Nizkocenovna avtomatizacija visokofrekvenčnih meritev

Peter Miklavčič, Boštjan Batagelj

Univerza v Ljubljani, Fakulteta za elektrotehniko, Tržaška cesta 25, 1000 Ljubljana
E-pošta: peter.miklavcic@fe.uni-lj.si

Low-cost automatization of radio-frequency measurements

In this paper a Python library in-development is presented that enables low-cost automatization of radio-frequency measurements using different instruments. The library supports instruments with omnipresent buses such as RS-232, USB and Ethernet/LXI. A cheap and practical solution for connecting legacy instruments with the GPIB bus is also discussed and identified. In the beginning, the background for the proposed solution is discussed and general software and hardware proposals are laid out. The proposed approach is then presented in detail, discussing the core library and its topics first and the developed measurement scripts are discussed next, followed by the postprocessing and graphing scripts. Remote and embedded execution are discussed as well.

1 Uvod

Avtomatizacija meritev v smiselni meri je pri raziskovalnem delu vedno dobrodošla. Poleg multiplikativnega prihranka časa pri ponavljajočih meritvah, jasna avtomatizacijska skripta omogoča boljšo ponovljivost in manj napak tako pri posameznih serijah meritev, kakor tudi pri znanstveni ponovljivosti poskusov na dolgi rok.

Radiofrekvenčne meritve pogosto zahtevajo zajem in obdelavo večje količine meritev in to iz različnih enokanalnih ali večkanalnih merilnih instrumentov (na primer visokofrekvenčnega merilnika moči, faznega detektorja, spektralnega analizatorja, ...). Profesionalne rešitve avtomatizacije takih meritev so poleg visoke cene programske in strojne opreme podvržene tudi nekompatibilnosti med programsko opremo različnih proizvajalcev ter različnimi namenskimi vodili.

Pri delu z najrazličnejšimi merilnimi instrumenti in pripomočki za potrebe meritev v visokofrekvenčni, optični in antenski tehniki imamo v laboratoriju potrebo po prilagodljivi, vsestranski in preprosti avtomatizaciji zajema, poobdelave ter prikaza meritev. V nadaljevanju je predstavljen predlog rešitve, naši dosedanji poskusi ter ugotovitve. Predlagana kombinacija programske in strojne opreme omogoča tekoče delo z različnimi merilnimi instrumenti, od profesionalnih komercialnih, do ročno izdelanih, kot so na primer nizkocenovni doma izdelani VF generatorji [1] in detektorji [2, 3] prof. dr. Matjaža Vidmarja. Z nekaj dodatne programske logike in obdelave meritev so taki eksperimentalni instrumenti primerni za grobe visokofrekvenčne meritve. Za meritve smernega diagrama antene se pri svojem delu v veliki meri zanašamo na ročno izdelane merilne vrtiljake, objavljene v [3]. Tudi v članku opisana programska oprema je primarno namenjena antenskim meritvam, tako da je bilo največ razvoja opravljenega v to smer.

2 Podatkovna vodila

Starejši merilni instrumenti tipično omogočajo krmiljenje naprave in branje izmerjenih vrednosti po 8-bitnem vzporednem vodilu IEEE 488, standardiziranem leta 1975. Vodilo je bilo razvito kakih 10 let prej pod imenom HP-IB (ang. *Hewlett-Packard Interface Bus*), po standardizaciji in splošni uporabi pa je postalo znano kot GPIB (ang. *General Purpose Interface Bus*). Danes GPIB nadomešča LXI (ang. *LAN eXtensions for instrumentation*), ki temelji na vodilu Ethernet, IP in protokolu IEEE 1588 za psevdo-realnočasovno uro, omogoča pa tudi pravo realnočasovno signalizacijo z uporabo dodatnih povezav. Pri napravah s podporo LXI je možno hitro in poceni avtomatizirati tudi kompleksnejšo meritev z več instrumenti, in to z uporabo le Ethernet stikala in nadzornega računalnika ali sorodne naprave ter ustrezne programske opreme. Večje težave so pri komercialno dostopnih vmesnikih med vodilom GPIB in modernimi vodili (na primer USB ali Ethernet), ki stanejo nekje od 100€ dalje, kar v določenih primerih lahko predstavlja previsok strošek. Poleg razmeroma visoke cene večina takih vmesnikov potrebuje namenske, včasih slabo podprte gonilnike, ki ne delujejo nujno na obeh glavnih družinah operacijskih sistemov, ki poganjata osebne računalnike in sorodne naprave.

Nekateri instrumenti omogočajo priklop preko drugih standardnih vodil, od katerih zagotovo velja omeniti RS-232 in USB. Pri teh vodilih za razliko od namenskih GPIB in LXI aplikacijska raven (protokol nadzora instrumenta in kodiranje podatkov) ni standardizirana, tako da nekateri instrumenti kljub uporabi standardnega vodila ne omogočajo preprostega in odprtega povezovanja z različno programsko opremo, pač pa v tem primeru zahtevajo (bolj ali manj dobro podprto) namensko programsko opremo z različnimi programskimi vmesniki in možnostmi za nadaljnjo avtomatizacijo meritev.

3 Programska oprema

Med poplavo komercialnih in odprtih programskih orodij ter množico programskih jezikov je bil s ciljem nizkocenovne rešitve izbran pristop z odprtokodnim tolmačitvenim programskim jezikom Python [4]. Ta z odprtim pristopom in široko (tudi akademsko) skupnostjo že leta doživlja hiter razvoj in ponuja vrsto kvalitetnih knjižnic za najrazličnejša opravila. Pri takih možnostih je v luči cilja znanstvene ponovljivosti danes verjetno smotrno izbrati tako ali primerljivo odprto okolje, kakor ga ponuja jezik Python.

4 Strojna oprema

Čeprav bi bilo zaradi znanstvene ponovljivosti verjetno prav, da bi bila odprtokodna strojna oprema v znanosti standard, je realnost toliko drugačna, da so bili kriteriji za primerne merilne instrumente (za razliko od rigoroznega pristopa pri izboru programske opreme) bolj pragmatično zastavljeni. Primerna naprava naj podpira vsaj enega izmed danes razširjenih vodil (RS-232, USB, Ethernet/LXI, in podobno), za naprave z vmesnikom GPIB pa se splača preveriti morebitne nizkocenovne in odprtokodne rešitve pretvornika. Naprava mora imeti dokumentirano ali vsaj znano komunikacijo na aplikacijski ravni. Na dolgi rok je smiselno stremeti k čim bolj odprtim in ponovljivim strojnim rešitvam, za dosego cilja nizkocenovnih ponovljivih meritev pa verjetno v smeri čim bolj univerzalnih merilnih instrumentov.

4.1 Povezava naprav z vmesnikom GPIB

Preizkušenih je bilo več različnih komercialnih in nekomercialnih rešitev pretvornikov za GPIB. Kot pomembna šibka točka so se pokazali že prej omenjeni slabo podprti in pomanjkljivo zasnovani gonilniki, ki onemogočajo zanesljivo delo. Tudi sicer pogosto uporabljena knjižnica PyVISA na primer ni polno podprta na kartičnih računalnikih s procesorji ARM. Kot robustna in preprosta rešitev so se izkazali vmesniki USB-GPIB, ki za povezavo na aplikacijski ravni uporabljajo navidezna serijska vrata. Obstajajo tudi odprtokodni projekti razvoja takih vmesnikov, od katerih se je dobro izkazal vmesnik AR488 [5], zasnovan na mikrokrmilniških razvojnih ploščah Arduino. Ponuja zmogljiv nizkocenoven vmesnik z odprtokodno strojno in programsko opremo.

5 Jedrna knjižnica

Pri razvoju predlagane programske rešitve so glavni cilji preprostost zasnove in uporabe, pravilna in ponovljiva uporaba okolja Python ter kompaktna in modularna programska koda, ki se ravno v pravi meri zanaša na zmogljive zunanje knjižnice. Rešitev naj ne vsebuje nepotrebnih funkcionalnosti, potrebne pa naj abstrahira na smiselnem nivoju za ponovljivo uporabo v različnih merilnih skriptah. Za večino opravil zadostuje tekstovni vmesnik (CLI, ang. *command-line interface*), ki je na voljo povsod, grafični vmesnik (GUI, ang. *graphical user interface*) pa se naj uporablja tam, kjer je to potrebno oziroma smiselno. Knjižnica je namenjena uporabi v okolju Python, zato se ji ni potrebno izolirati in razen kjer je to nujno potrebno ne poskuša ujeti napak, ampak se zanaša na Pythonov izpis in postopanje ob napakah.

Za dosego zadanega cilja je v razvoju nabor skript, od katerih glavna (z delovnim imenom `liblab.py`) vsebuje različne nizkonivojske in visokonivojske funkcije z namenom čim bolj praktične modularizacije programske (merilne) kode. Za shranjevanje meritev je bil izbran berljiv in vseprisoten format CSV (ang. *comma-separated values*), ki pri pravilni implementaciji dela s tekstovnimi datotekami ne predstavlja ozkega grla, poleg tega za delo s tem formatom obstaja standardna knjižnica `csv`. Tekom razvoja se je pokazalo tudi, da se ni smiselno omejevati na eno izmed obeh trenutno aktualnih različic jezika Python, ampak je programsko kodo dokaj trivialno napisati tako, da deluje z obema različicama, s tem pa je možno tudi izkoristiti prednosti posamezne. V praksi je pri pisanju programske kode za obe različici jezika potrebno najti način izpisa poljubnega niza, ki deluje enako pri obeh. V predstavljeni knjižnici je omenjeno rešeno z uporabo funkcij `stdout.write()` in `stdout.flush()`, za kateri je potrebno vključiti standardno knjižnico `sys`. Jedrna knjižnica `liblab.py` med drugim vsebuje:

1. Uporabljene matematične in fizikalne konstante.
2. Funkcije za izpis razhroščevalnih informacij.
3. Funkcije za delo s serijskimi vrati:
   - neblokirno (ang. *non-blocking*) branje serijskih vrat z medpomnilnikom delnih branj,
   - sprejem in obdelava identifikacijskih podatkov naprave, poslanih po serijski povezavi (detekcija tipa in/ali posamezne naprave),
   - določitev unikatnega identifikacijskega niza (sestavljenega iz VID, PID, serijske številke in poslanih identifikacijskih podatkov),
   - izpis serijskih vrat in njihovih lastnosti (namenjen razhroščevanju).
4. Splošne funkcije za delo z merilnimi instrumenti:
   - iskanje želenih naprav med zaznanimi serijskimi vrati,
   - vzporedno čakanje med koncem strojnih in začetkom programskih inicializacij naprav,
   - branje medpomnilnika delnih branj,
   - vpis v medpomnilnik posameznega kanala meritev (s poljubnim povprečenjem posameznega kanala),
   - osnovna obdelava vpisanih meritev (preverjanje območja, normalizacija vrednosti, detekcija ovojnice - minimalnih in maksimalnih vrednosti),
   - programska zaključitev (zapiranje odprtih serijskih vrat in povezav).
5. Namenske funkcije za delo z merilnimi instrumenti:
   - razčlenjevanje vhodnih parametrov meritev (glede na tip instrumenta),
   - inicializacija strojnega in programskega nivoja (glede na tip instrumenta),
   - filtriranje (s pomočjo knjižnice `re`) in razčlenjevanje prebranih vrstic v koristne meritve (glede na tip instrumenta),

- nastavljanje uporabljenih visokofrekvenčnih izvorov in sintetizatorjev,
- nadzor frekvenčnega preleta VF generatorja,
- nastavljanje uporabljenih merilnih vrtiljakov,
- nadzor vrtenja in korakanje koračnega vrtiljaka,
- splošna omejitev hitrosti zajema.

6. Obdelava meritev (anten):
   - izračun veličin, potrebnih za uspešne meritve anten (impedančna prilagoditev, Fraunhoferjeva razdalja, premer prve Fresnelove cone, izgube zaradi razširjanja v praznem prostoru),
   - analiza smernega diagrama ter integracija smernosti.
7. Funkcije za delo z datotekami CSV:
   - branje glave in razčlenjevanje stolpcev,
   - tekstovni vmesnik za izbor želenih stolpcev.
8. Funkcije za generiranje zvoka (samo v Python 3):
   - vključitev zunanje knjižnice `simpleaudio`,
   - (ne)blokirno igranje poljubnega tona,
   - opozorilni pisk.

Ob izvršitvi skripte `liblab.py` je na voljo nekaj osnovnih razhroščevalnih funkcij: izpis serijskih vrat in zaznanih naprav, poskusno branje podprtih instrumentov, preizkus zvoka, sicer pa je skripta (knjižnica) namenjena vključitvi v *višjenivojske* merilne skripte. Pri idealno modularizirani kodi te tako vsebujejo le za posamezno meritev pomembno programsko logiko.

5.1 Vhodni podatki in nastavitve

Jedrna knjižnica in pripadajoče merilne skripte za svoje delovanje potrebujejo določene podatke, kot na primer identifikacijske nize vseh instrumentov, parametre vodil in povezav, privzete nastavitve instrumentov (na primer moč izvora, hitrost vrtenja vrtiljaka, ...), seznam merilnih kanalov posamezne naprave s pripadajočimi merilnimi enotami in območji, in podobno. Našteti podatki so zapisani v datoteki `config.py`, ki je namenjena vključitvi s strani jedrne knjižnice. Na ta način so uporabniški podatki zelo preprosto ločeni od programske kode. Pri zagonu katerekoli merilne skripte uporabnik poda le še obvezne (in neobvezne) vhodne parametre posamezne skripte, ki jih skripte nato razčlenijo s standardno Python knjižnico `argparse`.

6 Zbiranje meritev

Zaenkrat je bilo razvitih le nekaj osnovnejših merilnih skript, ki pa s svojim nastavljivim pristopom in modularizirano kodo omogočajo že kar nekaj različnih merilnih scenarijev v visokofrekvenčni tehniki.

6.1 Časovni prelet: `recorder.py`

Merilna skripta `recorder.py` je namenjena dolgotrajnemu zbiranju meritev iz različnih merilnikov, amplitudnih in faznih detektorjev, dodatno pa omogoča še zajem podatkov iz senzorja Bosch Sensortec BME280, ki zna meriti temperaturo okolice, relativno zračno vlažnost in zračni pritisk. Senzor je na računalnik moč priklopiti z uporabo katere izmed mikrokrmilniških razvojnih plošč [6] in preprostim programom, ki prebrane vrednosti pošilja po (navidezni) serijski povezavi po vodilu USB.

Skripta `recorder.py` ima z namenom dolgoročne stabilnosti preprosto zasnovo in uporablja POSIX funkcijo `os.fork()` za dvonitno delovanje, s katerim usklajuje zajem meritev iz različno hitrih instrumentov. Narejena je kot avtomat končnih stanj (ang. *finite-state machine*). V primarni niti programa se vrši branje vseh instrumentov, razen najbolj počasnega, tekstovni vmesnik, pisanje izhodnih datotek, morebitna omejitev hitrosti zajema in komunikacija s sekundarno nitjo (za kar je uporabljena knjižnica `mmap`). V sekundarni niti se vrši branje najpočasnejšega instrumenta, ki po končanem branju z medprocesno komunikacijo da takt vpisa v izhodno datoteko primarni niti. Skripta se izvaja dokler uporabnik ne pošlje prekinitve (ujete s pomočjo standardne knjižnice `signal`), nakar zaključi izhodno datoteko in zapre povezave do instrumentov. Omogoča tudi nastavljivo povprečenje posameznih kanalov, zajete meritve pa poskuša časovno čim bolj poravnati in jih po potrebi označi z milisekundno uro.

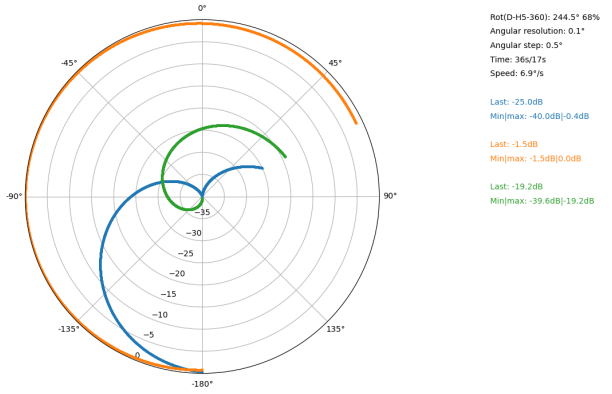
6.2 Frekvenčni prelet: `sweeper.py`

Skripta `sweeper.py` je namenjena zajemu meritev pri diskretnem frekvenčnem in/ali amplitudnem preletu nastavljivega visokofrekvenčnega generatorja. Omogoča zajem meritev individualno izbranih in povprečenih merilnih kanalov v vsaki točki preleta in jih zapiše v izhodno datoteko, med izvajanjem pa izpisuje stanje preleta. Tekom preleta spremlja stanje VF izvora in opozarja na nedosegljive točke merjenja izven frekvenčno-amplitudnega območja generatorja, preverja pa tudi merilno območje izmerjenih vrednosti posameznih merilnih kanalov.

6.3 Prostorski prelet: `rotator.py`

Skripta `rotator.py` je namenjena meritvam z merilnim vrtiljakom. V našem primeru gre največkrat za meritve smernega diagrama antene, katerim je skripta primarno namenjena. Glavna zanka programa upravlja in spremlja vrtenje vrtiljaka, povezanega preko vodila USB. Omogoča nastavljiv korak in sektor preleta. Med vrtenjem skripta zbira želene meritve (kanali se izbirajo enako kot pri skripti `sweeper.py`) in po potrebi nastavlja tudi visokofrekvenčni izvor ali ustavi vrtiljak (na primer za opravljanje frekvenčnega preleta). Poleg zajema meritev v izhodno datoteko in tekstovnega izpisa stanja, pa `rotator.py` za razliko od drugih opisanih skript, vsebuje tudi grafični vmesnik s polarnim izrisom za lažje spremljanje meritve v živo (prikazan na sliki 1) ter zvočno opozorilo ob začetku in koncu meritve.

Grafični vmesnik poleg spremljanja izmerjenih vrednosti izbranih kanalov (podobno kot tekstovni) izpisuje nastavitve in stanje vrtiljaka, pretečeni in preostali čas ter kotno hitrost vrtenja. Na koncu meritve prikaže na maksimalno vrednost poravnan snop in rezultate analize izmerjenega snopa. Ti obsegajo širino glavnega snopa (v primeru usmerjene antene), odklon maksimalne vrednosti od sredine izbranega sektorja in smernost izmerjenega smernega diagrama.



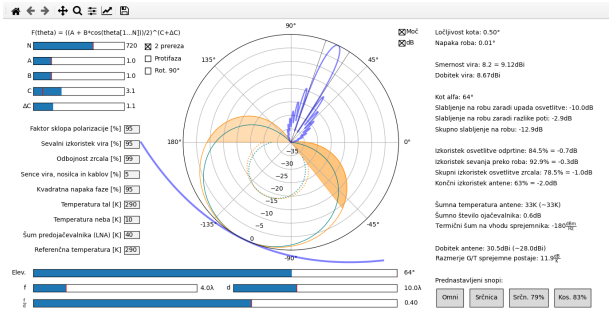
Slika 1: Grafični prikaz meritve s skripto `rotator.py`.

7 Obdelava in prikaz meritev

Knjižnica `liblab.py` v osnovi ni namenjena poobdelavi in prikazu meritev, saj v ta namen obstajajo primerni ter zmogljivi odprtokodni programi, kot je na primer LibreOffice Calc. Prav tako v ta namen jezik Python že ponuja različne zmogljive knjižnice, kot je na primer Matplotlib. Ta omogoča tudi risanje kvalitetnih polarnih grafov, kar se v praksi pokaže kot šibkost drugih odprtokodnih rešitev. Ker se v praksi kaže potreba po programski opremi za risanje polarnih grafov, je bil del razvoja opravljen tudi v to smer. Skripta `csvplot.py` (podobno kot grafični način skripte `rotator.py`) omogoča izpis vrednosti analize izmerjenih smernih diagramov ter polarni izris poljubnih meritev z uporabo knjižnice Matplotlib. Dodatno omogoča tudi grobo in fino rotacijo prikazanih smernih diagramov ter polarni kurzor za hitro odčitavanje meritev.

V drugem načinu delovanja skripta `csvplot.py` izriše kartezični graf, namenjen hitremu vpogledu v daljše datoteke CSV (na primer pri dolgotrajnem zajemu s skripto `recorder.py`). Pri tem vsak kanal (stolpec) izriše na skupni ali svoji ordinatni osi in le-te označi z glavo posameznega stolpca v datoteki CSV. Posebnost tega načina je možnost delovanja s hitro knjižnico za branje tekstovnih datotek `cStringIO`, kar omogoča hiter izris (nekaj sekund za več milijonov vrstic).

Poleg zajema, obdelave in prikaza meritev, je bila za potrebe dela na doktorski nalogi razvita tudi skripta za naprednejšo analizo smernega diagrama kot primarnega vira za osvetlitev simetrične parabolične antene za satelitske komunikacije. Skripta `illumer.py` zaenkrat omogoča branje, sintezo in prikaz smernega diagrama vira, poleg tega shematsko prikaže tudi zrcalo glede na nastavljene vhodne parametre (goriščna razdalja, premer zrcala, elevacija nad obzorjem) in približen izračun končnega smernega diagrama zrcala. Vmesnik je prikazan na sliki 2. Upošteva tudi vhodne izkoristke, faktorje in temperature za izračun šumnih veličin. Poleg analize smernega diagrama vira in zrcala zna izračunati tudi slabljenje na robu zrcala, izkoristek osvetlitve zrcala, šumno temperaturo antene, spektralno gostoto moči sprejetega termičnega šuma in razmerje G/T (dobitek antene proti šumni temperaturi sprejemnega sistema).



Slika 2: Grafični vmesnik programa za analizo osvetlitve zrcalne antene (`illumer.py`).

8 Oddaljen dostop

Ker je primarni uporabniški vmesnik razvitih skript ukazna vrstica, lahko knjižnica `liblab.py` in pripadajoče merilne skripte tečejo na najrazličnejših napravah. Tekstovni vmesnik hkrati omogoča trivialen oddaljen dostop in zagon na katerikoli primerni napravi, na primer s pomočjo SSH (ang. *secure shell*), ki tipično omogoča tudi oddaljeno kopiranje datotek s SCP (ang. *secure copy*). Možnost poganjanja avtomatizirane meritve iz manjše naprave, kot na primer kartičnega računalnika Raspberry Pi, omogoča dodatno svobodo pri postavitvi merilnih instrumentov in ne okupira uporabnikovega računalnika med zajemom meritev.

9 Zaključek

V tem delu je predstavljen koncept knjižnice za odprtokodno avtomatizacijo zajema ter obdelavo in prikaz meritev v radiofrekvenčni tehniki. Predstavljene so programske smernice in ugotovitve pri povezovanju merilnih instrumentov preko standardnih vodil kot na primer RS-232, USB, Ethernet/LXI in GPIB. Taka knjižnica skupaj s primeri merilnih skript je trenutno še v fazi razvoja, z objavo dosedanjih poskusov razvoja pa se preverja odziv znanstvene skupnosti, ki bo usmerjal nadaljnje delo.

Zahvala

Raziskovalni program št. P2-0246 je sofinancirala Javna agencija za raziskovalno dejavnost Republike Slovenije iz državnega proračuna.

Literatura

[1] M. Vidmar: Visokofrekvencni vir z ulomkovno zanko, 2017, http://lea.hamradio.si/~s53mv/fpll/fpll.html

[2] M. Vidmar: Lock-in sprejemnik z virom do 12GHz, 2017, http://lea.hamradio.si/~s53mv/lockin/lockin.html

[3] M. Vidmar: Amaterska antenska merilnica, 2017, http://lea.hamradio.si/~s53mv/aam/aam.html

[4] Python.org - Documentation, https://www.python.org/doc/

[5] AR488 Arduino GPIB Interface, https://github.com/Twilight-Logic/AR488/

[6] J. Ivković in J. L. Ivković: Analysis of the performance of the new generation of 32-bit Microcontrollers for IoT and Big Data Application, *Proc. of the International Conference on Information Society and Technology (ICIST)*, 2017