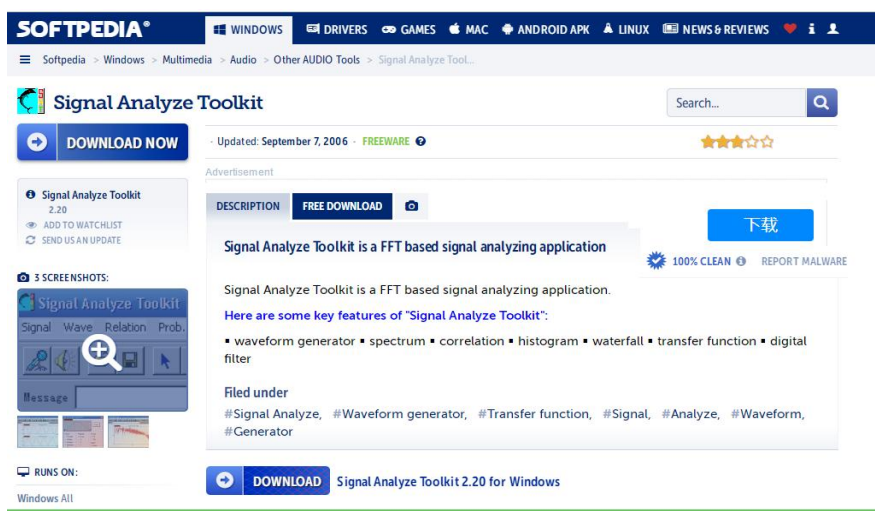


关于 Python DRVl 可重构虚拟仪器资源包

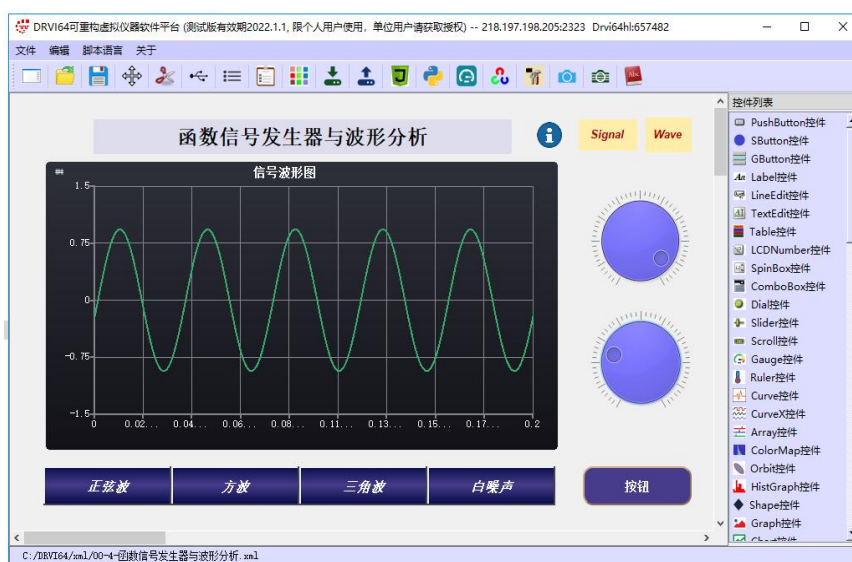
何岭松

华中科技大学机械学院仪器系

最近十年，一直用美国的 Matlab 进行教学和科研工作，十分方便；但 Matlab，需要隔三差五的激活后，才能授权使用；别人不让你继续用，那是分分钟的事。我的主业是计算机虚拟仪器，C++、Java、Asp、Php，懂的计算机编程语言多且杂，也会那么几招软件开发功夫，HaHa，共享软件的江湖上现在还能查到早年闯荡时留下的痕迹。



一直想写一个 Mini 版的 Matlab 软件，可以放在 U 盘中，随身携带到教室，用来进行测控类课程教学。N 年前，学习 QT_C++，将过去开发的 DRVl 虚拟仪器软件从 Visual_C++过渡到了 QT_C++。发挥 QT 优点的 DRVl 界面酷炫，速度快，体积小，总体效果不错。

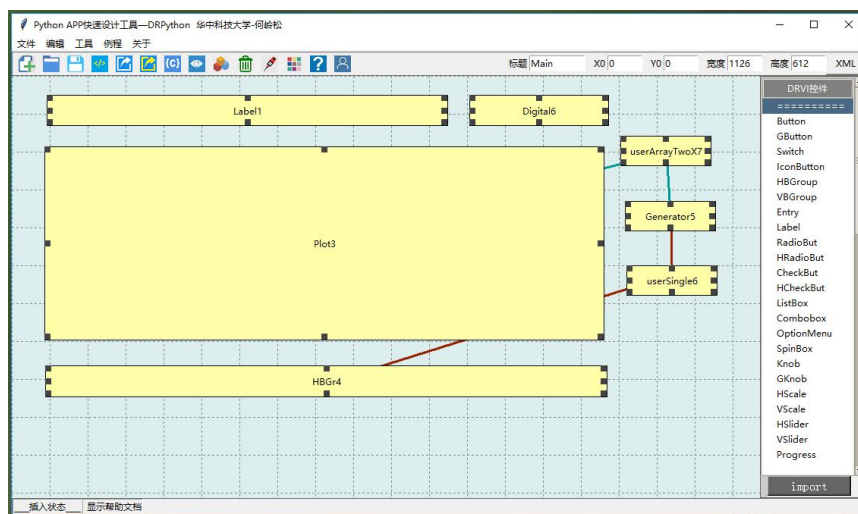


但 C++开发的软件有个硬伤，就是信号分析部分编程很难、功能扩展很难，功能实现非一般用户可以掌握。QT 版 DRVl 用于科研项目开发可以，但做为大众用途的工具软件，此路不通。

疫情仍未完全平息的这个暑假，异常炎热，无法外出观鸟，只能终日宅在空调屋里胡思乱想。忽然一天来了灵感，用 DRVl 中的软件总线和数据流驱动模型+Python 和 Python 旗下的 Numpy、SciPy、Matplotlib 科学三剑客,来打造一款信号分析和测控技术领域的开源“Matlab”。

闭门造车一个月，终于将 DRVI 数据流驱动模型嫁接到了 Python 和其自带的 Tkinter 库，以及第三方的 Matplotlib 库上，形成了 Python 可重构虚拟仪器界面库 DRVI(内核大约 3M)，并赶在研究生《数字信号分析理论与实践》开课前完成了 Python 版的 APP_Designer——DRPython。

学生可以用 Python 语言快速设计虚拟仪器风格的 Python 程序，从实践中探索信号分析的奥秘。这是用 DRPython 的图形化编程界面布局的一个小电子琴程序的界面：



布局结果仍然采用笔者提出的可重构虚拟仪器装配描述语言 XVIML 进行保存、加载、修改和运行：

```
<?xml version="1.0" encoding="utf-8"?>
<drvi4.1>
  <Main>
    <p1 v="0,0,0,1344,756"/>
    <p2 v=""/>
  </Main>
  <Label1>
    <p1 v="2,46,57,529,41"/>
    <p2 v="'#003355','#ffffff','小电子琴例程',20"/>
  </Label1>
  <Plot3>
    <p1 v="51,43,125,738,256"/>
    <p2 v="',''#003355','#000000','#ffffff','#00eeff','#ffff00',0,0.1,0,0,0,0"/>
  </Plot3>
  <HBGr4>
    <p1 v="101,44,414,741,42"/>
    <p2 v="'#003355','#ffffff','1,2,3,4,5,6,7,8,9',12,0"/>
  </HBGr4>
  <Generator5>
    <p1 v="79,808,197,120,40"/>
    <p2 v="0,44100,22050,0.6,100,0"/>
  </Generator5>
  <userSingle6>
    <p1 v="200,810,282,120,40"/>
    <p2 v=""/>
  <![CDATA[v=v*100
v=v+200
mDigital6.setValueSingle(v)]]>    </userSingle6>
  <Digital6>
    <p1 v="36,603,57,184,41"/>
    <p2 v="'#000000','#00ffaa',0,2,22"/>
  </Digital6>
  <userArrayTwoX7>
```

```

        <p1 v="203,802,111,120,40"/>
        <p2 v=""/>
<![CDATA[sa.play(1,44100,xDataArray,0)]]> </userArrayTwoX7>
    <ProbeLine>
        <p1 v="HBGr4,addCallBackSingle,userSingle6,setValueSingle,1"/>
    </ProbeLine>
    <ProbeLine>
        <p1 v="userSingle6,addCallBackSingle,Generator5,setVSFre,1"/>
    </ProbeLine>
    <ProbeLine>
        <p1 v="Generator5,addCallBack2DX,userArrayTwoX7,setValue2DX,3"/>
    </ProbeLine>
    <ProbeLine>
        <p1 v="userArrayTwoX7,addCallBack2DX,Plot3,setValue2DX,3"/>
    </ProbeLine>
    <Import>
<![CDATA[#==import related package====
import tkinter as tk
import tkinter.messagebox as msgbox
import numpy as np
from scipy.fftpack import fft,ifft
import matplotlib.pyplot as plt
import drvi.drviDSP as dsp
import drvi.drviControlls as dr
import drvi.drviSimpleAudio as sa
]]> </Import>
</drvi4.1>

```

对普通用户来说，XVIML 只是保存 Python 应用程序 GUI 布局设计的后台助记符，并不需要直接与其打交道。完成 GUI 布局 and 控件间的数据流连线后，就可以用 DRPython 的 GUI 界面导出功能，将 XVLML 助记符记录的 GUI 界面设计结果和为 DRVI 可重构虚拟仪器 GUI 界面库支持的用户可看懂的 Python 程序源代码框架。这时，用户就可以在其基础上用 Python 编辑器进行设计和修改，与一般的 Python 程序并无区别。

下面是直接导出的一个可运行的 Python 小电子琴程序：

```

import tkinter as tk
import tkinter.messagebox as msgbox
import numpy as np
from scipy.fftpack import fft,ifft
import matplotlib.pyplot as plt
import drvi.drviDSP as dsp
import drvi.drviControlls as dr
import drvi.drviSimpleAudio as sa

#==CallBack Function of MenuBar=====
def mExit():
    win.destroy()
    return 1

def mAbout():
    k=msgbox.askyesno('About','This is a APP frame by DRPython !')
    return 1

def muserSingle6(v):
    v=v*100
    v=v+200
    mDigital6.setValueSingle(v)

```

```

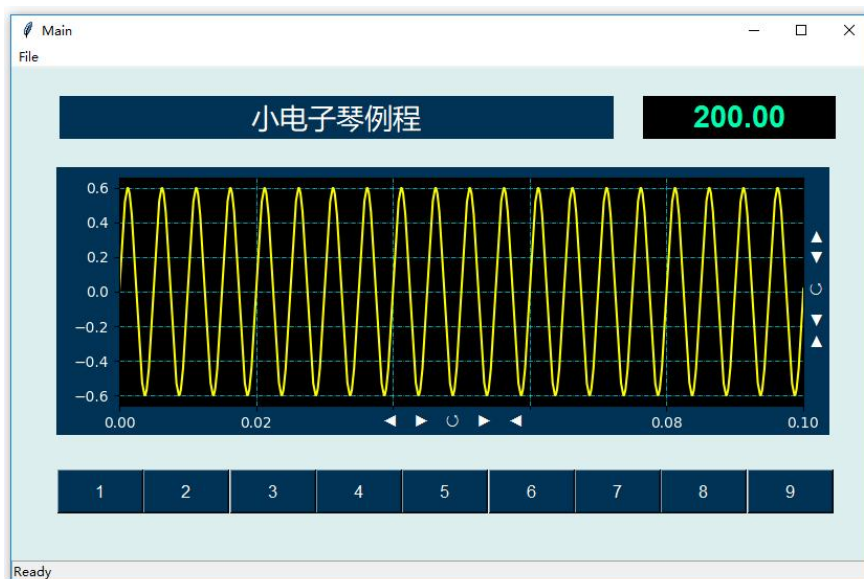
mGenerator5.setVSFre(v)
print("data ready")

def muserArrayTwoX7(dx,xDataArray):
    sa.play(1,44100,xDataArray,0)
    mPlot3.setValue2DX(dx,xDataArray)
    print("data ready")

#==Main Window=====
win= tk.Tk()
win.wm_title('Main')
winWidthCur=986
winHeightCur=572
win.geometry('986x572')
win.config(bg="#ddeeee")
#==MenuBar=====
menubar=tk.Menu(win)
fmenu=tk.Menu(menubar,tearoff=False)
fmenu.add_command(label="About",command=mAbout)
fmenu.add_command(label="Close",command=mExit)
menubar.add_cascade(label="File",menu=fmenu)
win.config(menu=menubar)
#==StatusBar=====
sbar=tk.Label(win,text="Ready",bd=1,relief=tk.SUNKEN,anchor=tk.W)
sbar.pack(side=tk.BOTTOM,fill=tk.X)
#==Layout of controls=====
mLabel1=dr.DRLabel(win,46,27,529,41,'#003355','#fffff','小电子琴例程',20)
mPlot3=dr.DRPlot(win,43,95,738,256,"'#003355','#000000','#fffff','#00eeff','#fff00',0,0.1,0,0,0,0)
mHBGr4=dr.DRHButtonGroup(win,44,384,741,42,'#003355','#fffff','1,2,3,4,5,6,7,8,9',12,0)
mGenerator5=dsp.DRGenerator(0,44100,22050,0.6,100,0)
mDigital6=dr.DRDigital(win,603,27,184,41,'#000000','#00ffaa',0,2,22)
#==User defined data probe=====
mHBGr4.addCallBackSingle(muserSingle6)
mGenerator5.addCallBack2DX(muserArrayTwoX7)
#==Main Loop=====
win.mainloop()

```

这是运行后的程序界面，初创作品的 GUI 界面虽然还不是很漂亮，但核心的图形化编程和数据流驱动的程序设计思想近乎完美的实现，十分满意。



虽然图形化编程创建出的一个带 GUI 界面的 Python 信号分析程序就几十行代码，但它背后站着的

是 Numpy、SciPy、Matplotlib 科学三剑客，在数学运算和曲线显示方面的深厚功力与 Matlab 可以说是旗鼓相当。这样学生就可以将精力专注在信号分析问题本身，而不是在程序的设计实现上。

DRVI/DRPython 有 Numpy、SciPy、Matplotlib 科学三剑客助剑，相信一定可以象 Don_Quijote_de_la_Mancha 一样，在共享软件的江湖上闯上出一小片自己的天地。HaHa，不自量力的 DRVI/DRPython 和我！

但天有不测风云，AI 的出现完全颠覆了传统编程技术发展路径，图形化编程多半会被 AI Coding 直接取代，费了很大劲实现的 DRPython 图形化编程工具，刚出道，就 Out 了；但做为基石，DRVI 库中的虚拟控件还是很有用的，可以让 AI 组装出的 Python 程序更简明、直观。

