УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

Душан Кнежевић

Реализација софтвера за управљање мрежним уређајима на бази СНМП протокола

МАСТЕР РАД

Нови Сад, 2016

	УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА** 21000 НОВИ САД, Трг Доситеја Обрадовића 6	Број:
	ЗАДАТАК ЗА МАСТЕР РАД	Датум:

(Податке уноси предметни наставник - ментор)

СТУДИЈСКИ ПРОГРАМ:	Рачунарство и аутоматика
РУКОВОДИЛАЦ СТУДИЈСКОГ ПРОГРАМА:	Проф. др Никола Јорговановић

Студент:	Душан Кнежевић	Број индекса:	E2 66/2014
Област:	Архитектуре и алгоритми ДСП-а 1		
Ментор:	др. Јелена Ковачевић		

НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА ИЗДАЈЕ СЕ ЗАДАТАК ЗА МАСТЕР РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА:

- проблем – тема рада;
- начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна;

НАСЛОВ МАСТЕР РАДА:

Реализација софтвера за управљање мрежним уређајима на бази СНМП протокола

ТЕКСТ ЗАДАТКА:

У оквиру задатка потребно је реализовати програмску подршку за управљање мрежним уређајима путем СНМП протокола у системима за управљање и надзор (аутоматизацију) у индустријским постројењима. Потребно је реализовати СНМП Агента који се састоји из два дела СНМП Агент и СНМП трап. Потребно је обезбедити интеграцију и високу поузданост решења, праћење V развојног модела. Валидирати решење унит тестовима, интеграционим тестовима и системским тестовима да би се испоштовао целокупан V модел развоја система.

Руководилац студијског програма:	Ментор рада:

УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:		
Идентификациони број, **ИБР**:		
Тип документације, **ТД**:		Монографска документација
Тип записа, **ТЗ**:		Текстуални штампани материјал
Врста рада, **ВР**:		Дипломски – мастер рад
Аутор, **АУ**:		**Душан Кнежевић**
Ментор, **МН**:		**др Јелена Ковачевић**
Наслов рада, **НР**:		**Реализација софтвера за управљање мрежним уређајима на бази СНМП протокола**
Језик публикације, **ЈП**:		Српски / латиница
Језик извода, **ЈИ**:		Српски
Земља публиковања, **ЗП**:		Република Србија
Уже географско подручје, **УГП**:		Војводина
Година, **ГО**:		**2016**
Издавач, **ИЗ**:		Ауторски репринт
Место и адреса, **МА**:		Нови Сад; трг Доситеја Обрадовића 6
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)		**7/49/15/6/22/0/0**
Научна област, **НО**:		Електротехника и рачунарство
Научна дисциплина, **НД**:		Рачунарска техника
Предметна одредница/Кључне речи, **ПО**:		**SNMP protokol, Texas Instruments TMS570, ethernet platforma, V модел развоја**
УДК		
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад
Важна напомена, **ВН**:		
Извод, **ИЗ**:		**У раду је представљена реализација програмске подршке за управљање мрежним уређајима путем СНМП протокола у системима за управљање и надзор (аутоматизацију) у индустријским постројењима. Реализован је СНМП Агент који служи за комуникацију и управљање уређајима и СНМП Трап који обавештава управљачку апликацију да је у његовој комуникацијској околини наступио неки нежељени догађај. Развој подразумева дефинисање формалних захтева, архитектуре, дизајна, имплементације и верификације према предложеном V моделу. Програмско решење је успешно интегрисано у постојећи систем, а поузданост система је верификована тестирањем низом унит, интеграционих и системских тестова.**
А,Датум прихватања теме, **ДП**:		
Датум одбране, **ДО**:		**23.02.2016.**
Чланови комисије, **КО**:	Председник:	**др Илија Башичевић**
	Члан:	**др Растислав Струхарик** — Потпис ментора
	Члан, ментор:	**др Јелена Ковачевић**

UNIVERSITY OF NOVI SAD ● **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, **ANO**:			
Identification number, **INO**:			
Document type, **DT**:		Monographic publication	
Type of record, **TR**:		Textual printed material	
Contents code, **CC**:		Master Thesis	
Author, **AU**:		**Dušan Knežević**	
Mentor, **MN**:		**PhD Jelena Kovačević**	
Title, **TI**:		**Implementation of software for managing network devices based on SNMP protocol**	
Language of text, **LT**:		Serbian	
Language of abstract, **LA**:		Serbian	
Country of publication, **CP**:		Republic of Serbia	
Locality of publication, **LP**:		Vojvodina	
Publication year, **PY**:		**2016**	
Publisher, **PB**:		Author's reprint	
Publication place, **PP**:		Novi Sad, Dositeja Obradovica sq. 6	
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		**7/49/15/6/22/0/0**	
Scientific field, **SF**:		Electrical Engineering	
Scientific discipline, **SD**:		Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, **S/KW**:		**SNMP protocol, Texas Instruments TMS570, ethernet platform, V model software development**	
UC			
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, **N**:			
Abstract, **AB**:		**This master work gives a description of softwer for managing network devices through SNMP protocol in systems for management and monitoring (automation) in industrial plants. The SNMP Agent is implemented for communication and device management and SNMP Trap that notifies the control application that has happend an error in communications environment. Development of this software requires definition of formal requirements, design architecture, implementation and verification according to V-model development process. Software is successfully integrated into existing system and reliability has been verified using unit, integration and system tests.**	
Accepted by the Scientific Board on, **ASB**:			
Defended on, **DE**:		**23.02.2016.**	
Defended Board, **DB**:	President:	**Ilija Bašičević, PhD**	
	Member:	**Rastislav Struharik, PhD**	Menthor's sign
	Member, Mentor:	**Jelena Kovačević, PhD**	

SADRŽAJ

SPISAK SLIKA

SPISAK TABELA

SKRAĆENICE

SNMP - *Simple Network Management Protocol,* Protokol za jedinstveno upravljanje mrežom

UDP - *User Datagram Protocol,* Korisnički datagram protokol

TCP - *Transmission Control Protocol*, Transmisioni kontrolni protokol

MIB - *Management Information Base,* Baza upravljačkih informacija

OID - *Object Identifier*, Identifikator objekta

IP - *Internet Protocol,* Internet protokol

RFC - *Request for Comments,* Zahtev za komentar

NMS - *Network Management System,* Sistem za upravljanje mrežom

PDU -*Protocol Data Units,* Jedinica podataka protokola

SPI - *Serial Peripheral Interface*, Serijska periferna sprega

ICMP - *Internet Control Message Protocol,* Protokol za internet kontrolu poruka

UART - *Universal Asynchronius Receiver/Transmiter* , Univerzalni asinhroni prijemnik/odašiljač

SPI - *Serial Peripheral Interface*, Serijska periferna sprega

1. Uvod

U ovom radu je prikazana realizacija softvera za upravljanje mrežnim uređajima putem protokola za jedinstveno upravljanje mrežom (SNMP protokol) u ugrađenim (*eng. embedded*) sistemima koji se koriste u industijskim postojenjima, u ovom slučaju taj sistem je Ethernet. Hardver za koji je softver realizovan je TMS570 platforma. Takođe i testranje je izvršeno na TMS570 platformi. Razvoj je podrazumevao definisanje formalnih zahteva, arhitekture, dizajna, implementacije i verifikacije prema V modelu.

Upravljanje i nadzor mrežne okoline izazovan je zadatak, najpre zbog kompleksnosti i raznolikosti današnjih mrežnih arhitektura. Standardizacija strategije i tehnika upravljanja je neophodna kako bi se omogućilo uspešno nadziranje današnjih mreža. Većina arhitektura za upravljanje mrežom zasnovana je na istim principima. Arhitektura tipičnog sistema za upravljanje mrežom (eng. Network Managment System) sastoji se od entiteta za upravljanje, entiteta kojima se upravlja i skupa veza između njih. [1] Programski agenti koji se nalaze u sistemima koji se nadziru predstavljaju vezu između fizičkog sistema i parametara koje je potrebno nadzirati te samog sistema za nadzor. Agenti prilikom prvog pokretanja stvaraju bazu podataka o parametrima koji se nadziru u sistemu, skladište ih u specijalizovanom obliku i po potrebi šalju potrebne podatke sistemu za nadzor putem protokola za upravljanje mrežom. Jedan od najrasprostranjenijih protokola za upravljanje mrežom je SNMP. SNMP je mrežni upravljački protokol dizajniran tako da olakša upravljanje i nadzor kompletne mreže, te svih njenih entiteta. Funkcionalnost i implementacija SNMP protokola je relativno jednostavna, no ipak dovoljno fleksibilna da pruži mogućnost kvalitetnog upravljanja velikim brojem različitih tipova uređaja u današnjoj distribuiranoj mrežnoj okolini. Kod jednostavnih računarskih mreža sa malim brojem članova uglavnom nije teško utvrditi da se javio problem i šta je uzrok problema. Međutim kod kompleksnih računarskih mreža koje čini veliki broj članova često je neophodno predvideti

moguće probleme, utvrditi da je do problema na mreži došlo i utvrditi njegovu lokaciju i uzrok. Uloga SNMP protokola jeste da administratorima obezbedi informacije vezane za rad računarske mreže, a koje je moguće iskoristiti za sprečavanje i rešavanje problema u njenom radu.
Rad se sastoji od sedam poglavlja.

Prvo poglavlje sadrži opis rada i osnove upravljanja u mrežnim sistemima.

U drugom sadrži opis TMS570 platforme. Opisan je SNMP protokol, osnovne funkcionalnosti kao i njegova arhitektura. Dat je prikaz V modela razvoja softvera i njegov kratak opis.

Treće poglavlje daje kratak opis ciljane platforme (celog sistema), pregled njegovih modula i ugradnju SNMP Agenta u postojeći embedded sistem. Opisana su dva posebna modula SNMP modul i SNMP Trap modul i način na koji funkcionišu.

Četvrto poglavlje sadrži detaljan opis programskog rešenja, odnosno realizaciju funkcija SNMP Agenta (SNMP modul i SNMP Trap modul)

Peto poglavlje je testiranje. U njemu su navedeni testovi kojima je modul testiran i kratak opis svake vrste testa i data je tabela sa izvršenim testovima.

Šesto poglavlje sadrži kratak pregled onoga što je urađeno u ovom radu.

U sedmom poglavlju je dat spisak korišćene literature tokom izrade ovog rada.

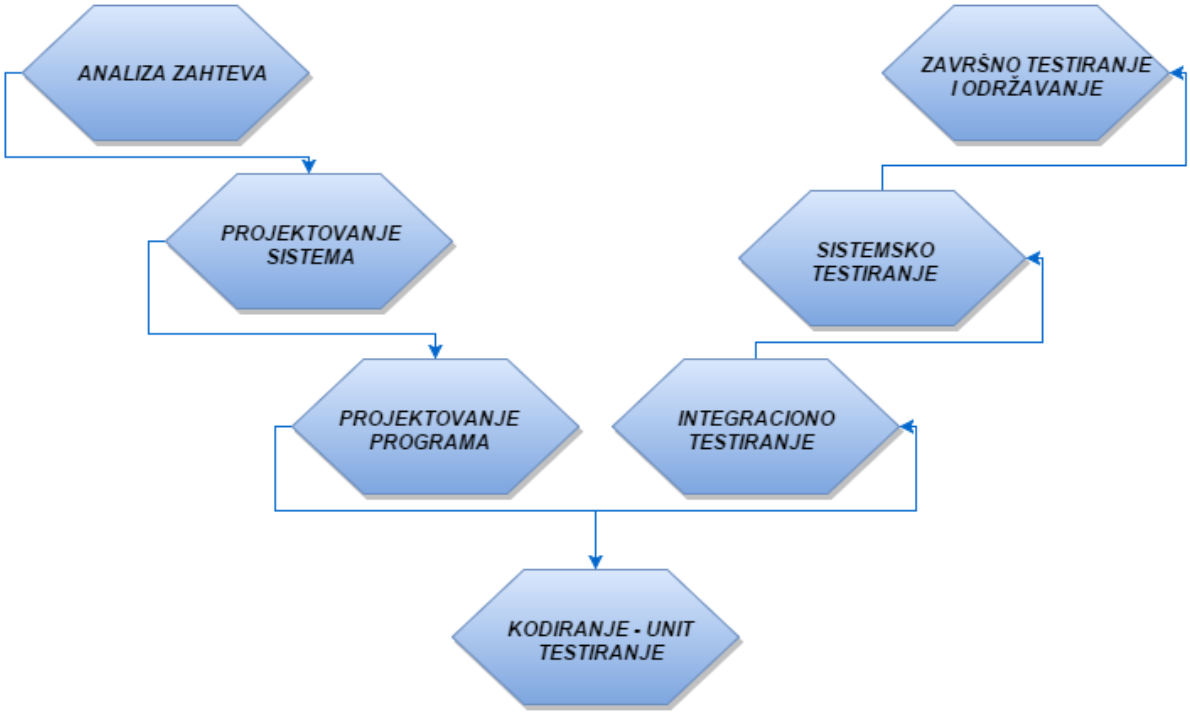
2. Teorijske osnove

U ovom poglavlju dat je opis V modela razvoja softvera, koji je korišćen za praćenje razvoja softvera kao i za njegovo testiranje. Opisane su prednosti i mane V modela. Dat je opis SNMP protokola, njegova uloga u sistemima, arhitektura i princip rada. Opisana je TMS570 hardverska platforma za koju je pisan ovaj softver.

2.1 V model razvoja softvera

V model razvoja softvera predstavlja odnos aktivnosti u okviru testiranja softvera sa fazama dizajna, analize i realizacije softvera. Naziv je dobio po izgledu dijagrama koji asocira na slovo V, gde se na levoj strani nalaze faze analize i dizajna sistema, a na desnoj strani nalaze se faze testiranja i održavanja sistema. [2] V model sugeriše da testiranje delova i testiranje pri integraciji mogu da se koriste za verifikovanje projektovanog programa. Tokom testiranja delova i testiranja pri integraciji programeri i članovi tima za testiranje treba da se osiguraju da su svi aspekti projektovanja ispravno implementirani u kodu. Slično tome, testiranje sistema treba da verifikuje projektovani sistem uz potrebu da su svi aspekti ispravno implementirani. V model se prvenstveno bavi aktivnostima i ispravnim radom sistema. [3]

Da bi sistem ispravno funkcionisao testiranje se vrši u tri faze: testiranje pojedinačnih modula sa integracijom, testiranje integrisanog sistema i završno testiranje.



Slika 1 - Šema V modela razvoja softvera

Osobine V modela:

- Testiranje delova, testiranje sistema pri integraciji i testiranje sistema utvrđuju ispravnost programa i mogu se koristiti za verifikovanje projekta programa
- Završni test služi za validaciju zahteva, odnosno proveru da li su svi zahtevi potpuno implementirani.
- Ako dođe do problema tokom verifikacije ili validacije, aktivnosti sa leve strane V modela mogu se ponoviti radi popravke ili poboljšanja zahteva, izgleda ili koda, pre nego što se ponove testiranja na desnoj strani modela

Prednosti V modela:

- Naredna faza počinje tek kada se prethodna faza završi
- Aktivnosti *planiranja testiranja* i *kreiranja testova* se izvršava pre kodovanja
- Greške koje su nastale u prethodnim fazama biće otklonjene u tekućoj fazi

Nedostaci V modela:

- Svaka promena u projektovanju povlači promene u dokumentaciji i test dokumentaciji (odnosno promene samog testa)
- Može se implementirati samo u velikim preduzećima
- Potrebno je mnogo sredstava za primenu i održavanje ovog modela

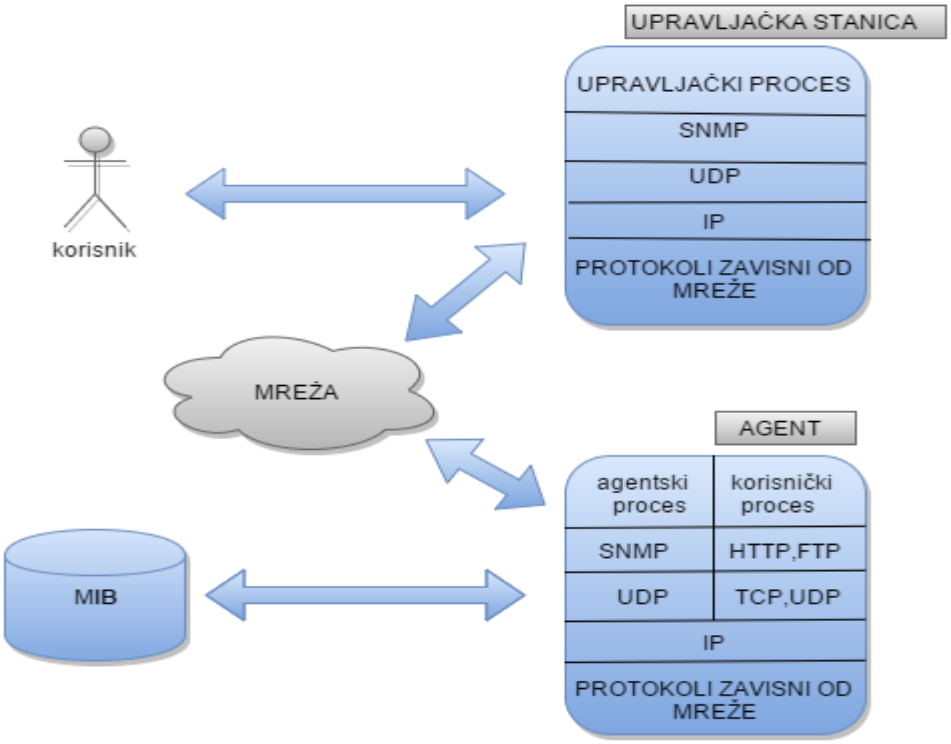
2.2 SNMP protokol

SNMP je nastao 80-tih godina sa ciljem da jednostavno integriše upravljanje heterogenim mrežama. Zasnovan je da radi na aplikativnom nivou koristeći IP kao transportni protokol i time ne zavisi od mrežne ciljane platforme. Svaki uređaj u sebi sadrži hardverski i softverski deo prilagođen prenosu podataka po unapred usaglašenom protokolu. Omogućen je nadzor različitih tipova uređaja i različitih proizvođača. SNMP je evoluirao od verzije v1, preko v2c do v3 u kojoj je zaštita podataka podignuta na viši nivo. [5]

SNMP je protokol koji omogućava kontrolnoj stanici (upravljaču) da upravlja kontrolisanim sistemom (agentom) kroz razmenu SNMP poruka. SNMP se oslanja na UDP.

SNMP poruka je paket poslat korišćenjem bespojnog prenosnog protokola UDP na pristup (port) 161. U OSI referentom modelu (eng. *Open Systems Interconnection Basic Reference Model*) SNMP spada u sedmi, odnosno sloj aplikacije. Na prvi pogled možda je neobičan izbor UDP protokola s obzirom da na prenosnom sloju postoji i spojni prenosni protokol TCP koji pruža pouzdan prenos informacija uz potvrdu prijema paketa. [6] SNMP na transportnom nivou namerno koristi UDP budući da se u ovom slučaju prednost daje brzini, odnosno što manjem zauzeću mreže.

Slika 2 prikazuje konfiguraciju sistema za upravljanje mrežom pomoću SNMP protokola.



Slika 2 - Kofiguracija sistema za upravljanje mrežom

SNMP je standardni i vrlo raširen protokol za upravljanje i administraciju mreže koji služi za prikupljanje informacija o subjektima na mreži i njihovo slanje administratoru. SNMP je veoma jednostavan protokol. Predviđene su samo dve operacije, a to su upit i zadavanje vrednosti neke promenljive. Proširenje protokola je u direktnoj zavisnosti od toga kako se definiše baza MIB. Na primer, ako korisnik želi da doda nove komande, najpre treba da ih definiše u MIB bazi. [7]

Upravljani podaci prikazani su u obliku varijabli (npr. Ime sistema, broj pokrenutih procesa) na agentima. Varijable su organizovane hijerarhijski u obliku stabla. SNMP koristi identifikatore objekta OID za određivanje mesta pojedine varijable u stablu. Svaki OID se sastoji od brojeva međusobno razdvojenim tačkom i naravno, jedinstven je. [8] Identifikatori objekta nalaze se zajedno sa još nekim informacijama u MIB-u odnosno bazi upravljanih informacija koja je opisana u sledećem poglavlju.

2.2.1 Baza upravljačkih informacija

Svaki SNMP agent sadrži popis svih svojih upravljanih objekata i taj popis predstavlja bazu upravljačkih informacija. MIB može da se definiše kao baza podataka u kojoj se čuvaju upravljani objekti čije se vrednosti kolektivno odražavaju na aktuelno stanje mreže. Svaka upravljačka stanica i Agent u SNMP-u održava lokalnu bazu podataka, koja je poznata kao MIB. Svaka vrsta statusa ili statička informacija kojoj se može pristupiti od upravljačke stanice je definisana u ovoj bazi. MIB baza podataka se sastoji od liste MIB modula, od kojih svaka predstavlja spisak objekata kojim se upravlja od strane Agenta. Svi MIB objekti unutar jednog MIB modula pripadaju istom čvoru.

Baza sadrži sledeće zapise:

- Ime
- OID
- Tip podataka
- Dozvole čitanja i pisanja
- Kratak opis za svaki objekat SNMP agenta

SNMP ne definiše unapred skup MIB varijabli, pa je dizajn veoma prilagodljiv. Svaki bitni protokol danas definiše i odgovarajući skup MIB varijabli:

- RFC 3418 opisuje bazu upravljačkih informacija za protokol SNMP
- RFC 4022 opisuje bazu upravljačkih informacija za protokol TCP
- RFC 4113 opisuje bazu upravljačkih informacija za protokol UDP
- RFC 4293 opisuje bazu upravljačkih informacija za protokol IP

Kao što se vidi, čak i sam SNMP protokol definiše svoj skup MIB varijabli koje su standardizovane odgovarajućim RFC-om.

2.2.2 Arhitektura SNMP-a

SNMP arhitektura se sastoji od dva ključna elementa: agenta i menadžera. Radi se o klijent-server arhitekturi u kojoj je agent server, a menadžer klijent.

- Agent je program koji se izvršava na svakom upravljivom ili nadziranom čvoru mreže i obezbeđuje spregu ka svim opcijama konfiguracije. Ove opcije se čuvaju u MIB bazi. Agent ima lokalno znanje o upravljačkim informacijama i prevodi ih u oblik kompatibilan sa SNMP. Omogućava udaljeni pristup opremi za upravljanje.
- Menadžer je softver. Uloga menadžera je da kontaktira razne agente i periodično prozove i prikupi podatke. To je klijent strana pri nadzoru i upravljanju.

SNMP sadrži i posebnu Trap (klopka/zamka) komandu, koja omogućava agentu da pošalje poruku u slučaju definisanih, obično kritičnih događaja (alarma). Upravljani uređaj je mrežni čvor koji sadrži SNMP agenta i koji se nalazi u upravljačkoj mreži. Uređaj za upravljanje sakuplja i čuva upravljačke informacije i čini ih dostupnima preko protokola SNMP. Upravljani uređaji (ponekad se nazivaju mrežni elementi) mogu biti ruteri, serveri za udaljeni pristup (*eng. access server*), komutatori, štampači ili uređaji energetske elektronike (ispravljači, punjači). Agent izvršava aplikacije koje prate i kontrolišu uređaje za upravljanje. NMS osigurava mnoštvo procesnih i memorijskih resursa, opremljenih za mrežno upravljanje. [9] Na upravljačkoj mreži mora postojati jedan ili više NMS. Uređaj za upravljanje prikupljene informacije šalje NMS preko SNMP.

2.2.3 Osnovna pravila kodiranja BER

BER (*eng. Basic Encoding Rules*) su originalna pravila postavljena od strane ASN1 (*eng. Abstract Syntax Notation One*) standarda za kodiranje apstraktne informacije u konkretnim tokovima podataka.

BER predstavlja sintaksu prenosa koju koristi ASN.1. Standardom ASN.1 obuhvaćena su i druga kodiranja, ali SNMP koristi samo BER. Neka od važnijih koje ASN.1 dopušta su CER (*eng. Canonical Encoding Rules*) i DER (*eng. Encoding Rules*). Glavna razlika između njih i BER kodiranja je u tome što je BER fleksibilniji, odnosno dopušta i alternativna kodiranja kao opciju pošiljalaca. Način na koji se svaki od podataka kodira pomoću BER kodiranja je sledeći: podatak se kodira pomoću identifikatora tipa, dužine podatka i same vrednosti podatka, oznake kraja ukoliko je potrebna.

Slika 3 - Format BER kodiranog polja

2.2.4 SNMP poruke

Protokol SNMP nudi tri operacije mrežnom sistemu upravljanja. To su:

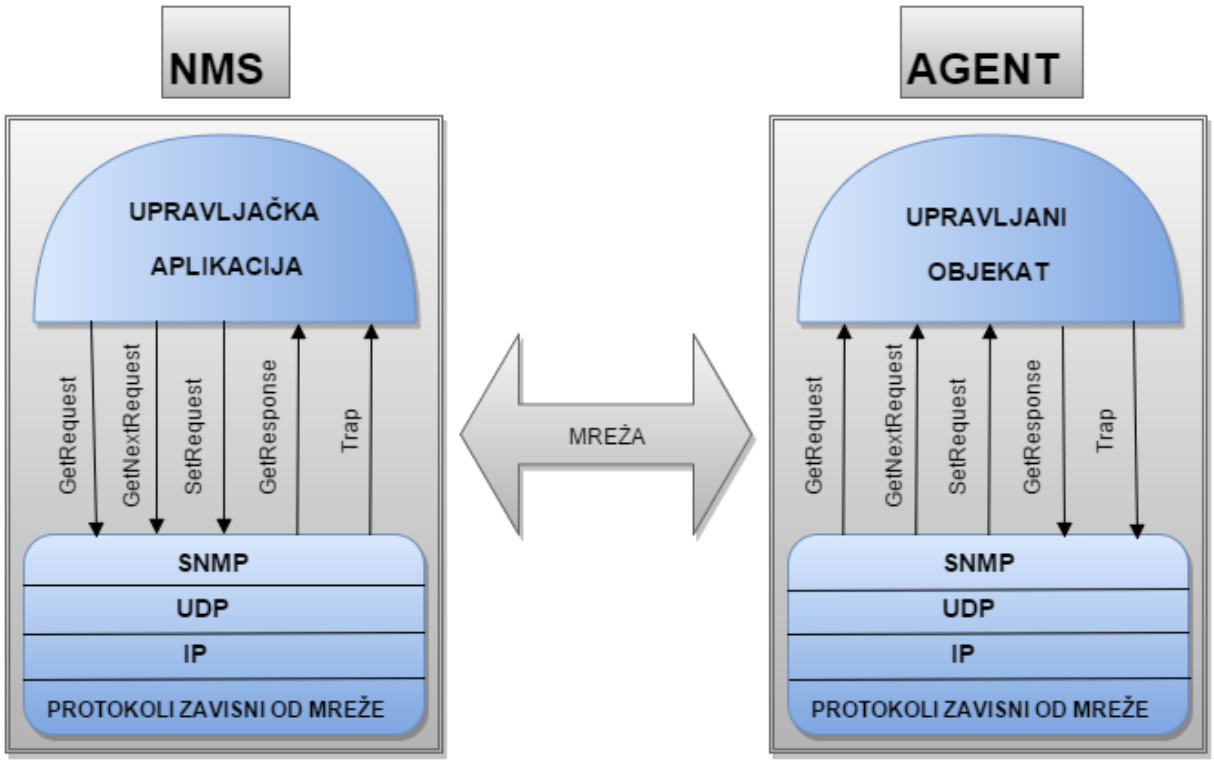
- Uzimanje (GET) – Uređaj upravljanja koristi poruku GetRequest za uzimanje vrednosti varijabli čiji se OID mora zadati u telu te iste poruke. Svaki objekat MIB-a je označen u obliku x.y, gde je x OID tog objekta a y oznaka instance. Agent odgovara slanjem SNMP poruke GetResponse i to tako da u GetResponse PDU polje upisuje odgovarajuće parove (OID, vrednosti)
- Postavljanje (SET) – služi za postavljanje vrednosti upravljanih objekata koji se nalaze u MIB-u agenta. Prilikom obavljanja operacije SET uredjaj upravljanja šalje agentu SetRequest PDU. Jedan SetRequest PDU može postaviti vrednosti jednog ili više objekata odjednom. Agent odgovara na poruku SetRequest slanjem poruke GetResponse.
- Zamka (TRAP) – koristeći ovaj mehanizam agent može obavestiti upravljačku aplikaciju da je u njegovoj komunikacijskoj okolini nastupio neki neželjeni događaj. Nakon nastupa takvog događaja agent šalje trap PDU poruku prema odredištu čija je adresa unapred konfigurisana u agentu.

Prve dve operacije se odnose na uzimanje vrednosti upravljanih varijabli, odnosno njihovo postavljanje. Postupak slanja poruka agentima u sklopu operacije uzimanja (GET) nazivamo prozivanje (*eng. Polling*) i on se odvija najčešće ciklično u nekom zadatom intervalu.

U tom intervalu uređaj za upravljanje proziva agente i nakon što sve prozove kreće ispočetka. Operacija postavljanja služi uređaju za upravljanje za postavljanje vrednosti upravljane varijable koja se nalazi u MIB-u agenta. I postavljanje (SET) i uzimanje (GET) uključuju odgovor agenta, što znači da se radi o dvosmernoj komunikaciji.

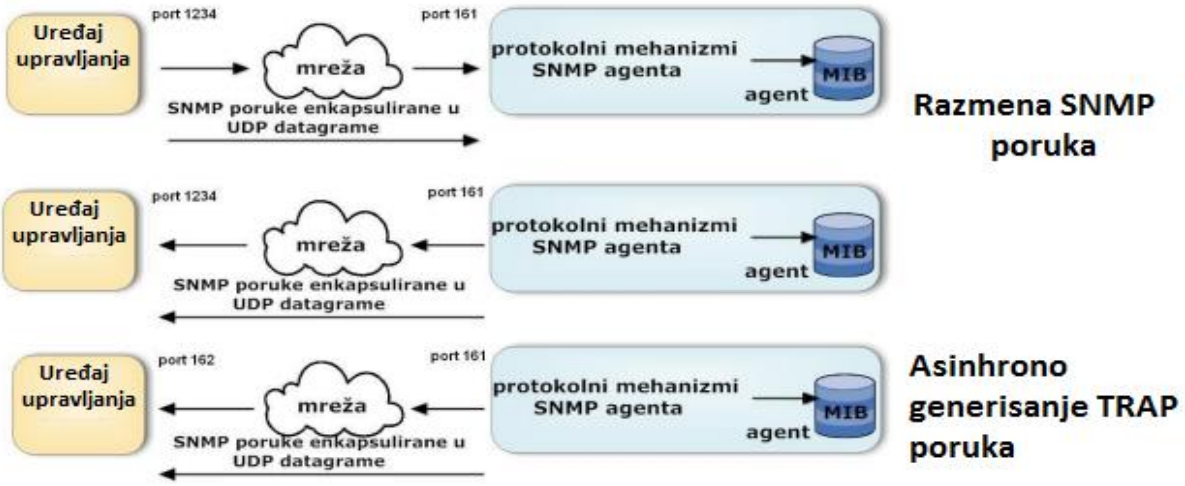
Zamka se sastoji od poruke koju šalje isključivo agent. Pomoću nje se obaveštava uređaj za upravljanje o nekom događaju koji se u tom trenutku dešava na strani agenta. [10] Uređaji

za upravljanje se mogu programirati da u slučaju ako primete obaveštenje učine neke automatske akcije (primer, ukoliko Ethernet preklopnik pošalje obaveštenje o čudnom ponašanju nekog pristupa, jedna od mogućih automatskih akcija je skidanje tog pristupa s mreže).



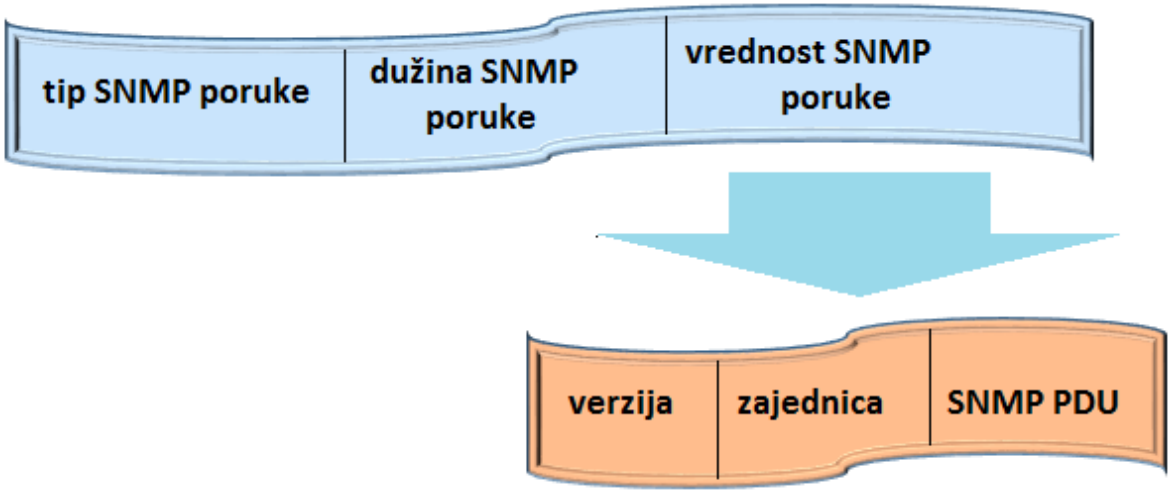
Slika 4 – SNMP poruke

Uređaj za upravljanje može agentu poslati jednu od poruka GetRequest, SetRequest i GetNextRequest. Nakon primanja bilo koje od tih poruka, agent odgovara sa porukom GetResponse. Poruku Trap agent šalje uređaju za upravljanje i time ga obaveštava o nekom događaju u svojoj okolini. Razmena poruka je prikazana na slici 5.



Slika 5 - UDP enkapsulirane poruke

Format SNMP poruke:

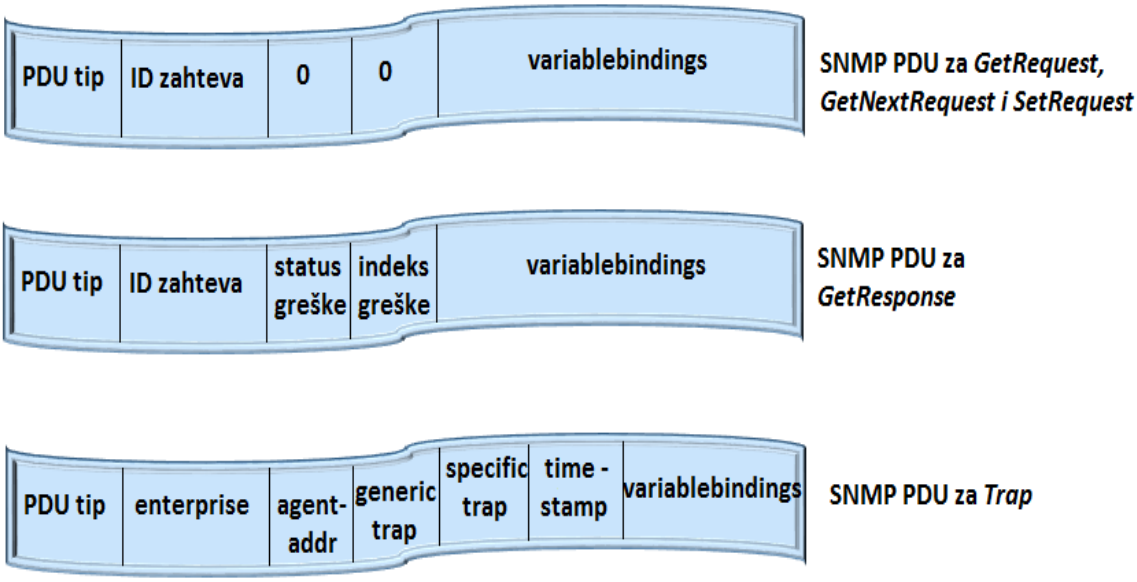


Slika 6 - BER kodiranje SNMP poruke

SNMP poruku čine tri osnovna polja:

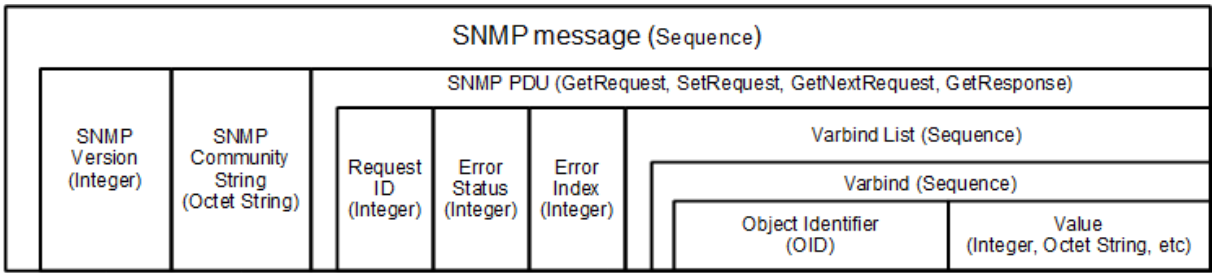
- Verzija – verzija je ceo broj i predstavlja redni broj SNMP protokola
- naziv zajednice – koristi se radi sigurnosti i predstavlja šifru.
- PDU

Za razliku od SNMP verzije i zajednice koji su jednostavni tipova podataka i nisu građeni od manjih polja, SNMP PDU je složeni tip i sastoji se od nekoliko manjih polja (slojeva). Postoje 3 vrste SNMP PDU-a koje su prikazane na slici 7.



Slika 7 – Vrste SNMP PDU

Pregled svih polja koji se mogu naći u prve dve vrste PDU-a, opisani su u tabli 1, a polja su pregledno prikazana na slici 8.



Slika 8 – Polja SNMP poruke

polje	opis
SNMP message	Niz koji predstavlja kompletnu SNMP poruku koja se sastoji od SNMP verzije, zajednice i SNMP PDU-a
SNMP version	Ceo broj koji predstavlja verziju SNMP-a (SNMPv1 = 0)
SNMP Community String	Octet String koji sadrži niz znakova korišćenih radi sigurnosti
SNMP PDU	Sadrži telo SNMP poruke. Postoji više vrsta PDU-a

Request ID	Ceo broj kojim se identifikuje pojedini SNMP zahtev. Vraća se natrag u odgovoru SNMP agenta omogućujući SNMP upravljaču da upari odgovor sa odgovarajućim zahtevom.
Error Status	Ceo broj koji je postavljen na 0x00 u zahtevu SNMP upravljača. SNMP agent stavlja kod greške u ovo polje ako se dogodila greška pri obradi zahteva.
Error Index	Ukoliko se dogodila greška, Index greške čuva pokazivač na objekat koji je uzrokovao grešku. Ako nema greške index je 0x00
Varbind List	Niz Varbind-a
Varbind	Variable binding, niz koji se sastoji od dva polja – identifikatora objekta i vrednosti objekta
Object Identifier (OID)	Pokazivač na pojedinu varijablu u SNMP Agentu
Value	*SetRequest* PDU – vrednost se postavlja na specifičan OID SNMP Agenta *GetRequest* PDU – vrednost je NULL i služi kao place holder za povratne podatke *GetResponse* PDU – povratna vrednost od specifičnog OID-a

Tabela 1 – Pregled polja SNMP poruke

2.3 TMS570 Platforma

TMS570 platforma je mikrokontroler visokih performansi koji se koristi za ugradnju u sisteme korišćene u industriji. TMS sadrži ARM Cortex R4 procesor. Ovaj procesor podržava BIG-ENDIAN (BE32) format. Platforma poseduje 384+256KB integrisane flash memorije i 32KB RAM memorije. [11] Obe memorije poseduju detekciju i korekciju grešaka.

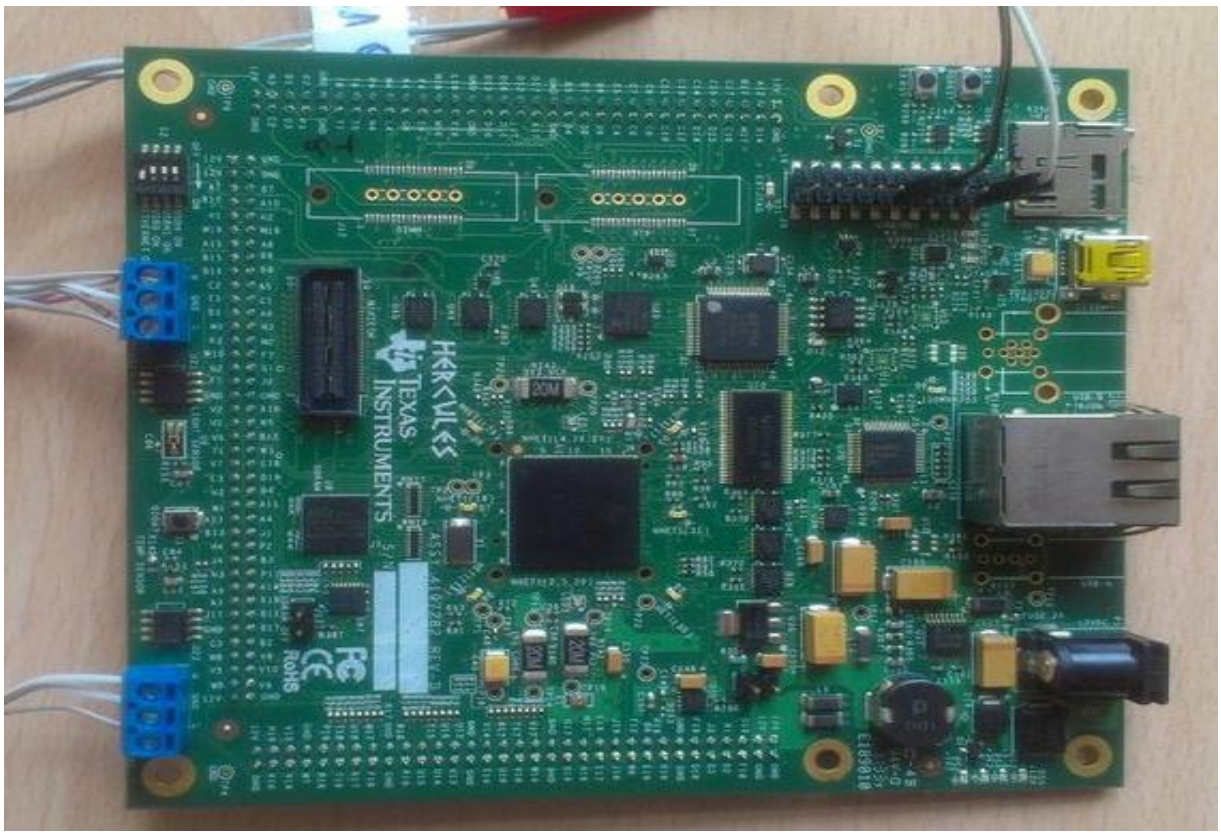
TMS570 platforma ima periferije za kontrolu aplikacija zasnovanih u realnom vremenu, uključujući dva N2HET (*eng. Next Generation High-End Timer*) koprocesora i dva ADCs (eng. *Analog-to-Digital Converters*) koji podržava do 24 ulaza. N2HET je napredni inteligentni tajmer, koji omogućava sofisticirane vremenske funkcije za aplikacije u realnom vremenu i pogodan je za aplikacije koje zahtevaju više informacija.

TMS570 ima i višestruku komunikacionu spregu :

- tri MibSPI sprege
- SPI
- LIN (*eng. Local Interconnect Network*)
- tri DCAN (eng. *Davidson Career Advisor Network)*
- I2C (*eng. Inter-Integrated Circuit*)
- Ethernet
- FlexRay kontroler

SPI magistrala omogućava brzi serijski prenos podataka između uređaja sa sličnim tipom podataka. LIN podržava Local Interconnect standard 2.0 i može se koristiti kao UART full-duplex prenos. DCAN podržava CAN 2.0 protokol standard i koristi serijski protokol koji efikasno distribuira kontrolu u realnom vremenu. DCAN je idealan za sisteme koji rade u teškim uslovima koji zahtevaju pouzdanu serijsku komunikaciju. On koristi serijski multimaster komunikacioni protokol koji uspešno vrši kontrolu u realnom vremenu.
FlaxRay kontroler koristi dvokanalni serijski, fiksno vremenski protokol. Ethernet modul podržava MII, RMII i MDIO sprege.
I2C je multimaster komunikacioni modul koji predstavlja spregu između mikrokontrolera i I2C kompatibilnog uređaja preko serijske I2C magistrale.

Sa integrisanim sigurnosnim funkcijama i širokim izborom komunikacionih i kontrolnih periferija TMS570 platforma je idealno rešenje za aplikacije u realnom vremenu visokih performansi. Platforma je prikazana na slici 9.



Slika 9 – TMS570 Platforma

3. Koncept rešenja

Ovim zadatkom je razvijen modul SNMP Agent na Ethernet platformi. Ethernet platforma je ugrađena platfoma (*eng. embedded*) koja ima brojne prednosti u okviru industrijske automatizacije i kontrole, kao i niže troškove opreme.

3.1 Arhitektura Ethernet platforme

Ethernet platforma se sastoji od sledećih blokova:

- Hardverska podrška
- Sistem datoteka (*eng. File System*)
- Podrška za komunikaciju i umrežavanje
- Mrežni servisi
- modul za upravljanje vremenom izvršavanja zadataka (*eng. Runtime System*)

Hardverska podrška obezbeđuje generički pristup drajverima sa višeg nivoa, tj. Ethernet End System drajveru da se izvršava.

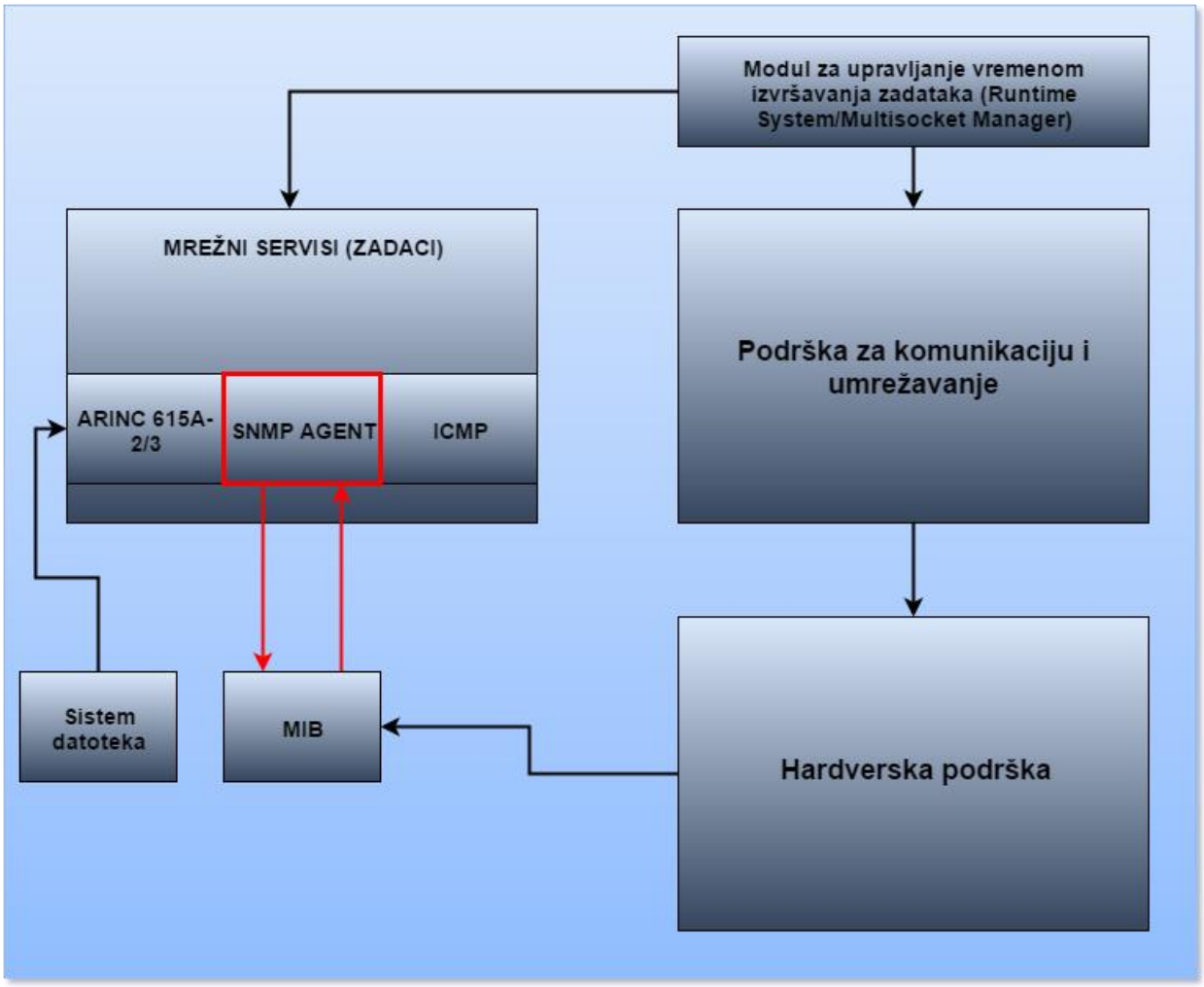
Sistem datoteka obezbeđuje pristup skladištu podataka, kao što je EEPROMs. ARINC 615a-2/3 Data Loader će ga koristiti u kontekstu ovog okvira, a korisnik aplikacije može koristiti ovu funkciju npr. za događaj logovanja.

Podrška za komunikaciju i umrežavanje se bavi zastupljenosti mrežnog uređaja, npr. Ethernet End System ili standardni Ethernet NICs (2 nivo u ISO/OSI referentnom modelu) i podrškom protokola višeg nivoa, odnosno IPv4 I UDP (3 i 4 nivo u ISO/OSI referentnom modelu).

Mrežni servisi će biti izgrađeni na vrhu podrške za komunikaciju i umrežavanje i obuhvataju:

- ARINC 615A-2/3 Data Loader
- SNMP
- ICMP

Modul za upravljanje vremenom izvršavanja zadataka (RTS) predstavlja osnovu za izvršavanje zadataka koji podržavaju vremenski-aktivne zadatke (fiksna aktivacija, fiksna raspodela vremena i rok izvršavanja). RTS – Ethernet sinhronizacija je pomoćna funkcija koja održava lokalni raspored modula za upravljanje vremenom izvršavanja zadataka zbog usklađenosti sa Ethernet komunikacionim ciklusom. Ovo omogućava čvrstu sinhronizaciju zadataka različitih mrežnih čvorova, a potom i malo kašnjenje end-to-end u komunikaciji između zadataka.



Slika 10 - Blok šema Ethernet platforme

Hardverska podrška obezbeđuje pristup drajverima sa višeg nivoa. Hardverska podrška ima dva cilja. Prvi je taj da će se softver razvijati tako da bude portabilan, a drugi cilj je integracija sprege koja omogućava korisnicima – klijentima da integrišu softver koji koristi ovu podršku na vlastitim okruženjima. Aplikacija hardvera se odnosi na prenosivost i ponovnu upotrebu u smislu da odvaja funkcionalnu opremu kontrole (mikročip) od opreme za konekciju i vezu sa procesorom (SPI ili PCI magistrala).

Sistem datoteka je vezan za prenosivost i ponovnu upotrebu i pruža prateće infrastrukture. Ova infrastruktura omogućava druge komponente, pre svega ARINC615-A2/3 Data Loader, da se dizajniraju na način prenosne i višekratne upotrebe. Sistem datoteka je objekat koji ima listu uređaja za skladištenje. To obezbeđuje spregu za upravljanje datotekama (inicijalizaciju, pisanje, čitanje, brisanje fajlova), odnosno ona pruža jedinstveno sredstvo za pristup skladištenim podacima u sistemu. Skladišteni podaci mogu biti Flash, NVRAM, FRAM itd.

U bloku podrška za komunikaciju i umrežavanje koriste se sledeći termini:

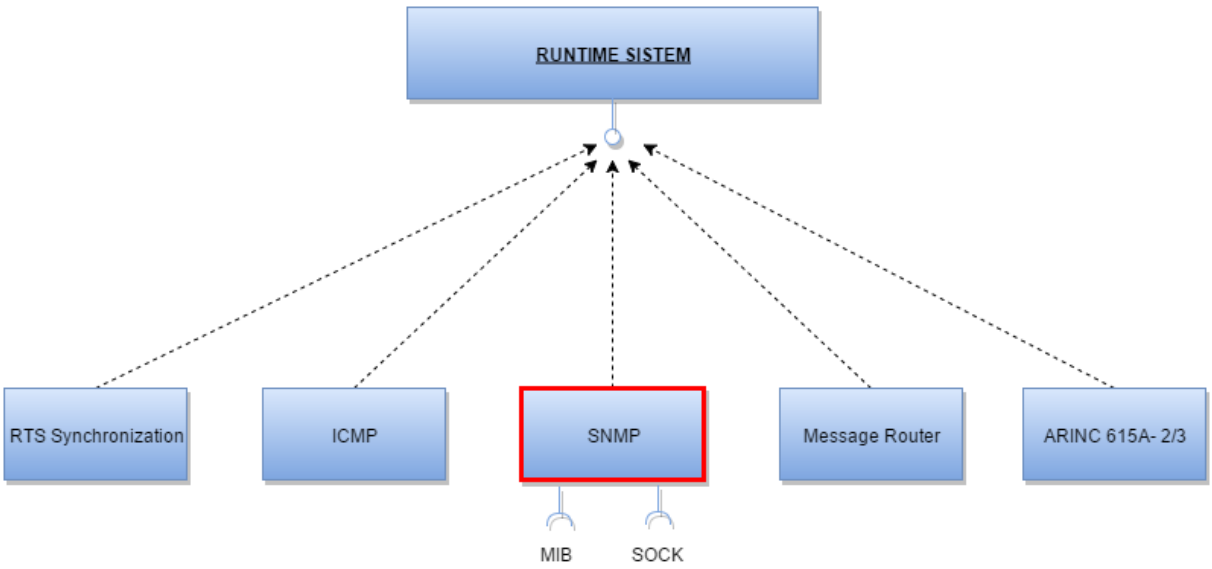
- **Mreža** je skup mrežnih čvorova (računara), koji su međusobno povezani preko mrežne veze, odnosno fizičkog medijuma. Mrežna sprega daje fizičku vezu između mrežnog čvora i mrežne veze. Pored računarskih elemenata, mreža može zahtevati dodatne elemente konekcije npr. prekidače u Ethernet-u.
- **Mrežna sprega** omogućava prenos podataka između mrežnih čvorova preko mrežne veze. To obično obrađuje nizak nivo informacija mreže, u Ethernet-u to su FCM (*eng. Frame check sequence*) ili IFG (*eng. Inter frame gap*). Mrežna sprega može biti standardni Ethernet NIC (*eng. Network Interface Controller*) i Ethernet End System Controller. Ethernet pruža i podršku viška, jedna mrežna sprega koristi više mrežnih veza.
- **Komunikaciona sprega** je generalizacija mrežne sprege, što znači da se ne mora navesti konkretan format za razmenu podataka (informacije o zaglavlju ili adresiranje).
- **Komunikaciona veza** je generalizacija fizičke mrežne veze, odnosno to je prenos podataka između komunikacionih sprega. Predstavlja vezu između dve komunikacine sprege.
- **Mrežni protokol** je dobro definisan skup pravila za razmenu podataka između računara koji čine sintaksu i semantika, npr IPv4. Svaki mrežni protokol će biti povezan sa jednom komunikacionom spregom
- **Mrežna konekcija** je logička sprega koja se sastoji od specifične kombinacije mrežne sprege, komunikacione sprege i protokola, uključujući i pomoćne informacije kao što su adresiranje i pakovanje podataka. Primer mrežne konekcije može biti: Ethernet (mrežna sprega), Generički Ethernet Adapter (komunikaciona sprega), IPv4 (mrežni protokol), UDP (mrežni protokol)
- **Mrežni servis** je aplikacija koja se izvršava na mrežnom čvoru i pruža čuvanje podataka, manipulaciju, prezentacije ili druge mogućnosti. Obično se specifični

protokol mrežnog servisa zasniva na mrežnoj konekciji i unapred utvrđenim potokolom (npr. UDP).

- **Mrežni servisi** – sve mrežne usluge koje pruža Ethernet platforma pružaju funkcije koje su od suštinske važnosti za mrežnu opremu. Sve mrežne usluge su aktivne komponente, takozvani zadaci koji potiču od modula za upravljanje vremenom izvršavanja zadataka.
- **Modul za upravljanje vremenom izvršavanja zadataka** daje osnov za izvršavanje svih zadataka.

3.2 Zadaci

Više od jednog zadatka radi na istom hardveru, kako bi modul za upravljanje vremenom izvršavanja zadataka, memorija i ostali resursi mogli da se upravljaju. Dok ARINC615A-2/3, ICMP, SNMP reaguju samo na određene zahteve, SNMP Trap i RTS Sinhronizacija reaguju na posebne zahteve. SNMP reaguje na uslove definisanih od strane određenih TRAP MIB-a, a RTS Sinhronizacija reaguje na odstupanje u lokalnom vremenu iz Ethernet vremena. Slika 2 pokazuje kako je SNMP modul uklapa u Ethernet platformu.



Slika 11 - Prikaz SNMP modula u Ethernet platformi

Kod koncepta zadatka primenjuje se multitasting. To znači da se zadaci koji su prekinuti ili zamenjeni drugim zadatkom, mogu nastaviti posle u nekom trenutku.

Zakazivanje (zasnovano na konfiguraciji) određuje trenutak početka izvršavanja i trajanje izvršavanja za svaki zadatak u sistemu.

Komponenta koja upravlja zadacima je modul za upravljanje vremenom izvršavanja zadataka. Modul za upravljanje vremenom izvršavanja zadataka pruža osnovu za izvršavanje sistemskih usluga i korisnički definisanih zadataka. Implementiran za ugrađene sisteme koje rade u realnom vremenu.

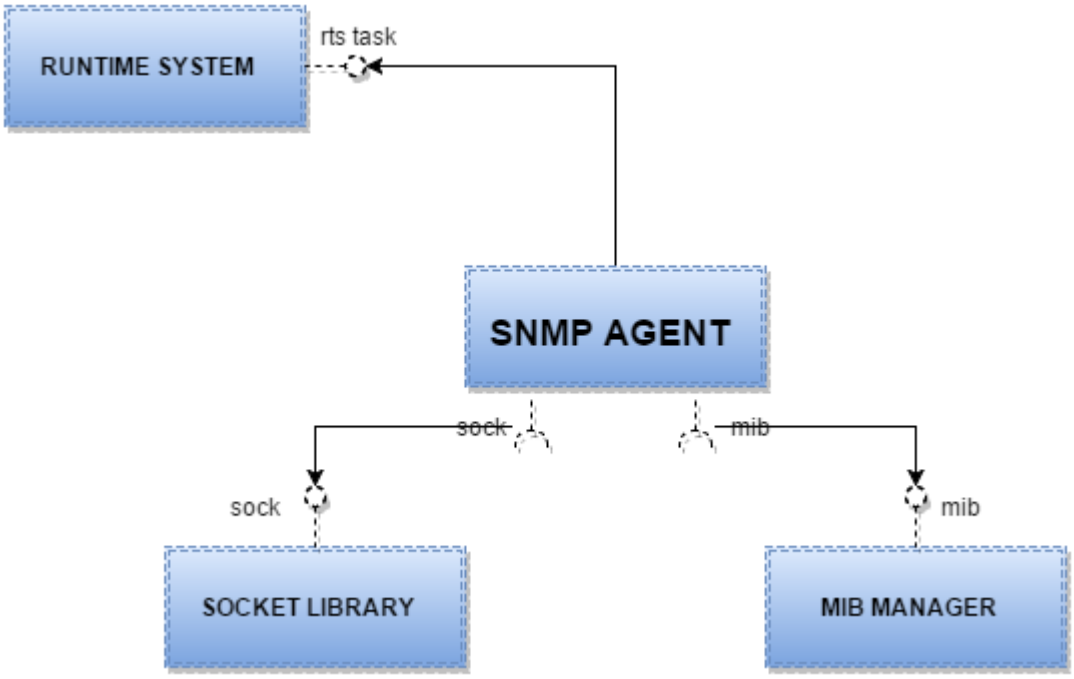
3.3 SNMP Agent

SNMP agent pruža informacije o upravljanju različitih uređaja u sistemu upravljanja mrežom. SNMP Agent koji se koristi na ovoj Ethernet platformi je podeljen u dva nezavisna modula: SNMP modul i SNMP Trap modul. Svaki modul predstavlja aktivnu komponentu (zadatak) koja se zakazuje od strane modula za upravljanje vremenom izvršavanja zadataka prema RFC 1157.

SNMP modul služi za komunikaciju menadžer – agent. On prima zahteve od menadžera i dekodira ih. U zavisnosti od vrste zahteva, SNMP modul preuzima drugačije korake u rukovanju zahtevima na meti. U suštini SNMP modul obezbeđuje sredstva za pronalaženje upravljanog objekta iz mete i da se te vrednosti menjaju tokom vremena rada sistema. Upravljani objekti su definisani u MIB bazi. SNMP modul izvršava standardne SNMP zahteve.

TRAP modul služi za upravljanje trap porukama i izvršava se kao poseban aktivni zadatak. Trap objekti, koje agent može generisati, definisani su u MIB bazi. Tokom vremena izvršavanja TRAP modula, TRAP modul proverava sve objekte u MIB bazi. Ako je došlo do nekog kritičnog događaja, TRAP modul sastavlja poruku i šalje je sistemu upravljanja mrežom.

Oba SNMP agent modula imaju isto arhitektonsko rešenje za zadatke, uključujući i sprege koje koriste da ostvare svoju funkcionalnost. Slika 3 pokazuje SNMP modul i svoju spregu. U SNMP modulu koriste MIB sprege za preuzimanje statusa promenljivih traženih od strane sistema upravljanja mrežom. Iako oni koriste istu spregu, ovi moduli koristite različite MIB baze podataka.



Slika 12 - Blok šema SNMP Agenta

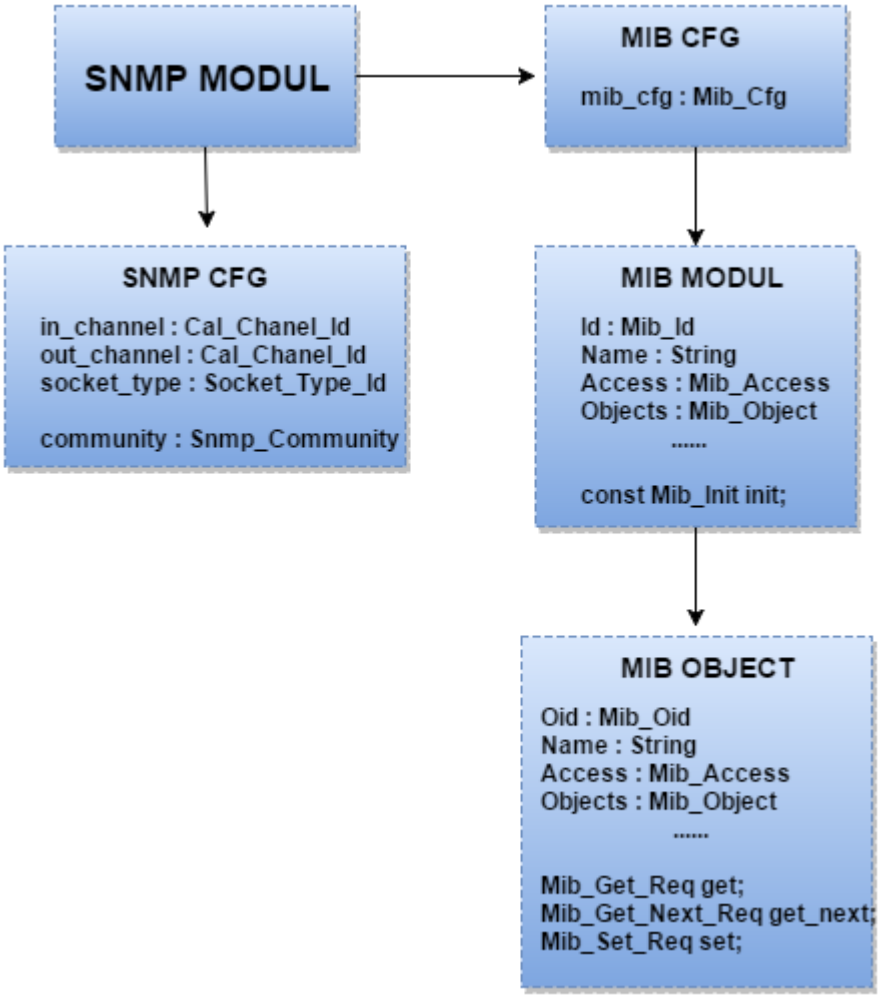
3.3.1 SNMP Modul

SNMP modul se sastoji od dve glavne funkcije koje su vidljive (sprežne funkcije) i određenog broja unutrašnjih funkcija koje se koriste od strane SNMP modula kako bi se obavljali neophodni koraci u rukovanju zahteva iz jednog sistema mrežnog upravljanja. Ove dve funkcije su:

- snmp_init ()
- snmp_ handler ()

Prva funkcija se poziva prilikom pokretanja procesa, dok se druga funkcija poziva iz niti i zapravo predstavlja SNMP modul. Inicijalizacija SNMP modula se vrši pozivom snmp_init funkcije. SNMP modul inicijalizuje SNMP komunikacijsku spregu i sve MIB module. Za svaki MIB modul poziva se funkcija za inicijalizaciju. Funkcija za inicijalizaciju MIB modula pokreće sve unutrašnje delove specifičnog MIB modula. MIB moduli su navedeni u konfiguraciji projekta. Oni se dodaju na sistem po kompajliranju.

Da bi smo mogli da rukujemo SNMP zahtevima iz Sistema mrežnog upravljanja i da se šalju SNMP odgovori, SNMP agent mora da se konfiguriše. Konfiguracija SNMP modula se vrši konfiguracijskom strukturom. Ova struktura konfiguriše komunikacione i parametre soketa, string komunikacije, maksimalnu veličinu SNMP poruke i maksimalan broj varbinds po SNMP zahtevu. Slika 5 prikazuje zavisnost između SNMP modula, SNMP MIB konfiguracija i MIB specifičnih konfiguracionih fajlova.

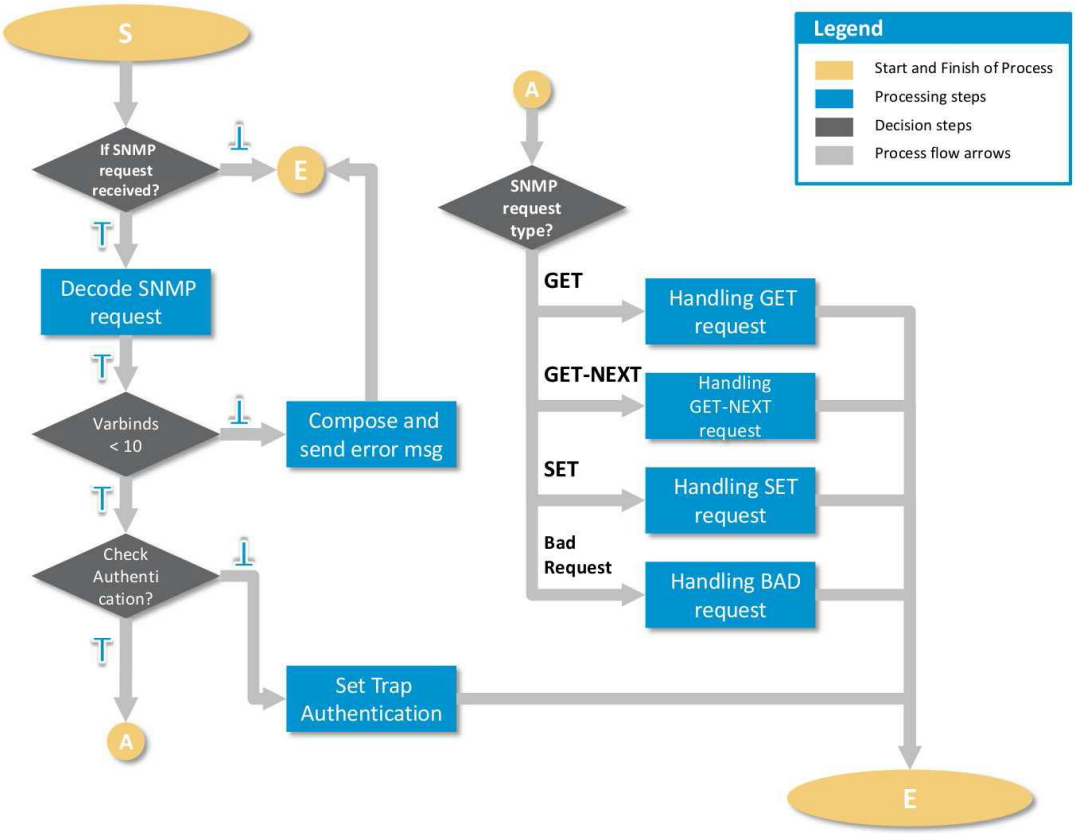


Slika 13 - Blok šema konfiguracije SNMP modula

Pomoću funkcije *snmp_handler* SNMP modul prima i obrađuje SNMP zahteve koje dobije od sistema mrežnog upravljanja. Rukovanje SNMP zahteva se vrši u nekoliko koraka.

Kada se poziva snmpv_handler funkcija, prvi korak je da proverite da li je stvarno SNMP zahtev primljen na definisanoj utičnici. Ako ništa nije primljeno na utičnici ili je greška otkrivena, izvršenje funkcija skače do kraja i funkcija vraća odgovarajuću povratnu vrednost za otkriveni događaj. Ako je nešto primio, izvršenje funkcija nastavlja da dekodiranje primljene poruke. Svi zahtevi SNMP imaju specijalni format poruka i dekodiranje se vrši prema tom formatu. Pre osnovanja zahteva se provera, SNMP modul vrši proveru autentičnosti koristeći komunikacione stringove izdvojene iz zahteva i upoređujući ga sa prethodno definisanom listom dozvoljenih komunikacionih stringova. Ako je provera uspešna, funkcija izvršenje nastavlja u pravcu rukovanja zahteva, u suprotnom SNMP modul signalizira neuspešnu autentičnost u TRAP modul. U zavisnosti od vrste zahteva SNMP (GET, GET-NEXT, SET), izvršava se jedan od moguća četiri zahteva. Iako SNMP ima samo tri vrste zahteva, četvrti se izvrši u slučaju

prijema lošeg zahteva iz mreže. Ako je zahtev odgovara jedan od poznatih SNMP zahteva, izvršenje nastavlja sa rukovanjem primljenog zahteva.



Slika 14 - Algoritam SNMP modula

Rukovanje zahteva se razlikuje u zavisnosti od tipa zahteva i zato postoje tri procedure:

- **Rukovanje GET zahteva -** za sve promenljive vezane za GET zahtev, agent izvršava sledeću proceduru. SNMP modul nalazi primljeni MIB objekat (OID) u MIB bazi podataka. Za svaki MIB modul proverava se pravo pristupa i lista MIB objekata se proverava u odnosu na primljeni OID. Provera se vrši na MIB objektu. Ako provera uspe, MIB objekat poziva get() funkciju koja se čuva u desktriptoru MIB objekta. Get() funkcija pronalazi objekat value (vrednost) iz sistema. Objekat value (vrednost) i status (povratna vrednost get() funkcije) vraća u SNMP modul. SNMP modul sastavlja SNMP GET odgovor od primljenih promenljivih i njegovih preuzetih vrednosti. SNMP GET odgovor se šalje sistemu mrežnog upravljanja.

Ako provera pravila ne uspe, SNMP modul sastavlja SNMP GET odgovor sa statusom greške error ID i indeksom greške postavljen na Varbind ID.

- **Rukovanje GET-NEXT zahteva** - za sve promenljive vezane za GET-NEXT zahtev, agent izvršava sledeću proceduru. SNMP modul nalazi primljeni MIB objekat (OID) u MIB bazi podataka. Za svaki MIB modul proverava se pravo pristupa i lista MIB objekata se proverava u odnosu na primljeni OID. Provera se vrši na MIB objektu. Ukoliko je provera uspešna , u zavisnosti od vrste pronađenog MIB objekata (skalar ili kolona), rukovanje SNMP GET-NEXT zahteva ide u različitim pravcima. Ako je pronađeni MIB objekat skalar, SNMP modul pronalazi sledeći element u MIB bazi podataka koji je dostupan za čitanje. Ovde postoje dva slučaja: ako je naredni pronađeni element skalar SNMP modul poziva get() funkciju tog MIB objekta, a ako je naredni pronađeni element kolona, SNMP modul poziva get-next() funkciju tog MIB objekta. Ako je prvi nađeni MIB objekat kolona, SNMP modul poziva get_next() funkciju tog MIB objekta. Ako SNMP modul ne pronađe primljeni objekat (skalar ili kolona) u MIB bazi podataka, SNMP modul traži prvi naredni pristupačni objekat. Funkcija koja pronalazi OID vraća informaciju o težini. Težina daje informacije o broju pronađenih objekata između nivoa primljenih OID-a i OID-a u bazi podataka. Pored težine, dobijamo informaciju o modulu i indeksu objekta (objekat reference) gde smo postigli tu težinu. Opet imamo dva slučaja: Ako je objekat skalar SNMP modul poziva get() funkciju tog MIB objekta, a ako je objekat kolona, SNMP modul poziva get_next() funkciju tog MIB objekta. Get() i get-next() funkcije preuzimaju objekat vrednosti iz sistema. Objekat vrednost, OID i status (povratna vrednost get()/get_next funkcije) se vraća u SNMP modul. SNMP modul sastavlja SNMP GET-NEXT odgovor od OID-a i njegovih preuzetih vrednosti. SNMP GET-NEXT odogovor se šalje sistemu mrežnog upravljanja. Ako provera pravila ne uspe, SNMP modul sastavlja SNMP GET-NEXT odgovor sa statusom greške error ID i indeksom greške postavljen na Varbind ID.
- **Rukovanje SET zahtevima** - za sve promenljive vezane za SET zahtev, agent izvršava sledeću proceduru. SNMP modul nalazi primljeni MIB objekat (OID) u MIB bazi podataka. Za svaki MIB modul proverava se pravo pristupa i lista MIB objekata se proverava u odnosu na primljeni OID. Provera se vrši na MIB objektu. Ako provera uspe, MIB objekat poziva set() funkciju koja se čuva u desktriptoru MIB objekta. Set() funkcija menja objekat vrednosti u sistemu. Novi objekat vrednosti i status operacija(povratna vrednost funkcije set()) se vraća u SNMP

modul. SNMP modul sastavlja SNMP SET odgovor od primljenih promenljivih i njegovih novih zadatih vrednosti. SNMP SET odogovor se šalje sistemu mrežnog upravljanja. Ako provera pravila ne uspe, SNMP modul sastavlja SNMP SET odgovor sa statusom greške error ID i indeksom greške postavljen na Varbind ID.

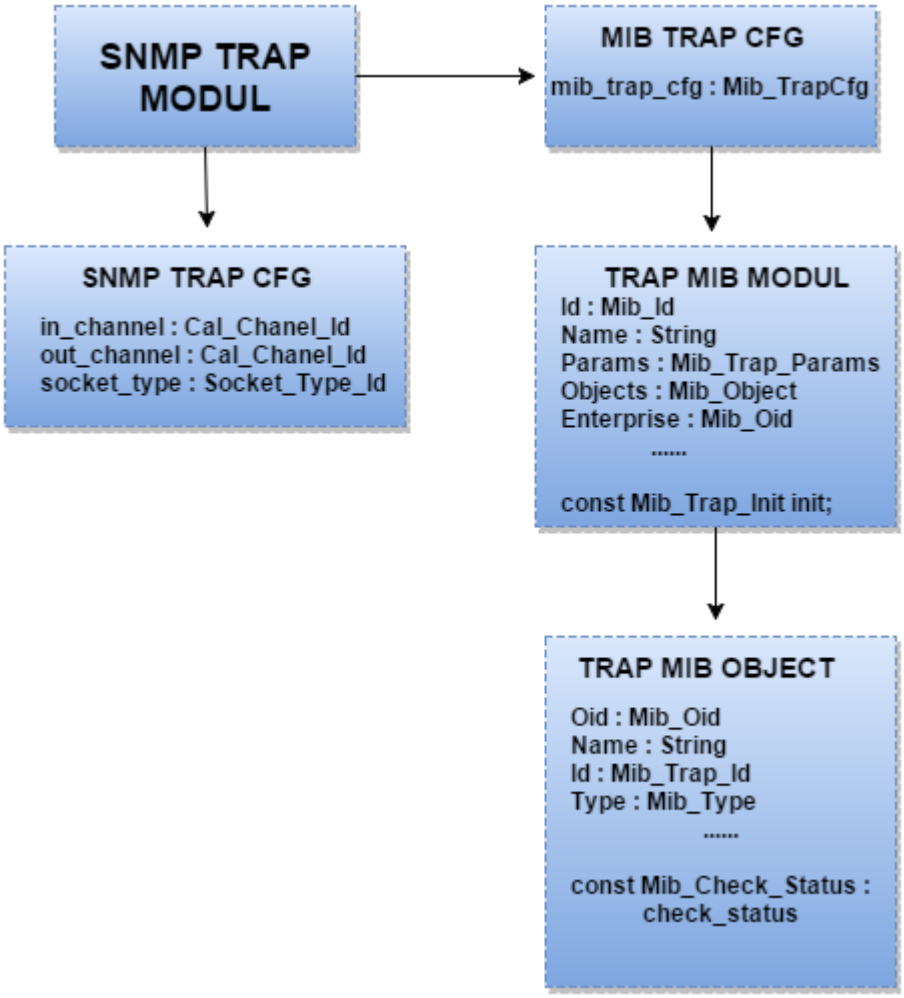
3.3.2 SNMP Trap modul

SNMP Trap modul je odgovoran za zamke. Zamka potiče iz agenta i šalje se do odredišta zamke, koji mora biti konfigurisan unutar samog agenta. Destinacija zamke je IP adresa sistema mrežnog upravljanja. Sastoji se od dve glavne funkcije koje su vidljive (sprežne funkcije) i određenog broja unutrašnjih funkcija koje se koriste od strane SNMP Trap modula u cilju obavljanja neophodnih koraka za slanje trap poruke. Ove dve funkcije su:

- snmp_trap_init ()
- snmp_ trap_handler ()

Inicijalizacija na SNMP Trap modula se vrši pozivom snmp_trap_init funkcije. SNMP Trap modul inicijalizuje SNMP Trap komunikacijsku spregu i sve Trap MIB module. Za svaki Trap MIB modul poziva se funkcija za inicijalizaciju. Funkcija za inicijalizaciju Trap MIB modula pokreće sve unutrašnje delove specifičnog MIB modula. To uključuje postavljanje prava pristupa Trap-MIB modula, Trap-MIB destinaciju IP adrese i Trap-MIB destinaciju ID zajednice. Trap MIB moduli su navedeni u konfiguraciji projekta. Konfiguracija na nivou Trap modula vrši se konfiguracijskom strukturom. Ova struktura konfiguriše TRAP komunikaciju i parametre utičnice vezane za sprežne utičnice na Ethernet platformi i maksimalnu veličinu Trap poruke. Oni se dodaju na sistem po kompajliranju.

Lista TRAP MIB modula u okviru projekta određena TRAP MIB konfiguracijom i TRAP konfiguracijskom strukturom i prenosi se na TRAP modul u početnom vremenu(prilikom inicijalizacije).



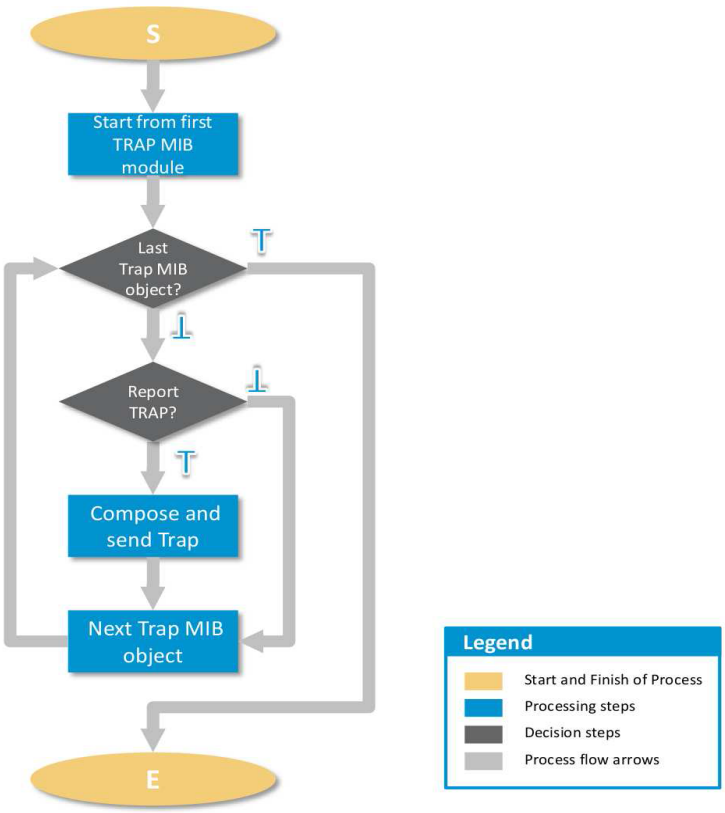
Slika 15 – Blok šema konfiguracije SNMP Trap modula

Funkcija trap_handler zapravo predstavlja SNMP Trap modul. Posao ove funkcije a samim tim i SNMP TRAP modula je provera da li se prijavio trap (zamka). Slika 6 prikazuje grafički prikaz SNMP Trap modula.

Za svaki TRAP-MIB objekat u TRAP Modulu, poziva se funkcija check_status(). Povratna vrednost ove funkcije ukazuje da li se prijavio trap(zamka). Ako se prijavila zamka, SNMP Trap modul sastavlja trap poruku uključujući promenljive koje je primio od odgovarajućeg TRAP-MIB objekta (od funkcije check_status) i izveštaj zamke šalje koristeći funkcionalnost utičnice.

Poseban slučaj slanja zamke je ako se javio neuspeh potvrde identiteta. U ovom slučaju imamo drugačiju situaciju. Oba modula, SNMP modul i SNMP Trap modul imaju pristup TRAP-MIB modulu. Ako neki događaj (npr. Neuspešna potvrda identiteta) mora biti prijavljen, SNMP modul obaveštava namenski TRAP-MIB modul o događaju da prijavi (poziv funkcije tt_set_auth_trap_flag()). Ova funkcija će u TRAP-MIB modulu ukazati da je provera identiteta neuspešna i da trap treba poslati. Pored toga, destinacija IP adrese i destinacija komunikacijskog

ID čuva se u namenskom TRAP-MIB modulu (privatni podaci). Zamka za neuspešnu potvrdu identiteta se prijavila od strane SNMP Trap modula tokom narednog pozivanja.



Slika 16 - Algoritam SNMP Trap modula

4. Programsko rešenje

Programsko rešenje se odnosi na konkretnu realizaciju SNMP Agenta i njegovo ugrađivanje na Ethernet ugrađenu platformu. Pri izradi ovog modula korišćen je programski jezik C.

SNMP Agent je integrisan u okviru Softverske Ethernet platforme i sastoji se od dva funkcionalna modula:

- SNMP modul
- SNMP Trap modul

SNMP modul je odgovoran za rukovanje operacija GetRequest, SetRequest, GetNextRequest i Walk (na osnovu GetNextRequest). SNMP Trap modul je odgovoran za rukovanje trap poruka. Realizacija ova dva funkcionalna modula je grupisana u osam C izvornih datoteka koje su opisane u tabeli.

SNMPv1_API.c	SNMP modul , sprežne i interne funkcije
SNMPv1_Trap_API.c	SNMP Trap modul, sprežne i interne funkcije
SNMPv1_Stat.c	Interne funkcije vezana za SNMP MIB sprege
SNMPv1_Generic_Traps.c	Interne funkcije koje se odnose na rukovanje zamkama
SNMPv1_i_Helper.c	Interne uslužne funkcije
SNMPv1_i_Set.c	Interne uslužne funkcije za dekodovanje
SNMPv1_i_Get.c	Interne uslužne funkcije za enkodovanje
SNMPv1_task.c	SNMP zadaci

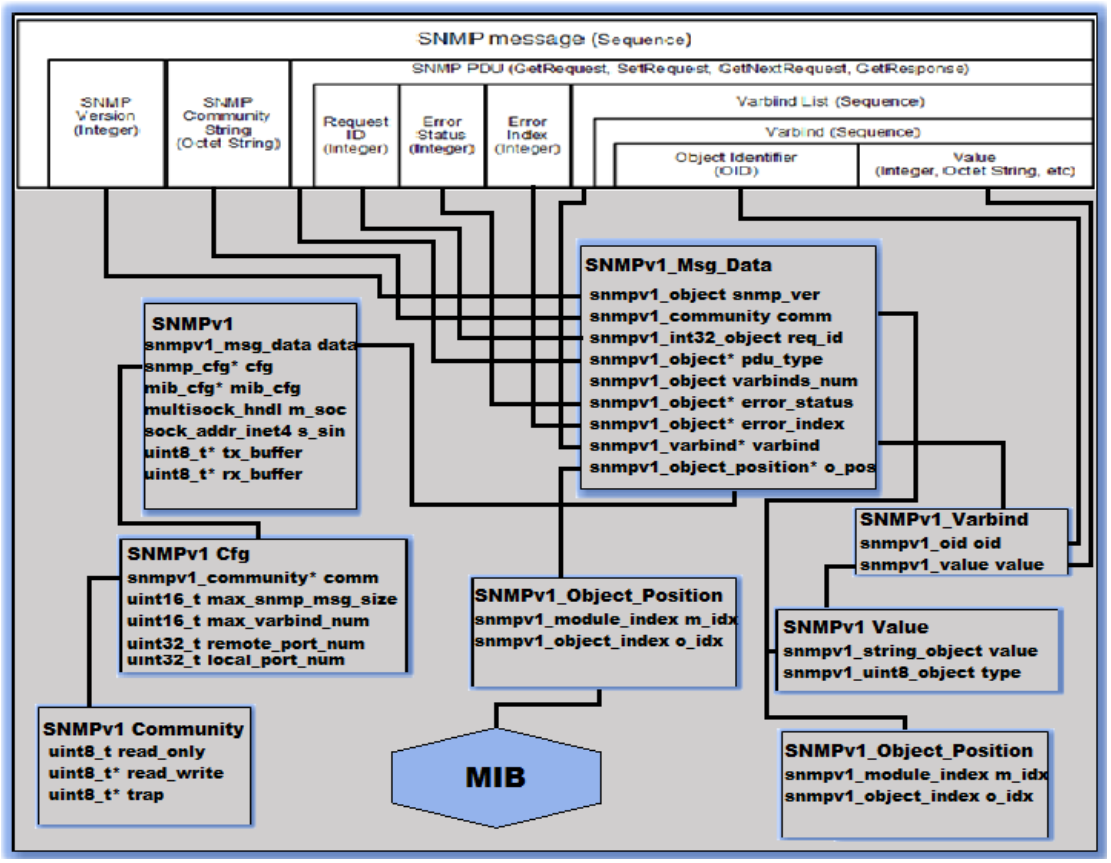
Tabela 2 – C izvorne datoteke SNMP Agenta

4.1 Strukture

Opis struktura:

- **SNMPv1** – sadrži sve neophodne informacije za rukovanje SNMP zahtevima.
- **SNMPv1_Trap** – sadrži sve neophodne informacije za zamke (*eng. traps*)
- **SNMPv1_Msg_Data** – sadrži informacije o primljenom SNMP zahtevu i izvršavanju istog
- **SNMPv1_Varbind** - sadrži sve neophodne informacije SNMP varbind polja
- **SNMPv1_Value –** sadrži sve neophodne informacije SNMP value polja
- **SNMPv1_Oid –** sadrži informacije Objekt idetifikatora (OID)
- **SNMPv1_String_Object –** sadrži informacije o nizu objekata koji postoje u SNMP zahtevu
- **SNMPv1_Object_Position –** sadrži informacije u vezi sa pozicijom objekta u MIB bazi

Strukture podataka kao i odnos između njih prikazan je na slici ispod.

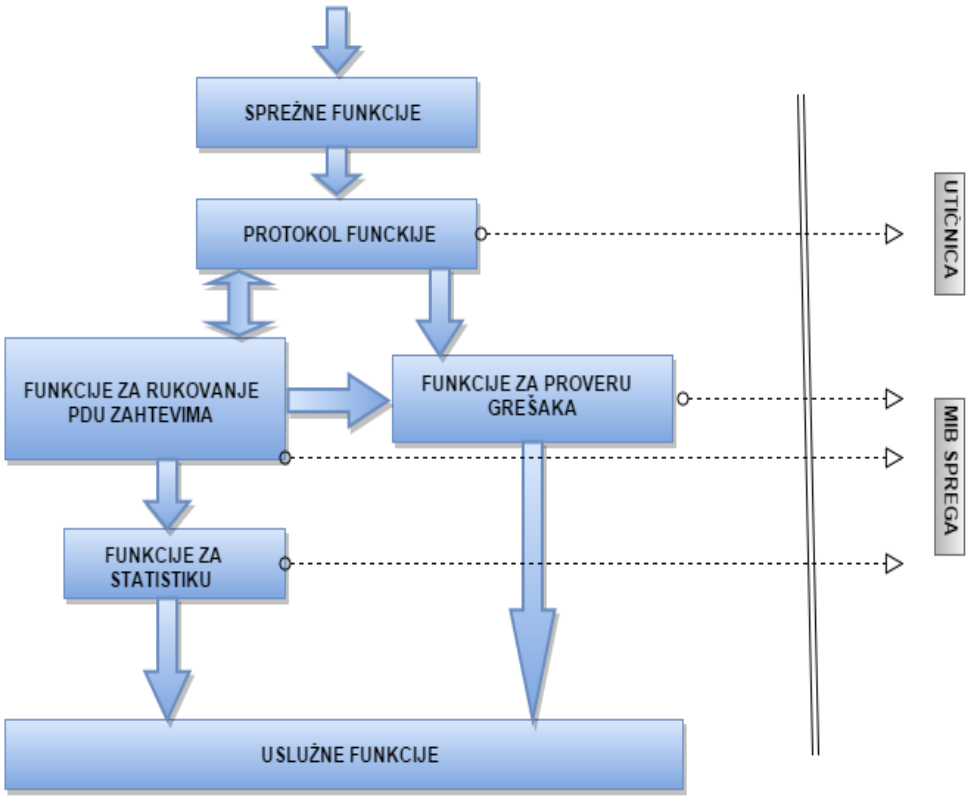


Slika 17 - Strukture podataka SNMP Agenta

4.2 Funkcije

Funkcije koje predstavljaju SNMP Agenta (SNMP modul i SNMP Trap modul) su grupisane na 6 logickih delova:

- Sprežne funkcije – ovaj deo sadrži funkcije koje su vidljive korisniku i služe kao sprega iz SNMP agenta
- Protokol funkcije – ovaj deo sadrži SNMP implementaciju protokola
- Funkcije za proveru grešaka – sadrže funkcije koje služe za proveru grešaka
- Funkcije za rukovanje PDU – sadrži sve definicije i funkcije vezane za SNMP zahteve rukovanja
- Funkcije za statistiku – ovaj deo sadrži sve funkcije koje se odnose na SNMP statistiku
- Uslužne funkcije – ovaj deo sadrži funkcije koje su za višekratnu upotrebu za različite logičke jedinice.



Slika 18 - Funkcije SNMP Agenta

4.2.1 Sprežne funkcije

- **snmp_init** - Ova funkcija inicijalizuje SNMP modul i ceo MIB modul. Funkcija je implementirana u SNMPv1_API.c modulu

- **snmp_handler** - Ova funkcija upravlja komunikacijom Menadžer (NMS) – SNMP Agent. Komunikacija se sastoji od prijema zahteva, njegovog dekodovanja i obavljanje odgovarajuće aktivnosti u zavisnosti od vrste zahteva (Get, Set ili Get-Next). Funkcija je implementirana u SNMPv1_API.c modulu
- **snmp_trap_init** - Ova funkcija inicijalizuje SNMP trap modul i sve Trap MIB module. Funkcija je implementirana u SNMPv1_Trap_API.c modulu
- **snmp_trap_handler** - Ova funkcija rukuje Trap porukom u Agentu. Za svaki Trap MIB objekat, Trap modul obavlja trap proveru. Ako se pojavio trap, Trap modul sastavlja Trap poruku i šalje je NMS stanici. Funkcija je implementirana u SNMPv1_Trap_API.c modulu.

4.2.2 Protokol funkcije

- **snmp_req_dec** - Ova funkcija dekodira SNMP zahtev. Svaki SNMP zahtev se čuva u prijemnom baferu. Funkcija je implementirana u SNMPv1_API.c modulu.
- **snmp_msg_enc** - Ova funkcija će sastaviti SNMP poruku odgovora koristeći informacije iz glavne SNMP strukture. Funkcija je implementirana u SNMPv1_API.c modulu.
- **snmp_trap_msg_enc** - Ova funkcija će sastaviti Trap poruku u Trap prenosnom baferu. Funkcija je implementirana u SNMPv1_Trap_API.c modulu.
- **snmp_err_enc** - Ova funkcija enkoduje poruku o grešci. Funkcija je implementirana u SNMPv1_API.c modulu.
- **snmp_send_response** - Ova funkcija šalje SNMP poruku odgovora. Funkcija je implementirana u SNMPv1_API.c modulu.
- **snmp_send_err_msg** - Ova funkcija šalje poruku greške SNMP-a. Funkcija je implementirana u SNMPv1_API.c modulu.
- **snmp_get_uint8_object** - Ova funkcija izdvaja uint8 podatke (Data) čuvane u BER formatu(Tip-Dužina-Podatak) iz prijemnog bafera. Adresa uint8 objekata će se čuvati u promenljivoj *obj*. Funkcija će pomerati pokazivač na sledeći objekat nakon dekodovanja postojećeg. Funkcija je implementirana u SNMPv1_i_Get.c modulu.
- **snmp_get_int32_object** - Ova funkcija izdvaja int32 podatke (Data) čuvane u BER formatu(Tip-Dužina-Podatak) iz prijemnog bafera. Adresa int32 objekata će se

čuvati u promenljivoj *obj*. Funkcija će pomerati pokazivač na sledeći objekat nakon dekodovanja postojećeg. Funkcija je implementirana u SNMPv1_i_Get.c modulu.

- **snmp_get_str_object** - Ova funkcija izdvaja string podatke (Data) čuvane u BER formatu(Tip-Dužina-Podatak) iz prijemnog bafera. Funkcija će pomerati pokazivač na sledeći objekat nakon dekodovanja postojećeg. Funkcija je implementirana u SNMPv1_i_Get.c modulu.
- **snmp_get_oid_obj** - Ova funkcija izdvaja OID objekte čuvane u BER formatu(Tip-Dužina-Podatak) iz prijemnog bafera. Funkcija će pomerati pokazivač na sledeći objekat nakon dekodovanja postojećeg. Funkcija je implementirana u SNMPv1_i_Get.c modulu.
- **snmp_get_value** - Ova funkcija izdvaja vrednost (Value) objekta čuvanog u BER formatu(Tip-Dužina-Podatak) iz prijemnog bafera. Funkcija će pomerati pokazivač na sledeći objekat nakon dekodovanja postojećeg. Funkcija je implementirana u SNMPv1_i_Get.c modulu.
- **snmp_get_varbind_list** - Ova funkcija izvlači VarbindList iz SNMP zahteva. Nakon uspešnog izvlačenja , funkcija vraća ukupan broj varbind-a iz VarbindList. Funkcija je implementirana u SNMPv1_i_Get.c modulu.
- **snmp_get_seq_length** - Ova funkcija izvlači SNMP dužinu sekvence kao uint16 objekt (lenght) čuvanog u BER formatu(Tip-Dužina-Podatak) iz prijemnog bafera. Funkcija će pomerati pokazivač na sledeći objekat nakon dekodovanja postojećeg. Funkcija je implementirana u SNMPv1_i_Get.c modulu.
- **snmp_get_pdu** - Ova funkcija izvlači SNMP PDU tip kao uint8 objekt (Type) čuvanog u BER formatu(Tip-Dužina-Podatak) iz prijemnog bafera. Funkcija će pomerati pokazivač na sledeći objekat nakon dekodovanja postojećeg. Funkcija je implementirana u SNMPv1_i_Get.c modulu.
- **snmp_get_len_handler** - Ova funkcija izdvaja dužinu SNMP poruke segmenta. Funkcija će pomerati pokazivač na sledeći objekat nakon dekodovanja postojećeg. Funkcija je implementirana u SNMPv1_i_Get.c modulu.
- **snmp_set_uint8_object** - Ova funkcija dodaje uint8 objekat (podatke) u BER formatu u poruci odgovora. Funkcija je implementirana u SNMPv1_i_Set.c
- **snmp_set_int32_object** - Ova funkcija dodaje int32 objekat (podatke) u BER formatu u poruci odgovora. Funkcija je implementirana u SNMPv1_i_Set.c
- **snmp_set_uint32_object** - Ova funkcija dodaje uint32 objekat (podatke) u BER formatu u poruci odgovora. Funkcija je implementirana u SNMPv1_i_Set.c

- **snmp_set_string_object** - Ova funkcija dodaje string objekat (podatke) u BER formatu u poruci odgovora. Funkcija je implementirana u SNMPv1_i_Set.c
- **snmp_set_ip_address** - Ova funkcija dodaje IP adresu(podatke) u BER formatu u poruci odgovora. Funkcija je implementirana u SNMPv1_i_Set.c
- **snmp_set_oid** - Ova funkcija dodaje OID objekat (podatke) u BER formatu u poruci odgovora. Funkcija je implementirana u SNMPv1_i_Set.c
- **snmp_set_value** - Ova funkcija dodaje vrednost objekat (Value) u BER formatu u poruci odgovora. Funkcija je implementirana u SNMPv1_i_Set.c
- **snmp_set_ber** - Ova funkcija dodaje dužinu i tip podatka u BER formatu na SNMP poruku. Funkcija je implementirana u SNMPv1_i_Set.c
- **snmp_set_len_handler** - Ova funkcija rukuje BER pravilom kodiranja za dužinu segmenta (sekvence) u SNMP poruci. Funkcija je implementirana u SNMPv1_i_Set.c

4.2.3 Funkcije za proveru grešaka

- **snmp_authenticate** – Ova funkcija proverava potvrdu identititeta SNMP zahteva. Funkcija vrši poređenje dobijenog stringa iz SNMP zahteva sa konfigurisanim stringom. Funkcija je implementirana u SNMPv1_API.c
- **snmp_get_rule_check** – Ova funkcija pronalazi i obavlja proveru SNMP pravila za svaki OID u VarbindList-i u jednom GET zahtevu. Ako ne pronađe OID u MIB bazi ili ne zadovoljava pravila odgovarajućih polja (status greške i indeks greške) utvrđuje se poruka o greški. Funkcija je implementirana u SNMPv1_API.c
- **snmp_set_rule_check** - Ova funkcija pronalazi i obavlja proveru SNMP pravila za svaki OID u VarbindList-i u jednom SET zahtevu. Ako ne pronađe OID u MIB bazi ili ne zadovoljava pravila odgovarajućih polja (status greške i indeks greške) utvrđuje se poruka o greški. Funkcija je implementirana u SNMPv1_API.c
- **snmp_get_next_rule_check** - Ova funkcija pronalazi i obavlja proveru SNMP pravila za svaki OID u VarbindList-i u jednom GET-NEXT zahtevu. Ako ne pronađe OID u MIB bazi ili ne zadovoljava pravila odgovarajućih polja (status greške i indeks greške) utvrđuje se poruka o greški. Funkcija je implementirana u SNMPv1_API.c
- **snmp_set_req_value_check** – Ova funkcija proverava da li je vrednost zahteva SET u prihvatljivim granicama utvrđenim minimalnim i maksimalnim vrednostima za određeni objekat (OID). U zavisnosti od vrste objekta različiti tipovi provere se

koriste. U slučaju greške sto odgovara statusu greške i indeksu greške, utvrđuje se poruka o greški. Funkcija je implementirana u SNMPv1_API.c

- **snmp_check_value_type** – Ova funkcija vrši proveru tipa vrednosti primljenom u SNMP zahtevu sa unapred definisanim SNMP tipovima. Funkcija je implementirana u SNMPv1_i_Get.c
- **snmp_check_value_length** – Ova funkcija vrši proveru dužine za vrednosti primljene SNMP zahtevom. Za svaku vrstu vrednosti, dužina vrednosti je unapred određena. Vrednost se definise sa *value_type* i *value_length*. Funkcija je implementirana u SNMPv1_i_Get.c

4.2.4 Funkcije za rukovanje PDU

- **snmp_req_handler** – Ova funkcija proverava vrstu zahteva i zavisno od vrste zahteva (SET, GET, GET-NEXT) poziva funkciju koja izvršava te operacije. Funkcija je implementirana u SNMPv1_API.c
- **snmp_handle_get_req** – Ova funkcija upravlja GET zahtevom. SNMP vrši proveru SNMP GET zahteva, pre nego što objekat vrednosti preuzet iz MIB baze. Poruka GET odgovor se sastavlja posle uspešnog preuzimanja vrednosti. Funkcija je implementirana u SNMPv1_API.c
- **snmp_handle_set_req** - Ova funkcija upravlja SET zahtevom. SNMP vrši proveru SNMP SET zahteva, pre nego što se objekat vrednosti smesti u MIB bazu. Poruka SET odgovor se sastavlja posle uspešnog postavljanja vrednosti. Funkcija je implementirana u SNMPv1_API.c
- **snmp_handle_get_next_req** - Ova funkcija upravlja GET-NEXT zahtevom. SNMP vrši proveru SNMP GET-NEXT zahteva, pre nego što objekat vrednosti preuzet iz MIB baze. Poruka GET odgovor se sastavlja posle uspešnog preuzimanja vrednosti. Funkcija je implementirana u SNMPv1_API.c
- **snmp_module_search** – Ova funkcija je funkcija pretrage. Ona pretražuju traženi OID u svim modulima (MIB bazi podataka) i čuva indeks modula u namenskoj strukturi. Funkcija vraća indeks objekta u zatečenom modulu. Funkcija je implementirana u SNMPv1_API.c
- **snmp_module_search_next** - Ova funkcija je funkcija pretrage. Ona pretražuju sledeći veći OID od traženog OID u svim modulima (MIB bazi podataka) i čuva

indeks modula u namenskoj strukturi. Funkcija vraća indeks objekta u zatečenom modulu. Funkcija je implementirana u SNMPv1_API.c

- **snmp_find_oid** – Ovo je funkcija pretrage za traženi OID unutar jednog MIB modula, koji je naveden ulaznim parametrom. Funkcija je implementirana u SNMPv1_API.c
- **snmp_find_next_oid** – Ova funkcija će tražiti sledeći veći OID unutar jednog MIB modula, na osnovu traženog OID-a. Modul je naveden ulaznim parametrom. Funkcija je implementirana u SNMPv1_API.c
- **snmp_common_handler** – Ova funkcija obavlja jednu od sledećih operacija u zavisnosti od PDU (zahtev) tipa: Prva operacija, pronalazi objekat vrednosti, stvara i sastavlja varbind u GET odgovoru. Druga operacija, postavlja objekat vrednosti , stvara i sastavlja varbind u GET odgovoru. Funkcija koristi objekte OID-a i njene vrednosti (preuzet ili izmenjen) za stvaranje varbind-a. Funkcija je implementirana u SNMPv1_API.c
- **snmp_get_next** – Funkcija pronalazi objekat vrednosti nađenog OID-a i stvara varbind iz OID vrednosti para. Funkcija je implementirana u SNMPv1_API.c
- **snmp_get_next_object** – Funkcija traži sledeći veći pristupačni OID unutar jednog MIB modula. Funkcija poziva get() ili get_next() funkcije u zavisnosti od vrste objekata i vraća poziciju objekta u modulu. Funkcija je implementirana u SNMPv1_API.c
- **snmp_go_to_next_oid** - Funkcija traži sledeći veći pristupačni OID u MIB bazi, počev od onog OID-a koji je izazvao "end of table" situaciju. Ako postoji sledeći dostupan objekat u MIB bazi podataka, funkcija preuzima vrednost objekta i vraća TRUE, u suprotnom vraća FALSE. Funkcija je implementirana u SNMPv1_API.c
- **snmp_get_tbl_idx** – Funkcija uzima OID indeks od primljenog OID-a ako postoji. Indeks je predstavljen kao niz elemenata. Funkcija je implementirana u SNMPv1_API.c
- **snmp_value_dec** – Ova funkcija dekodira objekat vrednosti iz SET zahteva na unapred definisanim globalnim promenljivima koje se koriste kao argument u određenim MIB set funkcijama. Funkcija je implementirana u SNMPv1_API.c
- **snmp_value_enc** – Ova funkcija sastavlja privremenu varbind vrednost u strukturi koja će kasnije biti kodirana u SNMP GET poruci odgovora. Funkcija je implementirana u SNMPv1_i_Helper.c
- **snmp_make_varbind** – Ova funkcija kreira varbind u prenosnom baferu ako ima dovoljno prostora. Funkcija je implementirana u SNMPv1_i_Helper.c

- **snmp_trap_check** – Ova funkcija vrši proveru statusa za sve TRAP objekte unutar jednog TRAP modula. Ako se prijavila zamka (Trap) funkcija će stvoriti TRAP poruku i poslati je unapred definisanom menadžeru. Funkcija je implementirana u SNMPv1_Trap_API.c
- **snmp_trap_create_msg** – Ova funkcija stvara TRAP poruku. Funkcija je implementirana u SNMPv1_Trap_API.c

4.2.5 Funkcije za statistiku

- **snmp_inc_snmp_cnt** – Ova fukncija povećava brojač iz MIB SNMP grupe. Funkcija je implementirana u SNMPv1_Stat.c
- **snmp_get_snmp_cnt** – Ova funkcija vraća vrednost MIB SNMP brojača, definisanog kao ulazni parametar funkcije. Funkcija je implementirana u SNMPv1_Stat.c
- **snmp_get_auth_trap_permit** – Ova funkcija će pokazati da li je SNMP Agent proces dozvoljen da generiše autentifikaciju-otkaz zamku. Funkcija je implementirana u SNMPv1_Stat.c
- **snmpv1_set_auth_trap_permit** – Ova funkcija omogućava/onemogućava stvaranje autentifikaciju-otkaz zamke od strane SNMP Agenta. Funkcija je implementirana u SNMPv1_Stat.c

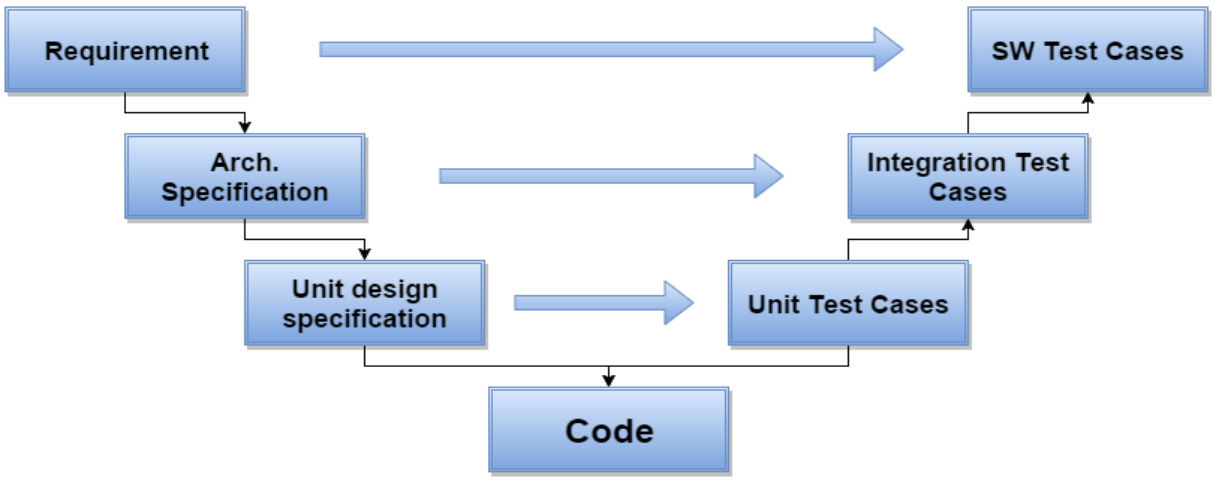
4.2.6 Uslužne funkcije

- **snmp_ber_dec** – Ova funkcija vrši dekodiranje bajova OID segmenta koji predstavlja ceo broj veći od 127. Koristi pravila BER kodiranja. Funkcija je implementirana u SNMPv1_i_Helper.c
- **snmp_ber_enc** - Ova funkcija vrši kodiranje bajova OID segmenta koji predstavlja ceo broj veći od 127. Koristi pravila BER kodiranja. Funkcija je implementirana u SNMPv1_i_Helper.c
- **snmp_i2b –** Ova funkcija će pretvoriti celobrojni orijentisan OID iz MIB baze u bajt orijentisan OID (OID niz okteta). Funkcija je implementirana u SNMPv1_i_Helper.c
- **snmp_b2i** – Ova funkcija konvertuje primljeni OID iz mreže u obliku bajta (OID niz okteta) na celobrojni oblik. Funkcija je implementirana u SNMPv1_i_Helper.c
- **snmp_get_time** – Funkcija pronalazi trenutno vreme sistema u formatu stotog dela sekunde. Funkcija je implementirana u SNMPv1_Trap_API.c modulu.

- **snmp_int_convert_host** – Ova funkcija konvertuje niz bajtova (maksimalno 4 bajta) iz mrežnog uređaja u bajtove domaćina. Funkcija je implementirana u SNMPv1_i_Get.c modulu.
- **snmp_int_convert_network –** Ova funkcija konvertuje niz bajtova domaćina (maksimalno 4 bajta) u celobrojnu vrednost bajtova mreže. Funkcija je implementirana u SNMPv1_i_Set.c modulu.

5. Ispitivanje i verifikacija

Modul koji je implementiran (SNMP Agent) je u potpunosti razvijen u skladu sa V modelom razvoja sistema. Analiza zahteva je detaljno opisana kroz specifikaciju zahteva sistema i specifikaciju dizajna sistema, koji su bile polazne tačke pri razvoju. Provera ispunjenosti ovih zahteva je izvršena na način koji je definisan u sistemsko-integracijskoj test specifikaciji i kao rezultat je dobijen odgovarajući test izveštaj. Projektovanje sistema je sledeća faza razvoja i njeni zahtevi su definisani u specifikaciji zahteva softvera dok je rezultat sistemskog testiranja koji je pokrio specifikaciju opisan u softver test izveštaju.[4] Zatim se prelazi na projektovanje programa u skladu sa dokumentom koji sadrži softversku arhitekturu, a nakon toga se kroz integraciono testiranje opisano u integracionoj test specifikaciji potvrđuje ispravnost date operacije. I na samom kraju se vrši kodiranje na osnovu dizajn dokumenta, izvršava testiranje svake jedinice, a rezultat toka testiranja se dokumentuje u posebnu *unit-test* specifikaciju.



Slika 19 - Različiti nivoi zahteva i verifikacija

5.1 Jedinično testiranje SNMP Agenta

Jedinično testiranje predstavlja fazu testiranja nekog softvera u kojem se svaka programska komponenta testira nezavisno od ostatka sistema. Ovaj tip testiranja se svodi na proveru funkcionalnosti neke komponente sistema za unapred zadati skup ulaznih podataka koji treba da pokrije sve situacije. [12] Testiranjem se proverava da li se za skup datih ulaznih vrednosti podataka dobijaju očekivani rezultati na izlazu sistema ili se uspešno izvršavaju očekivane akcije.

Za jedinično testiranje ovog modula kao testna platforma korišćen je ACT (*eng. Automatic Certification Test*). Ova platforma služi za izvršavanje jediničnih testova na računaru uz prikaz izvršenja testova u komandnom prozoru kao i upis detaljnih informacija o izvršenom testu u log datoteke. Testovi se mogu izvršavati pojedinačno, ali se može pokrenuti i grupa testova, u zavisnosti od potrebe. Testovi se mogu simulirati na računaru, ali po potrebi uz određena podešavanja i opremu moguće ih je izvršiti i na ciljnoj platformi (korišćen je TMS570).

Testovi se sastoje iz niza pajton skripti koji se izvršavaju sekvencijalno. Skripte opisuju sve akcije neophodne za konfigurisanje testa, preuzimanje rezultata kao i proveru pass/fail kriterijuma. U testnoj skripti se definišu funkcije koje poziva testirana funkcija. Njima je moguće zadati vrednosti koju će zadata funkcija vratiti kako bi se na taj način uticalo na dalje izvršavanje programa. Takođe je moguće menjati vrednosti lokalnim promenljivim i lokalnim strukturama sa ciljem određivanja toka izvršavanja programa.

Ovim načinom testiranja preko skripte moguće je proveriti:

- Povratnu vrednost testirane funkcije
- Da li je u toku izvršavanja funkcije pozvana ili nije pozvana neka druga funkcija
- Parametre pozvanih funkcija
- Da li je u toku izvršavanja funkcije došlo do promene vrednosti neke lokalne promenljive

Prilikom testiranja modula SNMP Agent kreirano je ukupno 281 testnu varijatnu, koje su proveravale ispravnost celokupnog modula (svih funkcija). Svi testovi su uspešno izvršeni (PASS). Pokrivenost koda testovima je verifikovana *Cantata* alatom za proveru pokrivenosti. [13] Pokrivenost koda testovima je 100%. Postoje tri vrste testiranja sa aspekta nivoa testiranja jediničnim testovima: testiranje na niskom nivou (kada se testira samo jedna funkcija), testiranje na visokom nivou (kada se odjednom testira više funkcija kako bi se testirao odnos između tih funkcija) i testiranje performansi (merenje najdužeg vremena izvršenja funkcije). Postoje dve vrste testiranja sa aspekta svrhe testiranja

jediničnim testovima: testiranje robusnosti (testiranje reakcije funkcija na nedozvoljene vrednosti) i regularno testiranje (testiranje ponašanja sa dozvoljenim vrednostima).

```
from names import *

g = global_data
a = addresses

tc = Test_Case \
    ( number      = 1
    , uut         = "snmp_init"
    , description = "Test unsuccessful allocation."
    , setup       = [ Function ("mmu_reserve", ["NULL", "NULL"
                                                , "NULL", "NULL"])
                    , Function ("snmp_set_gen_trap_flag", [])
                    ]
    , call        = Call ( "snmp_init", [ a.cfg ()
                                        , a.mib_cfg ()
                                        ]
                         )
    , exit        = [ g.snmp.cfg()
                    , a.cfg ()
                    , g.snmp.mib_cfg()
                    , a.mib_cfg ()
                    ]
    , verbose = SHOW_DATA
    , criteria    = [ check_for_variable ( "exit [0]", "exit [1]" )
                    , check_for_variable ( "exit [2]", "exit [3]" )
                    , check_return_value ( "snmp_init"
                                         , "SNMP_RET_ERR_NOMEM"
                                         )
                    , check_for_function_parameters (
                                          "snmp_set_gen_trap_flag"
                                          , "[ SET_GEN_TRAP, \
                                              SNMP_COLD_START_TRAP ]"
                                          )
                    ]
    )
```

Slika 20 - Primer testne skripte za testiranje SNMP Agenta

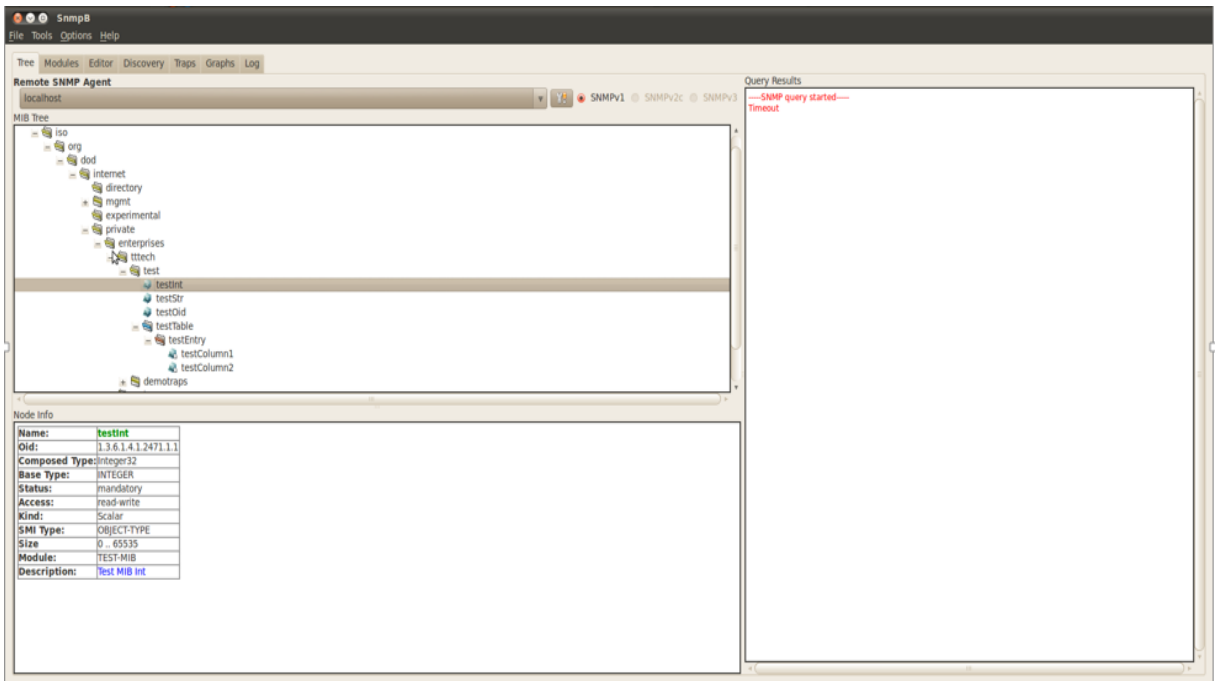
5.2 Integraciono testiranje SNMP Agenta

Nakon završenog jediničnog testiranja kreće se sa povezivanjem modula u jednu celinu i formira se sistem namenjen korisniku. Taj proces se naziva integracija, i ona se vrši dodavanjem komponenata za koje je utvrđeno da ispravno funkcionišu. [15] Iako smo ispravnost individualnih komponenti utvrdili kroz jedinične testove to nam ne garantuje ispravan rad komponenti kada se one nađu u sprezi sa drugim komponentama.

Integracioni testovi za SNMP Agent modul služe za proveru funkcionalnosti SNMP modula i SNMP Trap modula. Ovi testovi treba da obezbede konkretnu primenu funkcija, uključujući procenu stabilnosti softvera i ograničenja. Svaki test slučaj uključuje:

- Definiciju testa
- Proceduru i konfiguraciju koja je potrebna da bi se test izvršio
- PASS/FAIL kriterijume

Za testiranje SNMP Agenta potrebna su nam tri softverska paketa, SNMP Agent, SNMPB Manager i pajton skripta udp_client. SNMP Agent je implementiran modul koji se pokreće na konkretnoj ciljnoj platformi. SNMPB Manager je softver koji radi na računaru. Postoje dve verzije SNMPB Manager-a u zavisnosti od operativnog sistema (Linux i Windows). Pajton skripta udp_client se koristi za neke testove kada SNMPB Manager ne može pomoći. Skripta radi na Linux operativnom sistemu i za proveru rezultata koristi se Wireshark.



Slika 21 –SNMPB Manager

Prvi tip test slučajeva je vezan za GET, SET, GET-NEXT i WALK zahteve, drugi tip test slučajeva se bavi problemima zaštite. Poslednji tip testova pokazuje slučaj gde se premašuje maksimalan Varbind broj (maksimalan Varbind broj je postavljen na 10). Svi testni slučajevi su dati u tabelama , a tabela se sastoji iz sledećih polja:

- **ID testa** (GR – Get_Request test, GNR – Get_Next_Request test, SR – Set_Request test, GRT – Get_Request Table test, GNRT – Get_Next_Request Table test itd)
- **Ime testa**
- **Opis** – objašnjenje šta test predstavlja
- **Konfiguracija** – najvažniji deo tabele. U suštini, svi detalji konfiguracije ovde su dati.
- **Identifikator objekta** – OID na kojem će se obavljati akcija
- **Ime objekta**
- **Vrednost objekta** – početna promenljiva vrednost koja će se postaviti u Agentu
- **Vrednost testa** – vrednost testa za Set_Request
- **Akcija –** opis kako da se izvrši test
- **PASS kriterijum** – očekivana reakcija Manager-a

Primeri testa u kojem se koristi SNMPB Manager, kao i kojima se koristi pajton skripta i Wireshark prikazani su u tabelama ispod.

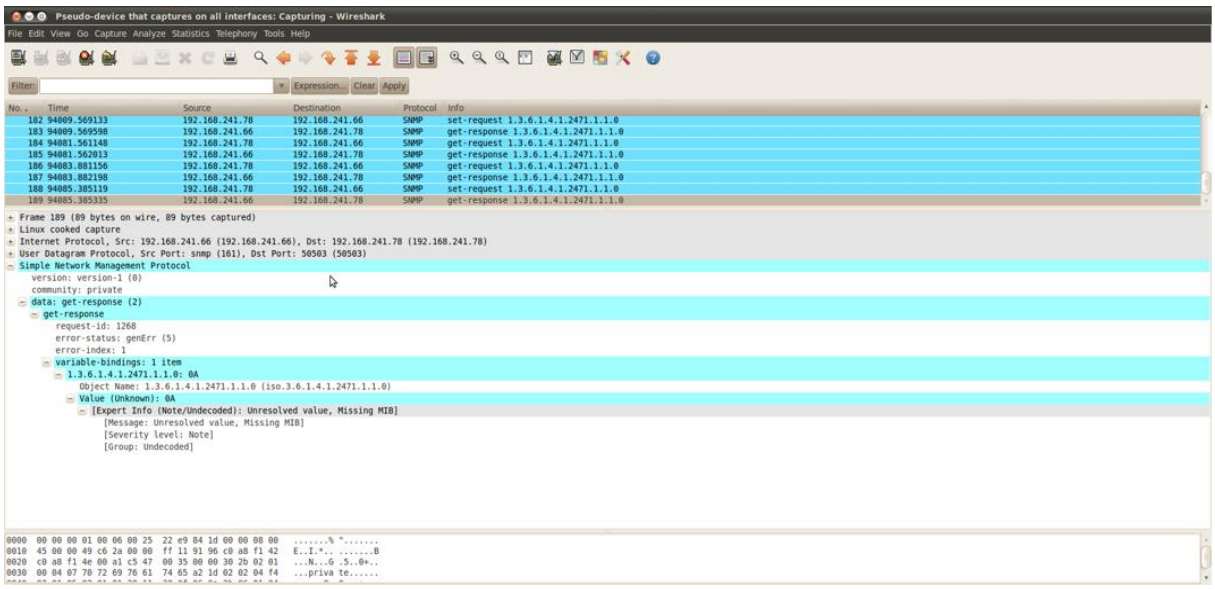
ID Testa	SR001
Ime Testa	Set zahtev, sve ispravno
Opis	Ovaj test pokazuje ponašanje izvođenja Set zahteva. Objekat je dostupan za postavljanje vrednosti i test vrednost je u opsegu. Rezultat testa: Manager bi trebao da dobije poruku sa novom celobrojnom vrednošću.
Konfiguracija	**Module access**: snmp_mib_module access level – MIB_ACC_RW, test_ mib_module access level – MIB_ACC_RW. **Object access**: testInt access levell – MIB_ACC_RW
Identifikator objekta	1.3.6.1.4.2471.1.1
Ime objekta	testInt
Vrednost objekta	testInt = 100
Vrednost testa	50
Akcija	Izvršavanje SET_REQUEST na testInt sa ovom konfiguracijom i sa vrednošću testa
PASS kriterijum	Manager ispisuje: 1:testInt.0 50

Tabela 3 – Primer 1 integracionog testiranja

ID Testa	VAL001
Ime Testa	Loš tip vrednosti
Opis	Ovaj test pokazuje ponašanje obavljanja Set_Request zahteva sa pogrešnim tipom vrednosti. Svi objekti su za upis ali će Agent odbiti analizu paketa zbog nepodržanog tipa vrednosti. Ovaj test se vrši pomoću pajton skripte – udp_client.py. Preuzeti rezultati se mogu videti preko Wireshark-a. Rezultat testa: Wireshark će uhvatiti SNMP paket sa greškom statusa : genErr

Konfiguracija	**Module access**: snmp_mib_module access level – MIB_ACC_RW, test_ mib_module access level – MIB_ACC_RW. **Object access**: testInt access levell – MIB_ACC_RW
Identifikator objekta	1.3.6.1.4.2471.1.1
Ime objekta	testInt
Vrednost objekta	testInt = 100
Akcija	Otvoriti skriptu udp_client.py u tekst editoru i zameniti UDP_IP adresu sa IP adresom ciljne platforme gde agent radi. Pokrenuti skriptu iz komandnog prozora u Linux-u: python udp_client.py i zatim pritisnuti bilo koji taster i zatim enter (prvi pritisnut taster pošalje paket, ENTER izlaz)
PASS kriterijum	Pošto se u ovom testu ne koristi SNMPB Manager, najjedostavniji način da se proveri preuzeta poruka je snimanje preko Wireshark

Tabela 4 – Primer 2 integracionog testiranja



Slika 22 - Snimanje poruka preko Wireshark-a

Izvršeno je ukupno 40 testnih slučajeva i svaki test je prošao očekivano.

Vrsta testa	**Broj testova**	**Rezultat**

Get_Request	6	☑
Set_Request	6	☑
Get_Next_Request	3	☑
Walk	2	☑
Get_Request Error	8	☑
Set_Request Error	3	☑
Get_Next_Request Error	5	☑
TRAP enabling/disabling	2	☑
Authentification Failure (get_request and set_request)	2	☑
Get_Request maximum varbinds number	1	☑
Bad SNMP Version	1	☑
Bad Value type	1	☑

Tabela 5 – Rezultati integracionog testiranja

5.3 Sistemsko testiranje SNMP Agenta

Kada se spremni i testirani podsistemi sjedine u finalni sistem, moguće je izvršavati sistemske testove. Oni uključuju i funkcionalno ponašanje sistema i nefunkcionalne osobine sistema (brzina, efikasnost, povezanost sa drugim modulima). Cilj je utvrditi da li se sistem ponaša korektno kada se sve komponente koriste zajedno. [14] Spoljašnja sprega ka drugim modulima, pomoćnim programima, ciljnoj platformi ili operativnim okruženjima se takođe razmatra na ovom nivou. Sistemsko testiranje se sastoji od sledećih koraka:

- Funkcionalno testiranje - Analizira se samo izvršavanje specificiranih funkcija i vrši se provera ulaznih i izlaznih podataka. Tačnost izlaznih podataka proverava se na osnovu specifikacije zahteva za softver. U ovim testovima se ne vrši analiza

izvornog koda. Problem funkcionalnog testiranja može da se pojavi u slučaju dvosmislenih zahteva i nemogućnosti opisivanja svih načina korišćenja softvera.

- Testiranje performansi - obuhvata pronalaženje i otklanjanje problema koji degradiraju performanse softvera. Ispitivanje performansi najčešće obuhvata: iskorišćenje procesorskih resursa, protok podataka i vreme odziva. Tipični resursi koji se proveravaju su: propusni opseg, brzina procesora, iskorišćenje memorije, zauzetost prostora na disku.
- Testiranje prihvatljivosti – cilj ovog testiranja je provera da li sistem ispunjava zahteve i očekivanja naručioca.
- Instalaciono testiranje – poslednja faza sistemskog testiranja, ima zadatak da pokaže ispravnost sistema u okruženju u kojem zaista sistem i radi.

Sistemsko testiranje SNMP Agenta je izvršeno na ciljnoj platformi TMS570 koju smo već pominjali. Testiranja koja su izvršena na nivou celog sistema, a koja su vezana konkretno za ovaj modul navedena su u tabeli. Izvršeno je ukupno 34 testa i svaki test je prošao očekivano. Testovi su podeljeni u 3 podgrupe:

- SNMP modul
- SNMP Trap modul
- MIB

Vrsta testa	**Broj testova**	**Rezultat**
SNMP - General	1	☑
TRAP – Standard Operation	3	☑
MIB - Integration	2	☑
MIB – Read/Write Access Level	3	☑
MIB – Read Only Access Level	3	☑
MIB – Write Only Access Level	3	☑
MIB – Not Accessable Access Level	3	☑
MIB – Robustness Test Cases	4	☑

Ethernet End System MIB	6	☑
Ethernet Switch MIB	6	☑

Tabela 6 – Rezultati sistemskog testiranja

6. Zaključak

U ovom radu je predstavljeno jedno rešenje realizacije softvera za mrežno upravljanje putem SNMP protokola na sistemu zasnovanom na Texas Instruments TMS570 hardverskoj platformi. Programsko rešenje je uspešno integrisano u Ethernet platformu, a pouzdanost sistema je verifikovana testiranjem nizom unit, integracionih i sistemskih testova. Realizovan je SNMP Agent koji se sastoji iz dva modula:

- SNMP modul – predstavlja SNMP Agenta, obezbeđuje spregu ka svim opcijama konfiguracije koji se čuvaju u MIB bazi, ima lokalno znanje o upravljačkim informacijama i prevodi ih (dekoduje) u oblik kompatibilan sa SNMP protokolom.
- SNMP Trap modul – služi za upravljanje trap porukama

Realizovani softver SNMP Agent je deo ugrađene Ethernet platforme koja se koristi u industrijskim postrojanjima u cilju automatizacije i kontrole. Ovaj modul je veoma bitan u ovom sistemu jer odgovara na zahteve od menadžera u ovom slučaju (Multisocket Manager) i snabdeva ga podacima. Takođe je zadužen za kontrolu lokalnog MIB-a. SNMP Agent koristi odgovarajuće MIB-ove za prikupljanje podataka o mrežnim uređajima, u ovom sistemu konkretno za Ethernet Switch i Endsystem drajver.

Pošto se ovaj sistem koristi u industrijskim postrojenjima potrebno je detaljno testirati funkcionalnost svakog modula posebno, ali takođe i ceo sistem. SNMP Agent je implementiran i testiran korišćenjem V modela razvoja. Testiranje je izvršeno nizom jediničnih testova čime se ispitala ispravnost svake funkcije koja je implementirana, integracionim testiranjem čime je proverena funkcionalnost SNMP Modula i SNMP Trap modula kao i njihova integracija. Na samom kraju kada je SNMP Agent ugrađen u ceo sistem ispitana je funkcionalnost celog sistema, nizom sistemskih testova.

7. Literatura

[1] Network Management, MIBs and MPLS: Principles, Design and Implementation, Addison Wesley, 2003.

[2] Mario Essert: Using V model for testing, https://insights.sei.cmu.edu/sei_blog/2013/11/using-v-models-for-testing.html učitano 06.01.2016.

[3] Deuter, A., "Slicing the V-Model -- Reduced Effort, Higher Flexibility," in *Global Software Engineering (ICGSE), 2013 IEEE 8th International Conference on* , vol., no., pp.1-10, 26-29 Aug. 2013

[4] Liu Shuping; Pang Ling, "The Research of V Model in Testing Embedded Software," in *Computer Science and Information Technology, 2008. ICCSIT '08. International Conference on* , vol., no., pp.463-466, Aug. 29 2008-Sept. 2 2008

[5] D.Bruey: SNMP: *Simple Network Management Protocol*, 2005.

[6] CERT: SNMP FAQ http://www.cert.org/tech_tips/snmp_faq.html učitano 08.01.2016

[7] TECHNET : How SNMP Works https://technet.microsoft.com/en-us/library/cc783142(v=ws.10).aspx učitano 08.01.2016

[8] Mladen Veinović, Aleksandar Jevremović, *Računarske mreže*, Beograd 2011

[9] A. Bažant, G. Gledec, Ž. Ilić, G. Ježić, M. Kos, M. Kunštić, I. Lovrek, M. Matijašević, B. Mikac, V. Sinković, „*Osnovne arhitekture mreža*“, Zagreb, Element, 2004.

[10] Glavač M., “*Upravljanje mrežom putem SNMP-a*”, Seminar, ZZT-FER, Zagreb, 2001

[11] TMS570LS0x32 16- and 32-Bit RISC Flash Microcontroller Datasheet, Texas Instruments, June 2015

[12] Ellims, M.; Bridges, J.; Ince, D.C., "Unit testing in practice," in *Software Reliability Engineering, 2004. ISSRE 2004. 15th International Symposium on* , vol., no., pp.3-13, 2-5 Nov. 2004

[13] Cantata, http://www.qa-systems.com/cantata.html, učitano 03.10.2015.

[14] Karmore, S.P.; Mabajan, A.R., "Universal methodology for embedded system testing," in *Computer Science & Education (ICCSE), 2013 8th International Conference on* , vol., no., pp.567-572, 26-28 April 2013

[15] Brar, H.K.; Kaur, P.J., "Differentiating Integration Testing and unit testing," in *Computing for Sustainable Global Development (INDIACom), 2015 2nd International Conference on* , vol., no., pp.796-798, 11-13 March 2015