

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

Używając języka Python i biblioteki *modbus-tk*, napisać skrypt języka Python, który odczytuje i zapisuje różne rodzaje pamięci za pomocą protokołu ModbusTCP

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)
- <https://github.com/ljean/modbus-tk> repozytorium biblioteki *modbus-tk*

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 *pyserial*

Pakiet *modbus-tk* oferuje API do Python'a które pozwala na łatwą implementację urządzeń *master* oraz *slave* w Modbus-ie RTU i TCP. Dodatkowo, dzięki innym bibliotekom 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. *modbus_tcp* implementuje klasę *TcpMaster* opowiadającą za komunikację 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 instancję 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 instancję 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

3.1 Zadanie do wykonania

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 tej samej pamięci
 - Odczytaj komórkę pamięci *Input* o adresie 22 i odczytaną wartość, podzieloną całkowicie przez 7, zapisz pod adresem 33 tej samej pamięci
3. Zapisz całość pamięci *Holding* do pliku. Wskazówka: <http://www.python.rk.edu.pl/w/p/operowanie-na-plikach-w-pythonie/>