Politechnika Rzeszowska

Wydział Budowy Maszyn i Lotnictwa

Katedra Mechaniki i Robotyki Stosowanej

Instrukcja do Laboratorium z KSP:
Modbus w Python

Autor:
Paweł Penar

1 Cel ćwiczenia

- **Cel I:** Używając języka Python i biblioteki *modbus-tk*, napisać skrypt języka Python, który odczytuje i zapisuje różne rodzaje pamięci za pomocą protokołu ModbusTCP
- **Cel II:** Używając języka Python i biblioteki *modbus-tk*, napisać skrypt języka Python, który odczytuje i zapisuje różne rodzaje pamięci zdefiniowane w zewnętrznym plik XML

2 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)
- `https://github.com/ljean/modbus-tk` repozytorium biblioteki *modbus-tk*
- `http://www.tutorialspoint.com/python/python_xml_processing.htm` poradnik dot. parsowania plików XML w Python, EN
- `http://www.python.rk.edu.pl/w/p/parsowanie-xml-w-pythonie/` poradnik dot. parsowania plików XML w Python, PL
- `https://docs.python.org/2/library/xml.dom.html` DOM API (Document Object Model API)

Uwaga: Stosujemy **Python w wersji 2.7**

2.2 Python - obsługa błędów

W tym punkcie należy zapoznać się z obsługą błędów w języku Python, tj. z instrukcjami *try,except*. Do tego celu można skorzystać z `https://docs.python.org/2/tutorial/errors.html` lub innych zasobów internetowych

2.2.1 Zadania do wykonania

Korzystając z zasobów internetowych napisać przykładowy skrypt wykorzystujący obsługę wyjątków.

3 Pakiet *modbus-tk*

Pakiet *modbus-tk* oferuje API do Python'a które pozwala na łatwą implementacje urządzeń *master* oraz *slave* w Modbus-ie RTU i TCP. Dodatkowo, dzięki innym biblioteką języka Python, biblioteka *master-tk* oferuje m.in. możliwość zapisywania logów.

Rozpoczęcie pracy z biblioteką *modbus-tk* rozpoczyna się od zaimportowania odpowiednich bibliotek, co może mieć postać

```
import modbus_tk
import modbus_tk.defines as def
from modbus_tk import modbus_tcp
```

W pakiecie *defines*, do którego odwołujemy się korzystając z aliasu *def*, zawarto definicje stałych związanych z protokołem Modbus (m.in numery funkcji).(link do pliku: `https://github.com/ljean/modbus-tk/blob/master/modbus_tk/defines.py`).

Kolejny pakiet, tj. *modbustcp* implementuje klasę *TcpMaster* opowiadającą za komunikacje w protokole Modbus TCP.

Rozpoczęcie korzystania z biblioteki poprzedzimy inicjalizacją narzędzia do wypisywania logów. Stąd pierwsza linia kodu w sekcji

```
if __name__ == "__main__":
```

to

```
logger = modbus_tk.utils.create_logger("console")
```

Powyższa linia kodu tworzy instancje klasy *Logger*, która implementuje system log-ów i pochodzi z biblioteki *logging* dostarczonej z Pythonem (więcej: `https://docs.python.org/2/library/logging.html`)

Następnie w ramach *try...except* należy stworzyć instancje obiektu klasy *TcpMaster*, tj:

```
try:
        master = modbus_tcp.TcpMaster(host="127.0.0.1", port=510)
    master.set_timeout(5.0)
    logger.info("connected")
except modbus_tk.modbus.ModbusError, exc:
    logger.error("%s- Code=%d", exc, exc.get_exception_code())
```

Parametry występujące w argumentach konstruktora *TcpMaster*, tj. *host* i *port* to odpowiednio adres IP urządzenia slave, z którym chcemy się połączyć oraz port na którym działa.

Odczyt i zapis pamięci z użyciem obiektu klasy *TcpMaster* jest realizowany za pomocą metody *execute*, której przykłady użycia podano w `https://github.com/ljean/modbus-tk/blob/master/examples/tcpmaster_example.py`

4 Parsowanie pliku XML za pomocą DOM API

Dany jest plik XML postaci:

```
<movies>
<movie>
        <title>Titanic</title>
        <format>DVD</format>
        <stars>5</stars>
</movie>
<movie>
        <title>Transformers</title>
        <format>CD</format>
        <stars>8</stars>
</movie>
<movie>
```

```
        <title>Cars</title>
        <format>DVD</format>
        <stars>2</stars>
</movie>
</movies>
```

By korzystać z parsera plików XML potrzeba zaimportować odpowiednie pakiety, np.

```
from xml.dom.minidom import parse
import xml.dom.minidom
```

Następnie tworzymy obiekt reprezentujący parsowany plik i za pomocą metody *parse* z pakietu *xml.dom.minidom*, której parametr to ścieżka do parsowanego pliku, wczytujemy plik. Stąd po wykonaniu linii

```
fileDomTree = xml.dom.minidom.parse("moviesData.xml")
```

zmienna *fileDomTree* reprezentuje dane z pliku *moviesData.xml*

Następnie, korzystając z pola *documentElement* obiektu *fileDomTree*, tworzymy nowy obiekt, który będzie zawierał zawartość węzła *movies*, tj.

```
dataCollection = fileDomTree.documentElement
```

Dalej, korzystając z obiektu *dataCollection*, utworzymy kolekcje węzłów *movie*, które znajdują się w węźle *movies*. Czyli

```
movies = dataCollection.getElementsByTagName("movie")
```

Tak utworzony obiekt kolekcji może być przeglądany w pętli *for*

```
for m in movies:
        title = m.getElementsByTagName('title')[0]
        print "Type: %s" % title.childNodes[0].data
```

gdzie linia

pobiera element o tagu *title*, i węzeł o takim tagu zapisuje w zmiennej *title*. Z uwagi na fakt, że węzeł może posiadać wiele węzłów potomnych, potrzeba użyć dość zaskakującej konstrukcji by uzyskać tytuł filmu. Najpierw pobierany jest pierwszy potomek węzła *title* (pole *childNodes*). Następnie, korzystając z wiedzy o tym, że w potomku węzła *title* znajduje się poszukiwany tekst (a nie następny złożony węzeł), użyte zostaje pole *data*, które przetrzymuje poszukiwany tytuł filmu.

5 Zadanie do wykonania do celu I

1. Przy pomocy prowadzącego zapoznaj się z programem *ModbusSlave* (strona projektu: `http://sourceforge.net/projects/pymodslave/`)

2. Wykorzystując bibliotekę *modbus-tk* i program *ModbusSlave* wykonaj następujące czynności:

   - Odczytaj 10 komórek pamięci *Coils* począwszy od adresu 10
   - Zapisz 5 komórek pamięci *Coils* począwszy od adresu 40
   - Odczytaj 5 komórek pamięci *Holding* począwszy od adresu 40

- Zapisz do dwóch komórek pamięci *Holding* począwszy od adresu 10 dzień i miesiąc swoich urodzin
- Odczytaj komórkę pamięci *Input* o adresie 3 i odczytaną wartość, pomnożoną przez dwa, zapisz pod adresem 15 pamięci *Holding*
- Odczytaj komórkę pamięci *Input* o adresie 22 i odczytaną wartość, podzieloną całkowicie przez 7, zapisz pod adresem 33 pamięci *Holding*

3. Zapisz całość pamięci *Holding* do pliku. Wskazówka: `http://www.python.rk.edu.pl/w/p/operowanie-na-plikach-w-pythonie/`

6 Zadanie do wykonania do celu II

1. Zapoznaj się ze strukturą pliku *commands.xml* który jest zamieszczony w repozytorium
2. Dokonaj odczytu pliku XML korzystając z DOM API, i wyświetl kolejne węzły *command*
3. Na podstawie informacji zawartych w *command.xml* oraz korzystając z modułu *modbus-tk*, ustaw odpowiednie pamięci na odpowiednie wartości i sprawdź poprawność wprowadzonych zmian w *pyModSlaveQt*
4. Na podstawie `http://www.boddie.org.uk/python/XML_intro.html` lub innego zasobu internetowego, zapisz stan wybranej pamięci w pliku XML. Dobierz odpowiednią strukturę węzłów.