2022KO EKAINA/UZTAILA

Python-en bidezko LabVIEW-ren ordezkoa

Beñat Berasategui Migueliz

1 Sarrera

Garatutako programak, `Python_VISA_neurketa.py` [1], laborategian neurketak egiteko sentsore eta ordenagailu baten arteko komunikazioa kudeatzeko modu bat erakustea du helburu. Era honetako egoeretan `Python` programazio-lengoaia erabiliz egin daitekeenaren oinarrizko adibide bat da.

2 Aldez aurreko eskakizunak

Gai honi lotutako programekin lan egin ahal izateko, hainbat pakete izan behar ditugu ordenagailuan instalatuta.

Oinarrizko eskakizuna `Python` bera instalatuta izatea da. Horrekin batera, `pip` pakete instalatzailea eta `IDLE` editorea instalatu nituen nik [2]. Ondoren, `scipy` eta `matplotlib`, *Command Prompt*-ean komando hauek idatziz:

```
pip install scipy
pip install matplotlib
```

Interesatzen zaizkigun aplikazioetarako, beste bi liburutegi ezinbesteko izango zaizkigu: `PyVISA` [3], neurketa-gailuak `Python`-en bidez kontrolatu ahal izateko, eta programa erabilterrazagoa bihurtuko duen *erabiltzailearen interfaze grafikoa* (ingelesez GUI) eraikitzeko, `PySimpleGUI` [4][5].

```
pip install -U pyvisa-py
pip install PySimpleGUI
```

Azkenik, kodea exekutagarri bihurtzeko modu erraz bat `PyInstaller` modulua erabiltzea dela ikusita, hau ere instalatzea lagungarria izango da [6].

```
pip install pyinstaller
```

Honi esker berehalakoa da *Command Prompt*-ean ondoko komandoak idatzita gure programa (`test.py` adibidez) exekutagarri bihurtzea:

```
pyinstaller --onefile --noconsole test.py
```

`--onefile` zehaztapenak prozesu honetan sortutako fitxategien kopurua minimizatzea ahalbidetzen du, eta `--noconsole`-k exekutagarria irekitzean leiho bakarra zabaltzea.

3 ReFlex termometroa

Aurreko atalean aipatutako tresna guztiekin egin daitekeenaren adibide moduan, zuntz optiko bidezko ReFlex^{TM} termometroarekin informazioa eskuratzeko programa da nik idatzitakoa.

Prozesu honetan, ezinbestekoa izan da gailuarekin komunikatzeko bidea ulertzea, oinarrizko era batean izan bada ere. RS-232 kablearen bidez konektatzen da ordenagailura, eta komunikazioa gauzatzeko aginduak gailuaren gidan [7] azaldutakoak dira. Bertan esaten denez, softwarearen ezarpenak honakoak izan behar dira: *9600 Baud, 1 Stop-Bit and No-Parity.*

Hiru komando erabili ditut nik. Garrantzitsuena, `t` neurtutako tenperatura jasotzeko. Gailuaren tenperatura (*enclosure temperature*) eskuratzeko, `b` da erabili beharreko agindua. Azkenik, gailuari buruzko informazioa lortzeko komandoa `i` da.

Inportantea da jakitea komando bat era egokian exekutatu ondoren, "*" izartxo bat bidaltzen duela termometroak. Erroreren bat gertatzen duenean aldiz, "Errx" mezua bidaltzen da. Bi horietako bat jaso aurretik ezin da hurrengo komandoa bidali.

4 PyVISA bidezko komunikazioa

Egindako lehen saiakeren helburua `PyVISA` liburutegiaren bitartez ordenagailutik termometrora aginduak bidaltzen ikastea izan da. Horrekin lotutako kodearen hainbat zati azalduko ditut atal honetan.

Erabilitako liburutegiak hauek izan dira:

```
import pyvisa
import numpy as np
import matplotlib.pyplot as plt
import time
```

National Instruments-en *Measurement & Automation Explorer* programa (NI-MAX) erabili dut gailuaren zehaztapenak hobeto ulertu eta `Python`-en nola islatu ikusteko. Hauek dira komunikazioa hasteko sententziak:

```
rm = pyvisa.ResourceManager()
my_instrument = rm.open_resource('ASRL1::INSTR')
```

NI-MAX-eko zein aurreko ataleko gidan aipatutako ezarpenak era honetan aukera daitezke:

```
my_instrument.baud_rate = 9600
my_instrument.data_bits = 8
my_instrument.parity = 0 #None
```

```
my_instrument.stop_bits = 10 #one = <StopBits.one: 10>
my_instrument.flow_control = 0 #none
```

Garrantzitsua den beste detaile bat aginduak non bukatzen diren zehazten duten karaktere bereziak (`\n`, `\r`) era egokian erabiltzea da. Konbinazio honek ongi funtzionatzen du:

```
my_instrument.read_termination = '\n'
my_instrument.write_termination = '\r'
```

Gauzak horrela, prest gaude ordenagailu eta termometroaren arteko komunikazioa hasteko. Oinarrizko komandoak hauek dira:

```
my_instrument.write('command')
my_instrument.read()
my_instrument.clear()
my_instrument.close()
```

Aurreko atalean aipatutako moduan, gailuaren irakurketa "*" edo "Errx" mezuaz bukatzen da, ez lerro jauziarekin. Dena dela, `my_instrument.read()`. aginduak lerro bakarra itzultzen du aldiro. Horrez gain, *encoding*-arekin lotutako erroreak oso ohikoak direla ikusi dut. Hau modu praktikoan gestionatzeko (nahiz eta agian erarik ortodoxoena ez izan), ondoko funtzioak sortu ditut:

```
def full_print(my_instrument):
    #Reads and prints the device's whole answer
    while True:
        try:
            s=str(my_instrument.read()).strip()
            ss=''
            if '*' in s:
                my_instrument.clear()
                break
            else:
                print('    '+s)
        except UnicodeDecodeError:
            print('----UnicodeDecodeError----')

def full_read(my_instrument):
    #Reads and returns just the first line
    try:
        s=str(my_instrument.read())
        my_instrument.clear()
        return s
    except UnicodeDecodeError:
        print('----UnicodeDecodeError----')

def full_query(my_instrument, arg):
```

```
42      #Writes arg, reads and returns just the first line
43      my_instrument.write(arg)
44      try:
45          s=str(my_instrument.read())
46          my_instrument.clear()
47          return s
48      except UnicodeDecodeError:
49          print('----UnicodeDecodeError----')
```

`Prog_p1_datuak_irakurri.py` [1] fitxategian atal honetan aztertutako kode guztia dago bateratuta, bestelako gehigarri batzuekin. Esaterako, neurketak irudikatzeko lehen saiakera bat agertzen da. Berez, orain artekoa nahikoa da termometroaren irakurketa lortzeko, baina erabiltzen errazagoa izan dadin, interfaze grafiko bat sortzea izango da hurrengo pausua. Hori da hain zuzen ere ondoko atalean azalduko dena.

5 GUI baten eraikuntza

Iturri ezberdinetan kontsultatu ondoren, `Python`-ekin hau egiteko modurik oinarrizkoena `PySimpleGUI` liburutegia erabiltzea da. Nahiko zabala denez, egin nahi nuenaren arabera webgune ezberdinetatik kode zatitxoak biltzea izan da lan egiteko modua. Egia esan zaila da agindu bakoitzak zehazki zer egiten duen jakitea, baina gutxi gorabehera ulergarria da kode bloke bakoitzak egiten duena. Ikusi `Prog_p2_GUI.py` fitxategia [1].

Oinarri-oinarrizkoena azaltzen duten tutorial asko aurki daitezke sarean. Nik *Python GUI Development With PySimpleGUI* deitutako bat erabili dut [8]. Horrez gain, *The PySimpleGUI Cookbook* deituako webgune batean [9] liburutegi honekin egin daitekeenaren adibide sinple asko ematen dira, horiek konbinatuz gauza konplexuagoak egiteko.

Oraingoan hauek izan dira erabilitako liburutegiak:

```
1 import PySimpleGUI as sg
2 import os.path
3 import numpy as np
4
5 import matplotlib.pyplot as plt
6 from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
7 import matplotlib
8 matplotlib.use('TkAgg')
```

`Prog_p2_GUI.py` fitxategian [1] egindakoa, oro har, honakoa izan da:

1) Leihoa zabaltzean ageriko diren elementuak adierazi, `Pythoneko` listak erabiliz eta objektu bakoitzari dagozkion aukerak ongi zehaztu. Elementu motak

ezagutzeko, oso erabilgarria da guztiak biltzen dituen eredu hau erabiltzea: `Demo_All_Elements_Simple.py` [10].

2) Momenturen batean neurketak adierazten dituen grafikoak sortu nahiko ditugunez, `Matplotlib` erabiliz irudi sinple bat egingo dugu, interfaze grafikoaren baitan nola sar daitekeen ikusten hasteko.
3) *Layout* osoa definitu, aurrez zerrendatutako elementu guztiak bateratuz. Horrela, eta `sg.Window()` sententzia erabiliz, programa exekutatzean zabalduko den leihoa sortuko dugu.
4) *Event loop* delakoan sartu. Zati hau da benetan interfaze grafikoaren funtzionamendua zehazten duena. `While` egitura baten bitartez etengabe ematen da `event, values = window.read()` agindua. Leihoko elementuetako bakoitza definitzean, `key` aukerarekin etiketa bat esleitu diogu. Programa exekutatzean elementu horrekin interakziorik gertatu bada, `event` aldagaiak etiketa horren izena hartuko du, eta `values` zerrendak elementu ezberdinekin lotutako balioak bilduko ditu.
   Adibide moduan, kasu berezi bat: leihoa ixteko X botoia sakatzean, programa amaitzea interesatzen zaigu. Hori gerta dadin, kodean hau agertu behar da:

```
while True:
...
    if event == sg.WIN_CLOSED:
        break
...
window.close()
```

6 PySimpleGUI eta MatPlotLib konbinatzea

Aurreko bi ataletan azaldutakoa, PyVISAren bidezko komunikazioa eta GUI baten eraikuntza, konbinatzea nahiko erraza da, ez du arazorik ematen. Espero baino zailagoa izan dena GUIaren baitan matplotliben bidez sortutako grafikoa integratzea izan da, eta are gehiago neurketak egin ahala eguneratzen den grafikoa.

Azkeneko hau asko konplikatzen da eta ez dago bide errazik. Nik probak egin ditut funtzionatzen zuen zerbait lortu dudan arte, baina egia esan ez zait oso argi geratu. Beraz, ez naiz zati honen nondik norakoak azaltzen sartuko. Izan ere, programa osoaren zati txiki bat baino ez da, eta ez ezinbestekoa.

Informazio gehiago behar izatekotan, nik gehiengoa [11]-tik atera dut. Horrez gain, [10]-n adibide moduan ematen diren hainbat programetan behar nuenaren antzeko gauzak bilatzen saiatu naiz. `Python_VISA_neurketa.py` programa nagusian [1] dago inplementatuta.

Erreferentziak

[1] Beñat Berasategui Migueliz. *Python-en bidezko LabVIEW-ren ordezkoa*. URL: https://github.com/BeBerasategi/Pythonen-bidezko-LabVIEW-ren-ordezkoa. (kontsulta-data: 2022/07/15).

[2] Python Software Foundation. *Download Python*. URL: https://www.python.org/downloads/. (kontsulta-data: 2022/07/15).

[3] PyVISA Authors. *User guide*. URL: https://pyvisa.readthedocs.io/en/latest/introduction/configuring.html. (kontsulta-data: 2022/06/30).

[4] *PySimpleGUI*. URL: https://pysimplegui.readthedocs.io/en/latest/. (kontsulta-data: 2022/07/15).

[5] Python Software Foundation. *PySimpleGUI 4.60.1*. URL: https://pypi.org/project/PySimpleGUI/. (kontsulta-data: 2022/07/15).

[6] Python Software Foundation. *Pyinstaller 5.2*. URL: https://pypi.org/project/pyinstaller/. (kontsulta-data: 2022/07/15).

[7] Neoptix Canada LP. *Reflex^{TM} Fiber Optic Thermometer System*. URL: https://d3pcsg2wjq9izr.cloudfront.net/files/55901/download/646306/1-7.pdf. (kontsulta-data: 2022/07/15).

[8] Real Python. *Python GUI Development With PySimpleGUI*. URL: https://www.youtube.com/watch?v=-_z2RPAH0Qk. (kontsulta-data: 2022/07/15).

[9] *The PySimpleGUI Cookbook*. URL: https://pysimplegui.readthedocs.io/en/latest/cookbook/. (kontsulta-data: 2022/07/15).

[10] *PySimpleGUI Demo Programs*. URL: https://github.com/PySimpleGUI/PySimpleGUI/tree/master/DemoPrograms. (kontsulta-data: 2022/07/15).

[11] Real Python. *Integrating Matplotlib With PySimpleGUI*. URL: https://realpython.com/pysimplegui-python/#integrating-matplotlib-with-pysimplegui. (kontsulta-data: 2022/07/15).

[12] Graz University of Technology. *Establishing communication with PyVISA*. URL: http://lampx.tugraz.at/~hadley/num/ch9/python/9.2.php. (kontsulta-data: 2022/06/30).

[13] David Cortesi. *PyInstaller Manual*. URL: https://pyinstaller.org/en/stable/operating-mode.html. (kontsulta-data: 2022/07/15).