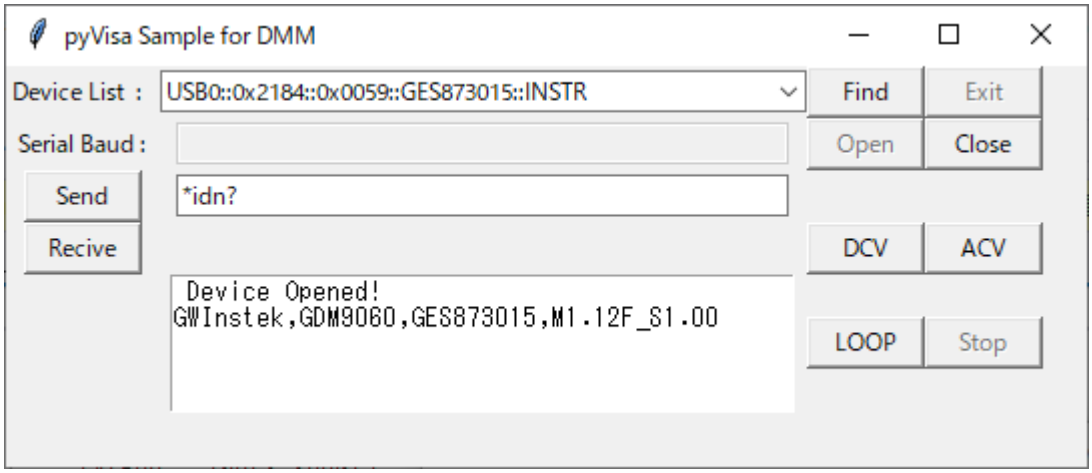
(2)　Python による VISA プログラミング

Python はプログラミングのための参考資料やサンプルが豊富にありますが、人口知能や Web アプリケーション、統計処理についてが多く、通信を使って計測機器などを制御する例はそれほど多くありません。

本ページでは GUI の作成とボタン処理を行う GUI ライブラリ(Tkinter)、通信を行う VISA ライブラリ(PyVISA)、一定周期で処理を行うインターバル動作、変数の共用を行うグローバル指定を説明いたします。

環境の準備などの手順はここでは扱いません、Python 本体の他に様々な開発ツールがありますのでインターネットで検索して Python3 の実行環境をそろえてください。また、ファイル処理やグラフ出力などはインターネットに豊富にサンプルがありますので必要に応じて追加してみてください。



今回作成する Python/Tkinter での GUI 表示

Python での画面の構成としては、ボタン 10 個、ドロップダウンリスト、テキストボックス 3 個、ラベル 2 個で、処理としてはボタンのほかに単純なインターバルでの動作の時計更新も盛り込んであります。

1. 初めに必要なライブラリとグローバル変数を登録します。

```
import pyvisa as visa
import tkinter as tk
import time
from time import sleep
from tkinter import ttk

rm = visa.ResourceManager()
instr = None
```

ライブラリ登録は import で行います、VISA のインポートは visa または pyvisa を登録します。Python の変数は何も指定しないとローカル変数になるようなので、リソースマネージャ用の変数 rm と計測器用の変数 instr を宣言します。複数の機器を制御する場合は、instr1、instr2 など変数を増やすことで対応します。

2. 画面のパーツを登録します。

```
#メイン画面の構築
root = tk.Tk()
root.geometry('530x160')
root.title('pyVisa Sample')

#ラベルの作成と配置
label1 = tk.Label(root, text = 'Device List  :  ')
label1.grid(row=0, column=0)

#コンボボックスの作成と配置
cbDevList= ttk.Combobox(root , width=50 )
cbDevList.grid(row=0 , column=1)
cbDevList.bind('<<ComboboxSelected>>',cmb_Sel)

#テキストボックスの作成と配置
txtOption = ttk.Entry(root , width=50 ,   state =tk.DISABLED )
txtOption.grid(row=1 , column=1)

#ボタンの作成と配置
btn3 = tk.Button(master=root, text='Open ',command=Open_clicked,state =tk.DISABLED )
btn3.grid(row = 1 , column=2)
```

画面パーツは grid で配置を行っています。画面の解像度や分解能、OS の種類によってサイズが変わってしまう症状があります。複数の環境で利用する場合はかなり余裕を持った配置が必要になります。細かいサイズ・位置を指定したい場合は place などを使用してください。ボタンやコンボボックスなどのイベントはここで登録しますが、種類によって登録方法が異なります。

3. イベントの処理を登録します。

```
def Close_clicked():                            #Closeボタン処理
    global instr                                 #Globalのinstr変数を利用

    btn2['state'] = tk.NORMAL     # OPEN/EXIT有効、CLOSE/Send/Recive禁止
    btn3['state'] = tk.NORMAL
    btn4['state'] = tk.DISABLED
    btn5['state'] = tk.DISABLED
    btn6['state'] = tk.DISABLED

    instr.close()                                #VISAデバイスをクロース

def Send_clicked():                             #送信ボタン処理
    global instr                                 #Globalのinstr変数を利用

    cbuff = txtSend.get()                        #テキストボックスから文字列を取得
    instr.write(cbuff)                           #文字列をVISAで送信
    txtRecive.delete('1.0',tk.END)               #テキストを消去
    if ('?' in cbuff) :                          #送信文字列に？がある場合
        txtRecive.insert (tk.END,instr.read())   #  文字列をVISA受信し表示

def Recive_clicked():                           #受信ボタン処理
    global instr                                 #Globalのinstr変数を利用

    txtRecive.delete('1.0',tk.END)               #テキストを消去
    txtRecive.insert (tk.END,instr.read())       #文字列をVISA受信し表示

def interval_work():                            # インターバルで行う処理を記述
    root.title('pyVisa Sample  ' + time.ctime()) # ここでは時刻をタイトルに表示
    root.after(1000, interval_work)              # インターバルの時間(ms)を指定
                                                 # 指定しなければインターバルは終了
```

ボタンの有効・無効は['state']で切替えます。
テキストの取得は get()、削除は delete()、追加は insert ()を使用します。
指定時間後にイベントを発生させる場合は after()を使用します、ここでは 1000ms 後に自分自身を指定しているので 1 秒ごとに処理が行われます。
デバイスを指定する instr は、それぞれの処理でグローバル宣言をしておかないと、ローカル変数が割り当てられ正しく動作しなくなりますので注意してください。

4. VISA 特有の処理を登録します。

```
def Find_clicked():                             # デバイスを検索し一覧表示します
    try:
        cbDevList['values']=['']
        instruments = rm.list_resources('?*')
        if len(instruments) > 0:
            cbDevList['values']=instruments
            cbDevList.current(0)                # Error処理
            btn3['state'] = tk.NORMAL
    except:
        print('Device Error')

def Exit_clicked():                             #Exitボタン処理

    rm.close()                                  # デバイスとの通信を終了します。
    root.destroy()
```

計測機との通信は
(1)初めにリソースを取得します。(グローバル変数で登録済)
(2)リソースからデバイス一覧を取得します。
(3)制御したいデバイスを見つけます。
の順番で開始します。

リソースが無いなどのエラーが発生した場合の対応を try/except で記述しています。

終了時はアプリケーションを終了すればリソースは開放されますが、終了前にリソースを開放します。

5. RS-232C 通信と Socket 通信では追加の設定を行います。

```
def Open_clicked():
    global instr
    visaAddr = cbDevList.get()
    if( visaAddr == '' ):
        print('No Device!')|
    else:
        try:
            instr = rm.open_resource(visaAddr)
            if instr.resource_name.startswith('ASRL'):
                instr.read_termination = '¥n'
                instr.write_termination = '¥n'
                instr.baud_rate = int(txtOption.get())

            if instr.resource_name.endswith('SOCKET'):
                instr.read_termination = '¥n'
                instr.write_termination = '¥n'

            btn2['state'] = tk.DISABLED     # OPEN/EXIT禁止、CLOSE/Send/Recive有効
            btn3['state'] = tk.DISABLED
            btn4['state'] = tk.NORMAL
            btn5['state'] = tk.NORMAL
            btn6['state'] = tk.NORMAL

            txtRecive.delete('1.0',tk.END)
            txtRecive.insert (tk.END,' Device Opened!')
# Error処理
        except:

            txtRecive.delete('1.0',tk.END)
            txtRecive.insert (tk.END,' Device Open Error!')
```

デバイス名の先頭に ASRL の文字がある場合が RS-232C、最後に SOCKET の文字がある場合が Socket 通信となります。ここでは送受信のデリミタを指定し、RS-232C ではさらにボーレートを指定します。

6. まとめ

実用アプリケーションには至っていませんが、アプリの最低限の内容を取り上げました。

- ・Python3 に対応しています、Python2 で使用する場合は多少の修正が必要です。
- ・GUI は標準の Tkinter を使用し、画面配置は単純な grid を利用しました。
- ・GUI のボタンやテキストなどの最低限の扱いを記述しました。
- ・簡単なインターバル処理の方法を記述しました。
- ・通信は PyVISA を使用しました。
- ・最低限のエラー処理を記述しました。
- ・グローバル変数の利用方法を記述しました。
- ・機器を増やす場合は機器のグローバル変数を追加して対応します。

後は使う方が必要なものを追加してご利用ください。

本資料のリンク
https://www.texio.co.jp/uploads/WebExpo/Study/Study_0001/python0002.html

完成版ソースコードのダウンロード
https://www.texio.co.jp/uploads/WebExpo/Study/Study_0001/Sample_VISA_DMM.zip