Itasdi

Innovative Teaching Approaches in development of Software Designed Instrumentation and its application in real-time systems

Praktikum iz merno-akvizicionih sistema

Co-funded by the Erasmus+ Programme of the European Union

Itasdi

Innovative Teaching Approaches in development of Software Designed Instrumentation and its application in real-time systems

Faculty of Technical Sciences

Ss. Cyril and Methodius University
Faculty of Electrical Engineering and Information Technologies

Zagreb University of Applied Sciences

School of Electrical Engineering
University of Belgrade

Faculty of Physics
Warsaw University of Technology

Co-funded by the Erasmus+ Programme of the European Union

Lekcija 13

Kreiranje grafičkog interfejsa u Python-u.

Cilj

Cilj lekcije je da se studenti upoznaju sa:

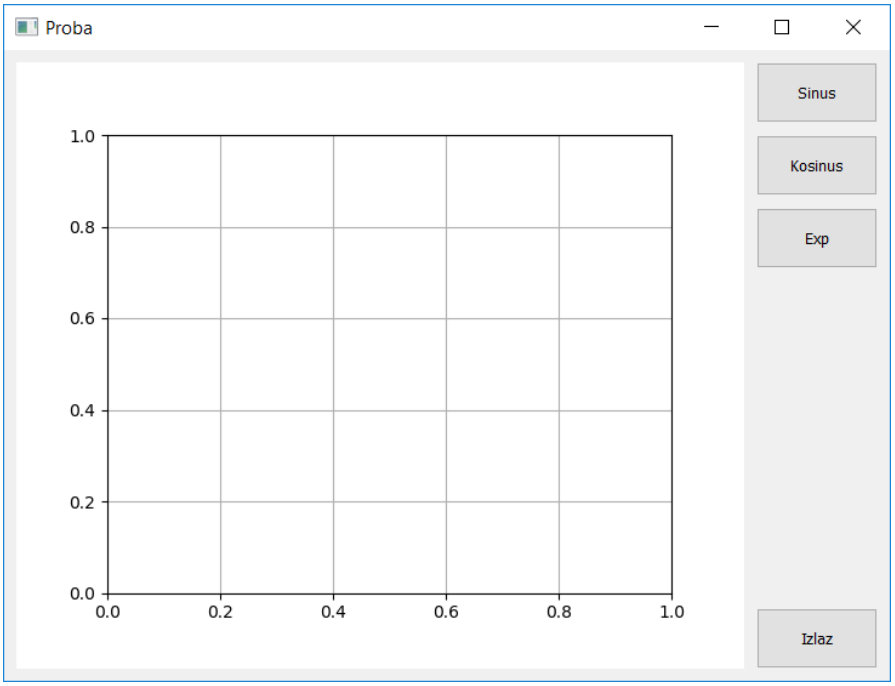
- Kreiranje grafičkog interfejsa – dodavanja tastera, polja za unos i prikaz teksta, prikaz grafika

Oprema

- Računar sa instaliranim ***Spyder*** softverskim okruženjem.

13.1 Grafički interfejs u Python-u

U Python programskom jeziku se može kreirati grafički interfejs koji omogućava komunikaciju između korisnika i programa. Funkcije za kreiranje grafičkog interfejsa se nalaze u biblioteci **PyQt5**. Primer grafičkog interfejsa je dat na Sl. 13.1.1.



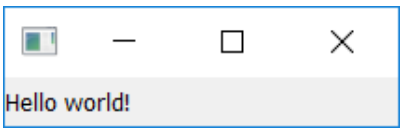
Sl. 13.1.1. *Primer grafičkog interfejsa kreiranog u Python programskom jeziku*

Zadatak 13.1.1. Kreiranje početne aplikacije.

1. U editoru otvoriti novu *.py* skriptu (*File>>New File*).
2. Potrebno je dodati biblioteku *PyQt5* u okruženje. Iz biblioteke je moguće učitati samo funkcije koje su potrebne za program koji se kreira. Iz biblioteke *PyQt5.QtWidgets* učitati funkcije *QApplication* i *Qlabel*.

```
from PyQt5.QtWidgets import QApplication, QLabel
```

3. Potrebno je kreirati aplikaciju komandom `app=QApplication([])`. Svaki Python GUI mora da ima tačno jednu instancu aplikacije. Većina delova aplikacije neće raditi ukoliko se ne izvrši ova linija koda. Ukoliko aplikacija koristi parametre, potrebno je unutar uglastih zagrada ubaciti parametre. Ako ne postoje parametri koji se prosleđuju aplikaciji, ostaviti prazne ove zagrade.
4. Zatim, kreirati tekstualni indikator u aplikaciji. Pomoću funkcije *QLabel* se definiše tekstualna labela na grafičkom interfejsu. Kao argument funkcije se prosleđuje string koji sadrži željeni tekst.
   `label = QLabel('tekst')`
   Da bi se tekstualni indikator pojavio na ekranu potrebno je reći `label.show()`.
5. Ukoliko želimo da se aplikacija izvršava sve dok je korisnik ne zatvori potrebno je dodati komandu `app.exec_()` na kraju programa.
6. Sačuvati skriptu pod imenom 13_01_01.py.
7. Pokrenuti i testirati program. Aplikacija treba da izgleda kao na Sl. 13.1.2.



Sl. 13.1.2. *Rezultat izvršavanja programa iz zadatka 13.1.1.*

Zadatak 13.1.2. Kreirati aplikaciju koja ispisuje poruku u konzoli kada se pritisne taster **Start**.

1. U editoru otvoriti novu *.py* skriptu (*File>>New File*).
2. Iz biblioteke *PyQt5.QtWidgets* učitati funkcije *QApplication* i *QPushButton*.
3. Kreirati *PyQt* aplikaciju: `app = QApplication([])`
4. Dodati taster u aplikaciju. Kao argument funkcije *QPushButton* se prosleđuje tekst koji želimo da piše na samom tasteru.
   `button = QPushButton('Start')`
5. Nakon kreiranja samog tastera, potrebno je dodati i akciju koja se izvršava prilikom pritiska tastera. Akcija se definiše u posebnoj funkciji koju je potrebno kreirati. Kreirati funkciju **button_fcn** koja nema ulaznih argumenata, a u telu funkcije se nalazi ispis poruke korisniku (funkcija *print*). *Napomena: funkcije u Python programskom jeziku se mogu definisati bilo gde u programu, pre pozivanja same funckije. Praksa je da se sve funkcije definišu na početku programa.*
6. Pomoću naredbe `button.clicked.connect(button_fcn)` se akcija definisana u funkciji **button_fcn** povezuje sa tasterom **button**.
7. Da bi taster mogao da se vidi u aplikaciji potrebno je dodati komandu `button.show()`.
8. Dodati komandu `app.exec_()` na kraju programa.
9. Sačuvati skriptu pod imenom 13_01_02.py.
10. Pokrenuti i testirati program.

Zadatak 13.1.3. Kreirati aplikaciju koja ispisuje tekst koji je uneo korisnik kada se pritisne taster **Prikazi**.

1. U editoru otvoriti novu *.py* skriptu (*File>>New File*).
2. Iz biblioteke *PyQt5.QtWidgets* učitati funkcije *QApplication, QPushButton* i *QEditLine*.
3. Kreirati *PyQt* aplikaciju: `app = QApplication([])`
4. Dodati polje za unos teksta u aplikaciju. Kao argument funkcije *QEditLine* se prosleđuje tekst koji želimo da piše u polju za unos teksta prilikom pokretanja aplikacije. Ova funkcija se može pozvati i bez argumenta.
   `text_input = QEditLine('Tekst poruke')`
5. Dodati taster **Prikazi** u aplikaciju.
6. Dodati funkciju koja će se izvršiti prilikom pritiska tastera. Potrebno je ispisati tekst koji je korisnik uneo u polje za upis teksta. Tom tekstu se može pristupiti pomoću naredbe `poruka = text_input.text()`. Kreirati funkciju **button_fcn** koja nema ulaznih argumenata, a u telu funkcije se nalazi ispis **poruke** korisniku (funkcija *print*). Povezati definisanu funkciju sa dugmetom.
7. Dodati komande `button.show()` i `text_input.show()` kako bi se oni prikazali u aplikaciji.

11. Dodati komandu `app.exec_()` na kraju programa.
12. Sačuvati skriptu pod imenom 13_01_03.py.
13. Pokrenuti i testirati program. Izmeniti sadržaj polja za unos teksta i pritisnuti opet taster **Prikazi**.

Primetiti da su se otvorila dva prozora. U jednom se nalazi taster, a u durom polje za unos teksta. Kako Python podržava objektno orijentisano programiranje, mnogo bolji način za kreiranje aplikacije je preko klase koja definiše samu aplikaciju.

Zadatak 13.1.4. Izmeniti zadatak 13.1.3. tako da se aplikacije definiše kao jedna klasa koja sadrži taster, polje za unos teksta i polje za prikaz poruke koju je uneo korisnik (Slika 13.1.3).

1. U editoru otvoriti novu *.py* skriptu (*File>>New File*).
2. Iz biblioteke *PyQt5.QtWidgets* učitati funkcije *QApplication, QPushButton, QEditLine* i *QWidget*. Dodati i biblioteku *system* kako bi se pristupilo sistemskim podacima.
3. Program počinje od kreiranja aplikacije kojoj se prosleđuju sistemski podaci `app = QApplication(sys.argv)`. Zatim potrebno je kreirati klasu koja sadrži elemente aplikacije. Program se završava kada korisnik izađe iz aplikacije, što je omogućeno pomoću funkcije `app.exec_()`.
4. Definisati klasu **App** kojoj se prosleđuje *QWidget* klasa koja u sebi sadrži funkcije za manipulaciju elementima u aplikaciji. U samoj klasi su definisane metode za definisanje aplikacije.
5. Metoda **__init__** predstavlja konstruktor klase **App**. On se poziva prilikom kreiranja objekta klase u glavnom programu. Ovde se podešava pozicija aplikacije na ekranu, veličina prozora i naziv aplikacije. Veličina i pozicija su izraženi u pikselima, a pozicija se definiše pozicijom gornjeg levog ćoška aplikacije. Da bi se definisali ostali elementi, u konstruktoru se poziva funkcija

initUI za inicijalizaciju korisničkog interfejsa. Različitim metodama i promenljivim unutar klase se pristupa preko ključne reči **self**.

6. U metodi **initUI** se pomoću funkcije `setGeometry` postavlja geometrija aplikacije (veličina i pozicija). Pomoću funkcije `setWindowTitle` dodaje se naziv aplikacije. Unutar ove funkcije je potrebno definisati sve elemente koje aplikacija treba da sadrži.
7. Svi elementi koji se nalaze u aplikaciji se definišu kao i ranije. Moguće je promeniti poziciji i veličinu svakog elementa pomoću funkcija `move()` i `resize()`, respektivno.
8. Funkcija koja se pridružuje tasteru se definiše kao zasebna metoda klase na isti način kao i što je ranije opisano.
14. Da bi se aplikacija prikazala na ekranu potrebno je unutar metode **initUI** napisati naredbu `self.show()`.
15. Sačuvati skriptu pod imenom 13_01_04.py.
16. Pokrenuti i testirati program.

```
from PyQt5.QtWidgets import QApplication, QPushButton, QLineEdit, QLabel, QWidget
import sys

class App(QWidget):
    def __init__(self):
        super().__init__()
        self.title = 'Zadatak 13.1.4'
        self.left = 200
        self.top = 200
        self.width = 120
        self.height = 190
        self.initUI()

    def initUI(self):
        self.setWindowTitle(self.title)
        self.setGeometry(self.left, self.top, self.width, self.height)

        self.text_input = QLineEdit('Tekst', self)
        self.text_input.resize(100, 50)
        self.text_input.move(10,10)

        self.button = QPushButton('Start', self)
        self.button.resize(100,50)
        self.button.move(10,70)
        self.button.clicked.connect(self.button_fcn)

        self.text_output = QLabel(' ', self)
        self.text_output.resize(100, 50)
        self.text_output.move(10,130)

        self.show()

    def button_fcn(self):
        poruka = self.text_input.text()
        self.text_output.setText(poruka)

app = QApplication(sys.argv)
ex = App()
app.exec_()
```

Sl. 13.1.3. *Kod koji realizuje aplikaciju definisanu zadatkom 13.1.4.*

Zadatak 13.1.5. Kreirati aplikaciju koja omogućava da se pritiskom na taster **Prikazi signal**, na grafiku prikazuje 1000 slučajno generisanih odbiraka sa srednjom vrednošću koju unosi korisnik.

1. U editoru otvoriti novu *.py* skriptu (*File>>New File*).

2. Iz biblioteke *PyQt5.QtWidgets* učitati funkcije *QApplication*, *QPushButton*, *QEditLine* i *QWidget*. Dodati biblioteku *system*. Biblioteka *numpy.random* služi za generisanje slučajnih odbiraka (`import numpy.random as r`). Za prikaz grafika u aplikaciji su potrebne funkcije iz biblioteke *matplotlib* (Sl.13.1.4).

```
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg as FigureCanvas
from matplotlib.figure import Figure
import matplotlib.pyplot as plt
```

Sl. 13.1.4. *Učitavanje funkcija za prikaz grafika u aplikaciji*

3. U glavnom delu programa se kreira aplikacija i objekat klase **App**. Na kraju glavnog programa se nalazi naredba `app.exec_()`.
4. Potrebno je definisati klasu **PlotCanvas** čiji su objekti grafici koji će biti prikazani u aplikaciji (Sl. 13.1.5).

```
class PlotCanvas(FigureCanvas):

    def __init__(self, parent = None, width = 2, height = 2, dpi = 100):
        fig = Figure(figsize=(width, height), dpi=dpi)
        self.axes = fig.add_subplot(111)
        self.axes.grid()

        FigureCanvas.__init__(self, fig)
        self.setParent(parent)

        FigureCanvas.setSizePolicy(self,
                QSizePolicy.Expanding,
                QSizePolicy.Expanding)
        FigureCanvas.updateGeometry(self)

    def plot(self, x):
        self.axes.cla()
        self.axes.grid()
        self.axes.plot(x, 'o')
        self.draw()
```

Sl. 13.1.4. *Učitavanje funkcija za prikaz grafika u aplikaciji*

Klasa se sastoji od konstruktora i jedne metode. Slično kao i pri kreiranju objekta aplikacije, u konstruktoru grafika se postavljaju dimenzije samog grafika i definiše se sama figura u kojoj će biti prikazan signal. Promenljiva **axes** definiše osu u odnosu na koju se crta grafik. Unutar jedne figure može biti više osa (*subplot*).

Metoda plot je vezana za objekat klase i služi za prikaz signala na grafiku. Na početku se izbriše sve što se nalazi na grafiku (*axes.cla*), a zatim se pomoću naredbe *axes.plot* definiše koja će biti prikazana na grafiku. Naredba *axes.draw* služi za crtanje signala.

5. Definisati klasu **App** kojoj se prosleđuje *QWidget* klasa.
6. U konstruktoru klase **App** (**__init__**) definisati da veličina prozora bude 550x500 piksela, a da se gornji levi ugao prozora nalazi na poziciji (150, 150). Nazvati aplikaciju **Zadatak 13.1.5**.
7. U metodi **initUI** postaviti geometriju aplikacije i njen naziv.
8. Kreirati polje za unos teksta dimenzija 200x75, sa početnom pozicijom u tački (10, 10). Početna vrednost u polju treba da bude 1.
9. Kreirati taster **Prikazi signal** koji ima dimenzije 300x75 piksela, sa početnom pozicijom u tački (240, 10).

10. Kreirati objekat **graphic** klase **PlotCanvas** (Sl. 13.1.5). Grafik treba da ima dimenzije 5.3x3.9, sa početnom pozicijom u tački (10, 100).

```
self.graphic = PlotCanvas(self, width = 5.3, height = 3.9)
self.graphic.move(10, 100)
```

Sl. 13.1.5. *Kreiranje objekta klase PlotCanvas*

11. Kreirati funkciju **prikaz_odbiraka** (Sl. 13.1.6) koju treba povezati sa tasterom **Prikazi signal**. Prvo treba pročitati koju vrednost je uzeo korisnik. Obratiti pažnju da funkcija *text()* vraća podatak tipa *string* tako da je potrebno izvršiti konverziju u *float*. Slučajno generisanje odbiraka sa normalnom raspodelom se vrši pomoću funkcije *normal* iz biblioteke *numpy.random*. Argumenti funkcije *normal* su srednja vrednost, standardna devijacija i broj odbiraka.
`odbirci = r.normal(srednja_vrednost, 1, N)`, pri čemu je N = 1000.
Prikazati odbirke na grafiku pozivanjem metode **plot** objekta **graphic**.

```
def prikaz_odbiraka(self):
    srednja_vrednost = float(self.text_input.text())
    N = 1000
    odbirci = r.normal(srednja_vrednost, 1, N)
    self.graphic.plot(x = odbirci)
```

Sl. 13.1.6. *Funkcija prikaz_odbiraka koja služi za prikaz slučajno generisanih odbiraka*

17. Da bi se aplikacija prikazala na ekranu potrebno je unutar metode initUI napisati naredbu `self.show()`.
18. Sačuvati skriptu pod imenom 13_01_05.py.
19. Pokrenuti i testirati program.

Zadatak za samostalan rad: Kreirati aplikaciju koja će imati četiri tastera, jedno polje za unos teksta, jedno polje za prikaz teksta i grafik. Korisnik treba prvo da unese vrednost u polje za unos teksta. Početna vrednost treba da bude 1. Ta vrednost predstavlja amplitudu signala koji će biti prikazan na grafiku. U zavisnosti od tastera koji je pritisnut se prikazuje jedna od četiri funkcije: $\sin(t)$, $\cos(t)$, $\sin(3t)$ i $\cos(3t)$. U polju za prikaz teksta potrebno je ispisati izabrani signal pomnožen sa zadatom amplitudom. Npr. ako korisnik unese vrednost 2 i izabere prikaz signala $\sin(3t)$ u polju za prikaz teksta će pisati „2*sin(3t)“.
Napomena: kreiranje vremenske ose je opisano u lekciji 12.