Erinevad signaalid ja nende sämplimine

Sisukord

Aastal 2021 on vastutav juhendaja: Kristo Allaje <kristo.allaje@ut.ee> 2021-02-19|19:31:00 +0200

1. Ülevaade

Juhendist on olemas ka PDF versioon.

Selle praktikumi käigus saab tutvuda tööriistadega, mida saab kasutada, et erinevatest allikatest eri liiki signaale digitaalsele kujule lugeda ja seejärel kujutada.

Juhendiga töötades pidage silmas

Praktikumijuhend sisaldab praktikumiülesannete kirjeldusi koos abistava infoga. Sealhulgas on toodud ka juhised oma süsteemi seadistamiseks. Töö õnnestumisele kaasa aitamiseks oleme teinud ka hulga märkmeid probleemsete kohtade osas. Selle info oleme jaganud tüübi järgi järgmise kolme kategooria vahel.

Siin toodud info võib aidata ülesande kiiremini või kavalamalt lahendada.

Siin toodud info võib aidata veaohtlikku kohta vältida.

Siin toodud info võib aidata vältida kahju riistvarale, tarkvarale või iseendale.

Töövahendid

1. Python 3.8
2. NumPy
3. PyQtGraph
4. PyVISA
5. SciPy IO
6. OpenCV 4.5
7. Audacity 2.4

Töö esitamine praktikumi lõppedes

Soovitame tungivalt hiljemalt iga alamülesande lõpetamise järgselt lisada uus versioon lahendusfailist oma harusse. GitLab keskkonnas on vaja luua *Merge request* ainult üks kord praktikumi jooksul.

1. Alustuseks veenduge, et olete oma kohaliku giti kausta **sees**
   a. Aktiivse kataloogi kontrollimiseks saate kasutada käsku `pwd` ning muutmiseks käsku `cd`
2. Kontrollige, et olete harul <**tudeng/eesnimi-perenimi**>
   a. Kasutage selleks käsku `git branch -l`
3. Sisestage käsk `git status`, et näha, kas on veel registreerimata muudatusi
   a. Kui mõni fail on muutunud, siis saate uue versiooni registreerimiseks kasutada käske `git add -p <muutunud-fail>` ja `git commit`
4. Kasutage käsku `git push`, et laadida kohalikud versioonid serverisse
   a. Lugege kindlasti käsu väljundit ning kahtluse korral kontrollige, kas failid jõudsid serverisse
5. Praktikumi lõppedes navigeerige veebilehele https://gitlab.ut.ee ning registreerige menüüde kaudu *Merge request* suunaga oma harust <**tudeng/eesnimi-perenimi**> harusse **master**
   a. Enne järgmist praktikumi vaatavad juhendajad teie pakutud versioonid üle ja liidavad need sobivuse korral master-nimelisse harusse

Enne alustamist

Repositooriumisse on lisatud uut sisu. Selle sisu kasutamiseks peate ta oma harusse liitma.

1. Alustuseks veenduge, et olete oma kohaliku giti kausta **sees**
   a. Aktiivse kataloogi kontrollimiseks saate kasutada käsku `pwd` ning muutmiseks käsku `cd`
2. Navigeerige master-nimelisele harule
   a. Kasutage selleks käsku `git checkout master`
3. Laadige alla ning liitke serveri versioon
   a. Kasutage selleks käsku `git pull`. Pange tähele ka käsu väljundina antud infot
4. Navigeerige tagasi oma harule
   a. Kasutage selleks käsku `git checkout tudeng/eesnimi-perenimi`. Muutke kindlasti käsus kasutatav haru nimi
5. Liitke kõik master-nimelisel harul olevad muudatused oma harule
   a. Kasutage selleks käsku `git merge master`

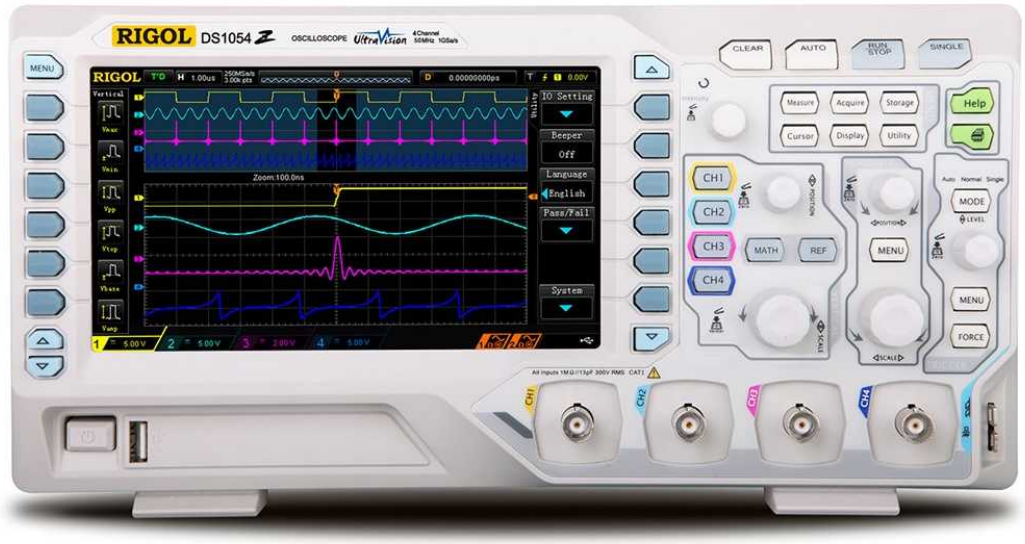
2. Kohustuslikud ülesanded

1. Ülesanne: Ostsilloskoobi kasutamine

Sissejuhatus

Eelmises praktikumis oli võimalus proovida signaali mõõtmist Arduino Nanoga. Kohati võib olla vaja sämplida signaale, mille pinge väärtus tõuseb üle 5V või mis on kiiremad kui mõned kilohertsid, ning sellega jääb tavapärane Arduino juba hätta.

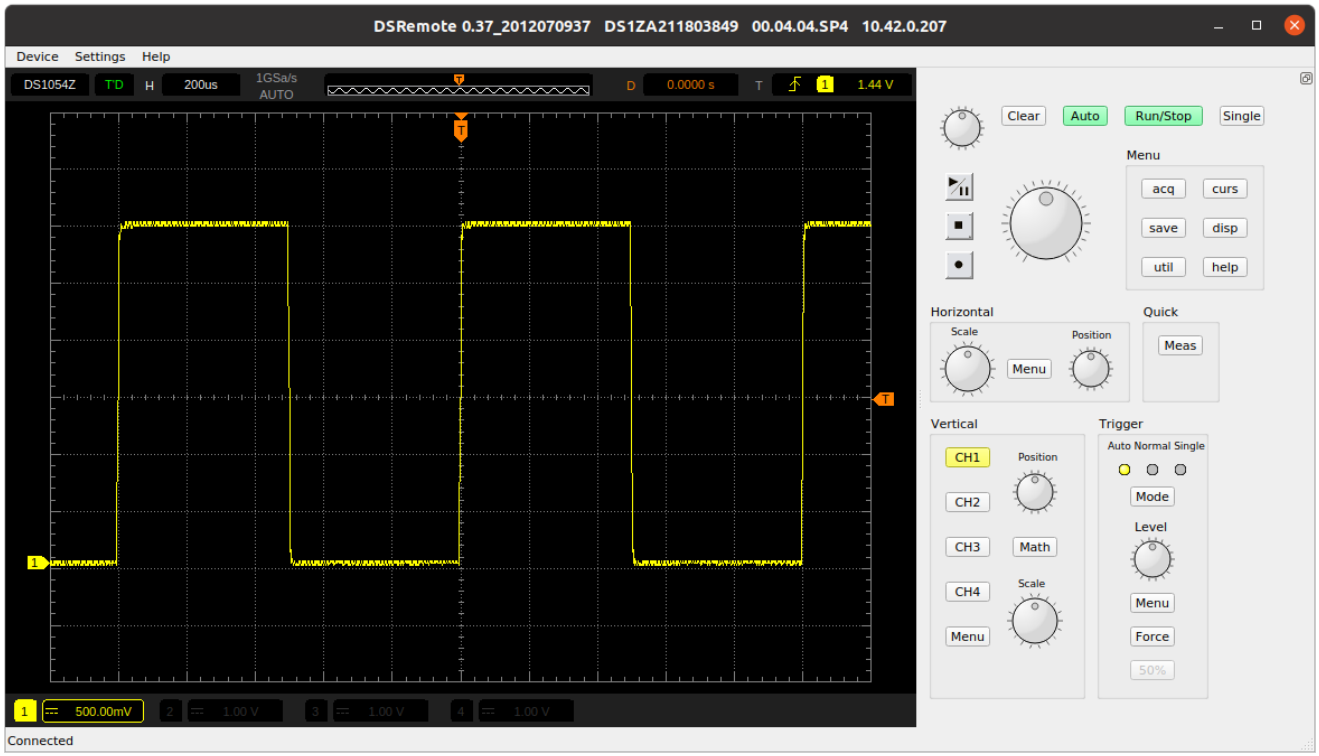
Praktikumis kasutame selleks aga just signaali mõõtmiseks disainitud riistvara, ostsilloskoopi. Klassis on meil kasutada Rigol DS1054Z, millest on huvi korral võimalik saada põhjalik ülevaade sellest videost. Rigol pakub ka endapoolseid materjale, mis tutvustavad klassiruumis olevate ostsilloskoopide võimalusi lühidalt. Rigoli pakutavate materjalidega saab tutvuda siin.



Joonis 1. Praktikumis kasutusel olev ostsilloskoop Rigol DS1054Z.

Selles ülesandes kasutame programmi DSRemote, et kontrollida klassiruumis olevaid ostsilloskoope üle LANi. Ostsilloskoobi kasutamiseks on teil vaja luua klassiarvutiga ühendus ning avada seal rakendus DSRemote. Hosti nimi TYTI-136-DS1054 tuleb sisestada DSRemote sätete alt valides **Settings › Settings › LAN**, sisestage oma ostsilloskoobi hosti nimi ning vajutage **Apply**. Ühenduse alustamiseks tuleb valida tööriistaribalt **Device › Connect** ning programm otsib eelnevalt määratud aadressilt ostsilloskoopi.

DSRemote on oma olemuselt natukene ebastabiiline ning võib kohati oma töö veateatega lõpetada. Sellisel juhul tuleb ühendus uuesti luua.



Joonis 2. DSRemote ostsilloskoobi kaugjuhtimiseks.

Uurige programmi poolt pakutavaid võimalusi ning juhtige ostsilloskoopi üle võrgu, kindlasti tasub lugeda kasutusjuhendit `help` nupu alt. Proovige mõõta esimesel ja teisel kanalil oleva sisendsignaali sagedust, tipp-tipp amplituudi (inglise k *peak-to-peak amplitude*) ning uurige võimalusi `acq`, `save`, `util`, `disp`, `math` nuppude all.

Ülesande kirjeldus

Selles ülesandes on keelatud kasutada ostsilloskoobi automaatset režiimi.

Paluge juhendajal genereerida ostsilloskoobi teisele kanalile uus sisendsignaal ning tehke järgnevat:

1. seadke ostsilloskoobi sünkronisatsiooni lävend, et sisendsignaal püsiks ekraanil paigal,
2. leidke sisendsignaali ligikaudne sagedus,
3. leidke sisendsignaali tipp-tipp amplituud,
4. leidke, mis on sisendsignaali nihe 0 voldi suhtes (*DC offset*).

Demonstreerige oma meisterlikku kontrolli ostsilloskoobi üle juhendajale.

Kodus tuleb lahendada Esimene koduülesanne

Uurige internetist klassiruumis olevate DS1054Z ostsilloskoopide kohta. Leidke vastused järgnevatele küsimustele.

1. Mis on nende ostsilloskoopide tootja poolt määratud kõrgeim sisendsignaali sagedus, mida see ostsilloskoop suudab mõõta? Mida see tehniline kirjeldus ostsilloskoopide puhul tähendab ning mis juhtub kiirema sisendi puhul?

   Eeldatav vastus:

   - selgitab lühidalt, mis on DS1054Z ostsilloskoobi kõrgeim sisendsignaali sagedus ning viitab allikale, millest see leiti;
   - vastab küsimusele, mis juhtub kiiremate sisendite puhul.

2. Üldjuhul on tavalise ostsilloskoobi proovikul olemas 1x ja 10x režiim. Mida need režiimid endast kujutavad ning millal oleks mõistlik kumbagi režiimi kasutada?

   Eeldatav vastus:

   - kirjeldab lühidalt, mida kujutavad endast 1x ja 10x režiimid;
   - seletab, millal oleks mõistlik kumbagi režiimi kasutada.

2. Ülesanne: PyVISA

Sissejuhatus

Ostsilloskoop on võimas tööriist, kuid selle saab muuta isegi võimsamaks, kui ühendada see arvutiga. Klassis olevate mõõteriistade ja arvutite ühendamiseks kasutame aines teeki nimega PyVISA. See annab võimaluse saata ostsilloskoobile SCPI (inglise k *Standard Commands for Programmable Instruments*) käske, millega on võimalik kontrollida selle erinevaid funktsioone. Hea ülevaate SCPI ja VISA käskudest leiab Rohde&Schwarz'i kodulehel olevast mõõteriistade kaugkasutuse juhendist.

Joonis 3. Ülevaade mõõteriistade kontrollimiseks vajalikest tarkvarakihtidest. (Rohde&Schwarz VISA and VISA Tools)

Teie kirjutatav kood kasutab SCPI käske, mis edastatakse läbi PyVISA ostsilloskoobile. SCPI käsud on lihtsalt ASCII sõned, mida saab läbi PyVISA instrumendile saata. Neid kasutades on võimalik seadme sätteid muuta või sealt andmeid lugeda. Allpool on näha minimaalset näidet, mis on vajalik klassis olevate mõõteseadmetega suhtlemiseks.

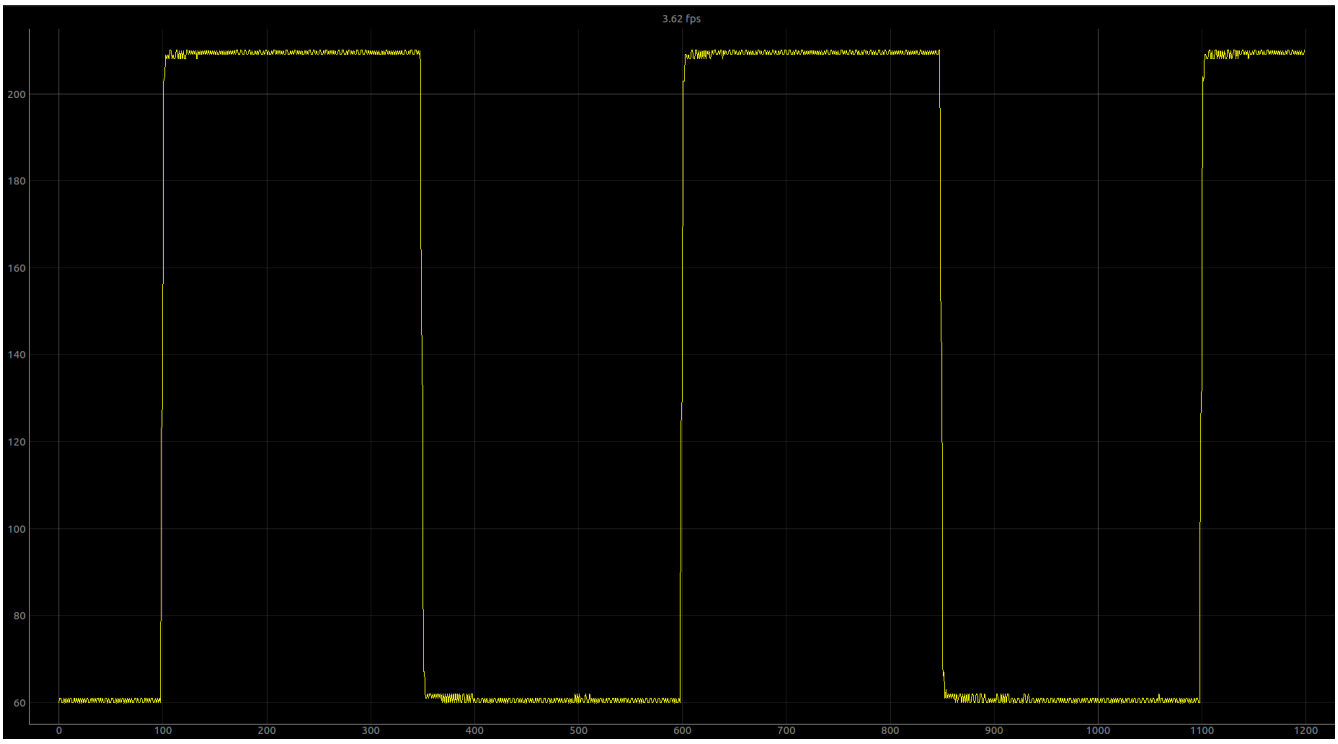
```
1 import pyvisa
2
3 # Loome PyVISA objekti, mis suudab ühenduda erinevate mõõteseadmetega
4 rm = pyvisa.ResourceManager()
5
6 # Loome ühenduse klassisruumis oleva ostsilloskoobiga
7 # Kõik seadmele saadetavad käsud ning sealt tulevad käsud lõpevad uue rea sümboliga
8 inst = rm.open_resource('TCPIP::TYTI-136-DS1054::INSTR', read_termination="\n",
  write_termination="\n")
9
10 # Küsime seadme unikaalset tuvastuskoodi
11 print(inst.query("*IDN?"))
12
13 # Sulgeme ühenduse seadmega
14 inst.close()
```

Ülesande kirjeldus

Selle ülesande eesmärk on sarnane eelmise praktikumi lõpuga. Teie ülesandeks on reaalajas küsida ostsilloskoobilt andmeid ning kujutada saadud andmepunkte graafikul. Soovitame tutvuda teie repositooriumis oleva failiga `yl2.py`. Probleemide korral on võimalik abi otsida ostsilloskoobi kasutusjuhendist ja selle programmeerimise juhendist, mis on kättesaadavad klassiarvuti töölaual või läbi eelpool toodud linkide.

Teie ülesandeks on järgnev:

1. kujutada ostsilloskoobilt küsitud andmeid reaalajas PyQtGraph'iga, värskendussagedusega 5 Hz,
    a. QTimer'i kohta leiate informatsiioni siit.



Joonis 4. Ostsilloskoobist küsitud andmepunktid kujutatuna PyQtGraph'i abil.

Kujutatu on juba päris lähedal sellele, mida on võimalik näha DSRemote ekraanil, kuid midagi on siin veel valesti. Lähemal uurimisel on selge, et X, Y-telgedel olevad väärtused on mööda ning järgnevas alamülesandes tegeleme just sellega.

2. kasutades seadmelt saadud infot, skaleerida kujutatav signaal X,Y-teljes õigesse skaalasse,
    a. X-telje andmepunkte on võimalik ümber muuta sekunditeks kasutades np.arange ning `decode_waveform_preamble` tagastatud infot X telje kohta.
    b. Y-telje andmepunkte on võimalik ümber muuta voltideks kasutades valemit: `(andmepunkt[n] - YOrigin - YReference) \cdot YIncrement`.
    c. Mõlema telje väärtuste arvutamiseks tuleb kasutada NumPy vektoriseeringut.
3. dünaamiliselt kajastada mõlema telje väärtusi, kui seadme juhtimisel neid muudetakse.
    a. Seda saab katsetada ühendudes DSRemote'ga ostsilloskoobi külge ning siis käivitades oma koodi. Muutes DSRemote'ga horisontaalset sammu peab ka teie reaalajas kuvatav graafik seda kajastama.

Näidake oma lahendust juhendajale ning tehke commit!

Kodus tuleb lahendada Teine koduülesanne

Tutvuge ostsilloskoobi kasutusjuhendiga ja programmeerimise juhendiga.

1. Mida muud on veel võimalik läbi SCPI käskude ostsilloskoobiga teha? Tooge kaks näidet ostsilloskoobi pakutavatest võimalustest, mis võiksid teid tulevikus signaalide uurimisel abistada.

   Eeldatav vastus:

   - kirjeldab lühidalt, mida valitud funktsioon teeb ning miks te just selle valisite;
   - demonstreerib pseudokoodiga, kuidas saata vastavat käsku mõõteriistale kasutades PyVISA teeki ja kuidas kasutada seadmelt saadud vastust.

3. Ülesanne: Audacity

Sissejuhatus

Praeguseks oleme tutvunud kahe erineva võimalusega, kuidas signaali lugeda. Samas ei ole need kindlasti ainukesed meetodid signaali salvestamiseks. Erinevaid liideseid, mille kaudu signaali igapäevaselt edastatakse, on väga palju. Üks neist on laialt levinud 3.5 millimeetrine audio pesa, millega saab ühendada tavalise arvuti külge kõlareid, kõrvaklappe, mikrofone või viimaseid kahte kombineerivaid peakomplekte. Tegelikult ei ole selle liidese puhul mikrofonist signaali lugemine midagi keerulisemat kui juhtmetel pingena esitatava signaali mõõtmine, kasutates analoog-digitaal muundurit (ADC) nagu ka eelmises praktikumiülesandes.

Selleks, et saada hea kvaliteediga heli, on olulised peamiselt kaks parameetrit.

Mõõtmise sagedus

Peab jälgima, et signaali mõõdetakse füüsilisest maailmast piisavalt tihti. Teisisõnu, on oluline, et sämplimissagedus oleks piisav. Kui sämplida liiga madala sagedusega, on vahepealne info puudu ja selle kohta tuleb taasesitamisel teha mingid eeldused. See, kui sagedasti tuleks sämplida ehk sooritada mõõtmisi, sõltub sellest, mis on taasesitamise võimekus ja kui palju on inimese kõrv võimeline vahet tuvastama.

Kvantimistasemete arv

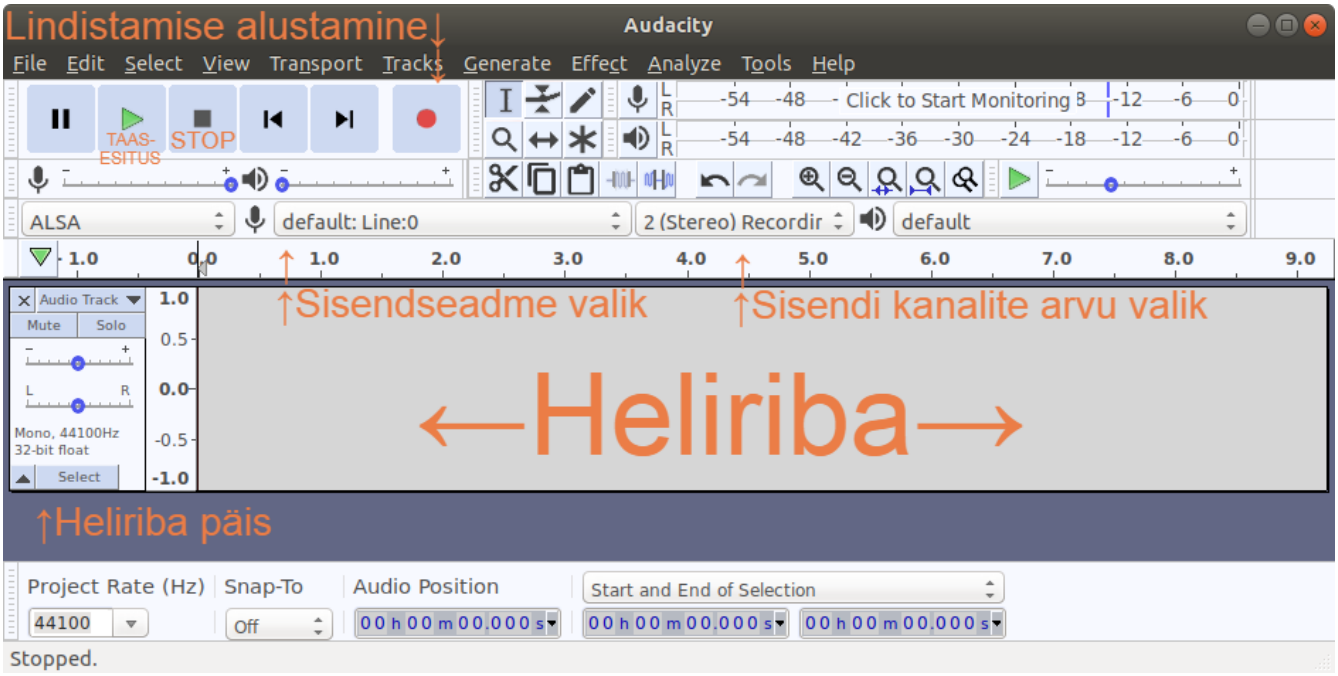
Protsessi, mis võtab füüsilisest maailmast väärtuse, mis on mingi punkt lõputus hulgas, ja seab sellele vastavusse lähima väärtuse lõplikus hulgas, nimetatakse kvantimiseks. Selle protsessi käigus toimub enamasti teatud lähendamine.

Lihtsa näitena võib kvantimise kohta tuua täisarvuks ümardamise. Kui tulemust on võimalik salvestada vaid täisarvulise väärtusena ja füüsilises maailmas on tulemuseks 3,156798329, siis saab selle salvestada kui 3. Samas on ka siin toodud esialgne väärtus 3.156798329 mingi lähend ja füüsiliselt on tulemusel veel täpsem väärtus. Olgu see kasvõi 3.156798329000000000001. Seega, kui lisada kvantimisel tasemeid, on tulemus küll täpsem, kuid siiski mitte kunagi lõpmatult täpne. Tasemeid on põhjust lisada ainult seni, kuni mõõtmist teostav riistvara nii täpselt mõõta suudab.

Käesolevas ülesandes saab vaadata, kuidas muutub lindistatud helifail, kui muuta selle diskreetimissagedust programmis Audacity.

Ülesande kirjeldus

Selles ülesandes tuleb uurida, mis muutub, kui muuta salvestatud helifaili sämplimissagedust ja esituskiirust. Selleks, et seda Audacity's teha, tuleb avada aluskoodis olev fail nimega `pale_blue_dot.wav`.



Joonis 5. Audacity elementide kirjeldus

Vajutades heliriba päises nuppu *Select* ning siis tööriistaribalt **Edit › Duplicate** (`ctrl` + `D`) on võimalik heliriba dubleerida. Allpool kirjeldatud sämplimissageduse ja esituskiiruse muudatused tuleb läbi viia kopeeritud heliribadel, säilitades algse faili!

Vajutades heliriba päises *Select* nuppu, saab heliriba esituskiirust muuta vajutades **Heliriba nimi › Rate**. Sämplimissagedust saab muuta valides tööriistaribalt **Tracks › Resample...**. Viimaks, ainult muudetud heliriba kuulamiseks või faili eksportimiseks tuleks valida päisest *Solo*. See muudab ainult valitud heliriba aktiivseks ning vaigistab kõik teised.

Loodud heliribade kuulamiseks soovitame need eksportida klassiarvutis faili ning saata need kasutades Secure Copy käsku enda arvutisse. Näidet *scp* kasutamisest on näha allpool.

```
1 # Serverist faili allalaadimine
2 scp dsp-student@arvuti_ip:~/kopeeritav_fail.txt asukoht_kohalikus_arvutis
```

Ülesande arvestatud saamiseks on vaja muuta helifaili järgnevalt:

1. Esimesel juhul on esituskiirus 44100 Hz (originaalfail).
2. Teisel juhul on esituskiirus 16000 Hz.
3. Kolmandal juhul on esituskiirus 96000 Hz.
4. Neljandal juhul on vaja faili sämplimissagedust vähendada, kuni helis olev kõne pole enam arusaadav.
5. Viiendal juhul tuleb kopeerida neljandas sammus loodud heliriba ning muuta selle sämplimissagedus tagasi 44100 Hz peale.

Oma repositooriumisse tuleb salvestada Audacity projektifail, kasutades menüüst valikut **File › Save Project As**. Enne salvestamist võib kustutada ära kogu üleliigse info. See vähendab projektifaili suurust ja seega ei muuda hilisemat repositooriumi kloonimist liiga aeglaseks.

Kontrollküsimusele, et kas salvestada projekt, tuleb vastata jaatavalt.

Demonstreerige oma loodud Audacity projekti juhendajale ning tehke commit!

Kodus tuleb lahendada Kolmas koduülesanne

1. Uurige internetist Nyquisti teoreemi kohta. Aruandes tehke **oma sõnadega** loetust kokkuvõte.

   Eeldatav vastus:

   - kirjeldab lühidalt, mida kujutab endast Nyquisti teoreem.

2. Kuulake ülesande käigus loodud helifaile. Viimase heliriba sämplimissageduse taastamisel 44100 Hz peale ei muutunud heli tagasi teravaks. Mis on selle põhjuseks?

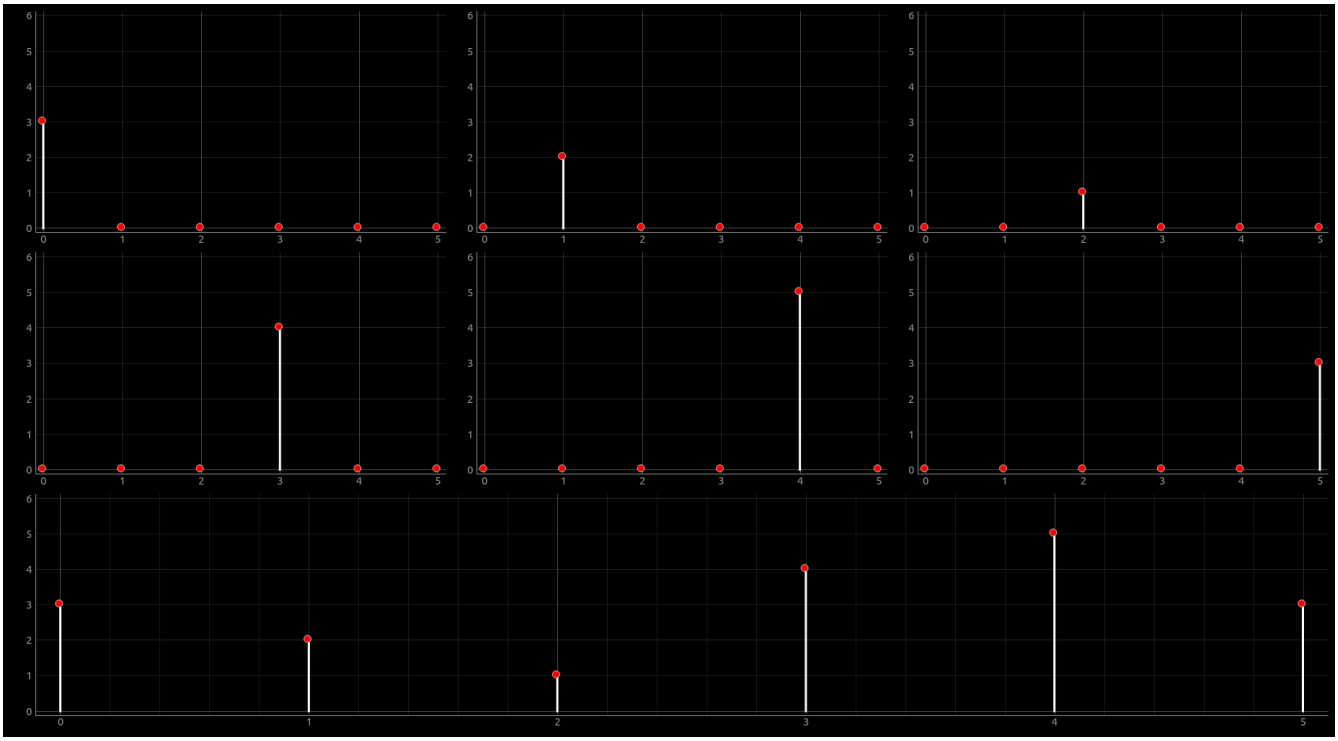
   Eeldatav vastus:

   - kirjeldab lühidalt, mis on erinevate heliribade erinevus kuulates;
   - pakub põhjuseid, miks see nii võib olla;
   - selgitab, miks viimane heliriba ei muutunud tagasi teravaks, kui muuta sämplimisagedus tagasi 44100 Hz peale.

4. Ülesanne: Impulsid

Sissejuhatus

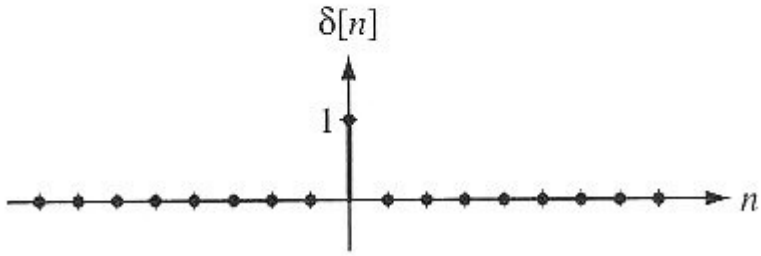
Selles ülesandes näeme, kuidas iga digitaalset signaali saab lahutada impulssideks. Impulss on signaal, mis koosneb kõikjal nullidest, välja arvatud ühes mittenullises punktis. Allpool oleval pildil on näha kuut erinevat impulssi, mis on omavahel kokku liidetud, et moodustada viimases aknas olev signaal. Eelnevates praktikumides oleme tekitanud selliseid diskreetseid signaale sämplimise abil. Kui eelmises ülesandes töödeldud signaali kuva Audacity's suurendada (kasutades `ctrl` ja hiire kerimisrullikut), on näha, et tegelikult on tegu lihtsalt signaaliga, mis on loodud väga paljude impulsside koosmõjul.



Sämplimisest võibki mõelda kui iga sämpli võtmisega sellise impulsi tekitamisest, mis on sämplimise algusaja suhtes nihkes ning kõik `N` sämplit kokku liites tekib digitaalne signaal pikkusega `N`. Digitaalsete signaalide impulssideks lahutamine on tähtis, sest see lubab signaale uurida üks sämpel korraga. Teades, kuidas süsteem reageerib impulsile, on võimalik välja arvutada selle süsteemi reaktsioon mistahes sisendile. Sellist lähenemist nimetatakse konvolutsiooniks ning see on ühe hilisema praktikumi teema.

Matemaatiliselt tähistatakse impulsse kui üht ühikimpulssi (Kronecker'i deltafunktsioon), mille amplituud on mingisuguse arvuga korrutatud ning kogu funktsioon on ajas nihutatud. Ühikimpulss on defineeritud järgnevalt:

$$\delta[n] = \begin{cases} 1, n = 0 \\ 0, n \neq 0 \end{cases}$$



Ainukene probleem, mis muudab impulsside teema keeruliseks, on nomenklatuur (nimetamise süsteem). Termineid Dirac'i delta, Kronecker'i delta ja ühikimpulss kasutatakse sagedasti samaväärsetena, kuigi see pole päris tõene.

Ülesande kirjeldus

Ülesande efektiivseks lahendamiseks soovitame uurida NumPy pakutavaid võimalusi.

Selles ülesandes kasutame teadmist, et iga impulss on oma olemuselt ühikimpulss, mida on skaleeritud ning ajas nihutatud.

Looge funktsioon, mis võtab sisendiks argumendi, mis kirjeldab ühikimpulsi pikkust. Funktsiooni väljund erinevate sisendite puhul peab välja nägema järgnev:

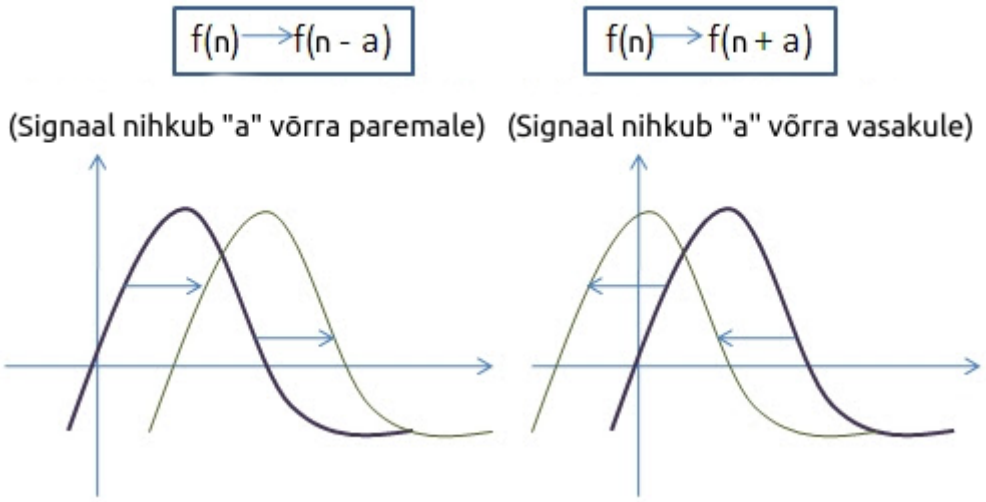
```
impulss = funktsioon(5)
# print(impulss)
[1. 0. 0. 0. 0.]

impulss = funktsioon(8)
# print(impulss)
[1. 0. 0. 0. 0. 0. 0. 0.]

impulss = funktsioon()
# print(impulss)
[1.]
```

Nüüd täiendame eelnevalt loodud funktsiooni, et lisada sellele ühikimpulsi nihutamine ajas. Matemaatiliselt on signaali nihutamine ajas defineeritud järgnevalt:

```
\begin{equation}
  f(n+a) =
  \begin{cases}
    \text{signaal liigub paremale nihke võrra} & a < 0 \\
    \text{signaal liigub vasakule nihke võrra} & a > 0 \\
    \text{signaal ei muuda oma asukohta} & a = 0
  \end{cases}
\end{equation}
```



Nagu pildi pealt näha, on signaali negatiivse arvu võrra nihutamine võrreldav viivitusega. Seda on näha ka pildilt, et nihutatud signaal (heledam) saavutab oma maksimaalse väärtuse hiljem kui algne signaal (tumedam), mida pole ajas nihutatud. See on väga sagedane, et süsteemi antud sisend ei tule kohe süsteemi väljundist välja, vaid selleks peab mingi aeg ootama ehk signaali lisatakse viivitus. Signaali positiivse arvu võrra nihutamine on võrreldav tulevikku vaatamisega ning seda pole võimalik teostada ilma signaali eelnevalt salvestamata. Alles siis, kui on salvestatud terve signaal, on võimalik ajahetkel `n` arvutada signaali väärtus ajahetkel `n+3`. Rohkem informatsiooni signaalide nihutamise kohta leiab sellest artiklist ja sellest videost.

Lisage oma eelnevalt loodud funktsioonile võimalus nihutada loodud ühikimpulssi, kui `n\leq0`. Funktsiooni väljund erinevate sisendite puhul peab välja nägema järgnev:

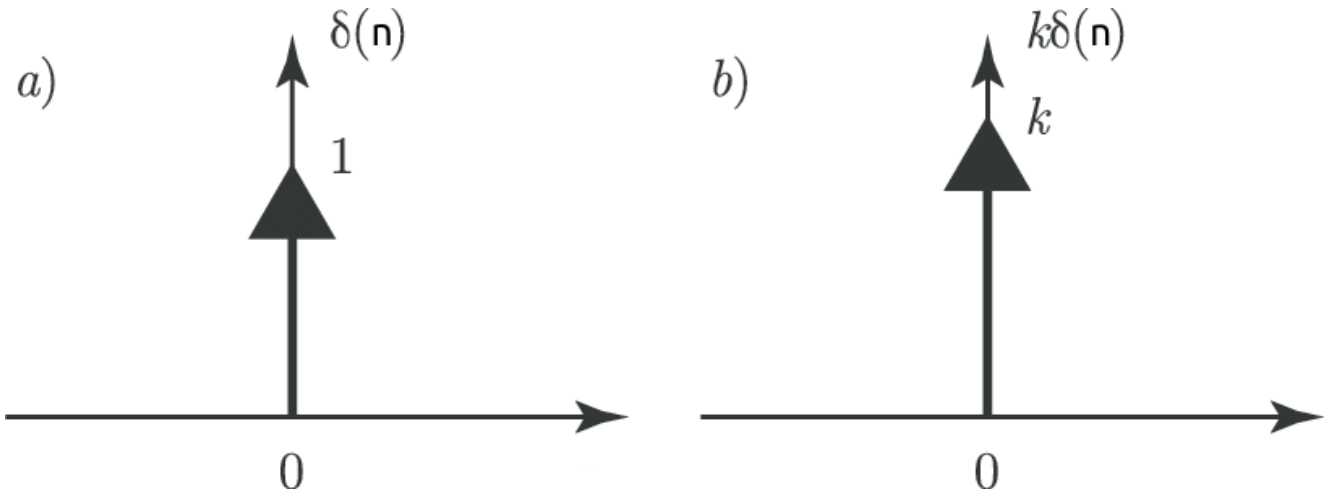
```
impulss = funktsioon(5, -2)
# print(impulss)
[0. 0. 1. 0. 0.]

impulss = funktsioon(8, -5)
# print(impulss)
[0. 0. 0. 0. 0. 1. 0. 0.]

impulss = funktsioon(1)
# print(impulss)
[1.]
```

Selleks, et oleks võimalik luua mistahes impulss, on veel vaja implementeerida ühikimpulsi

amplituudi skaleerimine. Kui tavalist ühikimpulssi tähistab tavaline `\delta[n]` siis impulssi, mille amplituud on `k`, tähistab järgnev notatsioon `k\delta[n]`.



Lisage oma eelnevalt loodud funktsioonile võimalus skaleerida loodud ühikimpulsi amplituudi `k` võrra. Funktsiooni väljund erinevate sisendite puhul peab välja nägema järgnev:

```
impulss = funktsioon(5, -2, -2)
# print(impulss)
[ 0.  0. -2.  0.  0.]

impulss = funktsioon(8, -5, 3)
# print(impulss)
[0. 0. 0. 0. 0. 3. 0. 0.]

impulss = funktsioon(2, -1)
# print(impulss)
[0. 1.]
```

Selle ülesande viimaseks alamülesandeks on luua funktsioon, mis võtab sisendiks kui tahes palju sama pikkusega impulsse ning moodustab neist ühe digitaalse signaali kasutades superpositsiooniprintsiipi. Superpositsiooniprintsiip on kõikides lineaarsetes süsteemides kehtiv reegel, mille järgi süsteemi reaktsioon mitmele mõjurile on sama, mis üksikute mõjurite poolt tekitatud reaktsioonide summa. Seda reeglit kujutab väga hästi selle ülesande alguses olev pilt sämplimisest, kus iga üksik sämpel kokku liidetuna teiste sämplitega moodustab digitaalse signaali.

Kasutades eelpool õpitut ning teile antud aluskoodi, moodustage järgnev signaal kasutades skaleeritud ja ajas nihutatud ühikimpulsse ning kujutage seda graafikul. Oma lahendused kirjutage faili `yl4.py`. Ühikimpulsside pikkuse võite valida ise.

```
\begin{equation}
  y[n] = -2\delta[n-1] + 4\delta[n] + 3\delta[n-2] + 4\delta[n-1] + \delta[n-2] -
\delta[n]
\end{equation}
```

Näidake oma lahendust praktikumi juhendajale ning tehke commit!

3. Lisaülesanded (Valikuline)

Enne lisaülesannete lahendamist on kohustuslik ette näidata kõigi põhiülesannete töötavad lahendused. Lisaülesannete lahendamiseks soovitame kasutada SciPy IO ja OpenCV teeke.

5. Ülesanne: Varieeruv heli (1 punkt)

Selle ülesande eesmärk on koostada Pythoni abil WAV fail, mis simuleerib häiresignaali, nagu selles videos. Signaal peab olema koostatud käsitsi, kasutades for-tsüklit, mis käib läbi ette määratud sämplite arvu ja annab sämplitele väärtused.

Näidake oma lahendust juhendajale ning tehke commit!

6. Ülesanne: Pildist heliks (1 punkt)

Kursuse repositooriumis on fail nimega `helipilt.png`. Sellel pildil on helisignaal kujutatud pikslitena. Ülesande eesmärk on koostada programm, mis konverteerib toodud pildi kuulatavaks WAV failiks. Heli on kodeeritud rida-rida haaval ja iga sämpli moodustavad 2 järjestikust pikslit, kus sellest paarist esimene piksel on helisämpli madalama kaaluga bait (*Least Significant Byte*).

Olulise parameetrina on helifaili sämplimissagedus 22050Hz.

Näidake oma lahendust juhendajale ning tehke commit!

7. Ülesanne: Helist pildiks (1 punkt)

Kursuse repositooriumis on toodud heliklipp `pildiheli.wav`, mis kõlab kaugelt kuulates nagu tavaline helifail. Tegelikult on selle faili madalama kaaluga baitidesse (LSB) peidetud pilt. Ülesande eesmärk on taastada see pilt, luues selleks Pythonis vastava konverteerimise programmi.

Helifaili kodeeritud pilt on 750 pikslit lai ja 400 pikslit kõrge.

Näidake oma lahendust juhendajale ning tehke commit!

4. Kodutöö ülesanded

1. ülesanne

Uurige internetist klassiruumis olevate DS1054Z ostsilloskoopide kohta. Leidke vastused järgnevatele küsimustele.

1. Mis on nende ostsilloskoopide tootja poolt määratud kõrgeim sisendsignaali sagedus, mida see ostsilloskoop suudab mõõta? Mida see tehniline kirjeldus ostsilloskoopide puhul tähendab ning mis juhtub kiirema sisendi puhul?

   Eeldatav vastus:

   - selgitab lühidalt, mis on DS1054Z ostsilloskoobi kõrgeim sisendsignaali sagedus ning viitab allikale, millest see leiti;
   - vastab küsimusele, mis juhtub kiiremate sisendite puhul.

2. Üldjuhul on tavalise ostsilloskoobi proovikul olemas 1x ja 10x režiim. Mida need režiimid endast kujutavad ning millal oleks mõistlik kumbagi režiimi kasutada?

   Eeldatav vastus:

   - kirjeldab lühidalt, mida kujutavad endast 1x ja 10x režiimid;
   - seletab, millal oleks mõistlik kumbagi režiimi kasutada.

2. ülesanne

Tutvuge ostsilloskoobi kasutusjuhendiga ja programmeerimise juhendiga.

1. Mida muud on veel võimalik läbi SCPI käskude ostsilloskoobiga teha? Tooge kaks näidet ostsilloskoobi pakutavatest võimalustest, mis võiksid teid tulevikus signaalide uurimisel abistada.

   Eeldatav vastus:

   - kirjeldab lühidalt, mida valitud funktsioon teeb ning miks te just selle valisite;
   - demonstreerib pseudokoodiga, kuidas saata vastavat käsku mõõteriistale kasutades PyVISA teeki ja kuidas kasutada seadmelt saadud vastust.

3. ülesanne

1. Uurige internetist Nyquisti teoreemi kohta. Aruandes tehke **oma sõnadega** loetust kokkuvõte.

   Eeldatav vastus:

   - kirjeldab lühidalt, mida kujutab endast Nyquisti teoreem.

2. Kuulake ülesande käigus loodud helifaile. Viimase heliriba sämplimissageduse taastamisel 44100 Hz peale ei muutunud heli tagasi teravaks. Mis on selle põhjuseks?

   Eeldatav vastus:

   - kirjeldab lühidalt, mis on erinevate heliribade erinevus kuulates;
   - pakub põhjuseid, miks see nii võib olla;
   - selgitab, miks viimane heliriba ei muutunud tagasi teravaks, kui muuta sämplimisagedus tagasi 44100 Hz peale.