Korisničko sučelje za programirljivi logički kontroler Modicon M340

Bošnjak, Marko

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ELEKTROTEHNIČKI FAKULTET

Sveučilišni studij

KORISNIČKO SUČELJE ZA PROGRAMIRLJIVI LOGIČKI KONTROLER MODICON M340

Diplomski rad

Marko Bošnjak

Osijek, 2016.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 26.09.2016.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Marko Bošnjak
Studij, smjer:	Diplomski sveučilišni studij Elektrotehnika, smjer Elektroenergetika
Mat. br. studenta, godina upisa:	D-781, 06.10.2014.
OIB studenta:	44381469798
Mentor:	Doc.dr.sc. Emmanuel Karlo Nyarko
Sumentor:	
Predsjednik Povjerenstva:	Izv.prof.dr.sc. Robert Cupec
Član Povjerenstva:	Doc.dr.sc. Damir Filko
Naslov diplomskog rada:	Korisničko sučelje za programirljivi logički kontroler Modicon M340
Znanstvena grana rada:	**Procesno računarstvo (zn. polje računarstvo)**
Zadatak diplomskog rada:	Izraditi program za osobno računalo koji omogućuje akviziciju, prikaz i pohranu mjernih podataka s programirljivog logičkog kontrolera Modicon M340 te pokretanje odgovarajućih akcija s osobnog računala.
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Vrlo dobar (4)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 Postignuti rezultati u odnosu na složenost zadatka: 2 Jasnoća pismenog izražavanja: 2 Razina samostalnosti: 2
Datum prijedloga ocjene mentora:	26.09.2016.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek, 03.10.2016.

Ime i prezime studenta:	Marko Bošnjak
Studij:	Diplomski sveučilišni studij Elektrotehnika, smjer Elektroenergetika
Mat. br. studenta, godina upisa:	D-781, 06.10.2014.
Ephorus podudaranje [%]:	1%

Ovom izjavom izjavljujem da je rad pod nazivom**: Korisničko sučelje za programirljivi logički kontroler Modicon M340**

izrađen pod vodstvom mentora Doc.dr.sc. Emmanuel Karlo Nyarko

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

Razvojem automatizacijske tehnologije dogodio se prijelaz s relejne tehnologije na PLC tehnologiju. PLC (engl. *Programmable Logic Controller*) je univerzalna programibilna upravljačka jedinica, razvijen kao zamjena za složene relejne upravljačke sklopove. Promjena iz jedne na drugu tehnologiju se dogodila iz funkcionalnih razloga. Releji su logički uređaji koji se moraju prespojiti vodičima ako se želi promijeniti upravljačka funkcija, dodati nova komponenta ili se dogodi greška u logici upravljanja. Kod PLC-a nije potrebno prespajanje vodiča, nego se sve potrebne promjene mogu vršiti izmjenom programa koji se prenosi u PLC. Program prema kojem programljivi logički kontroler upravlja procesima u ovom radu se sastavlja u programskom sučelju UNITY PRO. U sklopu ovog diplomskog rada napravljen je program koji omogućuje akviziciju, prikaz i pohranu mjernih podataka s programljivog logičkog kontrolera M340 te slanje podataka s osobnog računala na programljivi logički kontroler. Za izradu programskog sučelja za komunikaciju s PLC uređajem je korišten Microsoft Visual Studio programski paket unutar kojeg je korišten C# i Windows Form Aplication. Kao izvorni kod je korišten ModbusTCP Tester V1.1 od Stephan Strickera koji je preuzet s Code Project internet stranice [1] i ModbusTCP DataSampler v1.0 koji je napravio Mirko Gagro [2]. Funkcionalnost programskog sučelja je testirana uz pomoć Modbus TCP/IP simulatora koji simulira rad PLC uređaja [3]. Tema ovog diplomskog rada obrađena je kroz pet poglavlja. U drugom poglavlju je objašnjena struktura i princip rada PLC ureađaja. U trećem poglavlju su definirane paralalelna i serijska komunikacija. U četvrtom poglavlju je objašnjeno što je Modbus protokol i koji je princip njegovog rada. U petom poglavlju su prikazane funkcije čitanja i slanja podataka s ModbusTCP Tester programskim sučeljem. U šestom poglavlju je objašnjen princip rada programskog sučelja koji je napravljen u sklopu ovog diplomskog rada.

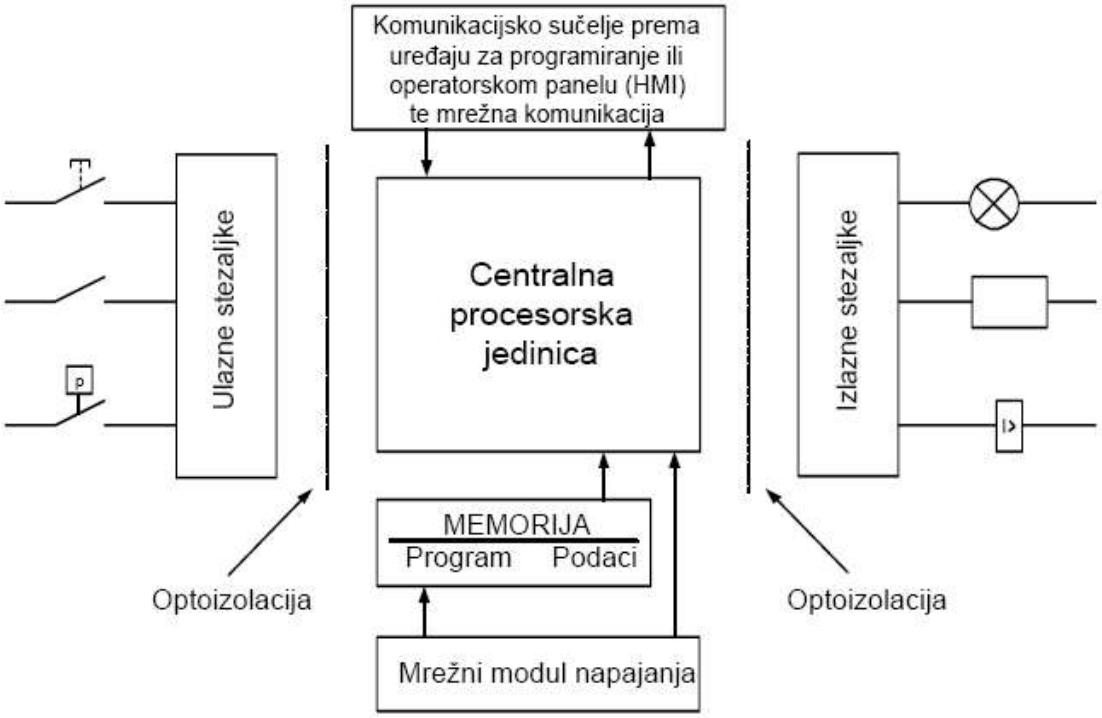
1.1. Zadatak diplomskog rada

Zadatak ovog diplomskog rada je napraviti program koji omogućuje akviziciju, pohranu mjernih podataka s programljivog logičkog kontrolera M340. Program treba imati mogućnost prikaza podataka u obliku grafa i slanje podataka s osobnog računala na programljivi logički kontroler u realnom vremenu.

2. STRUKTURA I PRINCIP RADA PLC-A

Svi PLC uređaji neovisno o njihovoj složenosti i cijeni imaju istu sklopovsku strukturu. Na slici 2.1. su prikazane osnovne cjeline PLC uređaja. Struktura PLC uređaja prema [4] sadrži:

- ulazni dio (digitalni, analogni ulazi)
- izlazni dio (digitalni, analogni izlazi)
- CPU, tj. Centralnu procesorsku jedinicu
- memorijski blok za program i podatke
- module za proširenje
- mrežni dio za napajanje
- komunikacijsko sučelje.



Slika 2.1. *Osnovne cjeline PLC uređaja* [4]

2.1. Ulazni dio

Ulazni dio PLC-a predstavljaju priključne vijčane stezaljke na koje se spajaju "dojavni" signali iz procesa čijim se radom upravlja, te su mjesto od kojeg počinje prilagodba vanjskog signala iz radne okoline, signalu kojeg razumije procesorska jedinica PLC-a. Informacije koje PLC prima na svojim ulaznim stezaljkama mogu biti digitalne (diskretne) i analogne. Digitalna ulazna informacija može biti npr. signal s krajnje sklopke, osjetila, tipkala i sl., dok analogna ulazna informacija može biti npr. naponski signal 0-10 VDC s mjernog pretvornika tlaka, temperature i slično [4].

2.2. Izlazni dio

Izlazni dio PLC-a su priključne vijčane stezaljke na koje se spajaju izvršni uređaji iz procesa kojima PLC šalje digitalne i analogne signale te na taj način upravlja procesom. Na digitalne izlaze iz PLC-a su najčešće spojeni magnetni svici, releji, sklopnici, motorske sklopke, signalne lampe, pneumatski razvodnici i sl., dok na analogne izlaze mogu biti spojeni npr. strujni signal za prikaz neke veličine na pokaznom instrumentu, referenca brzine za frekvencijski pretvarač, PID regulirana veličina itd. [4].

2.3. Centralna procesorska jedinica

Centralna procesorska jedinica s memorijom glavna je jedinica PLC uređaj. Procesorska jedinica čita stanja svih ulaza PLC uređaja (analognih i digitalnih), logički ih obrađuje u skladu s programom izrađenim od strane korisnika, te upravlja izlazima prema rezultatima dobivenim nakon logičke obrade [4].

2.4. Memorijski blok za program i podatke

PLC korisnik prilikom programiranja koristi dva segmenta memorije procesorske jedinice – programske datoteke i datoteke podataka. Programske datoteke koriste korisnički definirane programe, potprograme i datoteku za dojavu i obradu grešaka. Datoteke podataka služe za memoriranje programski ovisnih podataka kao što su U/I status, postavne i trenutne vrijednosti

brojača i vremenskih članova te ostale memorijske konstante i varijable. Podaci programske datoteke i datoteke podataka pohranjuju se u dvije vrste memorije; RAM (engl. *Random access memory*) i EEPROM (engl. *Electricaly erasable programable read only memory*) [4].

2.5. Modul napajanja

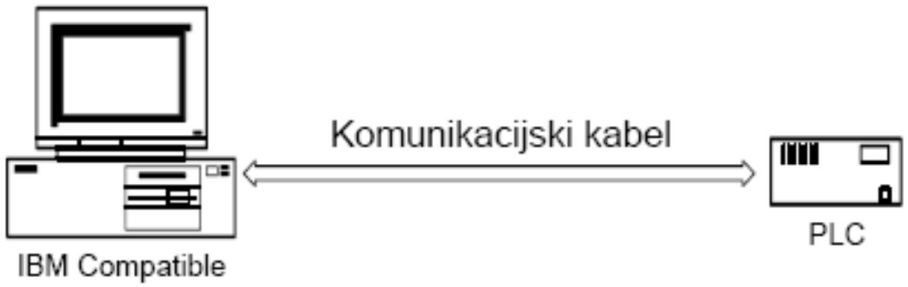
Kao i na svakom računalu, modul napajanja je najrobusniji i najteži sastavni dio. Neosjetljiv je na smetnje koje dolaze iz električne mreže kao i na kraće ispade mrežnog napona (trajanja 10-15 ms). Standardni ulazi napajanja PLC uređaja su: 120/230 VAC i 24 VDC [4].

2.6. Moduli za proširenje

Modul za proširenje je poseban uređaj koji se spaja na PLC i koji na sebi ima dodatne ulazne i/ili izlazne stezaljke. Na taj način se PLC uređaj uvijek može proširiti bez da se nabavlja novi. Najčešće se moduli za proširenje prodaju kao moduli za digitalne ulaze i/ili izlaze te moduli za analogne ulaze i/ili izlaze [4].

2.7. Komunikacijsko sučelje

Komunikacijsko sučelje ima višestruku namjenu. Prva i osnovna je komunikacija s nadređenim PC računalom na kojem se nalazi upravljački program, šalje u PLC te dijagnosticira stanje rada. Na slici 2.2. je prikazana komunikacijska veza između računala i PLC uređaja.

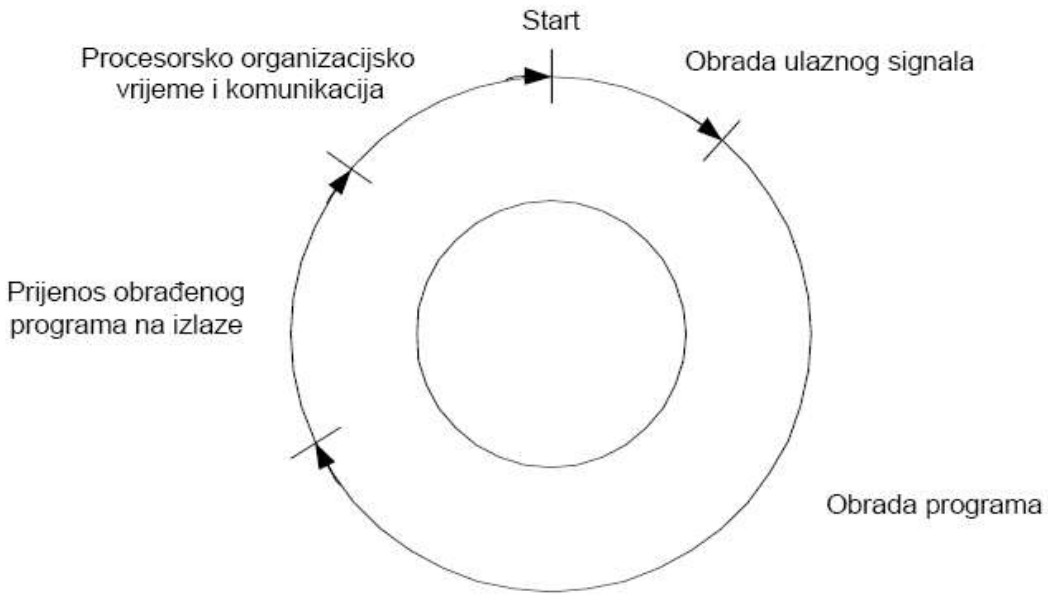


Slika 2.2. *Komunikacija računala s PLC-om* [4]

Ostale mogućnosti su komunikacija s ostalim PLC uređajima i raznim senzorima preko interne mreže (npr. DeviceNet), komunikacija s raznim vrstama operatorskih panela, te komunikacija modemskom vezom. Gotovo svi PLC uređaji imaju ugrađen serijski port za komunikaciju (RS-232 – električki standard), a komunikacija se vrši preko protokola koji ovisi o proizvođaču uređaja [4]. Komunikacijsko sučelje je bitan dio strukture PLC-a za ovaj diplomski rad te će se o njemu još govoriti u daljnjem nastavku.

2.8. Rad uređaja

Princip rada PLC-a zasniva se na registriranju ulaznih signala, obradi, te slanju izlaznih signala. Signali koje primaju moderni PLC-ovi mogu biti digitalni i analogni. Način na koji će PLC obraditi ulazne signale i na osnovu toga dati izlaze zadaje se programom, koji je pohranjen u memoriji PLC-a. Na slici 2.3. je prikazan ciklus rada PLC uređaja.



Slika 2.3. *Ciklus rada PLC-a* [4]

Prema slici 2.3., rad uređaja PLC prema promjeni stanja na njegovim ulazima mora kontinuirano korigirati stanja izlaza, na način određen logikom u korisničkom programu. PLC tu internu obradu podataka vrti ciklički u beskonačnoj petlji. Ciklus obrade podataka prema [4] podijeljen je na nekoliko dijelova:

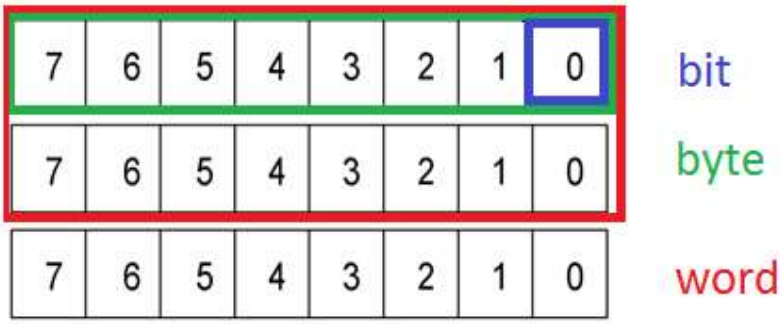
1. Obrada ulaznog stanja – očitanje stanja ulaza te prijenos podataka ulaznog stanja u ulazni memorijski registar procesorske jedinice;
2. Obrada programa – programska obrada ulaznih stanja prema logici korisničkog programa te slanje rezultata u izlazni memorijski registar procesorske jedinice;
3. Prijenos obrađenog programa na izlaze – prijenos obrađenih podataka iz izlaznog memorijskog registra na fizičke izlaze PLC-a;
4. Procesorsko organizacijsko vrijeme i komunikacija – odvijaju se operacije potrebne za funkcioniranje operativnog sustava PLC uređaja te komunikacija s vanjskim jedinicama.
   Vrijeme jednog ciklusa za oko 500 programskih naredbi se kreće oko 1,5 ms.

3. PODJELA KOMUNIKACIJE

Slanje i primanje podataka između PLC uređaja i upravljačke jedinice (PC-a) se može ostvariti na dva osnovna načina:
-paralelnom komunikacijom
-serijskom komunikacijom.

3.1. Paralelna komunikacija

Paralelna komunikacija je kada se svi bitovi od jedne riječi (engl. *Word*) prenose istovremeno preko paralelnih kabela. Tipovi podataka koji se koriste u komunikaciji su prikazani na slici 3.1. Ovaj način komunikacije omogućava da se podaci prenose preko malih udaljenosti velikom brzinom. Može se koristiti za spajanje laboratorijske opreme na sistem.



Slika 3.1. *Tipovi podataka koji se koriste u komunikaciji*

Standardno sučelje koje se najčešće koristi za paralelnu komunikaciju je IEE-488, za kojeg se koristi i naziv *General Purpose Instrument Bus* (GPIB). Paralelna komunikacija se može provoditi između slušatelja (engl. *Listener*), govornika (engl. *Talker*) i kontrolera (engl. *Controllers*).

3.2. Serijska komunikacija

Serijska komunikacija je kada se u jednom trenutku prenosi samo jedan bit. Podaci *word-a* se u svrhu prijenosa rastavljaju u sastavne dijelove bitove, te ponovono sastavlju u *word* kada su zaprimljeni. Serijska komunikacija se koristi za prijenos podataka preko velikih udaljenosti. Može se koristiti za spajanje i komunikaciju između kompjutera i PLC-a.

RS-232 kommunikacija je najčešće korištena komunikacija između PLC-a i eksternog uređaja. Omogućuje komunikaciju "Point to point", što znači da se komunikacija ostvaruje samo između dva uređaja.

Postoje dva tipa RS-232 uređaja, a to su:
-DTA (engl. *Data Terminal Equipment*), primjer takvog uređaja je kompjuter,
-DCE (engl. *Data Communication Equipment*), primjer takvog uređaja je modem.

PLC može biti ili DTA ili DCE uređaj. Standardi kao što su RS-422 i RS-423 su slični RS-232, samo što oni omogućavaju prijenos na više uređaja i na veće udaljenosti. Kao vrsta komunikacije u ovom diplomskom radu se koristi serijska komunikacija.

4. MODBUS PROTOKOL

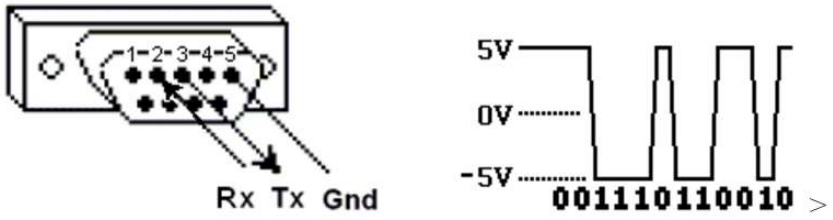
Modbus je serijski komunikacijski protokol koji je razvijen i objavljen od Modicona 1979. godine. Razvijen je za prijenos informacija preko serijskih linija između električnih uređaja. Uređaj koji zahtijeva informacije se naziva *Modbus Master,* dok se uređaji koji snabdjevaju informacije nazivaju *Modbus Slaves*. U standardnoj Modbus mreži postoji jedan *Master* i do 247 *Slave* uređaja. Svaki *Slave* uređaj ima jedinstvenu *Slave* adresu od 1 do 247. *Master* može i pisati podatke na *Slave* uređaj.

Modbus je otvoreni protocol, što znači da ga proizvođači mogu implementirati u uređaje bez plaćanja naknada. Postao je standardni komunikacijski protokol u industriji, te je danas uobičajeni način povezivanja industrijskih elektroničkih uređaja. Koristi se najviše za prijenos signala s instrumenata i kontrolskih uređaja do glavnog kontrolnog sistema ili sistema sakupljanja. Npr. sistem koji mjeri temperaturu i vlažnost i vraća rezultate kompjuteru. Modbus se često koristi za spajanje nadzornog kompjutera s *Remote Terminal Unit* (RTU) u nadzornoj kontroli i sistemima akvizicije podataka (SCADA).

Verzije Modbus protokola koje koriste serijske linije su Modbus RTU i Modbus ASCII, dok se za Ethernet konekciju koristi Modbus TCP.

4.1. Princip rada Modbus protokola

Podaci se šalju preko serijskih linija između uređaja. Najjednostvniji spoj je s jednim serijskim kabelom između serijskih spojeva dva uređaja, *Master-a* i *Slave-a*. Na slici 4.1. je prikazan prijenos podataka serijskom Modbus komunikacijom.



Slika 4.1. *Prijenos podataka serijskom Modbus komunikacijom* [5]

Podaci se šalju kao serija jedinica i nula koje se zovu *bit-ovi*. Svaki *bit* se šalje kao napon. Nule se šalju kao pozitivan napon a jedinice kao negativan napon. *Bit-ovi* se šalju velikim brzinama koja se mjeri u baudima (bitovi po sekundi). Uobičajena brzina slanja *bit-ova* je 9600 bauda.

Kod analize problema korisno je pogledati cijeli red podataka koji se prenosi. Dugi nizovi jedinica u nula su nepogodni za očitavanja, te se zbog toga bitovi spajaju i prikazuju u heksadecimalnom obliku. Svaki skup od četiri *bit-a* je određen jednim od šesnaest znakova od nula do F. Tablica 4.1. prikazuje heksadecimalni sustav.

Tablica 4.1. Heksadecimalni sustav [5]

0000 = 0	0100 = 4	1000 = 8	1100 = C
0001 = 1	0101 = 5	1001 = 9	1101 = D
0010 = 2	0110 = 6	1010 = A	1110 = E
0011 = 3	0111 = 7	1011 = B	1111 = F

Svaki blok od 8 *bit-ova* (1 *byte*) je predstavljen s jednim od 256 parova znakova od 00 do FF.

Kao što su četiri spojena *bit-a* predstavljena s jednim znakom heksadecimalne vrijednosti od 0 do F, tako se svakih osam *bit-ova* može spojiti i predstaviti s jednim od 256 ASCII (engl. *American Standard Code for Information Interchange*) znakova, koji uključuju uobičajene znakove s tipkovnice. Tablica 4.2. prikazuje jedan dio ASCII sustava.

Tablica 4.2. Jedan dio ASCII sustava [5]

decimal (base10)	binary (base2)	Hex (base16)	ASCII (base256)
0	0000 0000	00	null
1	0000 0001	01	"
34	0010 0010	22	#
35	0010 0011	23	$
36	0010 0100	24	%
47	0010 1111	2F	/
48	0011 0000	30	0
49	0011 0001	31	1
56	0011 1000	38	8
57	0011 1001	39	9
58	0011 1010	3A	:
64	0100 0000	40	@
65	0100 0001	41	A
66	0100 0010	42	B
89	0101 1001	59	Y
90	0101 1010	5A	Z
91	0101 1011	5B	[
95	0101 1111	5F	_
96	0110 0000	60	`
97	0110 0001	61	a
122	0111 1010	7A	z
123	0111 1011	7B	{
174	1010 1110	AE	®
255	1111 1111	FF	

Informacije se u *Slave* uređaju spremaju u 4 različite grupe (engl. *Tables*). Dvije grupe spremaju on/off diskretne vrijednosti koje se zovu *coil* i dvije grupe koje spremaju numeričke vrijednosti koje se zovu *register*. *Coil* i *register* vrijednosti imaju grupu samo za čitanje (engl. *Read-only*) i grupu za čitanje i pisanje (engl. *Read-write*).

Svaka grupa ima 9999 vrijednosti. Svaki *coil* ili kontakt je jedan *bit* i dodijeljen je podatkovnoj adresi između 0000 i 270E. Svaki *register* je jedan *word* koji ima vrijednost od šesnaest *bit-ova*, što je zapravo dva *byte-a* (1 *word* = 16 *bits* = 2 *bytes*). Njegov raspon adrese

je također od 0000 do 270E. Tablica 4.3. prikazuje grupe podataka koje se koriste kod *Slave* uređaja.

Tablica 4.3. Grupe podataka koje se koriste kod *Slave* uređaja [5]

Coil/Register Numbers	Data Addresses	Type	Table Name
1-9999	0000 to 270E	Read-Write	Discrete Output Coils
10001-19999	0000 to 270E	Read-Only	Discrete Input Contacts
30001-39999	0000 to 270E	Read-Only	Analog Input Registers
40001-49999	0000 to 270E	Read-Write	Analog Output Holding Registers

Coil i *register* broj (engl. *Coil/Register number*) se može smatrati imenima lokacija s obzirom da se oni ne pojavljuju u samoj poruci. Podatkovna adresa (engl. *Data Addresses*) se koristi u poruci. Razlika između ove dvije vrijednosti je što prva ima *offset* koji je različit za svaku grupu (1, 10001, 30001, 40001).

Svakom *slave* uređaju u mreži se dodjeljuje jedinstvena adresa od 1 do 247. Kada master uređaj zatraži podatke, prvi byte koji pošalje je adresa *slave* uređaja. Na taj način svaki *slave* uređaj zna treba li ignorirati poruku.

Drugi byte koji se pošalje od *master* uređaja je funkcijski kod (engl. *Function code*). Taj broj govori *slave* uređaju kojoj grupi pristupiti i da li da zapisuje ili da očitava iz grupe. U tablici 4.4. su prikazani funkcijski kodovi.

Tablica 4.4. Funkcijski kodovi [5]

Function Code	**Action**	**Table Name**
01 (01 hex)	Read	Discrete Output Coils
05 (05 hex)	Write single	Discrete Output Coil
15 (0F hex)	Write multiple	Discrete Output Coils
02 (02 hex)	Read	Discrete Input Contacts
04 (04 hex)	Read	Analog Input Registers
03 (03 hex)	Read	Analog Output Holding Registers
06 (06 hex)	Write single	Analog Output Holding Register
16 (10 hex)	Write multiple	Analog Output Holding Registers

Na kraju poruke se dodaje dva *byte-a* za provjeru greške. Taj dio poruke se naziva CRC (engl. *Cyclic Redundancy Check*). Svaki *byte* u poruci se koristi za kalkuriranje CRC-a. Zaprimljujući uređaj također izračunava CRC i uspoređuje ga sa CRC-om od uređaja koji šalje poruku. Ako je samo jedan *bit* zaprimljen nepravilno, CRC će biti drugačiji, što će rezultirati greškom (engl. *Error*).

4.2. Primjer Modbus naredbe i odziva

Ako prema tablici 4.4. odaberemo funkciju s brojem 3 i odredimo da želimo saznati *register* vrijednost od adrese 40108 do 40110 iz *slave* uređaja s adresom 17, tada će naredba zahtjeva izgledati kao sljedeće:

11 03 006B 0003 7687.

11 - Adresa *slave* uređaja (11 heksadecimalno = 17 decimalno)
03 - Funkcijski kod 3 (Read analog output holding registers)
006B - Podatkovna adresa od prve *register* vrijednosti koja se zahtijeva (006B heksadecimalno = 107 decimalno + 40001 offset = 40108)
0003 - Ukupni broj zahtjevanih vrijednosti *register* (očitava 3 *register* vrijednosti od 40108 do 40110)
7687 - CRC za provjeru greške

Odziv od *slave* uređaja na temelju zahtjeva bi tada bio:

11 03 06 AE41 5652 4340 49AD

11 - Adresa *slave* uređaja (11 heksadecimalno = 17 decimalno)
03 - Funkcijski kod 3 (Read analog output holding registers)
06 - Broj podataka u *byte-ovima* koji slijede (6 *byte* = 3 *register* × 2 *byte* , zato što se svaka *register* vrijednost sastoji od dva *byte-a*)
AE41 - Sadržaj *register* vrijednosti 40108
5652 - Sadržaj *register* vrijednosti 40109
4340 - Sadržaj *register* vrijednosti 40110
49AD - CRC za provjeru greške

U ovom primjeru vidimo da *register* vrijednost 40108 sadrži AE41. *Register* vrijednost može biti definirana kao jedna od sljedećih 16 bitnih vrijednosti, te bi prema tome poprimila različitu vrijednost.

Ako je 16 bit *unsigned integer*, tada je to broj između 0 i 65535, u našem slučaju je vrijednost AE41 = 44,609.

Ako je 16 bit *signed integer*, tada je to broj između -32768 i 32768 (AE41= -20,927).

Ako je ASCII vrijednost s dva znaka (AE41 = ® A).

Ako je *descrete* on/off vrijednost (AE41 = 1010 1110 0100 0001).

Kada bi *register* vrijednost 40108 kombinirali s *register* vrijednosti 40109, tada bi mogli formirati sljedeće tipove 32 bitnih podataka.

Tip podatka 32 *unsigned integer* koji predstavlja raspon brojeva od 0 do 4,294,967,295 (AE41 5652 = 2,923,517,522).

Tip podatka 32 *signed integer* koji predstavlja broj od -2,147,483,648 do 2,147,483,647 (AE41 5652 = -1,371,449,774).
Tip podatka *floating point* koji dopušta decimalnu vrijednost broja i do 7 decimalnih znamenki (AE41 5652 = -4.395978 E-11).

Modbus specifikacije ne definiraju točno kako se podaci spremaju u *register*. Zbog toga neki proizvođači implementiraju modbus u svoju opremu da prvo šalju i spremaju veće *byte-ove* koji su popraćeni s manjim *byte-ovima* (AE ispred 41). Istovjetno, drugi prvo spremaju i šalju manje podatke (41 ispred AE).

Slično tome kada su *register* vrijednosti kombinirane te prezentiraju 32 *bit-ni* podatkovni tip, neki uređaji spremaju većih 16 *bit-ova* (veći *word*) u prvi *register* i preostali manji *word* u drugi (AE41 ispred 5652), dok drugi rade suprotno (5652 ispred AE41).

Nema veze kojim redosljedom se *byte-ovi* ili *word-ovi* šalju, sve dok zaprimljujući uređaji znaju redoslijed kojim ih trebaju očekivati. Npr. ako se broj 2,923,517,522 šalje kao 32 *bit unsigned integer*, može se rasporediti na sljedeća četiri načina:

AE41 5652 - veći *byte* prvi veći *word* prvi
5652 AE41 - veći *byte* prvi manji *word* prvi

41AE 5256 - manji *byte* prvi veći *word* prvi
5256 41AE - manji *byte* prvi manji *word* prvi

Modbus mapa je lista za slave uređaj koja definira:
- što predstavlja podatak (npr. očitanje pritiska ili temperature)
- gdje se sprema podatak (koja grupa i podatkovna adresa)
- kako se spremaju podaci (tip podataka, raspored byte-ova i word-ova).

Neki uređaji su proizvedeni s fiksnom mapom koja je definirana od strane proizvođača, dok drugi uređaji omogućavaju da operator konfigurira ili programira posebnu mapu koja će odgovarati njihovim potrebama.

4.3. Modbus TCP/IP protokol

TCP (engl. *Transmission Control Protocol*) i IP (engl. *Internet Protocol*) su protokoli koji se koriste zajedno, te predstavljaju prijenosne protokole za internet. Kada se modbus informacija šalje koristeći ove protokole, podaci se prosljeđuju u TCP gdje se dodatne informacije dodaju na dodijeljeni IP. IP nakon toga stavlja podatke u pakete i transmitira ih.

TCP mora uspostaviti vezu prije prijenosa podataka zato što je konekcijom bazirani protokol. U TCP/IP protokolu *master* uređaje možemo zvati i *client,* a *slave* uređaje možemo zvati *server*. *Server* čeka na nadolazeću vezu sa *client* uređajem. Jednom kada se veza uspostavi *server* se odaziva na naredbe od *client* uređaja sve dok veza postoji.

Ovaj protokol je vrlo sličan RTU protokolu s nekoliko razlika. Podaci se prenose preko mreže umjesto preko serijskih kabela. Serveri nemaju SlaveID zato što umjesto toga koriste IP adrese.

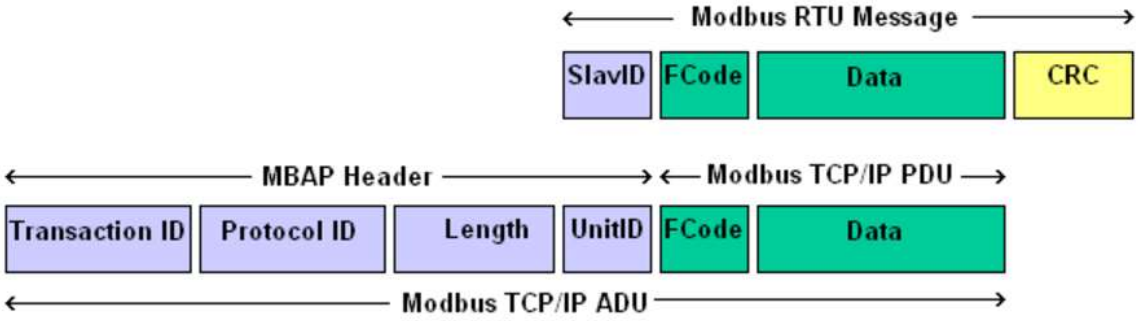
Osim razlike u prijenosu između serijskog kabela i mreže koje su predhodno navedene, postoje bitne razlike u sadržaju poruke. Ako iz RTU poruke maknemo SlaveID s početka i CRC s kraja, ostat će samo PDU (engl. *Protocol Data Unit*). To možemo vidjeti na sljedećem primjeru.

Ako se na primjeru u poglavlju 4.2. makne SlaveID i CRC dobije se PDU:

03 006B 0003

Na početak poruke se dodaje novi dio MBAP (engl. *Modbus Application Header*) koji ima vrijednost od 7 byte-ova. Na slici 4.2. je prikazana razlika između RTU i TCP/IP poruke. MBAP ima sljedeće podatke.

Transaction identifier koji je predstavljen s 2 *byte-a* od strane *client* uređaja kako bi drugačije identificirao svaki zahtjev.
Protocol identifier koji je predstavljen s 2 *byte* od strane *client* uređaja.
Lenght (2 *byte-a*) identificira koliko ima *byte-ova* u poruci koja slijedi.
Unit identifier predstavljen s 1 *byte-om* od strane *client* uređaja i koji se reflektira od strane *server* uređaja za identifikaciju postoje li drugi *slave* uređaji spojeni na *slave* uređaj.



Slika 4.2. *Razlika između RTU i TCP/IP poruke* [5]

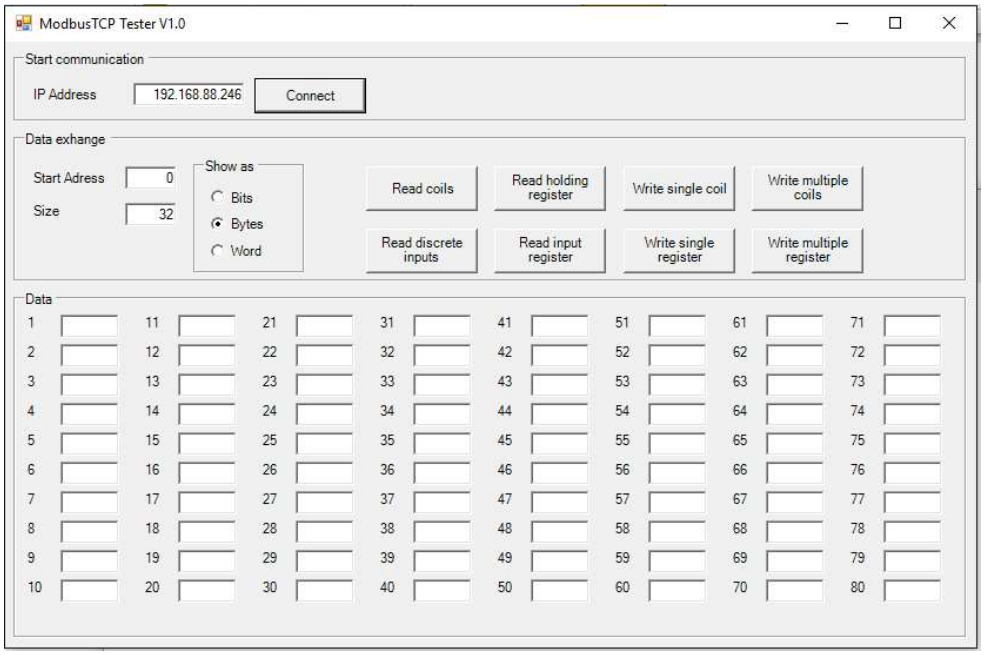
Ako se primjer u poglavlju 4.2. provede u TCP/IP protokolu umjesto u RTU, on bi izgledao kao sljedeći primjer.

0001 0000 0006 11 03 006B 0003

0001 - *Transaction Identifier*
0000 - *Protocol Identifier*
0006 - *Message Length* (6 *byte-ova* koji sljede)
11 - *Unit Identifier* (11 heksadecimalno = 17 decimalno)
03 - Funkcijski kod 3 (*read Analog Output Holding Registers*)
006B - Podatkovna adresa od prve *register* vrijednosti koja se zahtijeva (006B heksadecimalno = 107 decimalno + 40001 *offset* = 40108)
0003 - Ukupni broj zahtjevanih vrijednosti *register* (očitava 3 *register* vrijednosti od 40108 do 40110)

5. MODBUS TCP TESTER

Zadatak ovog diplomskog rada je napraviti korisničko sučelje koje ima mogućnost slanja i primanja informacija s PLC uređaja, spremanja tih informacija u memoriju, te prikazivanje primljenih informacija na grafu u realnom vremenu. Kao izvorni kod je korišten ModbusTCP Tester od Stephan Strickera, prikazan na slici 5.1., preuzet sa Code Project internet stranice koji je opisan u ovom poglavlju [1].

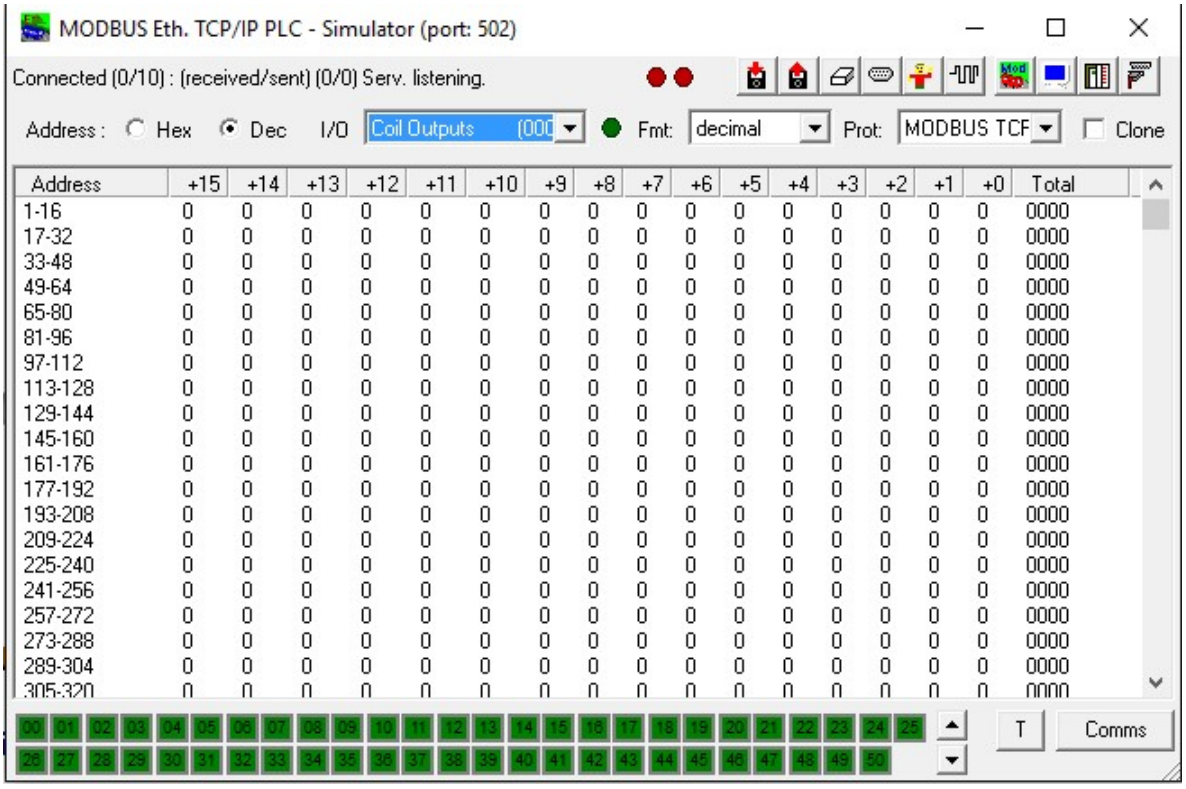


Slika 5.1 *ModbusTCP Tester aplikacija korištena kao izvorni kod*

Ova aplikacija podržava sljedeće funkcije:

- Read coils
- Read discrete inputs
- Write single coil
- Write multiple coils
- Read holding register
- Read input register
- Write single register
- Write multiple register

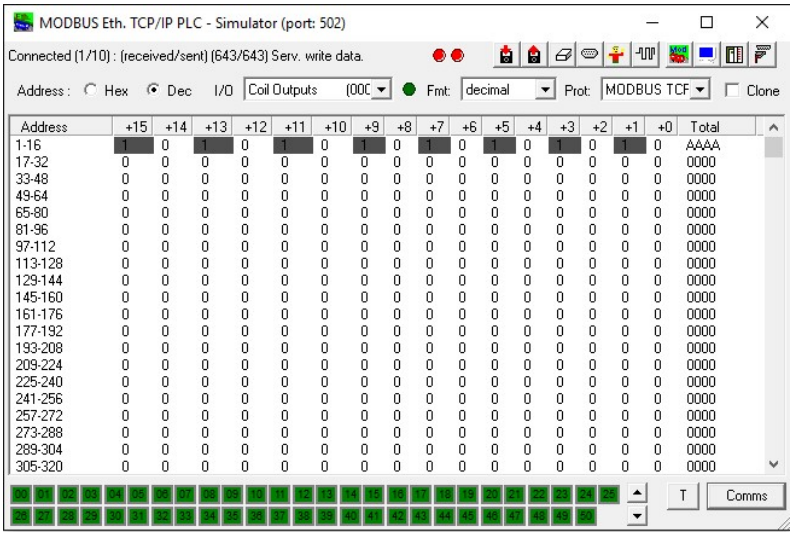
Navedene funkcije su prikazane u tablici 4.4., te su osnova svakog komunikacijskog sučelja između PLC-a (engl. *Server*) i PC-a (engl. *Client*). Za testiranje ove aplikacije korišten je Modbus TCP/IP simulator, prikazan na slici 5.2., koji simulira rad PLC uređaja [3].



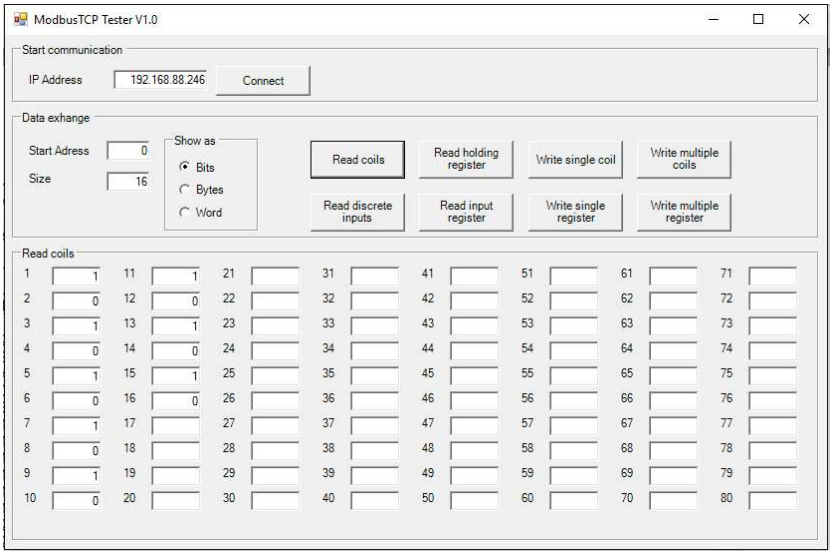
Slika 5.2. *Modbus TCP/IP simulator korišten za simuliranje PLC uređaja*

Kako bi se povezala ModbusTCP Tester aplikacija s Modbus TCP/IP simulatorom, potrebno je unijeti IP adresu računala u ModbusTCP Tester aplikaciju. Ovaj proces zapravo predstavlja spajanje računala s PLC uređajem, gdje bi se umjesto upisivanja adrese računala upisala adresa PLC uređaja. Nakon što se unese IP adresa te se pritisne tipka connect, dolazi se do stanja aplikacije na slici 5.1.

Read coils funkcija izvršava očitavanja *coil* vrijednosti ovisno o podešenoj početnoj adresi očitavanja i količini koliko *coil* vrijednosti (*bit-ova*) se želi čitati. Na Modbus TCP/IP simulatoru je potrebno podesiti I/O na *Coil Outputs*, dok je na ModbusTCP Tester aplikaciji potrebno podesiti tip podatka na bit, te odrediti početnu adresu i dužinu očitavanja. Na slici 5.3. je prikazano kako je u Modbus TCP/IP simulatoru podešeno da svaka druga adresa od prve pa do šesnaeste bude na vrijednosti jedan. Na slici 5.4. je prikazano da je ModbusTCP Tester aplikacija očitala podešene vrijednosti s simulatora.

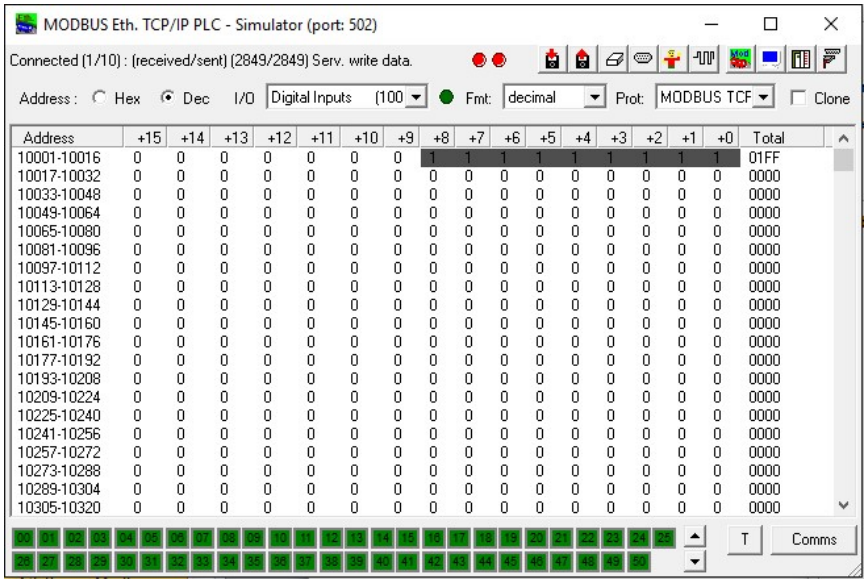


Slika 5.3. *Read coils funkcija u Modbus TCP/IP simulatoru*

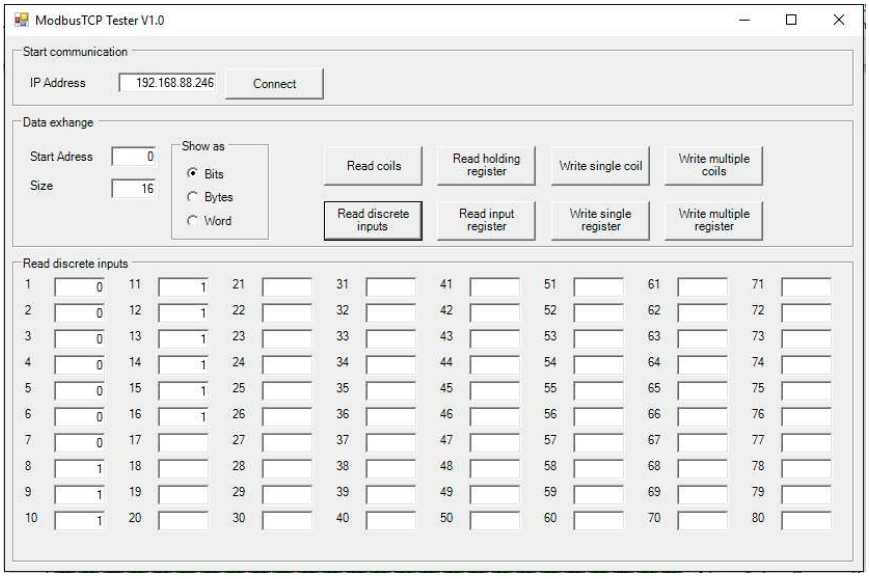


Slika 5.4. *Read coils funkcija u ModbusTCP Tester aplikaciji*

Read descrete inputs čita ulazne *coil* vrijednosti. Potrebno je na simulatoru promijeniti postavke sa *Coil Output* na *Digital Inputs*, dok su na ModbusTCP Tester aplikaciji sve postavke ostale kao u prethodnom primjeru. Na slikama 5.5. i 5.6. je prikazano izvršavanje funkcije *Read descrete inputs.*

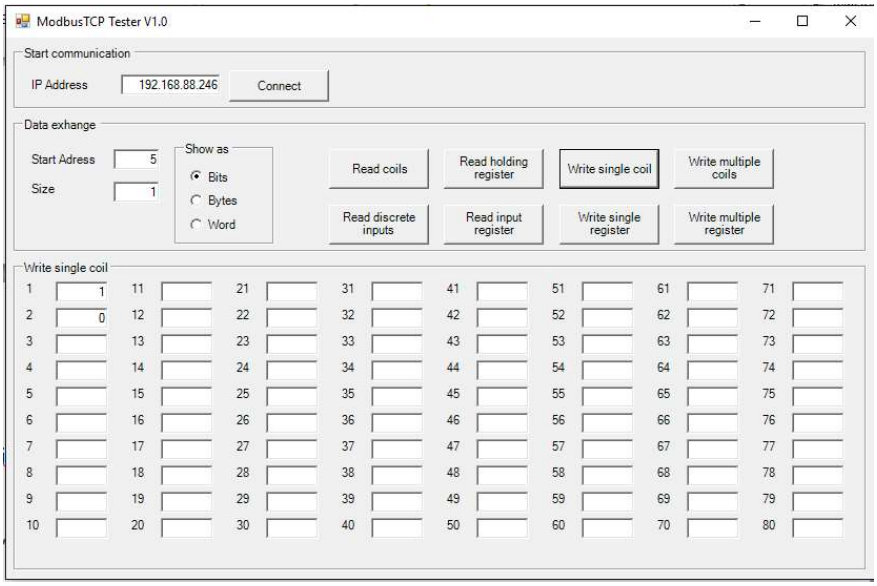


Slika 5.5. *Read descrete inputs funkcija u Modbus TCP/IP simulatoru*

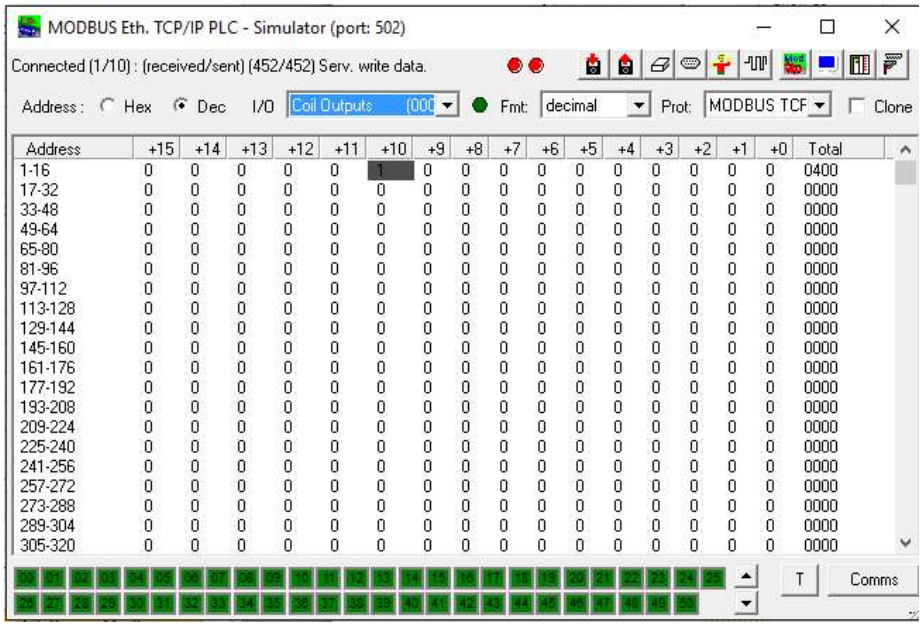


Slika 5.6. *Read descrete inputs funkcija u ModbusTCP Tester aplikaciji*

Write single coil funkcija omogućava slanje jedne *coil* vijednosti (jedan *bit*) s računala na PLC uređaj. Potrebno je postaviti, kao što se vidi na slici 5.7., *size* parametar na jedan zato što se šalje podatak veličine jedan *bit*, dok se sa *start address* parametrom određuje na koju adresu se šalje taj *bit*. Na simulatoru je potrebo podesiti I/O parameter na *coils output*, kao na slici 5.8.

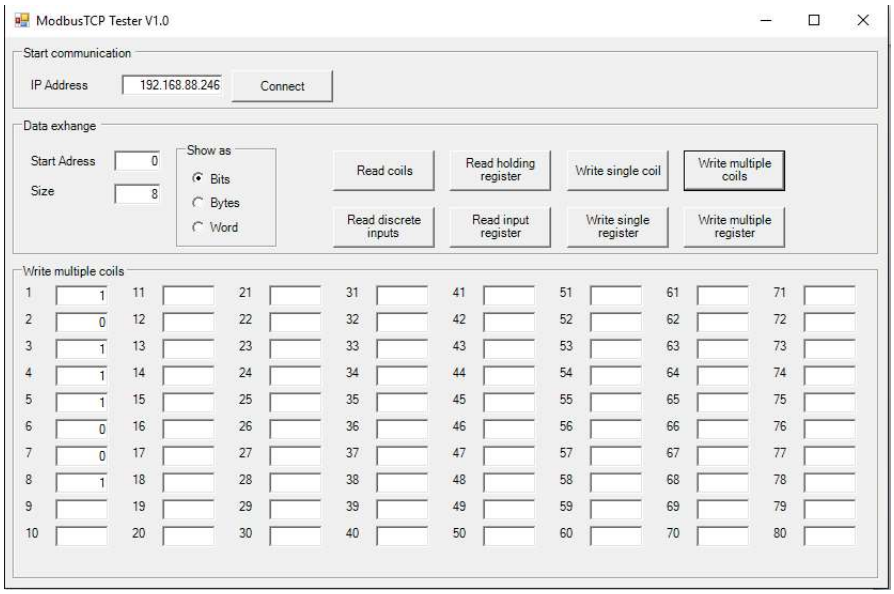


Slika 5.7 *Write single coil funkcija u ModbusTCP Tester aplikaciji*

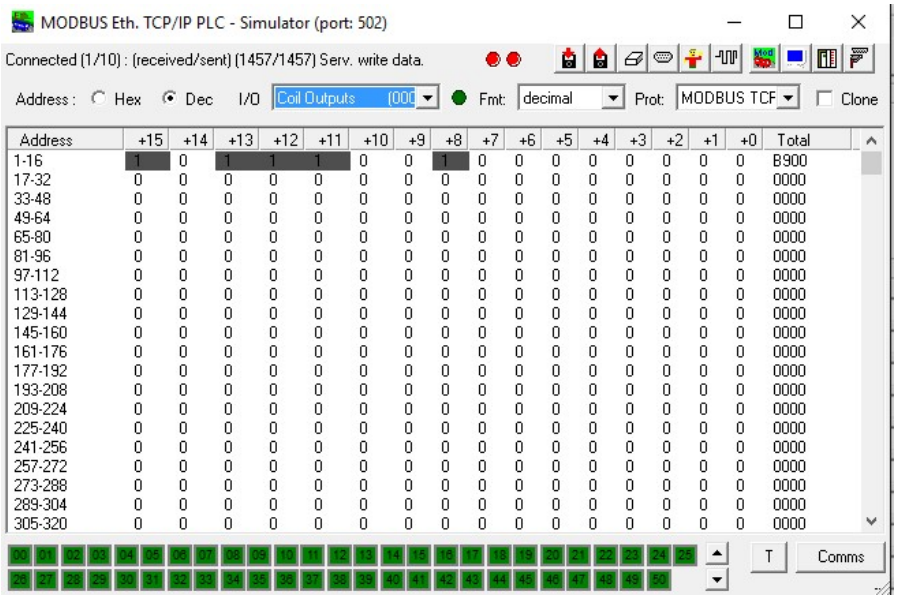


Slika 5.8. *Write single coil funkcija u Modbus TCP/IP simulatoru*

Write multiple coils funkcija šalje istovremeno više *coil* vrijednosti. *Start address* određuje početnu adresu na koju se šalju podaci, *size* određuje koliko bitova se šalje. I/O parameter na simulatoru je potrebno podesiti na *coil output*. Na slikama 5.9. i 5.10. je prikazano izvršavanje funkcije *Write multiple coils.*

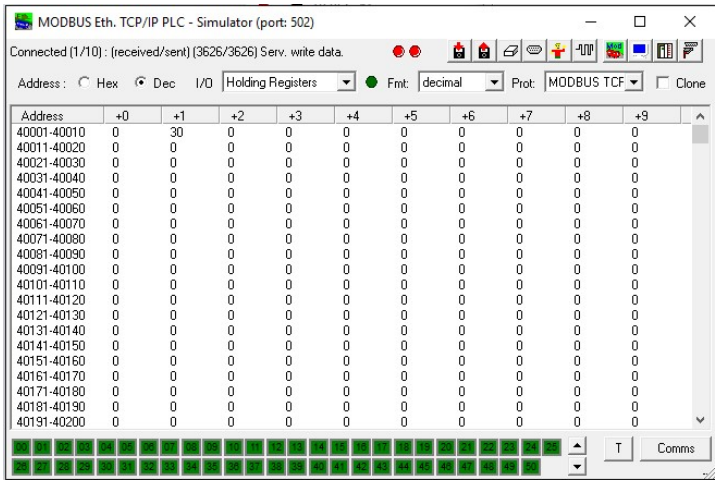


Slika 5.9. *Write multiple coils funkcija u ModbusTCP Tester aplikaciji*

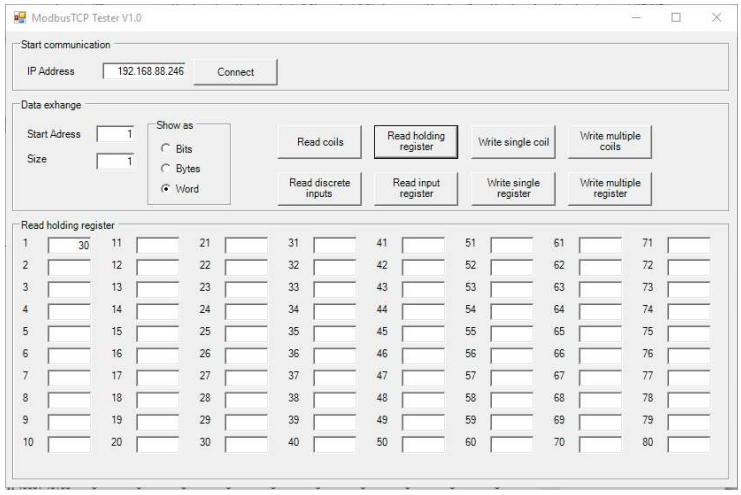


Slika 5.10. *Write multiple coils funkcija u Modbus TCP/IP simulatoru*

Read holding register funkcija očitava jednu *register* vrijednosti s PLC uređaja na određenoj adresi. Na simulatoru je potrebno podesiti I/O parametar na *holding registers.* U slučaju na slici 5.11. je na drugoj adresi *register* vrijednost podešena na trideset. Na slici 5.12. je prikazano da je na ModbusTCP Tester aplikaciji *size* parametar postavljen na jedan zato što se očitava jedna *register* vrijednost, dok je *start adress* podešen na jedan zato što se čita *register* na adresi jedan (brojanje se počinje od nule). Također je postavljen tip podatka koji se čita na *word* zato što je *word* jednak *register* vrijednosti.

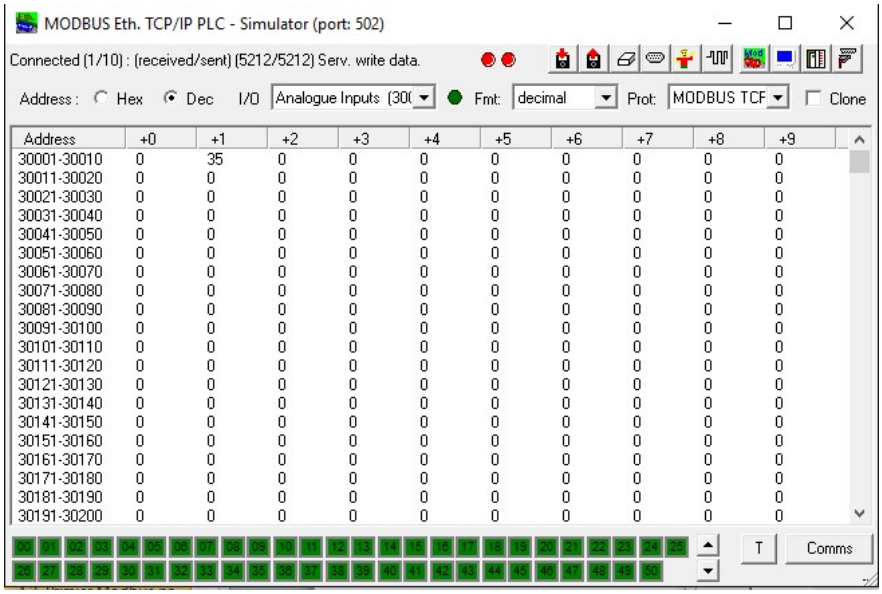


Slika 5.11. *Read holding register funkcija u Modbus TCP/IP simulatoru*

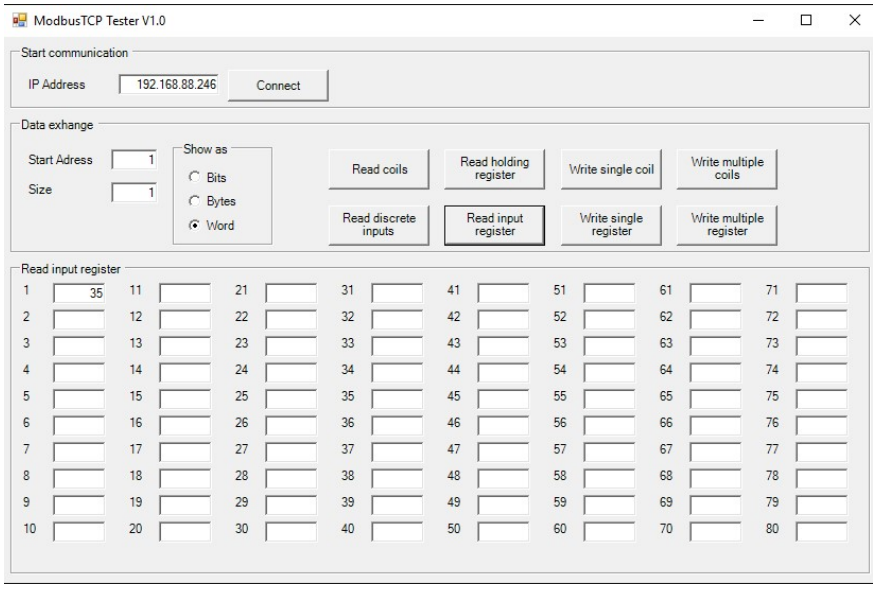


Slika 5.12. *Read holding register funkcija u ModbusTCP Tester aplikaciji*

Read input register funkcija ulazne vrijednosti PLC uređaja. Jedina razlika u postavkama u odnosu na prethodni primjer je što je potrebno u simulatoru promijeniti I/P parametar na *analogue inputs*. Na slikama 5.13. i 5.14. je prikazano izvršavanje funkcije *Read input register.*

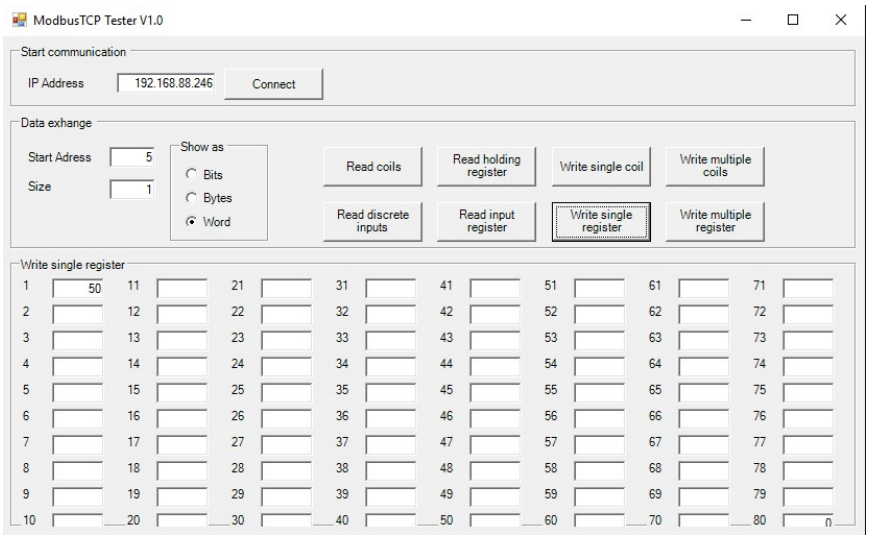


Slika 5.13. *Read input register funkcija u Modbus TCP/IP simulatoru*

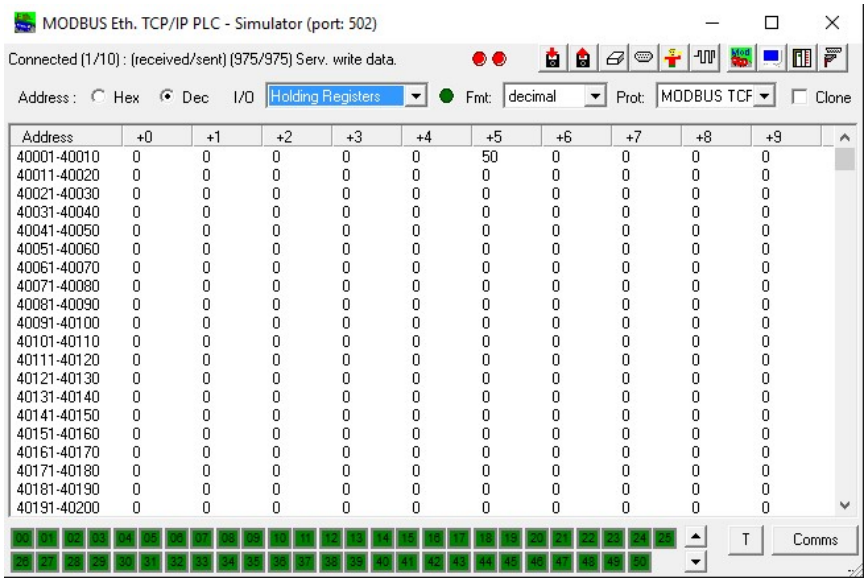


Slika 5.14. *Read input register funkcija u ModbusTCP Tester aplikaciji*

Write single register funkcija šalje jedan *register* tip podatka s računala na PLC uređaj. Na slici 5.15. je prikazano da je *Size* parametar podešen na jedan zato što se šalje samo jedan *register* tip podatka, dok *start adress* određuje na koju adresu se šalje *register*. Na simulatoru je I/O parametar postavljen na *holding register*, što je vidljivo na slici 5.16. gdje je očitana vrijednost s *ModbusTCP Tester* aplikacije.

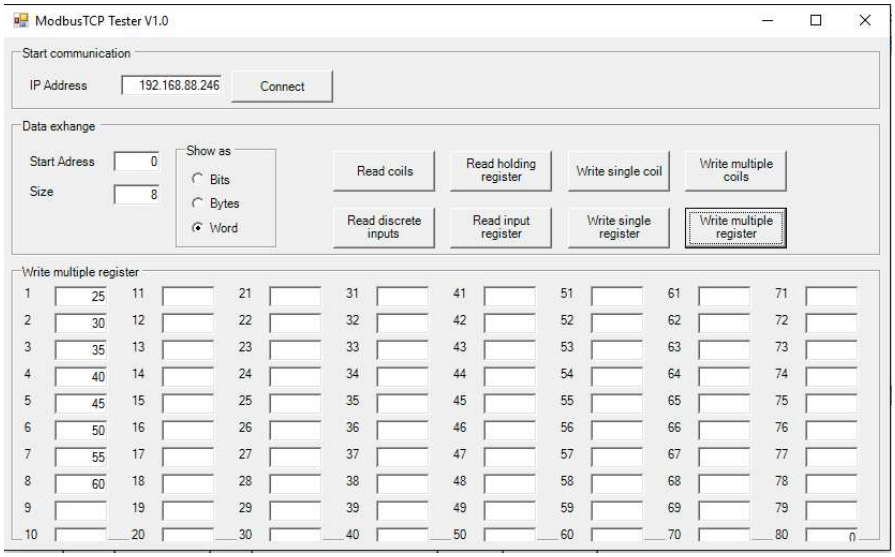


Slika 5.15. *Write single register funkcija u ModbusTCP Tester aplikaciji*

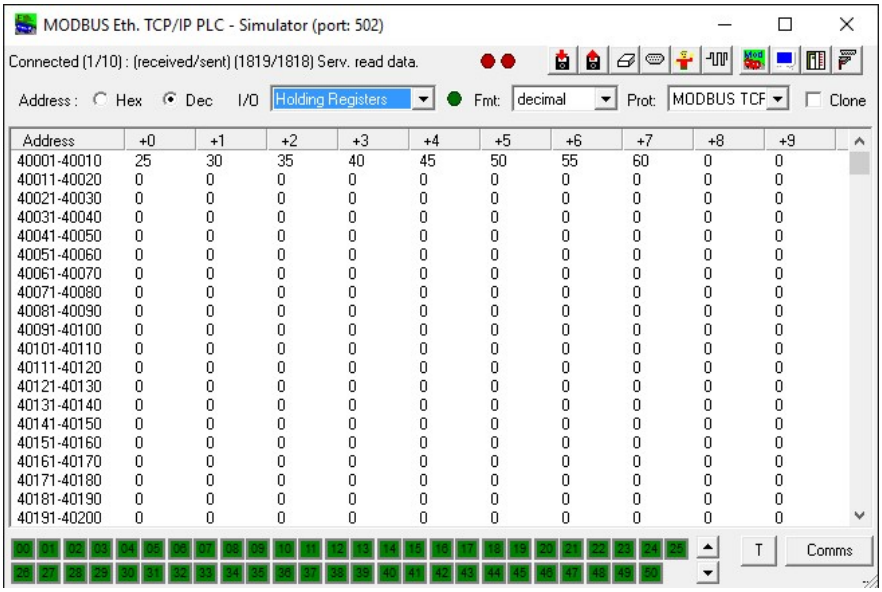


Slika 5.16. *Write single register funkcija u Modbus TCP/IP simulatoru*

Write multiple registers funkcija šalje istovremeno više *register* vrijednosti odjednom. *Start adress* određuje od koje adrese se zapisuju *register* vrijednosti, dok *size* određuje količinu podataka koje se šalju. Parametri na simulatoru su isti kao u predhodnom primjeru. Na slikama 5.17. i 5.18. je prikazano izvršavanje funkcije *Write multiple registers.*



Slika 5.17. *Write multiple registers funkcija u ModbusTCP Tester aplikaciji*



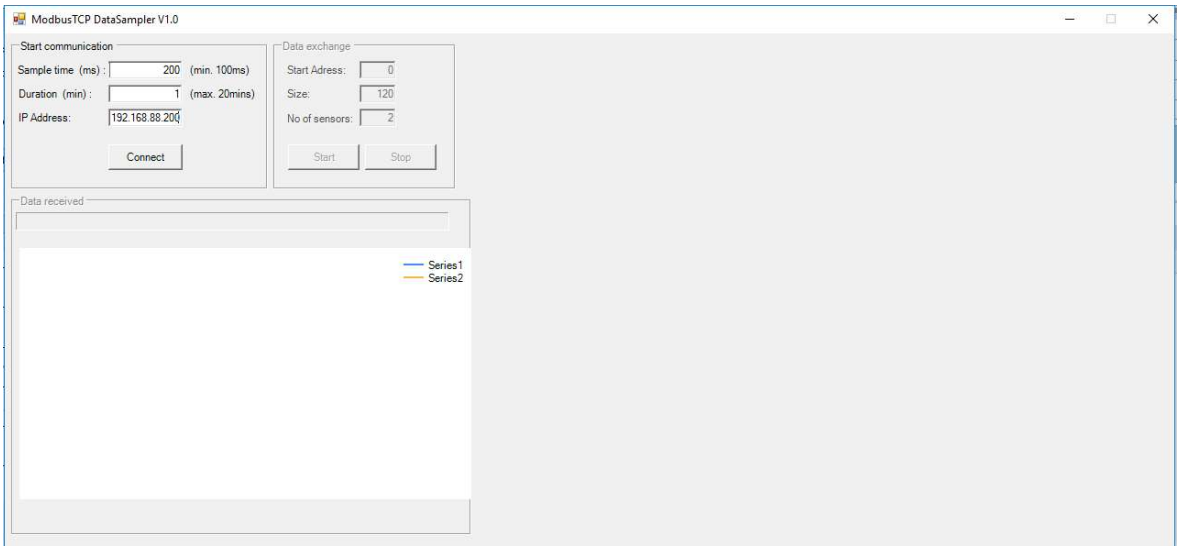
Slika 5.18. *Write multiple registers funkcija u Modbus TCP/IP simulatoru*

Na prethodnim primjerima je vidljivo kako se podaci šalju dvosmjerno između PLC ureađaja i računala. Na slikama koje prikazuju podatke Modbus TCP/IP simulatora je s lijeve strane vidljiv dio raspona adresa u grupi podataka koja se prenosi, koje su predhodno prikazane u tablici 4.3.

Dakle, ova aplikacija ima mogućnost slanja i primanja *coil* i *register* vrijednosti na razne načine ali ne i obradu tih podataka i njihovo spremanje. U daljnjem nastavku rada napravljen je program koji omogućava ispisivanje podataka u realnom vremenu u obliku grafa te spremanje tih istih podataka u txt file.

6. PROGRAMSKO SUČELJE

Program koji je napravljen u sklopu ovog diplomskog rada prima podatke koji su *register* vrijednosti, te ih sortira. Očitane i sortirane podatke prikazuje na grafu, te ih na kraju čitanja sprema u txt file. Prije, za vrijeme i nakon čitanja korisnik ima mogućnost slanja *register* i *coil* vrijednosti prema PLC uređaju. Kao izvorni kod programskog sučelja je korišten ModbusTCP Tester v1.1 autora Stephen Sticker-a [1] i ModbusTCP DataSampler v1.0 autora Mirka Gagre [2]. Ta dva programa su prerađena za potrebe ovog diplomskog rada i napravljen je program koji je prikazan na slici 6.1.



Slika 6.1. *Programsko sučelje*

U polje označeno s IP Adress se unosi TCP/IP adresa PLC uređaja. Adresa sa slike 6.1. je adresa koja je korištena pri testiranju programa.

Podaci se u memoriju PLC-a spremaju u paketima po 3 podatka kao što je prikazano u tablici 6.1. Uzorak predstavlja broj prikupljenog podatka, dok Senzor1 i Senzor2 predstavljaju podatke koji je prikupljaju. Vrijednosti u gornjem redu tablice 6.1. predstavljaju memorijske lokacije koje se čitaju.

U polje *Start Adress* se unosi početna adresa memorijskog bloka koji se čita s PLC-uređaja. U *Size* polje se unosi koliko memorijskih lokacija treba pročitati. Na primjeru sa slike 6.1. *Start Adress* je podešen na 0, dok je *Size* podešen na 120. U tom se slučaju čitaju podaci tipa *word* od memorijske lokacije %MW0000 do %MW0119.

Tablica 6.1. PLC vektori podataka

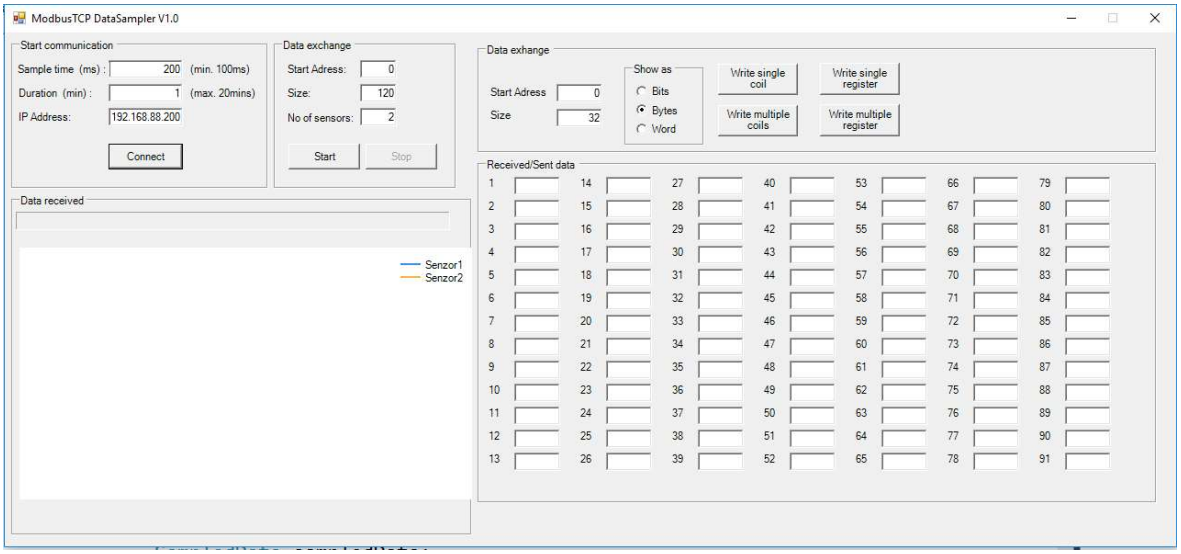
%MW0000	%MW0001	%MW0002	.....	%MW0117	%MW0118	%MW0119
Uzorak	Senzor 1	Senzor 2	.....	Uzorak	Senzor 1	Senzor 2

U vektoru od 120 podataka ima 40 paketa (120/3=40) kako je prikazano u tablici 6.2. Kako novi podaci ne bi brisali stare kad se vektor podataka napuni, program ima mogućnost podešavanja vremena uzorkovanja. U primjeru gdje je PLC podešen da se podaci zapisuju svakih 10 ms potrebno je *Sample time* podesiti na 200 ms kako bi se vektor podataka očitavao svakih 200 ms i kako ne bi došlo do gubljenja podataka pri čitanju vektora (40 paketa x 10 ms = 400 ms). Neki se podaci prenose više puta. Zbog toga je svakom podatku pridružen trenutak uzorkovanja za rekonstrukciju vremenskog slijeda podatka sa strane PC-a. Ako je na PLC-u vrijeme uzorkovanja podešeno na 100 ms, onda je potrebno *Sample time* podesiti na 2000 ms.

Tablica 6.2. PLC paketi podataka

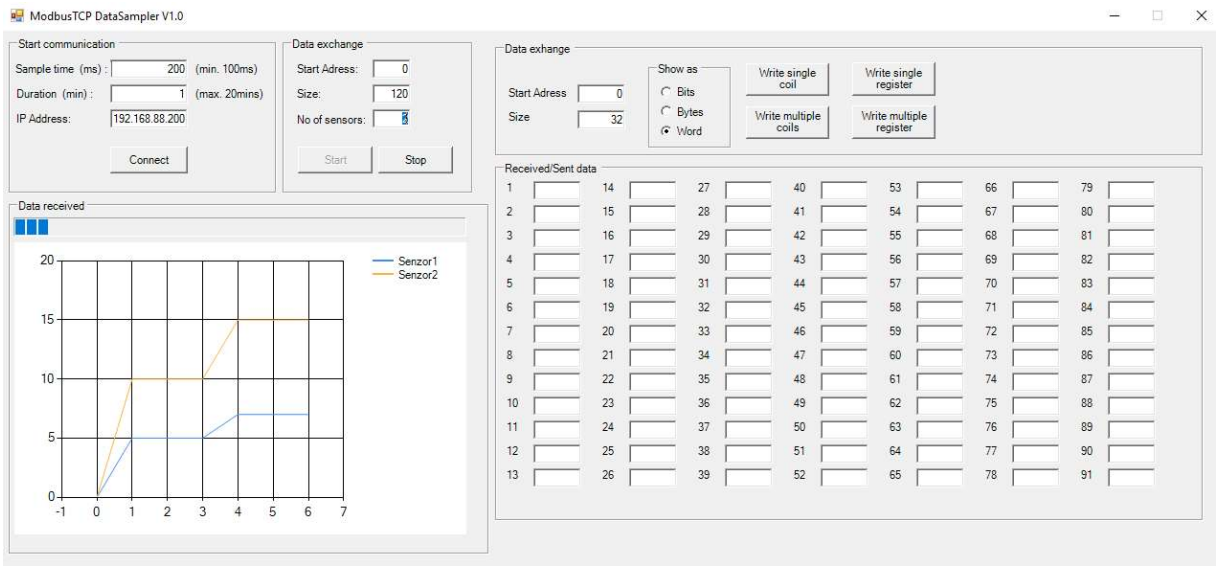
%MW0000	%MW0003	%MW0006	.....	%MW0111	%MW0114	%MW0117
Paket 1	Paket 2	Paket 3	.....	Paket 38	Paket 39	Paket 40

Polje *Duration* određuje vrijeme trajanja prikupljanja podataka, te je na primjeru sa slike 6.1. podešeno na 1 min kako je i preporučeno od autora programa [2].
No of sensors predstavlja broj senzora s kojih se čitaju podaci.
Pritiskom na tipku *Connect* uspostavlja se veza između PLC-a i PC-a, te se dobije mogućnost pokretanja čitanja ili slanja podataka, kao što je prikazano na slici 6.2.

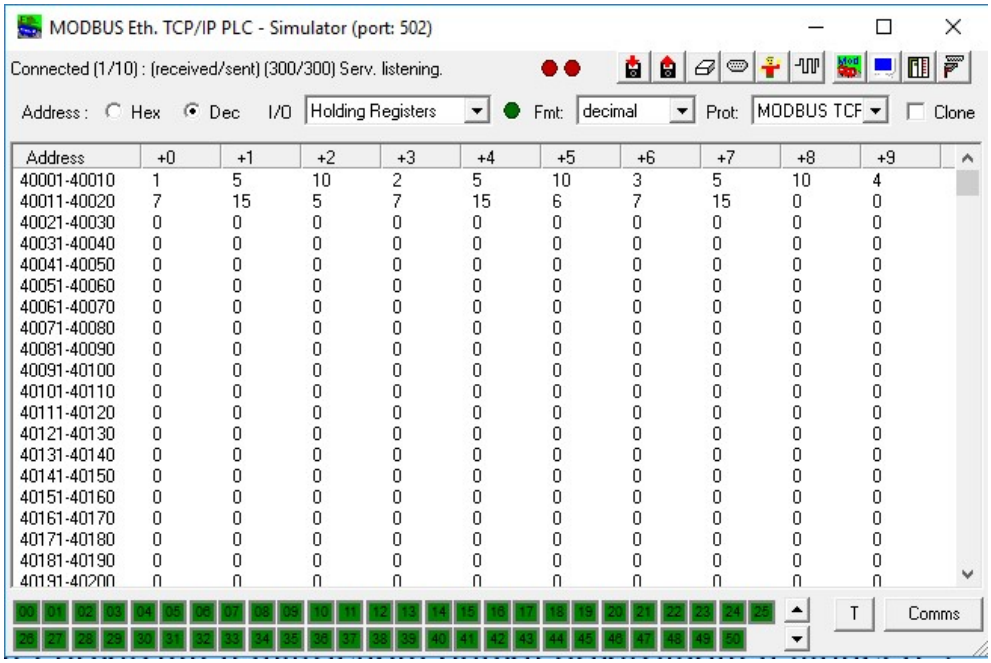


Slika 6.2. *Programsko sučelje spojeno s PLC-om*

Pritiskom na tipku *Start* počinje čitanje i sortiranje podataka. Očitani podaci se automatski prikazuju na grafu na kojem su na apscisi prikazani brojevi uzoraka, a na ordinati očitane *register* vrijednosti. Na slici 6.3. je prikazan primjer grafa gdje program čita podatke s Modbus TCP/IP simulatora koji su prikazani na slici 6.4.

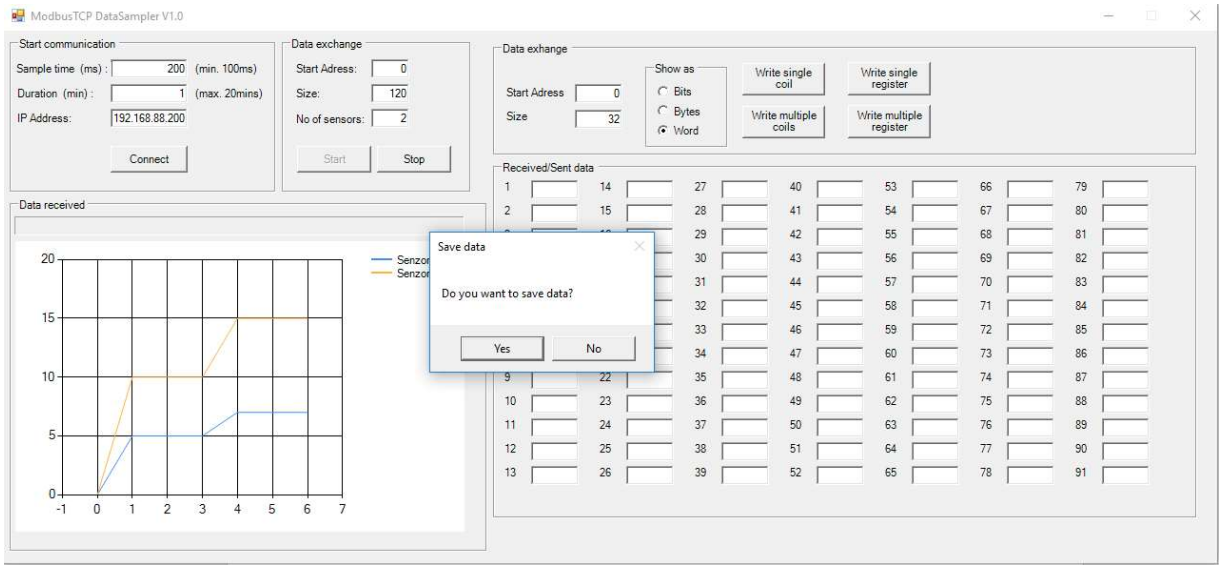


Slika 6.3. *Programsko sučelje s grafom*



Slika 6.4. *Podaci koje program čita s Modbus TCP/IP simulatora*

Bar graf *Data received* prikazuje vrijeme koje je preostalo do kraja čitanja podataka. Nakon završenog čitanja podataka korisnika se pita dali želi spremiti rezultate čitanja podataka kako nebi došlo do nepotrebnog punjenja memorije na računalu i nakupljanja nepotrebnih čitanja podataka kao što je prikazano na slici 6.5. Odabirom tipke Da (engl. *Yes*) podaci se prikazuju u matričnom obliku prikazanom u tablici 6.3. , te se spremaju u txt.file u folderu ModbusSampleCommon\bin\Debug. Odabirom tipke Ne (engl. No) iskočiti će prozor koji daje obavijest da podaci nisu spremljeni. Podaci su u txt.file-u u svakom redu odvojeni razmakom. Nakon odabira jedne od ponuđenih opcija graf se poništi i program je spreman za novo čitanje podataka.



Slika 6.5. *Programsko sučelje nakon završenog čitanja podataka*

Tablica 6.3. Format zapisa podataka u txt.file

Uzorak	Senzor 1	Senzor 2
⋮	⋮	⋮
n	Podatak1(n)	Podatak2(n)
n+1	Podatak1(n+1)	Podatak2(n+1)
n+2	Podatak1(n+2)	Podatak2(n+2)
n+3	Podatak1(n+3)	Podatak2(n+3)
n+4	Podatak1(n+4)	Podatak2(n+4)
⋮	⋮	⋮

7. ZAKLJUČAK

Cilj ovog diplomskog rada je bio napraviti programsko sučelje koje omogućuje akviziciju, prikaz u realnom vremenu i pohranu mjernih podataka s programljivog logičkog kontrolera te slanje podataka sa osobnog računala na programljivi logički kontroler. Riješavanjem toga cilja objašnjena je povijest, struktura i princip rada, te načini i protokoli komunikacije programljivih logičkih kontrolera. U radu je primijenjen C# programski jezik i Windows Form Aplication koji se nalaze u Microsoft Visual Studio programskom paketu. Program korisniku omogućuje da podatke čita, ispisuje na graf u realnom vremenu i sprema u memoriju računala, uz istovremenu mogućnost slanja podataka.

LITERATURA

[1] S. Stricker, Modbus TCP class, http://www.codeproject.com/Tips/16260/Modbus-TCP-class. (23. Siječanj 2016.).

[2] M. Gagro, Sinteza regulatora laboratorijske makete njihala metodom postavljanja polova, Elektrotehnički fakultet u Osijeku.

[3] Modbus PLC simulator, http://www.plcsimulator.org/ (27. 01. 2016.).

[4] G. Maličić, Programljivi logički kontroleri, Tehničko veleučilište u Zagrebu.

[5] Simply Modbus, http://www.simplymodbus.ca/index.html (14. 02. 2016.).

SAŽETAK

Ključne riječi: Programljivi logički kontroler (PLC), Modbus protokol, Programsko sučelje

U skopu ovog diplomskog rada je napravljeno programsko sučelje koje omogućuje akviziciju, prikaz i pohranu mjernih podataka s programljivog logičkog kontrolera te slanje podataka sa osobnog računala na programljivi logički kontroler. Kao uvod u rješavanje problema objašnjena je struktura i princip rada PLC uređaja, te su definirane paralelna i serijska komunikacija. Također je dan opis Modbus protokola i princip njegovog rada. Na kraju je dan opis programskog sučelja razvijen pomoću C# programskog jezika i Windows Form Aplication.

Key words: Programmable logic controller (PLC), Modbus protocol, Software Program

In this thesis, a user interface for communication between a PC and PLC is developed. The interface enables the acquisition, display and storage of measurement data from a PLC as well sending data from the PC to PLC. An overview of the structure and working principle of PLC devices, including parallel and serial communication is provided. A description of the Modbus protocol and the principle of its operation is given as well. Finally, details of the user interface developed using C # programming language and Windows Form Application is provided.

ŽIVOTOPIS

Marko Bošnjak, rođen 09.08.1991. u Zagrebu. Pohađao je Osnovnu školu Ivana Gorana Kovačića u Vinkovcima, nakon koje upisuje Tehničku školu Ruđera Boškovića u Vinkovcima. Nakon završene srednje škole 2010. godine upisuje stručni studij na Elektrotehničkom fakultetu u Osijeku, smjer elektroenergetika. Nakon stručnog studija 2013. godine upisuje razlikovnu godinu na istom fakultetu. Nakon razlikovne godine 2014. upisuje diplomski studij, smjer elektroenergetika.

Student:

Marko Bošnjak

_____