Configurar para medición de frecuencia
osciloscopio.write(':MEASure:FREQuency ON')
time.sleep(1)
freq_ch1 = osciloscopio.query(':MEASure:FREQuency?').strip()

# 5. REPETIR MEDICIONES PARA EL CANAL 2 (si es necesario)
print("\nTomando mediciones del Canal 2...")
# Limpiar mediciones previas
osciloscopio.write(':MEASure:CLEar ALL')
time.sleep(1)

# Configurar mediciones para el Canal 2
osciloscopio.write(':MEASure:SOURce1 CH2')
osciloscopio.write(':MEASure:SOURce2 CH2')
time.sleep(1)

# Activar y obtener medición pico a pico
osciloscopio.write(':MEASure:PK2PK ON')
time.sleep(1)
pk2pk_ch2 = osciloscopio.query(':MEASure:PK2PK?').strip()

# 6. MOSTRAR RESULTADOS OBTENIDOS
print("\n=== RESULTADOS DE LAS MEDICIONES ===")
print("\nGenerador de Funciones:")
print(f"- Configuración: {forma}, {frecuencia} Hz, {amplitud} Vpp")
```

```
        print("\nMediciones del Osciloscopio:")
        print(f"- Canal 1: Pico a Pico = {pk2pk_ch1} V, Frecuencia = {freq_ch1} Hz")
        print(f"- Canal 2: Pico a Pico = {pk2pk_ch2} V")

        # 7. FINALIZAR PRUEBA Y DESCONECTAR
        print("\nFinalizando prueba...")
        # Desactivar salida del generador
        generador.write('OUTPut1 OFF')

        # Cerrar conexiones
        generador.close()
        osciloscopio.close()
        print("Conexiones cerradas correctamente")
        print("=== PRUEBA AUTOMATIZADA COMPLETADA ===")

    except Exception as e:
        print(f"\nERROR: {str(e)}")
        print(f"Tipo de error: {type(e).__name__}")
        print("Verificar conexiones y direcciones IP")

        # Asegurar que las conexiones se cierren en caso de error
        try:
            generador.close()
            print("Conexión con generador cerrada")
        except:
            pass

        try:
            osciloscopio.close()
            print("Conexión con osciloscopio cerrada")
        except:
            pass

# Ejecutar la automatización
if __name__ == "__main__":
    automatizacion_integrada()
```

5.2. Análisis del Flujo de Automatización

El script anterior implementa un flujo de trabajo completo para la automatización de pruebas, siguiendo estos pasos clave:

1. **Inicialización de conexiones**: Establece comunicación con ambos instrumentos en paralelo, verificando la identidad de cada uno.
2. **Configuración del generador**: Configura el generador para producir una onda senoidal de 15 kHz con 50 mV de amplitud, estableciendo un estímulo de prueba controlado.
3. **Preparación del osciloscopio**: Configura el osciloscopio para capturar la señal mediante el comando AutoSet, que ajusta automáticamente escalas y trigger.
4. **Generación de la señal**: Activa la salida del generador de funciones para producir la señal de prueba.
5. **Medición y adquisición de datos**: Configura y ejecuta mediciones específicas en el osciloscopio (pico a pico y frecuencia) en ambos canales, obteniendo directamente los valores numéricos.

6. **Presentación de resultados**: Muestra los parámetros configurados en el generador y los valores medidos por el osciloscopio, proporcionando una visión completa del experimento.

7. **Finalización ordenada**: Desactiva las salidas y cierra las conexiones con ambos instrumentos de manera controlada.

Este enfoque estructurado garantiza que las mediciones se realicen de manera coherente y repetible, proporcionando datos precisos para su posterior análisis. La automatización elimina errores humanos en la configuración y lectura de instrumentos, mejorando significativamente la confiabilidad de los resultados experimentales.

5.3. Extensiones y Mejoras

El script presentado sirve como base para sistemas de adquisición de datos más complejos. Algunas extensiones potenciales incluyen:

- **Barrido de parámetros**: Modificar el script para realizar barridos automáticos de frecuencia, amplitud u otros parámetros, generando tablas de datos para análisis de respuesta.
- **Registro y exportación**: Añadir capacidades para guardar los resultados en archivos CSV o bases de datos para análisis posterior o generación de informes.
- **Interfaz gráfica**: Desarrollar una interfaz de usuario con Streamlit o Tkinter que facilite la configuración de parámetros y visualización de los datos adquiridos.
- **Captura de formas de onda**: Extender el script para capturar y visualizar las formas de onda completas, complementando las mediciones puntuales.
- **Automatización de pruebas repetitivas**: Implementar bucles de medición para realizar múltiples adquisiciones bajo diferentes condiciones, facilitando estudios estadísticos o de estabilidad.

La estructura modular del código permite integrarlo fácilmente en sistemas de adquisición de datos más amplios o adaptarlo a requisitos específicos de la aplicación.

6. Conclusiones

La automatización de equipos GW-Instek mediante Python y comandos SCPI representa una poderosa herramienta para laboratorios modernos, entornos educativos y aplicaciones industriales. A lo largo de este taller, hemos explorado las técnicas fundamentales para establecer comunicación con osciloscopios y generadores de funciones a través de interfaces de red, implementar scripts de control y desarrollar sistemas de adquisición de datos automáticos.

Los resultados obtenidos demuestran que, con un conocimiento básico de programación en Python y el uso de bibliotecas como PyVISA, es posible transformar procedimientos manuales repetitivos en flujos de trabajo automáticos, mejorando significativamente la eficiencia, precisión y reproducibilidad de las mediciones. La capacidad de controlar simultáneamente múltiples instrumentos abre posibilidades para implementar pruebas complejas y caracterizaciones de sistemas que serían impracticables con métodos manuales.

Entre los beneficios identificados durante el desarrollo del taller, destacan:

- **Reducción de error humano:** La automatización minimiza la variabilidad introducida por operadores, garantizando condiciones de prueba consistentes.
- **Aumento en productividad:** Las pruebas automatizadas pueden ejecutarse continuamente sin supervisión directa, liberando recursos humanos para tareas más creativas.
- **Mejora en documentación:** Los sistemas automatizados facilitan el registro sistemático de resultados, favoreciendo la trazabilidad y reproductibilidad.

- **Flexibilidad y escalabilidad:** Los scripts desarrollados pueden adaptarse fácilmente para acomodar nuevos requisitos o configuraciones de prueba.

Si bien la implementación inicial de sistemas automatizados requiere una inversión de tiempo para el desarrollo de scripts y configuración de interfaces, el retorno de esta inversión se materializa rápidamente en forma de ahorro de tiempo, reducción de errores y mayor capacidad de análisis. Particularmente en entornos educativos, la automatización permite a los estudiantes concentrarse en el análisis e interpretación de resultados, más que en los aspectos mecánicos de la recolección de datos.

Como direcciones futuras, resulta prometedora la integración de técnicas de aprendizaje automático para el análisis avanzado de datos capturados, el desarrollo de interfaces gráficas más sofisticadas, y la interconexión con sistemas de documentación automática para generar informes comprehensivos. Asimismo, la combinación de estas técnicas de automatización con plataformas IoT podría facilitar el monitoreo remoto y la operación distribuida de laboratorios.

En conclusión, la conjunción de equipamiento GW-Instek con técnicas modernas de programación en Python representa un paradigma poderoso para la modernización de laboratorios e implementación de sistemas de prueba avanzados, con beneficios tangibles en términos de eficiencia, precisión y capacidades analíticas.

7. Referencias

Referencias

[1] GW Instek (2023). *Multi-function Programmable Oscilloscope MPO-2000 series*. Recuperado de: https://www.gwinstek.com/en-global/products/detail/MPO-2000

[2] Python Software Foundation (2024). *PyVISA: Control your instruments with Python*. Recuperado de: https://pyvisa.readthedocs.io/

[3] National Instruments (2023). *Understanding VISA and SCPI for Instrument Control*. Recuperado de: https://www.ni.com/en/shop/data-acquisition-and-control/application-software/driver-software/visa.html

[4] Keysight Technologies (2024). *Instrument Automation with Python*. Recuperado de: https://www.keysight.com/us/en/assets/7018-06894/white-papers/5992-4268.pdf