Zajem podatkov akustične kamere na vgrajenem sistemu preko serijskega vmesnika

Jure Špeh, Andrej Trost, Damjan Zadnik

Univerza v Ljubljani, Fakulteta za elektrotehniko, Tržaška 25, Ljubljana
E-pošta: jure.speh@gmail.com

Acoustic camera data acquisition on the embedded system via serial interface

In the development of embedded systems with sensors, software on a personal computer is crucial for bidirectional communication with the device and data processing. We developed a prototype acoustic camera that captures signals from 16 microphones using a programmable FPGA board. For evaluation, we created Verilog HDL code and PC software to capture and process data via a serial interface.

The software handles communication settings, command sending, and data reception. The received data is automatically processed and stored for analysis with tools like Matlab and LabView. We also added functionality to convert raw data into WAV audio files. The modular software design is adaptable for other data acquisition applications with minor adjustments.

This article outlines the project's background and software development requirements, followed by the solution design and the concepts and technologies used. Finally, we present the results and suggest future improvements.

1 Uvod

V procesu razvoja vgrajenih sistemov s senzorji potrebujemo programsko opremo na osebnem računalniku, ki omogoča dvosmerno komunikacijo z napravo in prenos ter obdelavo podatkov. V sklopu razvoja prototipa akustične kamere smo razvili namizni program za komunikacijo in zajem testnih zvočnih vzorcev. Prototip akustične kamere z razvojno ploščo in programirljivim vezjem zajema ter shranjuje signale 16 mikrofonov. Za prenos podatkov smo uporabili serijski vmesnik in serijski pretvornik USB. Programska oprema omogoča nastavljanje parametrov komunikacije, pošiljanje ukazov in sprejemanje podatkov. Ti so avtomatsko obdelani in shranjeni v datoteke za poznejše analize z drugimi orodji, npr. Matlab in LabView. Dodali smo tudi funkcionalnost pretvorbe surovih podatkov v zvočne datoteke WAV (ang. Waveform Audio). Zasnova programske opreme je modularna, z manjšimi prilagoditvami uporabniškega vmesnika in logike pa je primerna za uporabo v drugih sorodnih aplikacijah zajema podatkov preko serijskega vmesnika.

V tem članku najprej opišemo ozadje projekta s poudarkom na opisu razvite vgrajene naprave in predstavimo zahteve za razvoj programske opreme. Sledi postopek načrtovanja rešitve z opisom konceptov in tehnologij, ki smo jih uporabili v fazi razvoja. Na koncu prikažemo rezultate in podamo predloge za nadaljnje izboljšave.

2 Zajem podatkov akustične kamere

Programska oprema za zajem podatkov akustične kamere je bila razvita v okviru projekta razvoja sistema za spremljanje hrupa in meritve zvokov na področju biodiverzitete [1]. Hrup predstavlja veliko tveganje za zdravje in kakovost bivanja v urbanih središčih. Za lažjo analizo podatkov se uporablja zemljevid hrupa, na katerih je mogoče locirati kritična območja, kjer je potrebno ukrepanje.

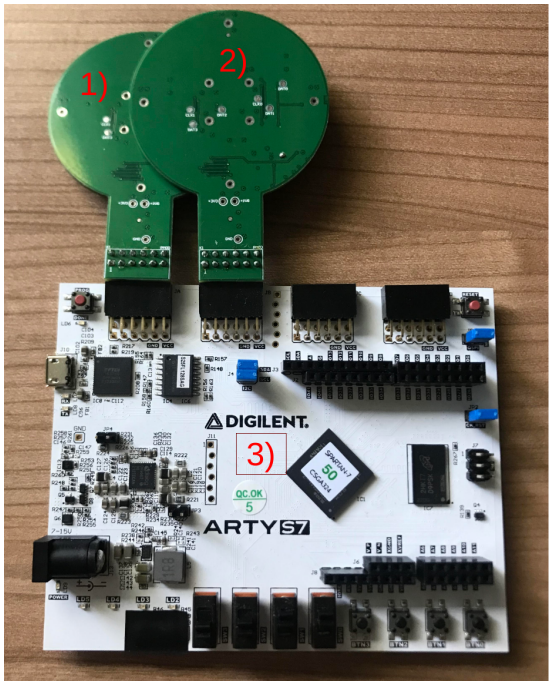
Merjenje zvoka na področju biodiverzitete se uporablja za učinkovit nadzor in zaščito ogroženih živalskih vrst. Z uporabo zvočnih podatkov se lahko izvaja lokalizacija živali [2]. Navadno se v ta namen uporabi več medsebojno časovno sinhroniziranih mikrofonskih enot. Na podlagi časovnih razlik med prihodom zvoka je mogoče oceniti lokacijo. Poleg tega zvok nudi tudi razpoznavanje posameznih primerkov, velikost in gostoto živalskih vrst na določenem območju.

2.1 Strojna oprema

Celotna strojna oprema se nahaja na sliki 1. Oznaki *1)* in *2)* predstavljata tiskanini z mikrofonskima poljema (ang. microphone array), oznaka *3)* pa tiskano vezje FPGA.

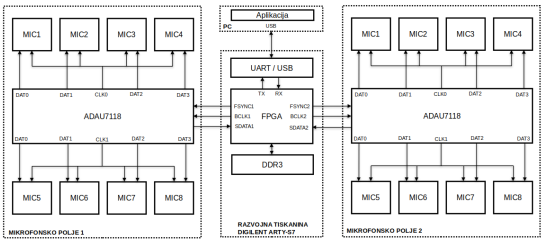
Na tiskanini mikrofonskega polja je nameščenih osem mikrofonov MEMS tipa Infineon IM73D122V01XTMA1, ki se je ob raziskavi trga izkazal kot optimalna rešitev glede na kombinacijo karakteristik, cene in dobavljivosti. Izbran mikrofon ima izhod tipa PDM, ki se uporablja pri pretvorbi analognega napetostnega signala v enobitni digitalni tok z modulacijo gostote pulzov. Višje amplitude vzorčenega signala se pretvorijo v večjo gostoto bitov. V podatkovnem listu je navedeno razmerje signal-šum (ang. signal to noise ratio - SNR) jakosti 73 dB [3].

Poleg mikrofonov se na tiskaninah nahaja tudi integrirano vezje ADAU7118, ki omogoča sočasen zajem podatkov z osmih mikrofonov in prenos podatkov. Blokovni diagram na sliki 2 prikazuje povezave in potek komunikacije med komponentami strojne opreme. Strojna oprema ima dva načina delovanja. V načinu snemanja mikrofonski polji pošiljata podatke na tiskanino FPGA (skupno 16 mikrofonov oz. kanalov). Ta podatke FPGA vpisuje v



Slika 1: Tiskanina FPGA (oznaka *3*) in tiskanini z mikrofonskima poljema (oznaki *1* in *2*).

zunanji pomnilnik DDR3. V načinu predvajanja FPGA pošilja podatke iz pomnilnika preko serijskega vmesnika na namizno aplikacijo na osebnem računalniku. Za prenos podatkov smo uporabili hitrost 1,625 Mbaud.



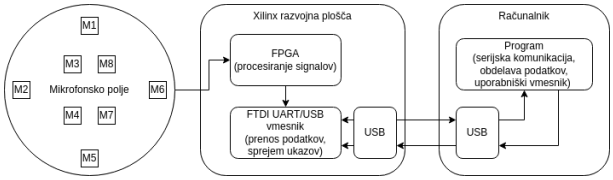
Slika 2: Blokovni diagram strojne opreme.

Za procesiranje in pošiljanje podatkov smo zaradi dobavljivosti, ustrezne zmogljivosti in ugodne cene uporabili tiskanino FPGA Arty-S7.

2.2 Zahteve za izdelavo programske rešitve

Na sliki 3 je prikazan diagram glavnih interakcij med programsko in strojno opremo, ki so bile določene v specifikaciji programske opreme. Mikrofonski polji pošiljata podatke na FPGA, ki procesira signale in preko vmesnika FTDI UART/USB komunicira s programom na računalniku. FPGA je z računalnikom povezan preko kabla USB.

Osnovne zahteve vključujejo sposobnost nastavljive interakcije s serijskimi vrati, dvosmerno komunikacijo, obdelavo in vizualizacijo prejetih podatkov, zapisovanje v tekstovne datoteke (TXT) ter pretvorbo shranjenih podatkov v glasovne datoteke (WAV).



Slika 3: Diagram interakcij med strojno in programsko opremo.

3 Načrtovanje programske rešitve

3.1 Uporabljeni koncepti

Programsko rešitev smo razvili v programskem jeziku Python z uporabo ogrodja za izdelavo uporabniških vmesnikov PySide6 [4], ki je uradni vmesnik Python ogrodja Qt. Za Python smo se odločili zaradi potrebe po hitrem razvoju in dela z veliko količino podatkov. Aplikacija je bila uporabljena med razvojem strojne opreme, zato so bile potrebne enostavne in hitre prilagoditve. Knjižnica Python PySerial za serijsko komunikacijo je dobro podprta na vseh glavnih operacijskih sistemih na katerih naj bi bila rešitev uporabljena.

Serijska komunikacija se navadno uporablja za prenos podatkov med računalniki in perifernimi napravami ali mikrokontrolerji. Uporabili smo protokol UART preko povezave USB. To je asinhron protokol brez ure za sinhronizacijo pri pošiljanju in branju podatkov. Za namen učinkovite komunikacije smo definirali višjenivojsko strukturo serijskih paketov, ki je predstavljena v tabeli 1. Vsak paket se začne in konča z začetnim in zaključnim bajtom. Začetnemu sledita bajta, ki predstavljata število podatkovnih bajtov. Ker lahko v sistemu preko serijske komunikacije komunicira več naprav, smo dodali oznaki za sprejemnika in izvor. Podatki so ASCII kodirane vrednosti, ki so v naprej dogovorjene. Sledi ukaz, ki je povezan z določenim dejanjem, ki ga želimo s sporočilom sprožiti. Za ukazom se nahajajo opcijski podatki, kjer lahko ukazu dodamo parametre ali pod ukaze. Za tem se na podlagi podatkovnih bajtov izračuna kontrolna vrednost z uporabo XOR logične funkcije na vseh bajtih razen STX, CRC in ETX bajta.

Tabela 1: Struktura komunikacijskega paketa in primer

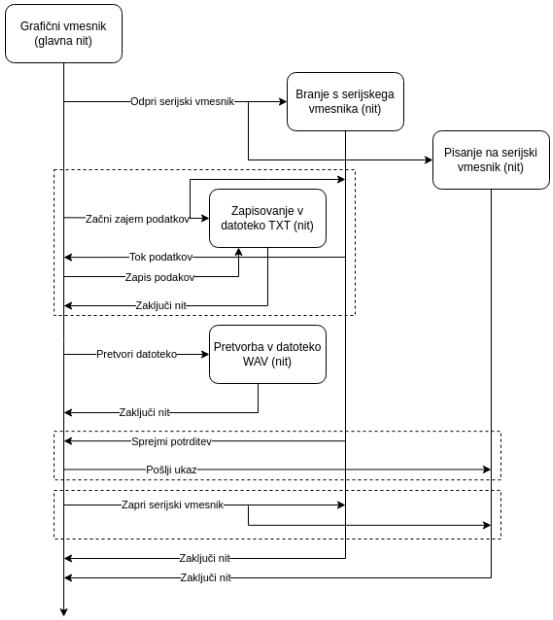
Oznaka	**Opis**	**Primer**
STX	Začetni bajt	0x02
NUMH	Št. bajtov (H)	0x00 (ni visokega bajta)
NUML	Št. bajtov (L)	0x01 (ni nizkega bajta)
DST	Sprejemnik	0x50 (‘P’) - PC
SRC	Izvor	0x46 (‘F’) - FPGA
COMM	Ukaz	0x41 (‘R’) - Recording
DATA	Podatki	(opcijsko)
CRC	XOR	0x69 (XOR)
ETX	Zaključni bajt	0x03

Program mora biti sposoben tudi zajema in pretvorbe nestrukturiranih podatkov akustičnih vzorcev. Pri tem načinu komunikacije se ne uporablja podatkovnih paketov iz tabele 1. S serijskega vodila se v vsakem ciklu zanke prebere vse podatke, ki so v tistem trenutku na

voljo. Podatki so nato v surovi obliki poslani v obdelavo z uporabo programskih signalov. Signali v knjižnici PySide6 so obvestila, ki jih oddajajo gradniki, ko se nekaj zgodi. V tem primeru se ob oddaji signala proži funkcija povratnega klica, ki poskrbi za zapis podatkov v tekstovno datoteko v ločeni niti.

3.2 Arhitektura aplikacije

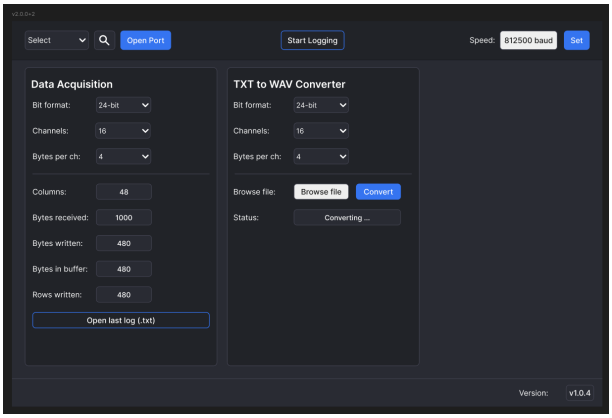
Aplikacija uporablja arhitekturo, ki je prikazana na sliki 4. Sestavljena je iz glavne niti v kateri teče uporabniški vmesnik. Ob odprtju serijskega vmesnika se ustvarita dve novi niti za komunikacijo z vmesnikom. Komunikacija med glavno in pomožnimi nitmi poteka preko deljenega spomina, signalov in komunikacijskih vrst. Pomožni niti za serijsko komunikacijo vsebujeta neskončni zanki, ki blokirata na branju komunikacijske vrste za zapisovanje na vmesnik in branju s serijskega vmesnika. Ker sta niti večino časa v blokiranem stanju, se sprostijo viri za izvajanje glavne zanke uporabniškega vmesnika. Na ta način se ustvari navidezno sočasno delovanje in izboljša odzivnost uporabniškega vmesnika. Za pretvorbo tekstovne datoteke s podatki v datoteko WAV smo uporabili večnitnost, ker gre v osnovi za vhodno izhodni proces branja in pisanja podatkov med datotekami. Vsak paket, ki je prebran s serijskega vmesnika mora biti ustrezno obdelan. Podatkovni paketi se zapišejo v tekstovno datoteko v svoji niti, saj gre tudi v tem primeru za vhodno izhodni proces.



Slika 4: Večnitna arhitektura aplikacije.

3.3 Načrtovanje uporabniškega vmesnika

Pred razvojem aplikacije smo prototip uporabniškega vmesnika izdelali z programskim orodjem Figma. To nam omogoča razmislek o dizajnu aplikacije, postavitvi gumbov, vnosnih polj in podatkovnih segmentov. Vse to prihrani čas pri samem razvoju in izboljša uporabniško izkušnjo. Izgled aplikacije se enostavno popravlja z manipulacijo grafičnih elementov. Prvotni izgled je prikazan na sliki 5. V osnovi smo vizualni del razdelili na dva dela: zgornji del, ki je namenjen interakciji s serijskim vmesnikom in osrednjem delu s segmentoma za zajem podatkov ter pretvorbo v zvočno datoteko. Tekom razvoja projekta in testiranja aplikacije so sicer prišle potrebe po dodatnih funkcionalnostih in spremembah postavitve elementov za učinkovitejše delo.



Slika 5: Prvotni izgled aplikacije v programskem orodju Figma.

4 Razvoj programske rešitve

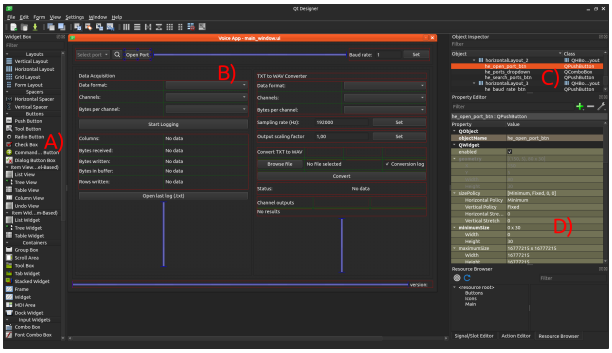
Uporabniški vmesnik smo razvili z uporabo orodja QtDesigner, ki je namenjeno grafičnemu razvoju uporabniških vmesnikov. Na sliki 6 so označene glavne komponente orodja:

A) Nabor standardnih komponent (postavitve, gumbi, polja,...) (Widget Box).

B) Glavno okno za umestitev komponent in definicijo relacij med njimi.

C) Okno za hierarhični pregled umeščenih komponent (Object Inspector).

D) Okno za nastavitev lastnosti posameznim komponentam (Property Editor).

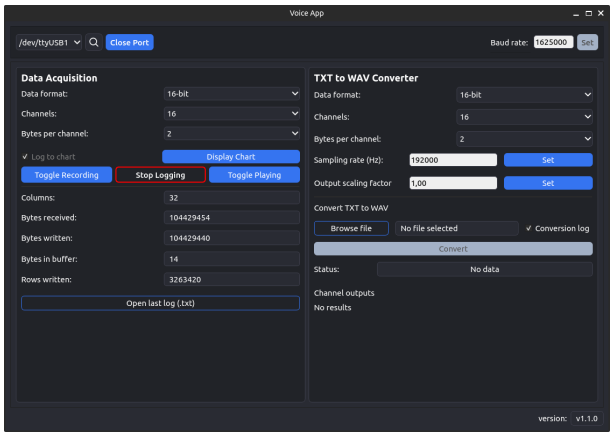


Slika 6: Uporaba orodja Qt Designer.

Končni izgled razvite aplikacije je prikazan na sliki 7. Uporabnik aplikacije preko kabla USB na osebni računalnik priključi tiskano vezje FPGA. Na spustnem seznamu

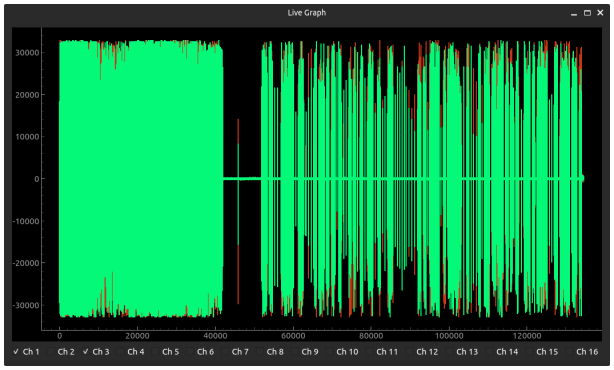
izbere ustrezno serijsko napravo, nastavi hitrost komunikacije in odpre vmesnik.

V segmentu *Data Acquisition* je mogoče nastavljati parametre zajema podatkov, ki se uporabljajo pri strukturiranju zajetih akustičnih podatkov v tekstovne datoteke. Z gumbom *Toggle recording* vezje FPGA začne zajem podatkov. Ob ponovnem pritisku se zajem ustavi. Z gumbom *Start logging* se odpre nova tekstovna datoteka, kamor se vpisujejo podatki. S pritiskom na gumb *Toggle playing* se začne prenos podatkov.



Slika 7: Uporabniški vmesnik delujoče aplikacije.

Med prenosom podatkov je mogoče spremljati parametre prenosa. Za lažji pregled zajetih podatkov lahko uporabnik uporabi tudi funkcionalnost vizualizacije na grafu (slika 8) ali odpre zadnjo podatkovno datoteko za hiter pregled podatkov.



Slika 8: Graf pretvorjenih zajetih vrednosti za kanala 1 in 3 v realnem času.

V segmentu *TXT to WAV Converter* je mogoče pretvarjati zajete tekstovne datoteke v glasovne datoteke WAV. Pred pretvorbo je potrebno definirati podatke za pretvorbo, ki se morajo ujemati s tistimi ob zajemu. V ločeni niti se ustvari proces, kjer se za vsak kanal zgenerira ločena zvokovna datoteka.

5 Rezultati

Velikost pomnilnika DDR3 je 256 MB in trenutno poteka razvoj hitrejšega komunikacijske vmesnika preko vrat USB 3.0. Pri testiranju programa smo izmerili povprečno hitrost prenosa podatkov 520 kB/s. Zvočni posnetek šestnajstih kanalov ob uporabi vzorčne frekvence 192 kHz se prenaša 19 minut in 30 sekund. Datoteka zasede 610 MB prostora v pomnilniku računalnika. Pretvorba te datoteke v 16 zvočnih posnetkov WAV traja okrog 15 sekund. Aplikacijo smo testirali na operacijskih sistemih Windows 10 in 11 (izvršljiva datoteka exe) ter Linux Ubuntu 22.04 LTS. Do razlik ali težav, ki bi bile povezane z različnimi operacijskimi sistemi ni prihajalo.

Največji izziv se je pokazal pri vizualizaciji podatkov na grafu v realnem času. V vsakem ciklu se ponovno izrišejo vse točke na grafu. Ta del smo nekoliko optimizirali, vendar se posodabljanje grafa kljub temu upočasnjuje s povečevanjem števila podatkov. Proces bi lahko pohitrili z izpuščanjem posodobitev, ki bi preveč zamujale. Izpuščeni slikovni okvirji ne bi predstavljali težav za uporabnika, saj se vsi podatki, ki so pripravljeni za izris shranijo v delovnem pomnilniku.

Jezik Python na splošno velja za enega počasnejših jezikov. Hitrost branja podatkov je bila ustrezna in zaradi uporabe jezika Python ni prihajalo do zastajanja podatkov na serijskem vmesniku. Hitrost prenosa podatkov bi lahko izboljšali s posodobitvami na nivoju strojne opreme, npr. z uporabo hitrejšega vmesnika USB 3.0.

6 Zaključek

Odločitev za izdelavo aplikacije po meri je pohitrila proces razvoja in omogočila testiranje strojne opreme in programske opreme ter algoritmov na vezju FPGA. Rešitev, ki smo jo razvili v programskem jeziku Python se je s finančnega, časovnega in zmogljivostnega okvirja izkazala za ustrezno. V nadaljevanju bi bilo potrebno optimizirati proces vizualizacije večkanalnih podatkov. Program je mogoče v prihodnosti z nekaj prilagoditvami uporabiti v sorodnih aplikacijah razvoja vgrajenih sistemov.

Zahvala

Delo je bilo sofinancirano iz programa ARIS “Spremljanje urbanega hrupa in biodiverzitete za zeleno prihodnost z akustičnim IoT radarjem s klasifikacijo dogodkov na osnovi UI”J7-50042.

Literatura

[1] J. Prezelj: Spremljanje urbanega hrupa in biodiverzitete za zeleno prihodnost z akustičnim IoT radarjem s klasifikacijo dogodkov na osnovi UI, Interna dokumentacija projekta ARIS, Fakulteta za strojništvo, Univerza v Ljubljani, 2023.

[2] Tessa A. Rhinehart, Lauren M. Chronister, Trieste Devlin, Justin Kitzes, Acoustic localization of terrestrial wildlife: Current practices and future opportunities, Ecology and Evolution, Vol.10, (2020), pp.6794–6818

[3] Infineon: IM73D122V01, podatkovni list, https://www.infineon.com/dgdl/Infineon-IM73D122-DataSheet-v01_00-EN.pdf?fileId=8ac78c8c83cd3081018409cafdfb46db

[4] PySide: Qt for Python, https://doc.qt.io/qtforpython-6/index.html