Politechnika Rzeszowska

Wydział Budowy Maszyn i Lotnictwa

Katedra Mechaniki i Robotyki Stosowanej

Instrukcja do Laboratorium z KSP:
Port Szeregowy w Python, cz. I

Autor:
Paweł Penar

1 Cel ćwiczenia

Używając dowolnego edytor tekstu i języka Python (+ biblioteki) napisać skrypt, który przeprowadza procedurę konfiguracji urządzenia podłączonego do portu szeregowego, które jest symulowane przez program *Serial Device Emulator*.

2 Wstęp do języka Python

2.1 Python - odnośniki do zasobów internetowych

Należy zapoznać się z następującymi stronami internetowymi

- `https://www.python.org/` strona domowa języka Python
- `http://winpython.github.io/` Paczka pod system Windows zawierający kompilator Pythona, zbiór pakietów (bibliotek) (m.in numpy, scipy, pyserial) w tym bibliotekę QT z programem QtDesigner oraz narzędzia do zarządzania pakietami (bibliotekami)
- `http://www.learnpython.org/pl/` samouczek Python'a
- `http://www.python.rk.edu.pl/w/p/podstawy/` Kurs Python'a

Uwaga: Stosujemy **Python w wersji 2.7**

2.2 Python - *Hello World*

W tym punkcie należy napisać swój pierwszy skrypt w Pythonie. Jednak nim to zostanie zrobione należy poczytać o wcięciach w Pythonie, np. tu: `http://pl.wikibooks.org/wiki/Zanurkuj_w_Pythonie`

Następnie, używając dowolnego edytora tekstu, w wybranym przez siebie miejscu (jednak najlepiej tam, gdzie umieszczono program *WinPython Command Prompt*) utworzyć plik *skrypt1.py* w którym powinien znaleźć się kod:

```
x=2
if x==2:
        print "To jest dwa"
else:
        print "To nie jest dwa"
```

By wykonać skrypt zapisany w pliku *skrypt.py* należy wykorzystać terminal *WinPython Command Prompt* i wywołać polecenie *python skrypt.py*.

Uwaga: Jeśli skrypt jest w innym folderze niż *WinPython Command Prompt*, należy w konsoli zmienić aktualny folder, korzystając z komendy *cd*, np. *cd c:*

Drugim sposobem na utworzenie pliku *skrypt1.py* jest użycie narzędzia IDLE, które koloruje składnie i pozwala na łatwe wykonywanie skryptów w nim stworzonych. Po jego uruchomieniu należy wybrać *File* → *New File* by stworzyć nowy plik skryptu. Po jego napisaniu i zapisaniu, wybierając z menu edytora tekstu *Run* → *Run Module* (skrót: F5), wykonamy program w konsoli.

2.3 Zadania do wykonania

Korzystając z zasobów strony `http://www.learnpython.org/pl/` napisać:

- Pętle *for* która wypisze liczby od 0 do 10
- Napisać dwie instrukcje warunkowe wykorzystujące trzy dowolne funkcje logiczne
- Napisać funkcje, która dla podanych współczynników równania kwadratowego wyznaczy wyróżnik kwadratowy, tj. tzw. Deltę.

3 Pakiet *pyserial*

Pakiet *pyserial* oferuje API do Python'a które pozwala na łatwy dostęp do portu szeregowego w różnych systemach operacyjnych. Więcej informacji o pakiecie znajduje się na stronie domowej projektu, tj. `http://pyserial.sourceforge.net/`. Co więcej, **pakiet jest domyślnie zainstalowany w pakiecie *WinPython***. Dodatkowo, pod adresem `http://pyserial.sourceforge.net/shortintro.html`, znajduje się krótki wstęp do pakietu pokazujący API oraz podstawowe metody klasy *Serial*, która odpowiada za połączenie z portem COM.

Korzystanie z pakietu należy rozpocząć od importu pakietu do pliku skryptu, tj.:

```
import serial
```

Następnie do zmiennej *port*, przypiszemy nową instancje klasy *Serial*, która zostanie podłączona do portu *COM1*.

```
port=serial.Serial('COM1')
```

Dokładny opis klasy i dokumentacja konstruktora (wraz z domyślnymi parametrami połączenia z portem COM) znajduje się na stronie `http://pyserial.sourceforge.net/pyserial_api.html`

Poprawność połączenia z portem COM można sprawdzić za pomocą metody *isOpen*, np.:

```
print port.isOpen()
```

By wypisać nazwę portu, z którym został połączona instancja klasy *Serial*, zapisana pod zmienną *port*, należy wypisać pole *name* klasy *Serial*, tj.:

```
print port.name
```

Jeśli zaś chcemy wysłać ciąg liter na port szeregowy należy skorzystać z metody *write*

```
port.write('KSP')
```

Funkcją odwrotną do funkcji wysyłania, jest funkcja czytania, która występuje z różnymi parametrami. Najprostsza jej wersja jest bezparametrowa, a jej użycie jest następujące:

```
x=port.read()
```

W tym wypadku metoda *read* odczytuje jeden bajt danych, który może zostać wypisany przez *print*. Należy dodać, że metoda *read* blokuje program, dlatego często za jej wykonanie odpowiada osobny wątek.

Zamknięcie połączenia z portem COM jest realizowane przez metodę *close*

```
port.close()
```

Podsumowując, prosty skrypt używający pakietu *pyserial* prezentuje poniższy listing:

```
import serial

port = serial.Serial('COM7')
if port.isOpen():
    port.write('hello')
    x=port.read()
    print x
    port.close()
```

3.1 Przydatne uwagi

Poniżej znajdują się dwie uwagi dotyczące przesyłania za pomocą pakietu *pyserial* liczb całkowitych w zakresie jednego bajta:

- By wysłać liczbę całkowitą przy pomocy metody *write* należy zastosować przekształcenia na tablice bajtów, np. kod *opakowuje* cyfrę dwa w jednoelementową tablice bajtów. Stąd, wysłanie cyfry dwa z użyciem pakietu *pyserial* sprowadza się do następującej linii kodu

  ```
  port.write(bytearray([2]))
  ```

  Podobnie można wysyłać cyfry w kodzie szesnastkowym

  ```
  port.write(bytearray([0x02]))
  ```

- By odczytać informacje przesłane na port szeregowy jako liczbę całkowitą, tj. jej typ danych to *int*, należy skorzystać z funkcji *ord*, której opis znajduje się tu:

  (https://docs.python.org/2/library/functions.html#ord). Można to zilustrować przykładem

  ```
  x=port.read()
  print ord(x)
  ```

3.2 Zadanie do wykonania

Korzystając z dokumentacji dostarczonej z programem *Serial Device Emulator*, napisać skrypt w języku Python, który przeprowadza procedurę konfiguracji urządzenia którą symuluje program. Do tego celu należy wykorzystać pakiet *pyserial*, instrukcje oraz źródła dostępne w Internecie.

Program *Serial Device Emulator* i jego dokumentacja są dostępne pod adresem `https://github.com/ppenar/PortDeviceEmulator`