Visoka tehnička škola strukovnih studija Novi Sad

seminarski rad na temu:

Modbus protokol

Mentor:
dr. Velibor Pjevalica

Student:
Ivica K

Zadatak

Napisati Windows aplikaciju u programskom jeziku Python, koja putem modbus protokola komunicira sa kompatibilnim uređajem, i iz njega čita podatke.

Uvod

Modbus je serijski protokol za komunikaciju razvijen 1979. godine, za upotrebu sa PLC kontrolerima firme *Modicon.* Od tada, modbus protokol je postao standard za upotrebu u industrijskim uređajima. Neke od značajnih osobina ovog protokola su:

- besplatan je za implementaciju i otvoren je
- lako se koristi i održava
- funkcioniše na nivou bitova
- u određenoj meri podseća na klijent-server arhitekturu

Kada kažem klijent-server arhitektura, želim da istaknem mogućnost da se na jedan "master" uređaj poveže do 246 "slave" uređaja. Master uređaj može da bude računar, ili SCADA[1] sistem.

Slave uređaji se nazivaju RTU (remote terminal unit – udaljeni terminal uređaj), a u osnovi su to elektronski uređaji koje kontroliše mikroprocesor, koji određene parametre iz fizičkog sveta koji nas okružuje, skupljaju, i putem određenog sistema prosleđuju master uređaju. Prosleđivanje ovih parametara se vrši koristeći neki od protokola, kao što su RS232, RS485, RS422 ili putem Etherneta, odnosno TCP/IP protokola.

Od svojih prvih primena pa do danas, modbus uređaji su koristili, i koriste sledeće protokole:

- Modbus RTU – klasična serijska komunikacija, šalje bitove kroz provodnik. Svaka naredba poslata uređaju, i podaci vraćeni na osnovu nje se nazivaju frejm (frame). Svaki frejm je odvojen periodom "tišine" kada nema komunikacije
- Modbus ASCII – takođe spada u klasičnu serijsku komunikaciju, ali se za komunikaciju koriste ASCII karakteri.
- Modbus TCP/IP – za komunikaciju se koristi TCP/IP protokol razvijen za potrebe povezivanja uređaja na velikim daljinama, osnova današnjeg interneta.
- Modbus UDP – kao što mu ime kaže, koristi se UDP protokol, koji eliminiše TCP *overhead*[2].
- Enron modbus – proširenje modbus protokola sa podrškom za 32bitne cele brojeve i 32bitne brojeve sa pokretnim zarezom.

---

1 SCADA (supervisory control and data acquisition) je tip industrijskog sistema za kontrolu procesa. Najčešće je to kompjuterski kontrolisan sistem koji kontroliše industrijske procese, kroz koji se skupljaju podaci iz fizičkog sveta. Ti podaci mogu biti spoljašnja temperatura, pritisak, protok fluida itd. Koriste se u naftnoj i gasnoj industriji, u rudnicima, na automatskim pumpama za gorivo, elektro industriji itd.

2 Svaki komunikacioni protokol koji zahteva potvrdu sa obe strane, kao TCP protokol sa takozvanim "trostrukim rukovanjem" sadrži overhead – otprilike 40 bajtova rezervisanih za startnu i odredišnu adresu.

Struktura modbus naredbe

Modbus komunikacija se odvija takozvanim porukama. Struktura svake poruke je nezavisna od fizičkog rasporeda i konfiguracije provodnika. To znači da su poruke kroz RS232 i TCP/IP protokol identične.

Svaka modbus poruka ima identičnu strukturu, koju čine četiri dela. Razmenu ovih poruka uvek inicira master uređaj, a u zavisnosti od slave uređaja naznačenog u samoj poruci, slave odgovara.

šematski prikaz modbus poruke

Polje	Opis
1 – adresa slave uređaja	Adresa uređaja za koji je poruka namenjena
2 – funkcija	Kod za funkciju koju uređaj treba da izvrši
3 – podaci	Blok sa podacima nad kojima se primenjuje funkcija
4 – provera grešaka	Blok podataka za proveru greške

tabela koja opisuje šematski prikaz modbus poruke

Modbus ASCII i Modbus RTU

Kod serijske konekcije postoje dva načina komunikacije, odnosno ASCII i RTU. Ovi načini, ili modovi komunikacije određuju kodiranje same modbus poruke. Kod Modbus/ASCII moda, poruke su kodirane prostim ASCII formatom (heksadecimalne vrednosti), čitljivim za čoveka. Modbus/RTU koristi binarno kodiranje što poruku čini nečitljivom za čoveka, ali je sama poruka manja, što omogućava slanje/primanje većeg broja poruka u istom vremenskom intervalu, ako ga poredimo sa ASCII načinom kodiranja.

Kao što sam ranije rekao, svaka modbus poruka je takozvani frejm, odnosno paket podataka. Dva pomenuta načina kodiranja, ASCII i RTU drugačije označavaju frejmove podataka.

Modbus ASCII

Start poruke označava dvotačka **:**, a kraj poruke je označen karakterima **CR/LF.** CR označava pomeranje kursora na početak reda teksta (carriage return) i jedan je od ASCII kontrolnih karaktera. LF (line feed) je instrukcija za prelazak u novi red teksta, koja u kombinaciji sa CR karakterom pomera kursor na početak sledećeg reda. Ukoliko ovaj primer prevedemo na modbus protokol, ova dva karaktera označavaju kraj jedne poruke, i početak čitanja druge.

Modbus RTU

Izostanak komunikacije, takozvana “tišina” ili “idle mod” označavaju početak i kraj svake poruke. Dužina ove tišine je definisana na minimalno 3.5 karaktera za početak i kraj poruke. Međutim, ukoliko uređaj detektuje tišinu u dužini od 1.5 karaktera, on automatski očekuje da pristiže nova poruka. Zbog ovog svojstva, RTU kodirane poruke moraju biti poslate bez prekida.

Modbus adresiranje

Prva informacija kod svake modbus poruke, što vidimo i sa šematskog prikaza, jeste adresa slave uređaja za koji je namenjena. Adresa zauzima jedan bajt, i u ASCII modu je kodirana kao dva heksadecimalna karaktera (svaki po 4 bita). U RTU modu se takođe koristi jedan bajt, sastavljen od 8 bitova, pri čemu svaki od bitova uzima vrednost 0 ili 1.

Adresa slave uređaja može da bude u opsegu 1 – 247, a adresa 0 je rezervisana kao broadcast adresa. Kada master uređaj pošalje poruku, svi slave uređaji je primaju, ali na nju odgovara samo slave čija adresa je navedena u prvom bajtu same poruke. Kada slave odgovara, on koristi istu adresu kao i master (svoju adresu) kako bi master znao sa kog slave uređaja je primio poruku.

Svaki modbus uređaj sadrži skupove memorijskih adresa, koje možemo posmatrati kao predefinisane promenljive u nekom programskom jeziku. Te memorijske adrese se nazivaju registri, i u zavisnosti od funkcije koju obavljaju mogu biti ulazni, izlazni i sadržajni registri. U zavisnosti od količine podataka sa kojom rade, ovi registri se dele na:

- Coils
- Discrete inputs
- Input registers
- Holding registers

Coils, odnosno kalemovi, su read-write promenljive, koje prihvataju vrednosti 0 ili 1. Ako napravimo analogiju sa programskim jezicima, možemo ih posmatrati kao Bulove promenljive, gde je 1 vrednost True, a 0 vrednost False.

Discrete inputs, kako im ime kaže, jesu ulazni read-only registri, koji takođe prihvataju jedan bit kao ulaz; 0 ili 1.

Input registers, odnosno ulazni registri, su takođe read-only registri, ali podaci u njima su dužine 16 bita.

Holding registers, odnosno sadržajni registri omogućavaju master uređaju da u njih upisuje, i iz njih čita podatke dužine 16 bita[3].

3 32 bita kod Enron modbus uređaja

Svaki skup registara (memorijskih adresa) je unutar samog uređaja logički odvojen. Ovo odvajanje možemo zamisliti kao skup nekih tabela, gde svaka tabela pripada određenom skupu registara. Dve tabele služe za diskretne vrednosti (0 ili 1), a dve tabele za brojevne vrednosti.

Takođe, svaka od ovih tabela može da sadrži 9999 vrednosti. Usvojen je standar gde su registri podeljeni na sledeći način:

Broj registra / coila	Adresa podataka	Tip	Ime “tabele”
1 - 9999	0000 - 270E	Read – write	Discrete output coils
10001-19999	0000 - 270E	Read – only	Discrete input contacts
30001 - 39999	0000 - 270E	Read – only	Analog input registers
40001 - 49999	0000 – 270E	Read – write	Analog output holding registers

Modbus naredbe

Svaka modbus poruka, bilo poslata ili primljena, je strukturno identična. Prvi bajt je adresa slave uređaja, a drugi bajt je rezervisan za modbus naredbu koju treba izvršiti. Taj broj, kodiran heksadecimalno, “govori” slave uređaju kojoj tabeli da pristupi, i koju radnju[4] nad podacima u njoj da izvrši. U tabeli ispod je prikaz osnovnih funkcija koje se koriste. Ukupan broj funkcija je veći, a neke od funkcija su proizvođači samih uređaja dodali.

Kod funkcije	Radnja	Ime “tabele”
01 *(01 heksadecimalno)*	Čitanje	Discrete Output Coils
05 *(05 heksadecimalno)*	Upis jednog	Discrete Output Coil
15 *(0F heksadecimalno)*	Upis više	Discrete Output Coils
02 *(02 heksadecimalno)*	Čitanje	Discrete Input Contacts
04 *(04 heksadecimalno)*	Čitanje	Analog Input Registers
03 *(03 heksadecimalno)*	Čitanje	Analog Output Holding Registers
06 *(06 heksadecimalno)*	Upis jednog	Analog Output Holding Registers
16 *(10 heksadecimalno)*	Upis više	Analog Output Holding Registers

4 Čitanje ili pisanje

Nakon svega rečenog, prikazao bih strukturu jedne modbus naredbe kodiranu ASCII kodom, kojom master sa slave uređaja sa adresom 17 čita jedan coil:

11 01 0013 0001 0E84

Kada bih rečima izgovorio ovu naredbu ona bi glasila ovako: “Sa slave uređaja 17 pročitaj samo coil 20”, a evo i objašnjenja nakon heksadecimalnog dekodiranja:

Heksadecimalno	Sistem sa osnovom 10	Značenje
11	17	Adresa slave uređaja
01	1	Funkcija koja se izvršava
0013	19[5]	Adresa registra na kom počinje izvršavanje funkcije
0001	1	Broj registara nad kojima će biti izvršena funkcija, brojeći od startnog registra
0E84		CRC kod za proveru greške

Ukoliko želimo sa slave uređaja sa adresom 48 pročitamo 55 coila, od 20. do 75. naredba bi bila vrlo slična:

30 01 0013 004C 0E84

Razlike u odnosu na naredbu objašnjenu kroz tabelu su:

- broj registara nad kojima će biti izvršena operacija čitanja, što je naznačeno sa *004C heksadecimalno,* odnosno 76[6] u sistemu sa osnovom 10.
- adresa slave uređaja, *30 heksadecimalno*, odnosno 48 u sistemu sa osnovom 10

5 Brojanje registar počinje od 0, tako da ukoliko želimo da pročitamo registar sa adresom 20, u samoj modbus naredbi zahtevamo coil 19.

6 Isto objašnjenje kao fusnota 5.