37. Саветовање CIGRE Србија 2025
СИГУРНОСТ, СТАБИЛНОСТ, ПОУЗДАНОСТ И RESILIENCE ЕЛЕКТРОЕНЕРГЕТСКОГ СИСТЕМА
МУЛТИСЕКТОРСКО ПОВЕЗИВАЊЕ У ЕНЕРГЕТИЦИ И ПРИВРЕДИ
26 - 30. мај 2025, Копаоник

DOI: 10.46793/CIGRE37.B5.19

B5.19

PRIMENA SOFTVERA OTVORENOG KODA U INŽENJERINGU I TESTIRANJU IEC 61850 SISTEMA U ELEKTROENERGETSKIM OBJEKTIMA

APPLICATION OF OPEN-SOURCE SOFTWARE FOR ENGINEERING AND TESTING OF IEC 61850 SYSTEMS IN POWER TRANSMISSION AND DISTRIBUTION

Vladan Cvejić*

Kratak sadržaj: Savremeni sistemi relejne zaštite i upravljanja u elektroenergetskim objektima predstavljaju složene softversko-hardverske sisteme specijalizovane za rad u realnom vremenu. Njih karakteriše integracija velikog broja komunikacionih protokola i uređaja, pri čemu dominira primena koncepata definisanih serijom standarda IEC 61850, kao temelja za izgradnju globalne pametne mreže (Smart Grid). Tokom svih faza razvoja i integracije ovakvih sistema, inženjeri se suočavaju s kompleksnošću implementiranih tehnologija i multidisciplinarnim pristupom. Zbog toga je ključno olakšati njihov rad, bilo da je reč o inženjeringu, testiranju tokom izgradnje (FAT i SAT) ili održavanju. Na tržištu postoje specijalizovani alati koji mogu značajno pomoći, ali su često skupi i usko orijentisani. Alternativno rešenje je razvoj prilagođenih softverskih alata koji automatizuju inženjerske procese i olakšavaju njihov rad. S obzirom na to da su resursi IEC 61850 sistema u osnovi dizajnirani za mašinsku obradu, njihova ručna interpretacija nije jednostavna. Zbog toga skripte i pomoćni softverski alati mogu olakšati analizu, vizualizaciju podataka i identifikaciju ključnih informacija u velikim skupovima podataka, optimizujući ali i ubrzavajući inženjerski rad. Razvoj naprednih programskih jezika i softverskih biblioteka otvorenog koda (open-source) u poslednjim godinama omogućio je brzo i efikasno kreiranje prilagođenih alata za inženjere. Ovaj rad analizira dostupne biblioteke softvera otvorenog koda, prikazuje njihove primene i razmatra prednosti, ali i potencijalne rizike i ograničenja koja ovakva rešenja mogu doneti.

Ključne reči: *IEC61850, Testiranje, Softver otvorenog koda, Inženjering, Lokalne mreže*

Abstract: Modern relay protection and control systems in power utility objects represent complex software and hardware systems specialized for real-time operation. They are characterized by the integration of a large number of communication protocols and devices, with a predominant application of concepts defined by the IEC 61850 standard series, serving as the foundation for building a global Smart Grid. Throughout all phases of the development and integration of such systems, engineers face the complexity of implemented technologies and a multidisciplinary approach. Therefore, it is essential to facilitate their work, whether it involves engineering, testing during construction (FAT and SAT), or maintenance.

* Vladan Cvejić, ENSACO Solutions, vladan.cvejic@itdepends61850.com

There are specialized tools available on the market that can significantly assist, but they are often expensive and narrowly focused. An alternative solution is the development of custom software tools that automate engineering processes and streamline workflow. Since IEC 61850 system resources are fundamentally designed for machine processing, their interpretation is not straightforward for manual analysis. As a result, scripts and helper software tools can facilitate data analysis, visualization, and the identification of key information within large datasets, optimizing and accelerating engineering tasks. The rapid advancement of high-level programming languages and open-source software libraries in recent years has enabled the swift and efficient development of tailored tools for engineers. This paper analyzes available open-source libraries, showcases their applications, and discusses their benefits, as well as potential risks and limitations that such solutions may impose.

Key words: *IEC61850, Testing, Open Source, Engineering, LAN*

1 UVOD

Osnovna funkcija jednog sistema relejne zaštite i upravljanja (RZU sistem) u trafostanici jeste brzo i selektivno isključenje kvarova, osiguravanje sigurnog rada mreže, omogućavanje daljinskog upravljanja i kontinuiranog nadzora sistema, čime se povećava pouzdanost elektroenergetskog sistema i smanjuju gubici usled kvarova i ispada. Kao takav, on prati primarne elemente sistema od samog početka njihove primene u prenosu električne energije. Unapređenja koja su, u takvim sistemima vremenom uvođena, su posledica naučnog i tehnološkog razvoja. Ubrzani tehnološki razvoj poslednjih decenija je pomerio granice u većini industrija. Tehnologije iz pojedinih domena su našle primenu u drugim, time omogućavajući dodatne funkcionalnosti. Tako, moderna rešenja sistema automatizacije i relejne zaštite predstavljaju zbir različitih tehnologija iz raznih domena. Prodor telekomunikacionih i informatičkih tehnologija u ove sisteme proširuje skup aktivnosti koje se moraju planirati i sprovoditi, u svim navedenim koracima izgradnje i održavanja sistema automatizacije elemenata elektroenergetskog sistema. Aktivnosti inženjera relejne zaštite, automatike i lokalnog upravljanja, čak ako se i limitiramo na osnovne funkcije sistema, podrazumevaju da on poseduje znanje iz raznih oblasti elektroenergetike, automatike i šire. Sa modernizacijom koja je uvela telekomunikacione i informatičke koncepte, ti isti inženjeri su primorani da te koncepte prate i razumeju, ako ne u potpunosti, ono barem do tačke da mogu da zaključe da je problem u tom segmentu.

Standard IEC 61850 je preuzeo dužnost da modeluje sve relevantne elemente pametne mreže i da stavi na raspolaganje mehanizme koji mogu da omoguće pouzdanu i brzu razmenu informacija. On u sebi sadrži veliki skup industrijskih komunikacionih protokola i koncept objektno-orijentisanog modelovanja opšte prihvaćen u informatičkoj industriji. A sve u cilju da se kompleksni zahtevi pametne mreže smeste u okruženje koje ih može optimalno ispunjavati. To su, naravno, složeni računarski sistemi sastavljeni iz zaštitno-upravljačkih uređaja, komunikacionih uređaja (ethernet svičevi, ruteri, …), servera za sinhronizaciju, servera za prikupljanje i čuvanje podataka, automatizaciju, komunikaciju, vizuelizaciju, i tako dalje.

Takav računarski sistem u svojoj biti i dalje sadrži osnovne funkcije sistema relejne zaštite i upravljanja u trafostanici, ali „obložene“ i „pretvorene“ u vidove informacija koji nisu lako razumljivi i proverljivi za jednog inženjera energetike.

Stoga se, uporedo sa usvajanjem koncepata standarda IEC 61850 u proizvodima, industrija trudila da obezbedi i sve neophodne alate za rad u jednom takvom okruženju.

Razvoj istih je uslovljen ne samo zahtevima različitih grupa korisnika već i poslovno-finansijskim predispozicijama u datom momentu. Tada i proizvodi koji se nude, mogu da reše tekuće probleme inženjera sa različitim uspehom. Ako se tome doda i cena koja neće biti popularna upravo iz tehnološki složenog procesa razvoja, dolazimo do niše samostalnog razvoja pomoćnih alata. Doseg i širina ovog segmenta nije uslovljena finansijskim potrebama korisnika da stekne direktnu novčanu dobit tim razvojem, već njegovom rešenošću da iznađe rešenje za svoje zadatke i probleme. Segment proizvodnje softvera je doživeo bum u više smerova. Kako u komercijalnom tako i u smeru softvera otvorenog koda koji je dostupan širokom krugu korisnika za dalje korišćenje i unapređenje.

U ovom radu je dat osvrt na potencijal samostalnog razvoja pomoćnih alata koji se mogu iskoristiti u skoro svakom segmentu životnog ciklusa RZU sistema – inženjeringu, testiranju, dijagnostici, analizi. To mogu biti male skripte za automatizaciju ili veći programi koji mogu imati i grafički interfejs, a sve shodno potrebi onog koji ga je razvio a nije softverski inženjer, već inženjera-ispitivač ili na primer inženjer-projektant.

2 SOFTVERSKE BIBLIOTEKE I ALATI OTVORENOG KODA

U ovom poglavlju će biti razmotrene softverske biblioteke i alati otvorenog koda koje je autor koristio u razvoju svojih pomoćnih alata.

Prema iskustvu autora - koji dolazi iz oblasti elektroenergetike i poseduje znanje programiranja na amaterskom nivou – u toku praktičnog rada iskazala se potreba za jednostavnim programerskim alatima a koji imaju sličnost sa Matlab okruženjem, u kojem su inženjeri najčešće radili tokom studija, poput Python programskog jezika, koji je značajno finansijski pristupačniji i dugoročno održiviji za svakodnevni inženjerski rad.

Python se pokazao kao izuzetno pogodan jezik za inženjere, pre svega zato što omogućava visok nivo produktivnosti uz minimalno vreme potrebno za učenje osnovnih koncepata. Dodatno, Python ekosistem obuhvata veliki broj specijalizovanih biblioteka - kako za naučne i inženjerske primene, tako i za obradu podataka, automatizaciju, simulacije i vizuelizaciju - što korisnicima omogućava brzu izradu kompleksnih alata i analiza.

Značajnu prednost predstavlja i izuzetno aktivna i brojna korisnička zajednica, koja obezbeđuje obilje dostupnih resursa, tutorijala, primera softverskog koda i foruma za rešavanje problema, čime se olakšava ulazak u svet programiranja čak i onima bez formalnog obrazovanja iz te oblasti.

2.1 Softverske biblioteke programskog jezika Python

Prilikom izbora softverske biblioteke za razvoj korisničkih softverskih alata, neophodno je razmotriti sledeće aspekte:

- “Zrelost” biblioteke – Koliko dugo biblioteka postoji i koliko je aktivna zajednica ili razvojni tim koji je održava? Da li se redovno ažurira (npr. ispravke grešaka, dodavanje novih funkcionalnosti)? Stabilna i dobro održavana biblioteka je poželjnija, naročito u profesionalnim okruženjima.

- Dokumentacije – Da li je biblioteka dobro dokumentovana? Kvalitetna dokumentacija, zajedno sa konkretnim primerima upotrebe, značajno olakšava učenje i implementaciju.

- Licenca korišćenja – Da li je biblioteka slobodna za korišćenje u komercijalnim i/ili nekomercijalnim projektima? Treba proveriti da li licenca (MIT, BSD, GPL itd.) odgovara planiranoj upotrebi da bi se izbegli potencijalni pravni problemi.

- Potrebne funkcionalnosti – Koje konkretne funkcionalnosti su potrebne korisniku? Da li ih je moguće samostalno razviti, ili bi bilo efikasnije koristiti postojeću biblioteku otvorenog koda koja ih već nudi?

Načelno, ukoliko se koriste biblioteke sa MIT, BSD ili GPL licencom u svojim skripting aplikacijama samo za ličnu upotrebu i bez distribucije – ne krši se nijedna licenca. Licencni zahtevi se odnose samo na javno deljenje ili prodaju aplikacije. MIT i BSD dozvoljavaju da se koristi, menja i uključuje njihov kod bez ikakvih obaveza, čak i ako želimo da ga distribuiramo. Nema ograničenja ni u komercijalnoj upotrebi. GPL licenca zahteva da se izvedeni kod distribuira pod GPL-om ali opet, samo ako ga distribuirate. Dodatno, moguće je koristite Python skripte/programe sa MIT, BSD, GPL bibliotekom za automatizaciju posla i naplaćivanje usluge.

U svakom slučaju, pre nego što se odabere određena biblioteka, preporučuje se detaljno istraživanje i poređenje više opcija, uzimajući u obzir navedene kriterijume.

2.1.1 Pregled korisnih Python biblioteka

GenerateDS (MIT licenca) - Biblioteka za generisanje Python struktura podataka (npr. klasa) na osnovu XML šeme. Omogućava:

- Parsiranje XML fajlova u generisane objekte pogodne za dalju obradu,

- Generisanje tzv. *stub* fajlova (definišu strukturu programa bez implementacije logike) sa podklasama koje se mogu proširiti sa korisničkim metode za dalji rad sa podacima.

Korišćena je za razvoj infrastrukture za parsiranje SCL (*Substation Configuration Language*) fajlova, što omogućava razne akcije nad konfiguracionim fajlovima RZU sistema (*IED Capability Description*), IID (*Instantiated IED Description*) ili SCD (*System Configuration Description* itd.) poput, na primer, automatskog pronalaženja svih GOOSE/SV kontrolnih blokova, kao i njihovih nadzornih logičkih čvorova.

PyShark (MIT licenca) - Python omotač za tshark, komandno-linijski alat Wireshark softvera. Karakteristike:

- Omogućava parsiranje mrežnih paketa koristeći Wireshark dekoder strukture protokola,

- Podržava analiziranje iz PCAP fajlova ili "uživo" tokom snimanja mrežnog saobraćaja,

Korišćena je za parsiranje protokola poput GOOSE, SV, MMS (*Manufacturing Message Specification*), PTP (*Precision Time Protocol*), PRP (*Paralel Redundancy Protocol*), SNMP (*Simple Network Management Protocol*) iz snimaka mrežnog saobraćaja i sprovođenje analize paketa, analiza performansi, detekciju grešaka i tako dalje.

Netmiko (MIT licenca) **-** Biblioteka za jednostavnu komunikaciju putem SSH protokola sa mrežnim uređajima (ethernet svičevi, ruteri i slično). Ključne funkcionalnosti:

- Uspostavljanje konekcije sa mrežnim uređajima,

- Izvršavanje i parsiranje *show* i *config* komandi,

Korišćena je za prikupljanje konfiguracija i dijagnostike sa mrežnih uređaja, njihovu analizu i upoređivanje.

Pysnmp (BSD licenca) - Biblioteka za rad sa SNMP protokolom (verzije 1, 2c i 3). Omogućava:

- Slanje SNMP zahteva (GET, SET, WALK),
- Prijem i obradu *trap* obaveštenja,
- Rad sa MIB (*Management Information Base*) bazama podataka radi lakšeg razumevanja OID-ova (*Object IDentifier*),

Korišćena je za automatizaciju provere povezanosti uređaja na odgovarajuće portove ethernet svičeva, zatim praćenje ponašanja RSTP protokola u toku testiranja prekida mrežnog prstena, prikupljanje metrike o performansama i detekciju abnormalnosti.

Python-nmap (GPL licenca) - Omotač za mrežni alat *nmap*, namenjen mrežnom skeniranju i bezbednosnim proverama. Omogućava:

- Pokretanje *nmap* komandi iz Pythona,
- Parsiranje rezultata,
- Automatizaciju procesa skeniranja mrežnih segmenata.

Korišćena je za proveru svih postojećih i dostupnih komunikacionih uređaja u mrežnoj infrastrukturi i otvorenih portova za pristup.

Pyasn1 (MIT licenca) - Biblioteka za ASN.1 kodiranje i dekodiranje. Koristi se kada je potrebno raditi sa standardizovanim binarnim protokolima ili komunikacijom koja zahteva ASN.1 format, poput GOOSE, SV, MMS i slično.

Openpyxl (MIT licenca) - Biblioteka za rad sa Excel .xlsx fajlovima iz Pythona. Ključne funkcionalnosti:

- Kreiranje, čitanje i izmena .xlsx fajlova,
- Rad sa ćelijama, redovima, kolonama, radnim listovima i radnim sveskama,
- Podrška za formatiranje ćelija (boje, fontovi, stilovi) i tako dalje.

Korišćena je za generisanje izveštaja na osnovu mrežnih analiza (npr. tabela GOOSE/SV emitera i pretplatnika ili poruka), automatizaciju pravljenja tehničke dokumentacije i slično.

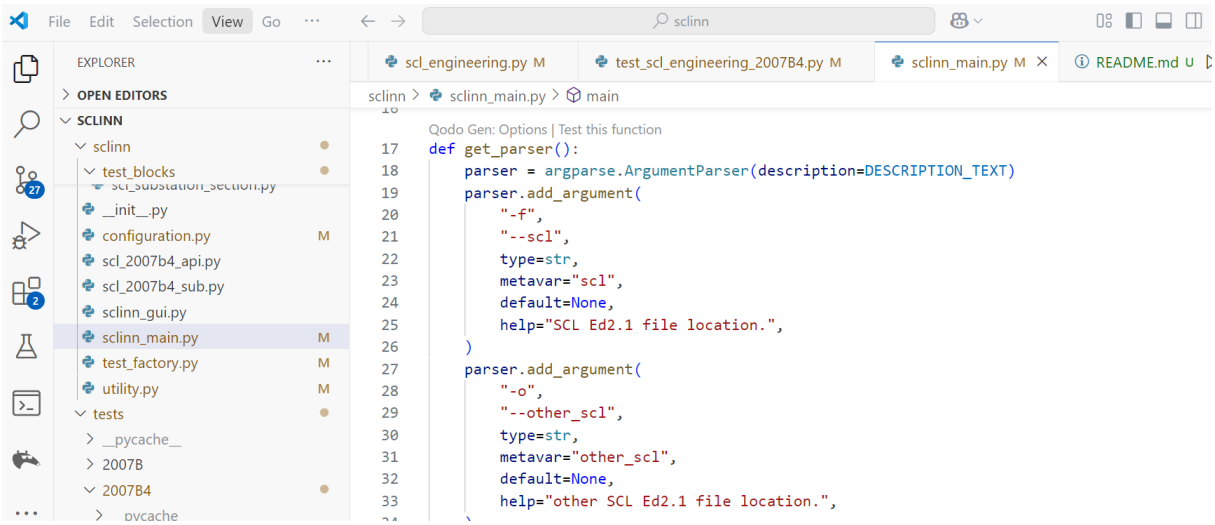
NiceGUI (MIT licenca) - Biblioteka za kreiranje grafičkog korisničkog interfejsa direktno u veb pregledaču koristeći Python. Karakteristike:

- Ne zahteva poznavanje HTML/CSS/JavaScript programskih jezika,
- Može se koristiti za razvoj lakih aplikacija, vizualizaciju podataka ili kao *front-end* za inženjerske alate.

2.2 Radno okruženje za razvoj alata

Inženjerima je danas na raspolaganju širok spektar radnih okruženja za razvoj sopstvenih alata i skripti. Ukoliko su potrebe jednostavne - na primer, osnovna obrada podataka sa tekstualnim prikazom rezultata - dovoljan je samo Python interpreter sa osnovnim standardnim bibliotekama, instaliran lokalno na računaru. U takvim slučajevima, skripte se mogu pisati u jednostavnim uređivačima teksta (poput Notepad++) i izvršavati iz komandne linije ili Python konzole.

Međutim, kada se razvijaju složeniji softverski alati, ili kada je potreban veći komfor tokom rada, preporučuje se korišćenje naprednijih razvojnih okruženja (IDE – *Integrated Development Environment*) kao što su Visual Studio Code, PyCharm i slični. Ova okruženja su veoma prilagodljiva, često podržavaju rad na projektima većih razmera, i sadrže funkcionalnosti koje značajno olakšavaju proces razvoja: automatsko formatiranje i bojenje sintakse, napredne alatke za otklanjanje grešaka (*debugging*), automatsko dopunjavanje koda (*autocomplete*), integrisane terminale, pa čak i podršku za rad sa virtuelnim okruženjima (*virtual environments*). Takođe, sve češće ovi alati imaju integrisanu podršku veštačke inteligencije (sistemi za generisanje koda, sugestije za optimizaciju i automatsko ispravljanje grešaka) što dodatno ubrzava razvoj i povećava kvalitet softverskih rešenja.



Slika 1: Grafički interfejs alata za razvoj softvera - Visual Studio Code

3 POTENCIJALNE PRIMENE KORISNIČKIH POMOĆNIH ALATA

Iako će se u ovom radu prevashodno obrađivati izrada alata za olakšavanje inženjeringa, testiranja i analize u RZU sistemima baziranim na IEC 61850 standardu, to ne ograničava primenu i na druge sisteme automatizacije.

3.1 Primena u oblasti inženjeringa RZU IEC 61850 sistema

Inženjering jednog RZU sistema baziranog na standardu IEC 61850 je zasnovan na definisanju i podešavanju mašinski procesibilnog softverskog koda u vidu XML fajla koji opisuje sve bitne funkcionalnosti. Što znači da je standard omogućio da se konfiguracija RZU sistema specificira u pomenutom formatu koji je moguće mašinski proveriti u skladu sa predefinisanim pravilima (IEC 61850-6 SCL jezik [1]).

Standard predviđa postojanje alata koji prate fazu inženjeringa i to – ICT – alat razvijen od strane proizvođača uređaja za konfigurisanje i komunikaciju sa uređajem, SCT – alat koji se koristi za konfigurisanje sistema baziranog na IEC 61850 u smislu podešavanja komunikacionih parametara, selekcije signala za razmenu koji će se distribuirati po različitim principima (izveštaji, logovi, GOOSE, SV, transfer fajlova...), SST – alat za specificiranje RZU sistema. Oni su već dostupni na tržištu sa neretko optimalnim setom funkcionalnosti koje su proizvođači smatrali da su neophodne korisniku. Sa druge strane, korisnik može da iskaže specifične i nestandardne zahteve u pogledu pomenutih alata, na primer, generisanjem dodatne dokumentacije projekta koja je prikazana u pogodnijim formatima za evaluaciju. Ako gorepomenuti alati to ne podržavaju, korisniku su dostupne biblioteke otvorenog koda koje on može primeniti u pravljenju svojih alata ili skripti koje mu u tome mogu pomoći.

Kao baza za razvoj sledećih primera alata korišćena je softverska biblioteka Generateds uz pomoć koje je dobijen API (*Application Program Interface*) direktno generisan iz IEC 61850 SCL XSD šeme (potrebno je ovo uraditi samo jednom za datu šemu). Na osnovu API-ja je dalje bilo moguće učitavati razne SCL fajlove koji predstavljaju konfiguraciju uređaja ili celog sistema, bilo da su to ICD, IID ili SCD i relativno lako pristupiti elementima konfiguracije – komunikacionim parametrima, podešavanjima logičkih čvorova, GOOSE/SV podešenjima, elementima DataSetova i svim drugim relevantnim parametrima a sve u cilju njihove dalje obrade prema zahtevima i željama inženjera.

Prvi primer je Python program za smeštanje raznih komunikacionih parametara RZU sistema u, za inženjere omiljenom, tabelarnom xlsx formatu. Program kao ulazni podatak koristi konfigurisani SCD fajl i kreira xlsx fajl sa radnim listovima u kojima se mogu naći detaljni podaci o pristupnim komunikacionim tačkama uređaja (*Access Points* - *AP*), GOOSE i SV kontrolni blokovi sa detaljima, logički čvorovi LGOS i LSVS sa podešavanjima (koje GOOSE i SV kontrolne blokove nadziru), raspodela pretplatnika na GOOSE i SV kontrolne blokove.

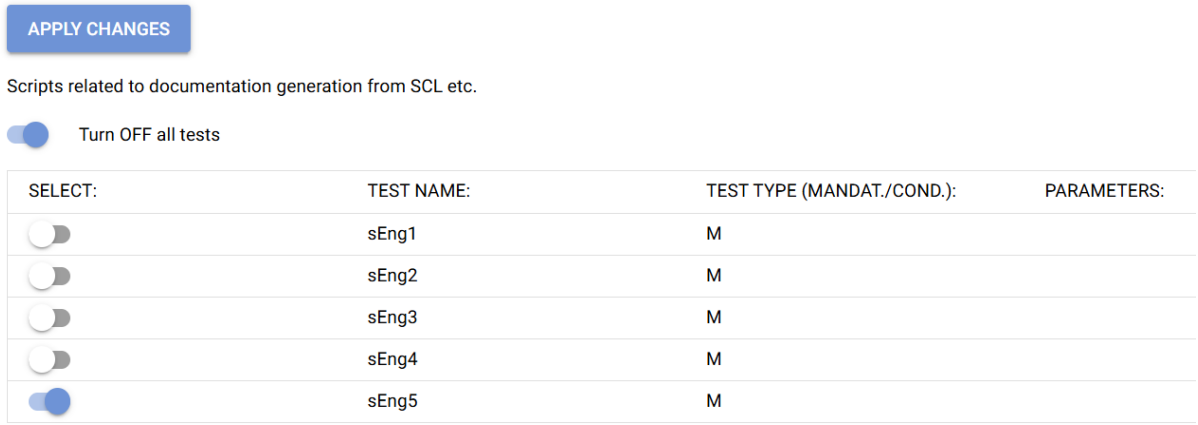
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	SubNetwork	iedName	apNam	redPro	route	cloc	KD	IP	IP-SUBNET	IP-GATEWA	OSI-NSA	OSI-TSE	OSI-PSE	OSI-SSE
2	Station Subnet	E26_F351_7SJ85	E		---	---	---	192.168.3.138	255.255.0.0	---	---	0001	00000001	0001
3	Station Subnet	E17_F351_7SJ85	E		---	---	---	192.168.3.93	255.255.0.0	---	---	0001	00000001	0001
4	Station Subnet	E16_F351_7SJ85	E		---	---	---	192.168.3.88	255.255.0.0	---	---	0001	00000001	0001
5	Station Subnet	E27_F351_7SJ85	E		---	---	---	192.168.3.143	255.255.0.0	---	---	0001	00000001	0001
6	Station Subnet	E11_A203_7SJ85	E		---	---	---	192.168.3.64	255.255.0.0	---	---	0001	00000001	0001
7	Station Subnet	E26_A203_7SJ85	E		---	---	---	192.168.3.139	255.255.0.0	---	---	0001	00000001	0001
8	Station Subnet	E03_F351_7SL87	E		---	---	---	192.168.3.23	255.255.0.0	---	---	0001	00000001	0001
9	Station Subnet	E12_F351_7SJ85	E		---	---	---	192.168.3.68	255.255.0.0	---	---	0001	00000001	0001
10	Station Subnet	E11_F351_7SJ85	E		---	---	---	192.168.3.63	255.255.0.0	---	---	0001	00000001	0001
11	Station Subnet	E18_F351_7SJ85	E		---	---	---	192.168.3.98	255.255.0.0	---	---	0001	00000001	0001
12	Station Subnet	E04_F351_7SL87	E		---	---	---	192.168.3.28	255.255.0.0	---	---	0001	00000001	0001
13	Station Subnet	E05_F351_7SL87	E		---	---	---	192.168.3.33	255.255.0.0	---	---	0001	00000001	0001
14	Station Subnet	E06_F351_7SL87	E		---	---	---	192.168.3.38	255.255.0.0	---	---	0001	00000001	0001
15	Station Subnet	E13_F351_7SL87	E		---	---	---	192.168.3.73	255.255.0.0	---	---	0001	00000001	0001
16	Station Subnet	E14_F351_7SL87	E		---	---	---	192.168.3.78	255.255.0.0	---	---	0001	00000001	0001
17	Station Subnet	E15_F351_7SL87	E		---	---	---	192.168.3.83	255.255.0.0	---	---	0001	00000001	0001
18	Station Subnet	E19_F351_7SL87	E		---	---	---	192.168.3.103	255.255.0.0	---	---	0001	00000001	0001
19	Station Subnet	E20_F351_7SL87	E		---	---	---	192.168.3.108	255.255.0.0	---	---	0001	00000001	0001
20	Station Subnet	E21_F351_7SL87	E		---	---	---	192.168.3.113	255.255.0.0	---	---	0001	00000001	0001
21	Station Subnet	E11_F350_7SS85	E		---	---	---	192.168.3.62	255.255.0.0	---	---	0001	00000001	0001
22	Station Subnet	C06_A203_7SJ85	E		---	---	---	192.168.1.39	255.255.0.0	---	---	0001	00000001	0001
23	Station Subnet	C02_F351_7SJ85	E		---	---	---	192.168.1.18	255.255.0.0	---	---	0001	00000001	0001

Slika 2: Detalji o komunikacionim pristupnim tačkama (AP) uređaja u RZU sistemu

	A	B	C	D	E	F
1	Publisher	Subcribers				
2	C02_A201Q1/LLN0.Control_DataSet[GO]	C06_A201(AP=E)				
3	C04_A201Q1/LLN0.Control_DataSet[GO]	C06_A201(AP=E)				
4	C05_A201Q1/LLN0.Control_DataSet[GO]	C06_A201(AP=E)				
5	C06_A201Application/LLN0.Control_DataSet[GO]	C02_A201(AP=E)	C04_A201(AP=E)	C03_A201(AP=E)	C10_A201(AP=E)	C09_A201(AP=E)
6	C08_A201Q1/LLN0.Control_DataSet[GO]	C06_A201(AP=E)				
7	C10_A201Q1/LLN0.Control_DataSet[GO]	C06_A201(AP=E)				
8	E03_A201Q1/LLN0.Control_DataSet[GO]	E26_A201(AP=E)	E11_A201(AP=E)	E17_A201(AP=E)	E18_A201(AP=E)	
9	E04_A201Q1/LLN0.Control_DataSet[GO]	E26_A201(AP=E)	E11_A201(AP=E)	E17_A201(AP=E)	E18_A201(AP=E)	
10	E05_A201Q1/LLN0.Control_DataSet[GO]	E26_A201(AP=E)	E11_A201(AP=E)	E17_A201(AP=E)	E18_A201(AP=E)	
11	E06_A201Q1/LLN0.Control_DataSet[GO]	E26_A201(AP=E)	E11_A201(AP=E)	E17_A201(AP=E)	E18_A201(AP=E)	
12	E11_A201Application/LLN0.Control_DataSet[GO]	E21_A201(AP=E)	E18_A201(AP=E)	E13_A201(AP=E)	E06_A201(AP=E)	E16_A201(AP=E)
13	E12_A201Q0/LLN0.Control_DataSet[GO]	C01_A201(AP=E)				
14	E12_A201Q1/LLN0.Control_DataSet[GO]	E26_A201(AP=E)	E11_A201(AP=E)	E17_A201(AP=E)	E18_A201(AP=E)	
15	E13_A201Q1/LLN0.Control_DataSet[GO]	E26_A201(AP=E)	E11_A201(AP=E)	E17_A201(AP=E)	E18_A201(AP=E)	
16	E14_A201Q1/LLN0.Control_DataSet[GO]	E26_A201(AP=E)	E11_A201(AP=E)	E17_A201(AP=E)	E18_A201(AP=E)	
17	E15_A201Q1/LLN0.Control_DataSet[GO]	E26_A201(AP=E)	E11_A201(AP=E)	E17_A201(AP=E)	E18_A201(AP=E)	
18	E16_A201Q0/LLN0.Control_DataSet[GO]	C03_A201(AP=E)				
19	E16_A201Q1/LLN0.Control_DataSet[GO]	E26_A201(AP=E)	E11_A201(AP=E)	E17_A201(AP=E)	E18_A201(AP=E)	
20	E17_A201Application/LLN0.Control_DataSet[GO]	E21_A201(AP=E)	E11_A201(AP=E)	E13_A201(AP=E)	E06_A201(AP=E)	E16_A201(AP=E)
21	E18_A201Application/LLN0.Control_DataSet[GO]	E21_A201(AP=E)	E11_A201(AP=E)	E13_A201(AP=E)	E06_A201(AP=E)	E16_A201(AP=E)
22	E19_A201Q1/LLN0.Control_DataSet[GO]	E26_A201(AP=E)	E11_A201(AP=E)	E17_A201(AP=E)	E18_A201(AP=E)	

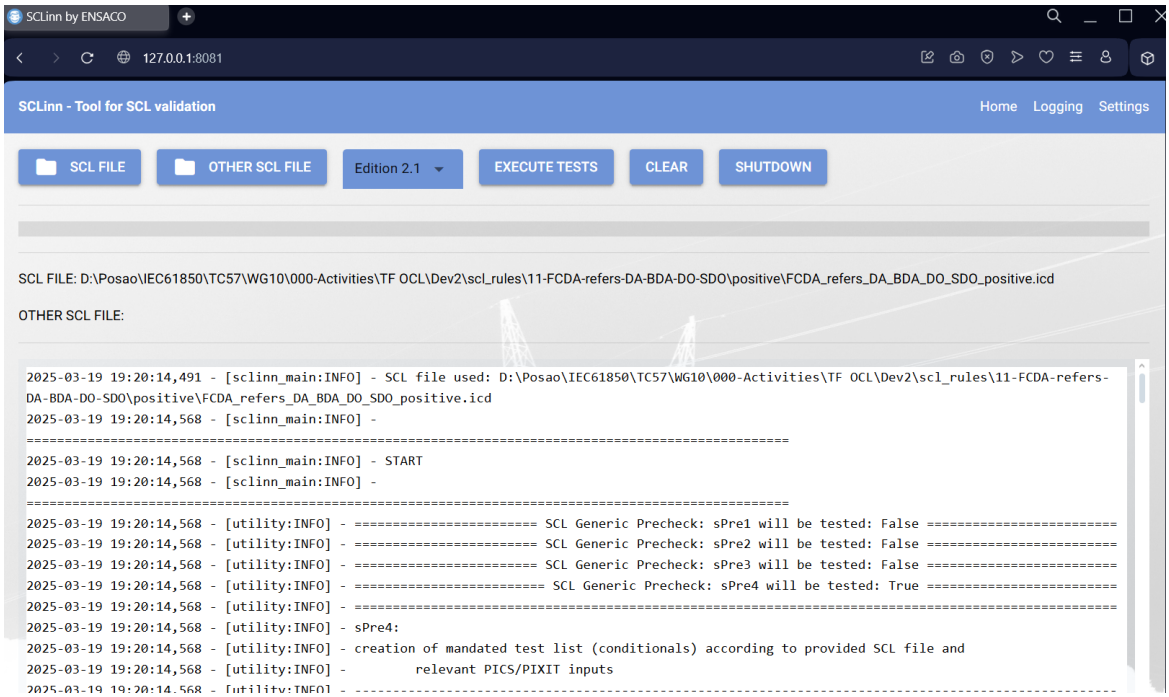
Slika 3: Detalji o *publisher/subscriber* raspodeli sa informaciji o korišćenom AP

Sledeći primer gde se inženjerski Python programi/alati mogu pokazati korisnim jeste dodatna validacija XML fajlova, tj. razvijanje dodatne logike koja može izvršiti proveru korisničkih zahteva. I ovde se kao osnova koristi API generisan uz pomoć biblioteke generateds. Velika količina napravljenih Python skripti je objedinjena i lako dostupna preko grafičkog interfejsa napravljenog uz pomoć nicegui biblioteke gde se više skripti može selektovati i primeniti na učitan SCL fajl. Detaljni rezultati se prikazuju u osnovnom prozoru na osnovu kojih inženjer može dijagnostikovati probleme i pronaći rešenja. Izvršavanje velikog broja testova kao i parsiranje velikih SCL fajlova može da bude zahtevno za računar i da bude relativno sporo (može da potraje i nekoliko minuta za 70-tak validacionih testova).



Slika 4: grafički interfejs za izbor željenih testova/skripti

S obzirom da se tzv. NSD (*Namespace definitions* – definisani prema IEC TS 61850-7-7 [2]) fajlovi koji su nosioci detalja IEC 61850 modela (opis logičkih čvorova, objekata, atributa itd) takođe baziraju na XML jeziku – opet uz pomoć generateds biblioteke se može iz XSD šeme definisati zaseban API uz pomoć kojeg se definicije modela (dostupni na sajtu IEC u redukovanoj verziji bez opisa) može učitati i dalje koristiti za provere, poređenja, pretraživanja modela korišćenjem prikaza u xlsx formatu.



Slika 5: grafički interfejs skupa alata za inženjering IEC 61850 RZU sistema

3.2 Primena u oblasti fabričkog/ispitivanja RZU IEC 61850 sistema na objektu

U toku fabričkog (FAT) ili ispitivanja na objektu (SAT) u fazi puštanja RZU sistema baziranog na IEC 61850 standardu, inženjer može sebi dodatno olakšati rad samostalnim razvojem programa i skripti. Dalje su dati primeri u kojima je autor koristio biblioteke otvorenog koda u kreiranju Python skripti/testova u toku svojih angažovanja na raznorodnih projekata.

Alati bazirani na pysnmp biblioteci:

1. Skeniranje postojeće mrežne infrastrukture, pronalaženje raspoloživih ethernet svičeva i dokumentovanje podešenih statičkih VLAN-ova

2. Skeniranje postojeće mrežne infrastrukture i dokumentovanje alokacije svih IP adresa (prema MAC adresama) po ethernet svičevima

3. Snimanje tekućih RSTP topoloških informacija u mrežnom komunikacionom prstenu

4. Snimanje PTP GrandMaster SNMP informacija (status sinhronizacije, status interfejsa, tačnost u ns, broj alociranih satelita, tačnost servisa vremenske sinhronizacije i sl.)

Alati bazirani na pyshark implementaciji:

1. Snimanje protoka na glavnim *trunk* portovima i određivanje totalnog zauzetog kapaciteta mreže (*bandwidth*) – minimalne, maksimalne i srednje vrednosti. Skripta je sa relativno lošim performansama. Druga varijanta je napravljena da proračunava zauzeti kapacitet mreže prema određenim tipovima protokola (SV, GOOSE, PTP, MMS, drugi protokoli). Probani su i drugi pristupi parsiranja zapisa ali bez značajnog poboljšanja u brzini izvršenja koda.

2. Pronalaženje svih VLAN ID-ova u PCAP snimku mrežnog saobraćaja

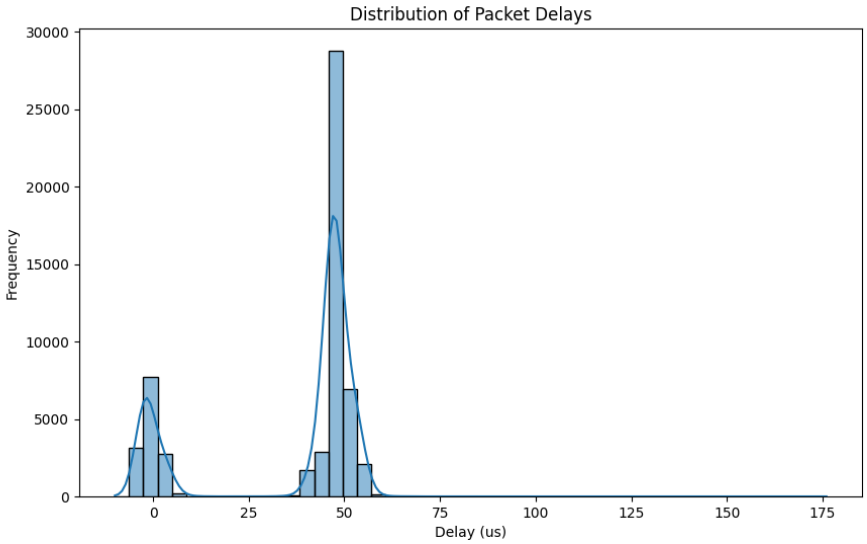
3. Proračun brzine RSTP rekonfiguracije na osnovu detekcije izgubljenih SV paketa od momenta prekida mreže pa do uspostavljanja druge funkcionalne topologije (rezolucija detekcije zavisi od brzine odabiranja tj. broja odbiraka u sekundi tako, na primer, za 4000 odbiraka u 1 sekundi se dobija da je vremenska razlika između slanja 2 odbirka 250 μs)

```
09:38:46,879 - [INFO] - tNet5:
09:38:46,879 - [INFO] - Calculate RSTP reconfiguration from missing packets (SV packets)
09:38:46,880 - [INFO] - ------------------------------------------------------------
09:38:46,880 - [INFO] - Network tests: tNet5 -  PCAP file: tNet5-RSTP-reconfig-capture_enp3s0_07-29-10-ALL.pcap
09:45:54,954 - [INFO] - SV stream - C06A1F4000S1I0U1:
09:45:54,973 - [WARNING] -     Gap detected: 3254 -> 3256, gap of 1 that is equal of lost time: 250.0 us
09:45:54,973 - [INFO] - SV stream - C05F4000S1I4U0:
09:45:54,993 - [INFO] -     No gaps detected.
09:45:54,993 - [INFO] - SV stream - C06F4000S1I4U0:
09:45:55,013 - [INFO] -     No gaps detected.
09:45:55,013 - [INFO] - SV stream - C02F4000S1I4U0:
09:45:55,033 - [INFO] -     No gaps detected.
09:45:55,033 - [INFO] - SV stream - C08F4000S1I4U0:
09:45:55,053 - [WARNING] -     Gap detected: 820 -> 903, gap of 82 that is equal of lost time: 20500.0 us
09:45:55,054 - [WARNING] -     Gap detected:p 2744 -> 2760, gap of 15 that is equal of lost time: 3750.0 us
09:45:55,054 - [INFO] - SV stream - C04F4000S1I4U0:
09:45:55,073 - [INFO] -     No gaps detected.
09:45:55,074 - [INFO] - SV stream - C09F4000S1I4U0:
09:45:55,093 - [WARNING] -     Gap detected: 820 -> 903, gap of 82 that is equal of lost time: 20500.0 us
09:45:55,094 - [WARNING] -     Gap detected: 2752 -> 2770, gap of 17 that is equal of lost time: 4250.0 us
09:45:55,094 - [INFO] - SV stream - C03F4000S1I4U0:
09:45:55,114 - [INFO] -     No gaps detected.
09:45:55,114 - [INFO] - SV stream - C01F4000S1I4U0:
09:45:55,135 - [INFO] -     No gaps detected.
```

Slika 6: Proračun brzine RSTP rekonfiguracije u mreži

4. Pronalaženje VLAN ID-jeva u snimku mrežnog saobraćaja na posredan način (SCL definicije GOOSE i SV kontrolnih blokova)

5. Poređenje vremena istih paketa koji su snimljeni na različitim mestima u lokalnoj mrežnoj infrastrukturi – određivanje distribucije kašnjenja paketa. Provera kašnjenja u zavisnosti od kvarova u mrežnom prstenu.



Slika 7: Analiza kašnjenja paketa u procesnoj mreži

6. Test provere postojanja PRP LAN A i LAN B sa istim sekvencionim brojem u PRP mrežnoj infrastrukturi na osnovu analize PCAP snimaka iz LAN A i LAN B.

7. Test provere da su svi LAN A paketi u LAN A PRP mreži i da su svi LAN B paketi u LAN B PRP mreži

Korišćenjem biblioteke netmiko, iako u njoj nije postojala podrška za ethernet svičeve koji su korišćeni u projektu, razvijen je pristup ka svičevima putem SSH (*secure shell* protokola za komunikaciju) zarad preuzimanja aktuelnih konfiguracija ali i raspoložive dijagnostike (npr. PTP statistike, dijagnostike napajanja, statistike RSTP protokola, VLAN statistike, ARP tabela itd). Ovaj pristup se pokazao pogodnijim od korišćenja SNMP biblioteke zato što je količina dostupnih informacija iz MIB bila manja u odnosu na sve raspoložive dijagnostičke izveštaje u CLI (*Command Line Interface*).

```
2024-09-11 18:01:14,880 - [base_connection:DEBUG] - Clear buffer detects data in the channel
2024-09-11 18:01:14,910 - [base_connection:DEBUG] - read_channel:
2024-09-11 18:01:14,910 - [base_connection:DEBUG] - [find_prompt()]: prompt is >
2024-09-11 18:01:14,910 - [base_connection:DEBUG] - write_channel: b'sql select from vlanStaticCfg\n'
2024-09-11 18:01:14,910 - [base_connection:DEBUG] - read_channel:
2024-09-11 18:01:14,925 - [base_connection:DEBUG] - read_channel:
2024-09-11 18:01:14,955 - [base_connection:DEBUG] - read_channel: sql select from vlanStaticCfg
2024-09-11 18:01:14,956 - [base_connection:DEBUG] - Pattern found: (sql\ select\ from\ vlanStaticCfg) sql select from vlanStaticCfg
2024-09-11 18:01:14,986 - [base_connection:DEBUG] - read_channel:

VID  VLAN Name           Forbidden Ports                                  IGMP  DHCP  MSTI Frm Size Mirror VLAN Status
1    Interswitch         None                                             Off   Off   0    Extended No          active
3    GOOSE_SV_NAP_A      2/1-2/2                                          Off   Off   0    Extended No          active
4    GOOSE_SV_NAP_B      0/3-2/4                                          Off   Off   0    Extended No          active
10   SV1                 0/3-1/4,2/2-2/4                                  Off   Off   0    Extended No          active
2024-09-11 18:01:15,017 - [base_connection:DEBUG] - read_channel:
20   SV2                 0/3-2/1,2/3-2/4                                  Off   Off   0    Extended No          active

5 records selected

ESC[0mESC[2K
>
```

Slika 8: Rezultat preuzimanja online podešenja iz eth. sviča (VLAN)

4 OGRANIČENJA I RIZICI

Pored prednosti koje su evidentne, alati zasnovani i razvijeni sa bibliotekama otvorenog koda su podložni određenim rizicima i imaju ograničenja koja će biti obrazložena u ovom poglavlju.

4.1 Podrška u održavanju

U zavisnosti od veličine tima volontera koji stoji iza razvoja biblioteka otvorenog koda koje inženjeri koriste u svojim alatima, oni kao krajnji korisnici istih se mogu naći u situaciji da izgube podršku u nekom momentu. Ta podrška podrazumeva rešavanje problematičnih ponašanja biblioteke, unapređenje u pogledu sigurnosti i novih funkcionalnosti, dodatna pojašnjenja o postojećim funkcionalnostima i tako dalje. Bez te podrške, korisnik je stavljen pred izazov da samostalno ili uz pomoć postojeće dokumentacije ili na primer, postavljanjem svojih pitanja forumima, neformalnim grupama, koji se bave problematikom date biblioteke razrešava tekuće nedostatke. To takođe dovodi do toga, da krajnji korisnik postaje, de fakto, novi „vlasnik“ izmenjenog dela programa i nadležan da ga, pa makar i samo za sebe – dokumentuje, održava, popravlja i slično.

Prilagođavanje biblioteke prema potencijalnim zahtevima korisnika je u rukama volontera koji mogu da ocene da je tražena promena nepraktična ili nepotrebna za biblioteku i ne prihvate je.

4.2 Mane korišćenja Python programskog jezika

Programi razvijeni u Python jeziku[4] imaju brojne prednosti, ali takođe i mane koje mogu biti bitne u zavisnosti od konteksta primene. Glavne mane Python programa:

- Sporije izvršavanje (loše performanse) - Python je interpretirani jezik, što znači da se kod ne kompajlira u mašinski kod, već se izvršava liniju po liniju. Zbog toga je značajno sporiji od jezika poput C, C++, Rust, Go, pa čak i Jave u mnogim slučajevima. Ovo može biti problem kod računski intenzivnih zadataka (npr. obrada velikih skupova podataka).

- Potrošnja memorije - Python koristi automatsko upravljanje memorijom i nije pogodan za sisteme sa ograničenim resursima.

- Slaba statička provera tipova podataka - Python je dinamički tipiziran jezik, što olakšava rad, ali greške u tipovima mogu ostati neotkrivene do samog izvršenja.

- Distribucija i zavisnosti - Deljenje Python aplikacije sa drugima može biti komplikovano: Zavisi od verzije Pythona, biblioteka i okruženja. Potrebno je praviti virtualna okruženja (venv, poetry, pipenv). Pakovanje za distribuciju (npr. u .exe fajl za Windows) zahteva dodatne alate i može biti problematično.

- Nije pogodan za sisteme u realnom vremenu - Python nije pogodan za sisteme sa strogim vremenskim ograničenjima.

- Bez direktnog pristupa hardveru - Python nije idealan za razvoj softvera koji efikasno upravlja hardverom, gde je potreban direktan pristup memoriji, portovima, hardveru – gde su jezici poput C/C++ bolji izbor.

- Sigurnosni rizici - Dinamička priroda jezika i mogućnost izvođenja koda „u hodu“ može dovesti do bezbednosnih propusta ako se na to ne obrati pažnja.

Ove mane se mogu zanemariti kada se Python koristi samo za automatizaciju, analizu podataka, pisanja skripti, i kada se aplikacija koristi u kontrolisanom okruženju (npr. unutar firme, na lokalnom serveru), ili na kraju kada je brz razvoj i fleksibilnost važnija od sirove performanse.

5 ZAKLJUČAK

Korišćenje široke palete dostupnih biblioteka otvorenog koda može značajno unaprediti i olakšati testiranje i validaciju implementacije modernih sistema relejne zaštite i upravljanja u objektima elektroenergetskog sistema. Na ovaj način se može postići veća fleksibilnost u inženjerskom radu, ostvariti značajna finansijska ušteda, povećati razumevanje složenih tehnoloških celina sistema i njegovih podsistema, a nadasve unaprediti usaglašenost sa internacionalnim standardom IEC 61850 i povezanim industrijskim standardima.

Iako ovakvi alati ne mogu uvek u potpunosti zameniti funkcionalnost, stabilnost i pouzdanost specijalizovanih komercijalnih softverskih rešenja, oni pružaju značajnu slobodu inženjeru da:

- lakše i brže eksperimentiše sa novim pristupima i algoritmima,

- razvija sopstvene softverske alate prilagođene specifičnostima projekta,

- bolje razume međuzavisnosti između slojeva sistema (komunikacioni, logički, fizički),

- i stiče dublje znanje o protokolima, formatima podataka i principima rada sistema koji čine osnovu moderne energetske automatizacije danas.

Uz to, rad sa alatima otvorenog koda podstiče razvoj programerskih veština, razumevanje principa softverske arhitekture, što inženjeru omogućava da prati i aktivno doprinosi aktuelnim tokovima u industriji automatizacije i energetike.

Na duže staze, ovakav pristup doprinosi stvaranju domaće ekspertize, smanjenju zavisnosti od zatvorenih rešenja i rastu inovativnosti kroz razmenu znanja i saradnju sa širom inženjerskom zajednicom.

6 LITERATURA

[1] SRPS EN 61850-6:2013/A1:2018 - Komunikacione mreže i sistemi za automatizaciju energetskih objekata – Deo 6: Jezik za opis konfiguracije za komunikaciju sa IED-ovima u energetskim postrojenjima

[2] IEC TS 61850-7-7:2018 ED1 - Communication networks and systems for power utility automation - Part 7-7: Machine-processable format of IEC 61850-related data models for tools

[3] IEC TR 61850-90-4:2020 ED2 - Communication networks and systems for power utility automation - Part 90-4: Network engineering guidelines

[4] Python 3.12 Documentation