Título do Documento		Projeto	
Comunicação Python e Gerador de Sinais			
Escrito/Atualizado por			Data
Eric Sonagli Abbade			
Revisado por		Assinatura	Data
Aprovado por		Assinatura	Data

1. Objetivo

Este documento tem como objetivo mostrar o funcionamento da comunicação de um código Python com um gerador de sinais.

2. Desenvolvimento

Foi desenvolvido um código Python que comunica com o gerador de funções. Esse código, pode ser encontrado no seguinte link do GitHub: **https://github.com/cnpem-emi/jiga-qds-swls/blob/master/jiga-qds-swls.py** e é descrito logo abaixo:

```
# NOTE: the default pyvisa import works well for Python 3.6+
# if you are working with python version lower than 3.6, use 'import visa'
instead of import pyvisa as visa

import pyvisa as visa
import time
visa.log_to_screen() #Facilita debug.

#Inicialização:
rm = visa.ResourceManager()
wg = rm.open_resource('USB0::0x0957::0x2507::MY57100781::0::INSTR')

#Exemplo de algumas configurações básicas:
string = wg.query('*IDN?')
a = wg.query('*OPT?')
b = wg.query('*OPC?')
print(string, a, b)

#Leituras de paramêtros:
read_all = wg.query(':SOURce%d:APPLy?' % (1))  #Leitura do formato de onda,
frequência(Hz), amplitude(V) e offset(V), atuais, antes da edição desses
parâmetros.
print(read_all)
#Leitura de cada um dos paramêtros, atuais, antes de suas edições:
```

```
volt = wg.query(':SOURce%d:VOLTage?' % (1))
print(volt)
freq = wg.query(':SOURce%d:FREQuency?' % (1))
print(freq)
offset = wg.query(':SOURce%d:VOLTage:OFFSet?' % (1))
print(offset)
phase = wg.query(':SOURce%d:PHASe?' % (1))
print(phase)
func = wg.query(':SOURce%d:FUNCtion?' % (1))
print(func)

wg.write(':OUTPut%d %d' % (1, 1)) #Liga a geração de onda no canal 1.

#Edição de parâmetros do gerador:
wg.write(':SOURce%d:VOLTage %G' % (1, 2)) #Amplitude (em V).
wg.write(':SOURce%d:FREQuency %G' % (1, 1000)) #Frequência (em Hz).
wg.write(':SOURce%d:VOLTage:OFFSet %G' % (1, 0)) #Offset (em V.) Lembrar que o
valor médio da senoide e o valor DC são: 2*OFFSet.
wg.write(':SOURce%d:PHASe %s' % (1, 58))#Valor de 0º a 360º.
wg.write(':SOURce%d:FUNCtion %s' % (1, 'SINusoid')) #Formatos aceitos: 'SINusoid'
'SQUare' 'TRIangle' 'RAMP' 'PULSe' 'NOIS' 'PRBS'  'DC'.

#wg.write(':OUTPut%d %d' % (1, 0)) #Desliga a geração de onda no canal 1.

read_all_new = wg.query(':SOURce%d:APPLy?' % (1)) #Leitura do formato de onda,
frequência(Hz), amplitude(V) e offset(V), após a edição desses parâmetros.
print(read_all_new)
#Leitura de cada um dos paramêtros, após a edição desses parâmetros:
volt_new = wg.query(':SOURce%d:VOLTage?' % (1))
print(volt_new)
freq_new = wg.query(':SOURce%d:FREQuency?' % (1))
print(freq_new)
offset_new = wg.query(':SOURce%d:VOLTage:OFFSet?' % (1))
print(offset_new)
phase_new = wg.query(':SOURce%d:PHASe?' % (1))
print(phase_new)
func_new = wg.query(':SOURce%d:FUNCtion?' % (1))
print(func_new)

#Finalização:
wg.close()
rm.close()
```

Esse código foi feito para se comunicar com o gerador de função do modelo: 33500B da marca KEYSIGHT. Foi desenvolvido na versão 3.11 do Python. Foi feito o download da biblioteca pyvisa, pelo comando: **pip install pyvisa** (versão mais recente atualmente: 1.14.1). Vale ressaltar que em versões do Python mais antigas que 3.6 deve-se importar a biblioteca pelo comando: import visa, ao invés do import pyvisa as visa usado. É possível gerenciar algumas configurações dessa biblioteca, usando o terminal, conforme o seguinte tutorial: **https://pyvisa.readthedocs.io/en/latest/introduction/shell.html** .

É importante ressaltar que talvez sejam necessárias mudanças nesse código para o controle de outro gerador, já que equipamentos diferentes, frequentemente, são acionados por comandos distintos. Um exemplo de um tutorial do controle de um gerador de funções pelo Python é dado por: **Keysight Automation with Python - OSH Garage .** Outro exemplo foi desenvolvido por membros do CNPEM para uma aplicação semelhante e é encontrado no seguinte link: **ldc-sw/LDC Board Test/SCPI_Commands.py at main · cnpem-emi/ldc-sw (github.com)** .

Como sugerido, os comandos para o acionamento do gerador nos códigos Python dos dois exemplos são diferentes daqueles usados no programa descrito nesse documento. Isso ocorre, justamente pelo fato de que nos exemplos se estavam usando equipamentos diferentes para o controle, além de possíveis diferenças nos softwares e sistemas operacionais.

Existe um tutorial mais complexo que auxilia na descoberta dos comandos específicos para cada gerador que pode ser encontrado no link: **PythonAutomationSeries 4 writing your first program (youtube.com)** . Esse tutorial completo pode ser visualizado na reprodução automática do vídeo referenciado.

Conforme explicado nesse vídeo, deve-se baixar o software: Command Expert (válido para equipamentos da KEYSIGHT e similares), acessível pelo link: **Command Expert Downloads | Keysight** . Nesse software podem ser usados comandos específicos para controlar o gerador. Esse mesmo software (Command Expert) tem uma funcionalidade para traduzir esses comandos para Python, de modo que eles podem ser copiados e colados para um arquivo .py, permitindo o controle.

Abaixo serão colocadas imagens confirmando o funcionamento do processo:

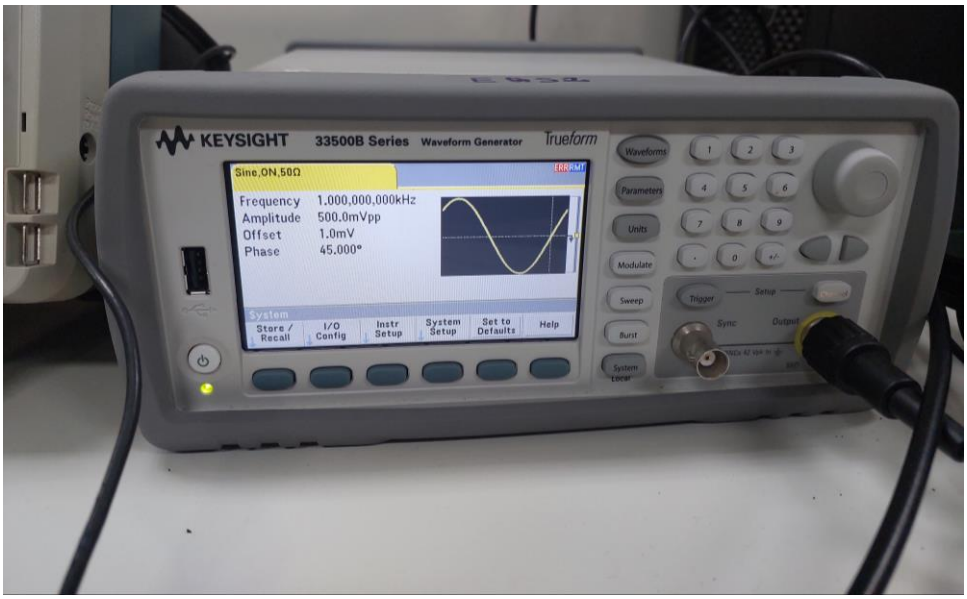

Figura 1: Configurações do gerador de função, similares as definidas no código.

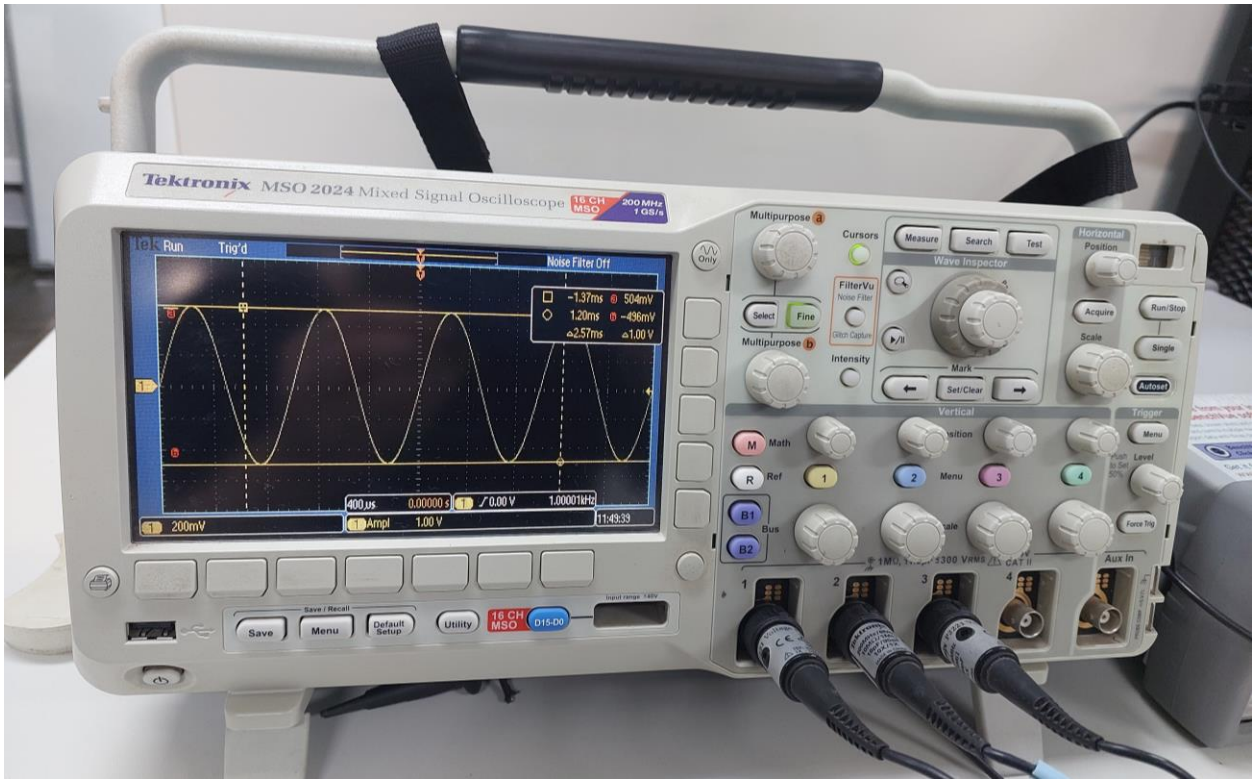

Figura 2: Leituras do osciloscópio comprovando o sucesso do experimento.

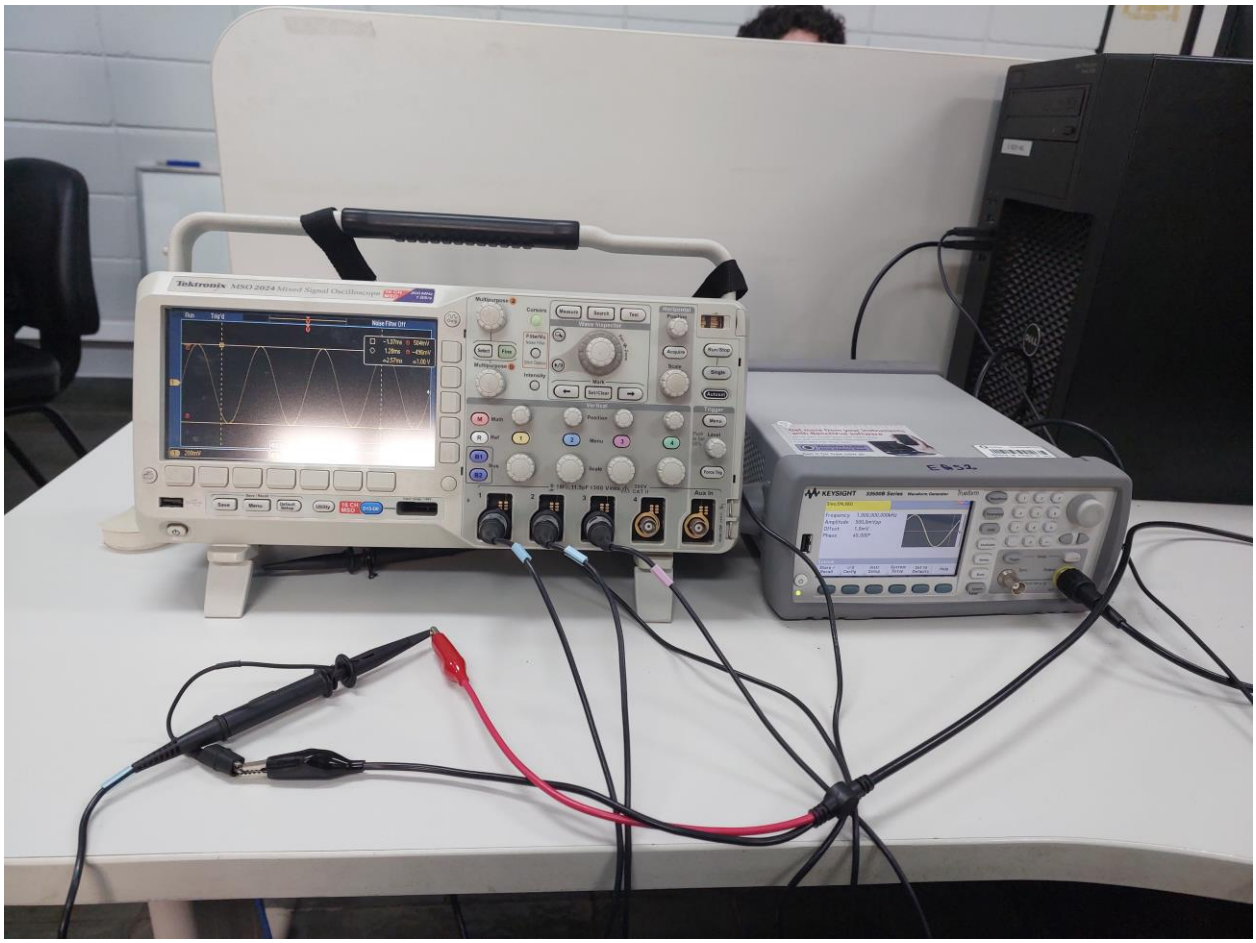

Figura 3: Conexão de gerador de função e osciloscópio, ilustrando o processo como um todo.

3. Conclusão

Pode-se concluir que a comunicação de um código Python com um gerador de sinais está funcional e, portanto, o objetivo desse documento foi atingido com sucesso.