TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Auris Prääm 206615IACB

Mikroarvutite aja sünkroniseerimine GPS ja NTP abil

Bakalaureusetöö

Juhendaja: Priit Roosipuu
MSc

Tallinn 2024

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Auris Prääm

13.05.2024

Annotatsioon

Antud lõputöö käsitleb NTP ehk *Network Time Protocol* ja GPS ehk *Global Positioning System* kasutust kahe mikroarvuti ajasünkronisatsiooniks eesmärgina kasutada neid signaaliviite mõõteseadmena. Töös võrreldakse antud teenuseid erinevatel alustel nagu sünkronisatsiooni stabiilsus ja kiirus. Lisaks uuritakse sünkronisatsiooni mõjutavaid tegureid nagu operatsioonisüsteem.

Katsetest järeldus, et nii NTP kui ka GPS on parimates tingimustes sünkronisatsiooni osas peaaegu võrdväärsed. Nii NTP kui GPS saavutasid väga hea stabiilsuse: standardhälve jäi mõlemal alla ühe mikrosekundi. Võrguühenduse halvenedes vähenes NTP stabiilsus aga märgatavalt.

NTP peamine eelis on selle kasutajamugavus: ei ole vaja eraldi GPS moodulit vaid ainult internetiühendust. Kui internetiühendus pole aga väga stabiilne või puudub üldse või NTP server on kliendist kaugel, siis on GPS parem valik. Lisaks on NTP parem valik, kui ei ole otsest vaadet taevale, et GPS moodul saaks hea ühenduse.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 32 leheküljel, 12 peatükki, 21 joonist.

Abstract

Microcomputer time synchronisation using GPS and NTP

This thesis deals with the use of NTP (Network Time Protocol) and GPS (Global Positioning System) for the time synchronization between 2 microcomputers with the goal of using them as a measurement device for signal latency. In addition, the impact of the operating system on the synchronization is evaluated. Finally, recommendations for the use of NTP and GPS are made, and conclusions are drawn.

The first part of the thesis (chapters 2 to 6) gives insight to the workings and use cases for NTP and GPS. In addition, the programmes and their configuration used in the project and the reasons for their selection are outlined. Chapter 7 explains the signal sending and receiving programs.

In the second part (chapters 8 to 10), the experiments and their results are described. It was concluded that both NTP and GPS are almost equivalent in terms of synchronisation stability under the best conditions. Both services achieved very good stability: the standard deviation was less than one microsecond for both. However, with the increase in network latency, the stability of NTP decreased noticeably.

The main advantage of NTP is its ease of use: no separate GPS module is needed, just an internet connection. However, if the internet connection is not very stable or non-existent, or the NTP server is far away from the client, GPS is a better choice. In addition, NTP is a better choice if there is no direct view of the sky for the GPS module to get a good connection.

The project can be further developed from several angles. Firstly, the use of PTP can be explored for even more accurate synchronisation. In addition, it is still possible to further optimise the Linux operating system or to rebuild the whole system as a bare metal solution.

The thesis is in Estonian and contains 32 pages of text, 12 chapters, 21 figures.

Lühendite ja mõistete sõnastik

GPIO	*General-Purpose Input/Output*. Tarkvara kontrollitav sisend/väljund viik
GPS	*Global Positioning System*. Satelliitnavigatsiooni süsteem asukoha määramiseks
LAN	*Local Area Network*. Kohtvõrk
NTP	*Network Time Protocol*. Andmesideprotokoll süsteemikella sünkroniseerimiseks teiste serveritega.
PPS	*Pulse Per Second*. Signaal, mille nivoo muutub järsult iga täissekund.
PTP	*Precision Time Protocol*. Protokoll süsteemikella sünkroniseerimiseks.
RIO	*Register Input/Output*. Registrid seadme viikude kontrollimiseks.
RMS	*Root Mean Square*. Ruutkeskmine.
RTC	*Real Time Clock*. Seade aja mõõtmiseks. Tihti peaarvutist eraldiseisev ja eraldi toitega.
TAI	*International Atomic Time*. Reaalaja ajastandard.
UDP	*User Datagram Protocol*. Transpordikihi andmeside protokoll.
UTC	*Coordinated Universal Time*. Maailmaaja ajastandard.
WAN	*Wide Area Network*. Laivõrk

Sisukord

Jooniste loetelu

1 Sissejuhatus

NTP ehk *Network Time Protocol* protokoll ja GPS ehk *Global Positioning System* satelliidivõrgustik on peamised viisid kuidas tänapäeval arvutisüsteeme omavahel sünkroniseeritakse. NTP protokolli kasutavad operatsioonisüsteemid nagu *Windows*, *MacOS* ja *Linux*, et sünkroniseerida arvuti kellaaeg, mis muudab NTP kõige populaarsemaks ajapäringu protokolliks [1]. GPS-i kasutatakse omakorda, et sünkroniseerida servereid, mis NTP protokolli kasutades aega pakuvad.

Sünkronisatsioon oleneb aga paljudest tingimustest, alustades internetikiiruse ja satelliitide nähtavusega ning lõpetades sünkronisatsiooni programmide ja operatsioonisüsteemi valikuga. Nende tingimuste mõju hindamine on aga vajalik parima lahenduse valimiseks.

1.1 Taustainfo

Lõputöö teema tuleneb vajadusest mõõta viidet omavahel ühendatud kaamera ja kaamerapilti kuvava ekraani vahel, mis asuvad geograafiliselt erinevates kohtades. Selle jaoks saadetakse valgusimpulss läbi kaamera, mis seejärel tuvastatakse ekraanil. Sellega mõõdetakse signaali hilistust. Õigete mõõtetulemuste jaoks peavad aga nii signaali saatja kui ka vastuvõtja olema sünkroniseeritud, et vastuvõtja teaks, millal signaal välja saadeti. [2]

Sellest tulenevalt uuritakse antud töös just seadmete omavahelist sünkronisatsiooni ning selle jaoks kasutatavaid meetodeid. Täpsemalt võrreldakse GPS ja NTP sobivust antud sünkronisatsiooni teostada.

Varasemalt on uurimise all olnud peamiselt süsteemid, kus ühte seadet sünkroniseeritakse UTC (*Coordinated Universal Time*) aja järgi [3], [4], [5]. Projekte, kus uuritakse kahe seadme omavahelist sünkronisatsiooni kombineerituna GPIO (*General Purpose Input/Output*) impulssidega, pole autor leidnud.

1.2 Probleemi püstitus

Lõputöö eesmärk on võrrelda GPS ja NTP teenuste aja sünkroniseerimisvõimekust ja uurida lähemalt nende iseärasusi. Täpsemalt keskendutakse kahe mikroarvuti omavahelisele sünkroniseeritusele. Antud teenuseid võrreldakse erinevates tingimustes ning vastavalt tulemustele tuuakse välja soovitused nende kasutamiseks. Tööks püstitatud uurimisküsimused on järgmised:

1. Kuidas sünkroniseerida mikroarvutite ajad kasutades GPS ja NTP teenuseid?
2. Aja sünkroniseerimisvõimaluste võrdlus: mis on GPS ja NTP eelised ja puudused? Milline on GPS ja NTP aja sünkroniseerimistäpsus kahe mikroarvuti vahel?
3. Millised on soovitused GPS ja NTP teenuste kasutamiseks mikroarvutite aja sünkroniseerimisel?
4. Kuidas operatsioonisüsteem mõjutab mikroarvutite aja sünkronisatsiooni?

1.3 Ülevaade tööst

Teema uurimispool on töös jagatud peamiselt kolmeks osaks. Esiteks kirjeldatakse lühidalt lahti kasutatud tehnoloogiaid. Seletatakse lahti, mis on NTP ja GPS ning kuidas neid kasutatakse ajasünkronisatsiooniks. Tuuakse välja programmid, mida kasutati NTP ja GPS andmete analüüsiks ning nende abil süsteemikella sättimisteks. Lisaks kirjeldatakse lahti tööks vajaliku mikroarvuti karakteristikud ning miks valiti just Raspberry Pi 5 seade. Seletatakse lahti antud mikroarvuti ja programmide valikust tulenevad probleemid nagu näiteks operatsioonisüsteemi hilistused ning kuidas neid hilistusi minimeeriti.

Järgmiseks kirjeldatakse lahti signaali saatmiseks ja vastuvõtmiseks kasutatud programmid, mille alusel hakatakse katseid läbi viima ning teenuseid omavahel võrdlema. Tuuakse välja katsete läbiviimistingimused, mis puudutab kasutavate programmide konfiguratsiooni ning operatsioonisüsteemi modifikatsioone.

Viimase teemana kirjutatakse kõikidest läbiviidud katsetest ning nende tulemustest. Seletatakse täpsemalt lahti iga katse olemus ning sellest tehtud järeldused.

Katsetulemustest tulenevalt antakse ka soovitusi kuidas NTP ja GPS tehnoloogiat ajasünkronisatsiooniks kasutada ning millistes olukordades oleks parim kasutada mida. Lõpuks võetakse tehtu üleüldiselt kokku, analüüsitakse tööd ning tuuakse välja edasiarenduskohad.

2 Network Time Protocol

Network Time Protocol ehk NTP on andmesideprotokoll, mida kasutatakse arvutite kellaaja sünkroniseerimiseks kellaserveritega. NTP on suuteline pakkuma alla millisekundilise täpsuse LAN (*Local Area Network*) ehk kohtvõrkudes ja mõne millisekundilise täpsuse WAN (*Wide Area Network*) ehk laivõrkudes. Enamasti kasutab klient mitut üksteisest sõltumatut kellaserverit, et tagada parem täpsus ja töökindlus. [6], [7]

NTP toetub UTC maailmaaja standardile, mis ajavööndi ei muutu. UTC omakorda toetub TAI (*International Atomic Time*) ehk rahvusvahelisele aatomajale, mida hoitakse sadade aatomkelladega üle maailma. Täpse kohaliku kellaaja peab UTC järgi määrama operatsioonisüsteem. Kuna UTC on määratud maa pöörlemiskiiruse järgi, siis peab TAI jälgimiseks NTP kasutama ka *leap* sekundeid. [7]

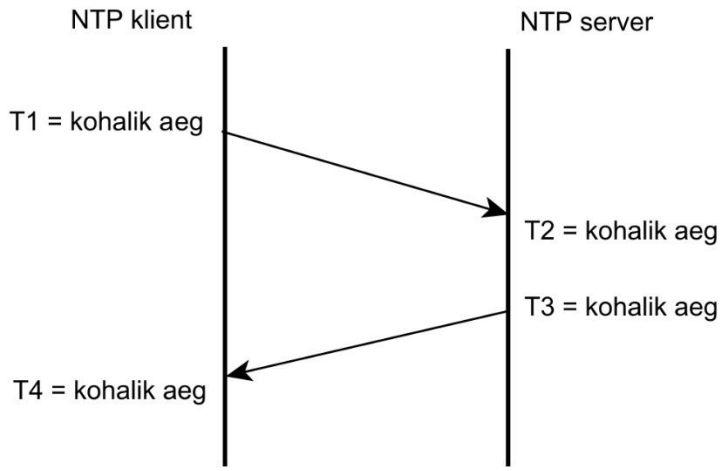
2.1 Protokolli kirjeldus

NTP kasutab kahte erinevat ajatempli formaati. Esiteks *datestamp*, mis on 128 bitine täisarv, kus esimesed 64 bitti tähistavad sekundeid ajastu ehk *epoch* ajast ning teine pool murdosa täissekundist. Seda formaati kasutatakse seadmesisesteks arvutusteks. Andmepakettides kasutatakse poole väiksemat ehk 64 bitist *timestamp* ajatemplit, et hoida kokku ruumi. *Timestamp* formaadi teoreetiline ajaresolutsiooni on 232 ps. [8], [6]

NTP kasutab andmete edastamiseks UDP ehk *User Datagram Protocol* protokolli, mida tavaliselt kuulatakse pordil 123. NTP pakett sisaldab nelja ajatemplit, mis on järjestatud järgmiselt:

1. Päringu saatmine kliendi poolt (T1)
2. Päringu kättesaamine serveri poolt (T2)
3. Vastuse saatmine serveri poolt (T3)
4. Vastuse kättesaamine kliendi poolt (T4)

Joonis 1 kujutab NTP ajatemplite salvestamise aega. T1 ja T2 tähistavad ajatempleid, millal päring kliendi poolt teele pandi ja millal serveri poolt kätte saadi. T3 on vastuse edastamise ajatempel ning T4 näitab aega, millal klient vastuse kätte sai. [6]



Joonis 1. NTP päringu ajatemplite saamise kohad. [9]

Kliendi ajanihe arvutatakse välja järgmise valemiga. [8], [6]

$$ajanihe = \frac{(T2 - T1) + (T3 - T4)}{2}$$

Antud ajanihe võimaldab kliendil arvutada oma umbkaudse kellaerinevuse serverist, mille tulemusel on võimalik sättida süsteemi kella kiirust nii, et kompenseerida antud ajanihe ilma kellaaega järsult muutmata. [8], [6]

2.2 Stratum tasemed

NTP arhitektuur töötab struktureeritud hierarhilisel süsteemil, mida nimetatakse *stratum*-iks. Referentskellasid, tüüpiliselt näiteks aatomkellasid, nimetatakse *stratum* 0 ehk kõrgeima täpsusega kelladeks. *Stratum* 1 serverid on sünkroniseeritud otse *stratum* 0 referentskellade nagu näiteks GPS satelliitide järgi. *Stratum* 2 serverid on sünkroniseeritud *stratum* 1 serverite järgi jne. Iga *stratum* tasemega väheneb kella täpsus tulenevalt võrgu hilistustest ja ebastabiilsustest. [6]

Stratum tasemete täpsus pole üheselt määratud, sest see sõltub mitmetest teguristest nagu näiteks võrguühendusest, riistvarast ja tarkvarast ning andmesidekiirusest. Kuigi *stratumi* tase annab üldise idee kellade täpsusest ja nende positsioonist ajahierarhias, ei taga see alati täpset ajaarvestust. [6]

3 Global Positioning System

GPS ehk *Global Positioning System* on mõnekümnest satelliidist koosnev võrgustik, millega on GPS signaali vastuvõtjatel võimalik määrata oma asukohta. Nende parameetrite alusel on võimalik vastuvõtjal võrrelda enda asukohta mitme teise satelliidiga ja selle toel välja arvutada enda positsioon. Lisaks pakub GPS ka väga täpseid ajatempleid, mida saab kasutada ajasünkronisatsiooniks. [10], [11]

GPS satelliidid sisaldavad aatomkellasid, mida hoitakse sünkroonis UTC ajaga. Aja täpsuseks lubatakse vähem kui 30 ns, 95% ajast, kui kasutatakse selleks mõeldud vastuvõtjat staatilises asukohas. Säärane täpsus teeb GPS signaalist odava ja mugava viisi saada otsene juurdepääs *stratum* 0 referentskelladele. [12]

3.1 PPS signaal

Jadasiini andmeühendus on täpse aja edastamiseks liiga aeglane ning lisab signaalile viidet. Parem viis ajaedastuseks on PPS ehk *Pulse Per Second* signaal. Selle tööpõhimõte on omandada kõrge nivoo võimalikult täpselt iga täissekund. Niimoodi eemaldatakse kõik jadaühenduse viited signaaliedastusest, sest seade peab tuvastama ainult ühe nivoo muutuse. [13]

PPS signaali lihtne tööpõhimõte võimaldab edastada väga täpset ajasignaali, mille järgi on võimalik sünkroniseerida erinevaid süsteeme. PPS signaali väljastus on näiteks sisse ehitatud paljudele GPS vastuvõtjatele ning seda kasutatakse ka antud töös mikroarvuti sünkroniseerimiseks GPS mooduli poolt väljastavate PPS signaalide järgi. [13]

3.2 LinuxPPS ja gpsd programmid

Kasutades ainult GPS moodulit ajasünkronisatsiooniks on vaja vähemalt 200 ms täpsusega ajatemplit, et *Chrony* tarkvara saaks aru, mis sekundile PPS signaali järgi kell sättida. Töös kasutatakse *gpsd* programmi, mis töötleb GPS mooduli pealt tulevat informatsiooni ning muudab selle lihtsasti loetavaks teistele programmidele. Antud programmi on vaja, et lugeda mooduli jadakommunikatsioonist välja ajatempel, mida

kasutatakse umbes paarisajamillisekundilise täpsusega sünkronisatsiooniks enne PPS signaali kasutuselevõttu. See on vajalik, et teada, millisele sekundile PPS signaali sünkroniseerida. [14]

LinuxPPS on rakendusliides, mille abil saavad teised programmid kasutada ühte või mitut PPS signaali. Liides lisab PPS signaalidele süsteemikella ajatemplid, mille järgi saavad teised programmid näiteks jälgida süsteemikella stabiilsust või seda sättida. [15]

4 Chrony programm

Chrony on NTP implementatsioon, mis võimaldab kella sünkronisatsiooni NTP serveritega ja referentskelladega. Lisaks võimaldab ta muuta arvuti ka NTP serveriks, mille poole saavad kliendid pöörduda. *Chrony* töötab UNIX-i laadsetel operatsioonisüsteemidel nagu näiteks *Linux* ja *macOS*. [16]

4.1 Võrdlus teiste programmidega

Peamised valikud *Chrony* kõrval on *ntpd* ja *openntp*. *Chrony* sisaldab võrreldes mainitud valikutega rohkem funktsionaalsust ja paremaid sünkronisatsiooninäitajaid. *Chrony* saab paremini hakkama ebastabiilse internetiühendusega situatsioonides, suudab kiiremini kohanduda süsteemikella kõikumistele ning on enamus situatsioonides ka täpsem. [17], [18], [19]

Lisaks suudab *Chrony* kasutada osade võrgukaartide võimekust panna internetipakettidele riistvaralisi ajatempleid. Säärane võimekus on olemas kasutusel olevatel Raspberry Pi 5 arvutitel, mis vähendab võrguhilistusi ja suurendab ajatemplite täpsust. [17], [19]

Vähesed *Chrony* puudused võrreldes konkurentidega on sisseehitatud referentskellade ohjuri puudumine, mida omab näiteks *ntpd*. See tähendab, et *Chrony* peab kasutama eraldi programmi nagu näiteks *gpsd*, et referentkellade andmeid töödelda. [17], [19]

5 Nõuded seadmetele

Töö raames peab kasutama seadet, mis oleks kompaktne ehk käsikantav, omaks NTP jaoks *Ethernet* porti ning GPS mooduli jaoks vajalikku ühendusvõimekust. Töötaks kas *Linux*, *BSD* või *macOS* operatsioonisüsteemiga, mida nõuab *Chrony* sünkroniseerimistarkvara, ning omaks GPIO viikusid signaaliimpulsside väljastamiseks ja vastuvõtmiseks [16]. Lisaks on seadme valimisel oluline selle taskukohasus, kasutuslihtsus ja dokumenteeritus.

GPS seade peab olema ühendatav valitud arvutiga, olema võimeline väljastama PPS signaali ning jätma arvutil vabaks vähemalt 2 GPIO viiku signaali saatmiseks ja vastuvõtmiseks.

5.1 Raspberry Pi 5 mikroarvuti

Töö jaoks valiti Raspberry Pi seeria seade. Raspberry Pi on seeria monoplaatarvuteid, mis on välja töötatud Ühendkuningriigis asuva Raspberry Pi Foundation poolt [20]. Antud seeria arvutid on väga laialdaselt kasutuses, mis on toonud kaasa mahuka kasutajate tagasiside nii seadme kui ka selle peal töötavate programmide kohta [21]. Lisaks olid Raspberry Pi arvutid lihtsasti kättesaadavad ning neid kasutati antud tööle eelneva süsteemi idee tõenduse (*proof-of-concept*) loomisel [2].

Töös on kasutusel kaks Raspberry Pi 5 arvutit, mis on antud seeria uusim ja kiireim mudel. Täpsemalt kasutatakse kahte 8 GB vahemäluga mudelit. Seadmed on varustatud tarkvara poolt juhitud jahutusventilaatoriga. [22]

5.2 Adafruit Ultimate GPS HAT moodul

Töös kasutati Adafruit Ultimate GPS HAT moodulit, mis kasutab CD-PA1616D GPS moodulit ning on mõeldud kasutamiseks Raspberry Pi mudelitel. Moodul lubab PPS täpsust ± 20 ns (RMS ehk *Root Mean Square* väärtus). Lisaks sisaldab moodul RTC ehk

Real Time Clock moodulit, mis talletab GPS satelliitide andmed ka ilma vooluta olekul. See parandab järgmisel käivitamisel sünkronisatsiooni kiirust. [23]

Konkreetne moodul valiti kuna see on otseselt disainitud töötama Raspberry Pi seadmetega ning omab RTC moodulit, sest seadme kasutuseemärkide pärast on sünkronisatsioonikiirus tähtis. Lisaks olid need lihtsasti kättesaadavad. [23], [24]

6 Linux operatsioonisüsteem

Raspberry Pi 5 ametlik operatsioonisüsteem on Raspberry Pi OS, mis põhineb *Debian* distributsioonil. Antud töös kasutati Raspberry Pi OS Lite 64-bitist versiooni, mis jookseb *kerneli* versioonil 6.6. Lite ehk ilma graafilise kasutajaliideseta versioon on kasutusel, et minimaliseerida operatsioonisüsteemi suurust ja selle jooksmiseks potentsiaalselt vajaminevate protsesside hulka. [25]

6.1 Reaalajavõimekus

Linux ei ole reaalaja operatsioonisüsteem ehk kasutatakse protsesside ajastajat, mis jagab ressursse ja arvutusaega protsesside vahel. See võib tekitada aga protsessides soovimatuid ja ebakonstantseid viiteid, mida antud seadme puhul on vaja minimeerida. [26]

PREEEMPT_RT on *Linuxi kerneli* ehk tuuma modifikatsioon, mille eesmärk on anda võimalus muuta *Linuxi* peal jooksvaid protsesse täielikult ennetavaks ehk minimaliseerida operatsioonisüsteemi osasid, millele ei saa sättida prioriteete. See vähendab protsesside latentsust ja suurendab nende stabiilsust. [26], [27]

Mõlemad Raspberry Pi 5 seadmed töötavad „6.1.77-rt24-v8+ PREEMPT_RT“ *patch*-i peal ning signaali saatmise ja vastuvõtmise programmid on sätestatud jooksma reaalajalõimedena. Lõimede sätestamiseks on kasutatud ametliku dokumentatsiooni näidiskoodi [28].

7 Signaali saatmine ja vastuvõtmine

Lisaks kellade sünkronisatsioonile on oluline tekitada ajaliselt täpsed signaali saatmise ja vastuvõtmise protsessid. Katsete jaoks peavad signaali saatmise ja vastuvõtmine olema võimalikult väikese viitega ning stabiilsed, et need mõjutaksid mõõtetulemusi võimalikult vähesel määral.

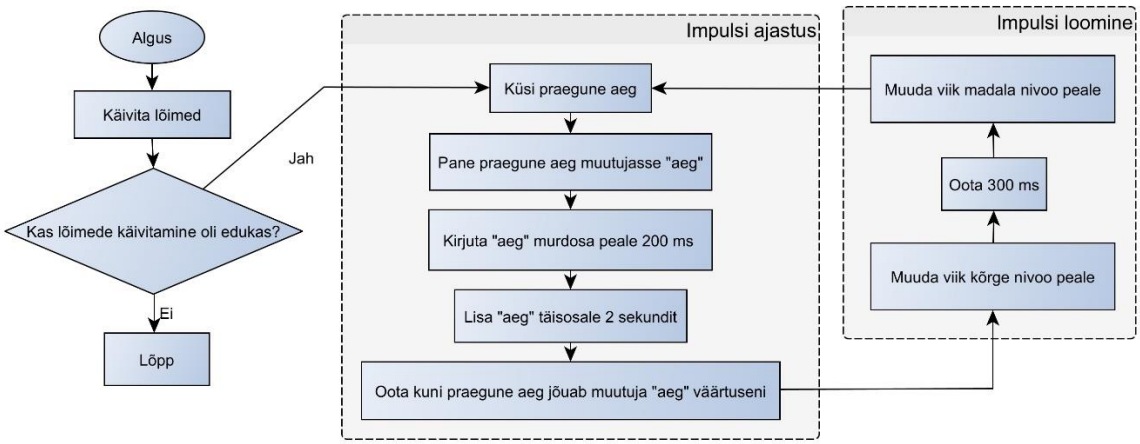
Lõpptootes mõõdetakse viidet, mis jääb ühe sekundi ümbrusesse ehk mõõtesignaali hakatakse väljastama iga kahe sekundi tagant, et need oleksid vastuvõtja poolt eristatavad. Antud töös käsitletakse signaali väljastust ja vastuvõtmist seadme GPIO viigult ning ei tegeleta viikudest väljaspool toimuvaga.

7.1 Registrite kasutamine

Otse mälust registrite lugemine ja kirjutamine on kiireim viis pääseda ligi Raspberry GPIO viikudele, sest see minimaliseerib tarkvaralise abstraktsiooni, mis esineks näiteks mõne teegi kasutamisel. Lisaks on viikudele ligipääs realiseeritud C keeles, mis vähendab ka programmeerimiskeelest tulenevaid abstraktsioone. [29]

Mälu registrite kaardistamiseks loetakse mällu vajalikud väljad. Viigu väärtuste lugemiseks kasutatakse programmi loetud registrite väärtusi, kuid viigule kirjutamiseks kasutatakse RIO ehk *Register Input Output* plokke, mis muudavad viigule kirjutamise lihtsamaks ja kiiremaks. [30]

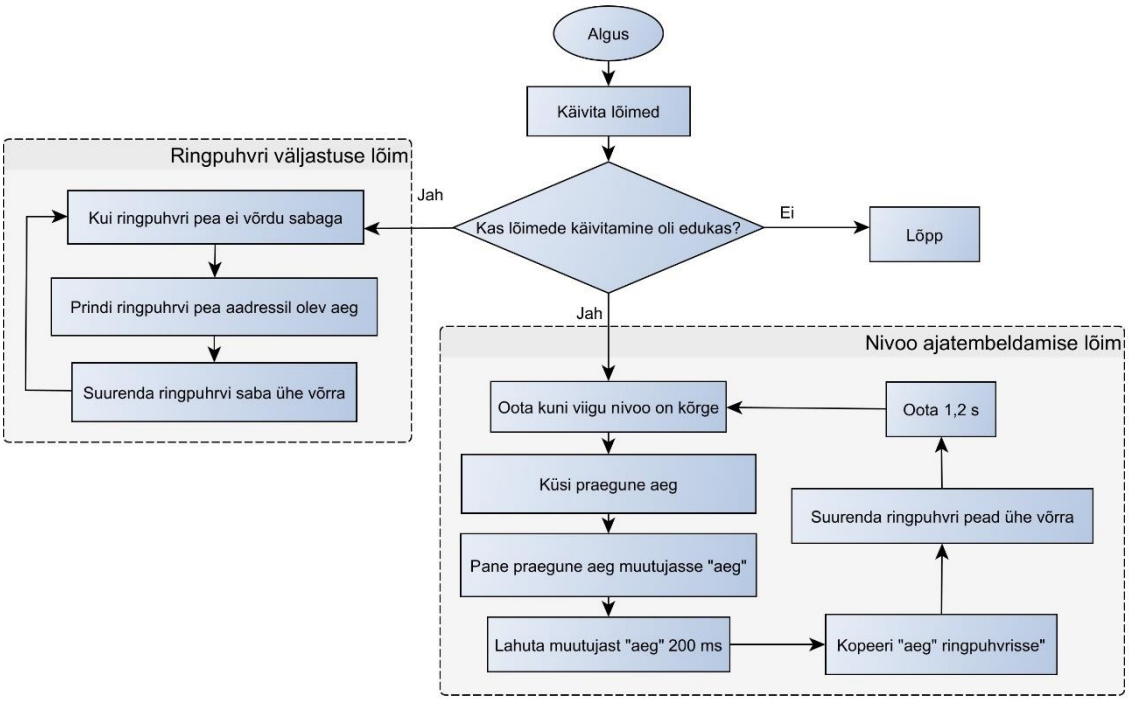
Impulsi saatmisel käivitatakse alguses reaalajalõim (Joonis 2) ning sisenetakse lõpmatusse tsüklisse, kus ajastatakse iga impulss kahe sekundi tagant ning 200 ms täissekundist, et hoida ära potentsiaalsed viited igal täissekundil vastuvõetava PPS signaali töötlemisega.



Joonis 2. Impulsi saatmise programm.

Impulsi vastuvõtmiseks (Joonis 3) käivitatakse kaks erinevat lõime. Üks reaalaja lõim, mis tegeleb kõrge nivoo tuvastamise ja ajatembeldamisega ning üks tavaline lõim, mis tegeleb ajatemplite välja printimisega. Säärane kombinatsioon on vajalik, sest reaalajalõim ei luba kasutada failidesse või standardväljundisse kirjutamist, sest see käivitab mitte-reaalajalisi protsesse. Lõimede vaheliseks andmete edastuseks kasutatakse ringpuhvrit.

Vastuvõtuajast lahutatakse 200 ms, sest see oli liidetud juurde saatmise protsessis. Lisaks ootab kood enne uue kõrge nivoo otsingut 1,2 s, mille jooksul uut impulssi katsete käigus ei oodata, et vähendada nivoo staatuse kontrollimiste arvu, mis omakorda vähendab protsessori koormust. Konkreetse ooteaja valikul pole otsest põhjust, sest testimiseks antud arv töötab ning vähendab protsessori kasutust. Programmi praktiliseks kasutuselevõtuks võib ooteaja kas täielikult eemaldada või sättida vastavalt vajadusele.



Joonis 3. Impulsi vastuvõtu programm.

Mõlemaid koode jooksutatakse *Super User* ehk kõrgendatud õigustega, mis on reaalajalõimede töötamiseks vajalik [31]. Vastuvõtmise viiteid salvestatakse suunates standardväljundi tekstifaili *tee* käsuga, et tulemusi oleks näha reaalajas nii terminalis kui ka failis hilisemaks töötlemiseks. Koodid koos nende erinevate versioonidega on saadaval GitHubi repositooriumis (Lisa 2).

8 Testimise metoodika

Katsete peamine eesmärk oli leida NTP ja GPS eelised ja puudused signaali viite mõõtmiseks kahe Raspberry Pi vahel. Lisaks leida täpseim ja stabiilseim sünkroniseerimisviis. Tähtis oli just Raspberry Pi-de omavaheline sünkronisatsioon mitte otseselt UTC või mõne muu ajastandardi vastu sünkronisatsioon.

Lisaks oli tähtis testida impulsside saatmist ja vastuvõtmist, et need oleksid võimalikult stabiilsed ja ettearvatavad.

8.1 Linuxi konfiguratsioon

Reaalaja *kernelil* kasutati enamjaolt selle standardkonfiguratsiooni. Reaalajas jooksvate protsesside prioriteet oli pandud maksimum lubatud väärtusele, mis on 99.

Taustaprotsesside vähendamiseks oli välja lülitatud *WiFi, Bluetooth* ja audio ning paljud väiksemad protsessid nagu näiteks osade tarkavauuenduste otsimine, mida antud töö raames vaja ei lähe. Lisaks oli välja lülitatud *kerneli* energiasäästu režiim ning protsessor oli sunnitud jooksma alati enda maksimum sagedusel, mis on 2,4 GHz, et vähendada potentsiaalseid viiteid [22].

Vähendatud oli ka Raspberry võrguadapteri pakettide väljasaatmise ja vastuvõtmise viidet. Võrgukaardid koguvad tavaliselt riistvaraliste katkestuste modereerimiseks kokku mitu internetipaketti enne kui need süsteemile edastatakse. See lisab aga NTP pakettidele viited. Nende viidete vähendamiseks oli viidud katkestuse ooteaeg standardselt 49-lt mikrosekundilt neljale mikrosekundile, mis on Raspberry Pi 5 võrguadapteri miinimum. [32]

8.2 Chrony konfiguratsioon

Peamine viis *Chrony* konfigureerimiseks käib läbi „chrony.conf“ faili. Kõikidel testidel oli *Chrony*-l aktiveeritud andmete logimine. Algsel sünkroniseerimisel esimesel kolmel kella uuendusel oli lubatud kella hüpata, et enne ainult kella kiiruse muutmisele üle

minekut oleks see suurusjärgus õigel kohal. Lisaks kompenseeriti TAI ja UTC vahe *leap* sekunditega. Kogu protsess lukustati ka operatiivmällu käsuga *lock_all*. Kõikidel testimistel oli lubatud ka riistvaralised ajatemplid. [33]

8.2.1 NTP pärimine

Chrony annab NTP päringute koostamisele palju valikuvabadust. Töös on kasutusel kõik *Chrony* poolt ja siia töösse sobivad NTP täpsuse parandamiseks mõeldud valikud. Joonis 4 esitab NTP päringute konfiguratsiooni. „server“ märksõna tähistab, et tahetakse pärida aega NTP serverilt ning „ip_address“ asendatakse serveri ip-aadressiga. [34], [33]

Olulisemad parameetrid, mida kasutati, on *minpoll* ja *maxpoll*, mis määravad vastavalt minimaalse ja maksimaalse saadetavate päringute intervalli. Need määratakse kahe astmega ehk „minpoll 2“ tähendab neljasekundilist intervalli ja „minpoll -2“ tähendab 0,25 sekundilist intervalli. Minimaalne intervall on -7 ehk 128 päringut sekundis ja maksimum on 24 ehk ligikaudu kord poole aasta jooksul. [33]

„filter“ kogub kokku määratud arvu päringu vastuseid ja võtab nende mõõtetulemustest mediaani, mida siis kasutati süsteemikella sättimiseks. See on mõeldud kasutamiseks ainult väga lühikeste päringuintervallidega, kus mitme vastuse kokku panemine on mõttekas. [33]

„xleave“ parameeter võimaldab kasutada täpsemaid ajatempleid, kui server samuti jookseb *Chrony* tarkvara peal. „extfield“ kasutab NTPv4 võimalust lisada päringutele lisainformatsiooni. „F323“ on *Chrony* eksperimentaalne päringu laiendusväli informatsiooniga, mis on pakutud NTPv5 ehk järgmise generatsiooni standardisse. See sisaldab kõrgema resolutsiooniga *root delay*, dispersiooni ja monotoonseid ajatempleid. Sellel on potentsiaal oluliselt parandada kella täpsust. [33]

“hwtimestamp” lubab arvutil kasutada riistavaralisi ajatempleid, kui arvuti enda riistvara seda võimaldab. Selles juhul salvestatakse paketi ajatempel juba võrgukaardil, mis vähendab hilistusi veelgi. [33]

```
server ip_address minpoll x maxpoll x xleave extfield F323 filter x
hwtimestamp * minpoll x
```

Joonis 4. Chrony NTP konfiguratsioon.

8.2.2 GPS päringud

GPS konfiguratsioon on jagatud kaheks osaks (Joonis 5). Esimene rida tähistab GPS jadaühendusest tulevaid andmeid, mis annab esialgse sünkronisatsiooni enne PPS ühendusele üleminekut. *Chrony* määrab allika kasutatavaks, kui see on maksimaalselt 200 ms nihkes hetkel kasutatava allikaga. See tähendab, et GPS jadaühenduse ajatemplid peavad olema 200 ms PPS signaalist, et toimuks üleminek PPS allikale. Selleks oli kasutatud „offset“ parameetrit, mis nihutab jadaühenduse ajatemplid 0,85 s võrra, mis katsetamise käigus oli piisav täpsustus, et *Chrony* läheks üle PPS signaalile. Antud väärtus on iga süsteemi koosluse puhul individuaalne. [33]

PPS signaali kasutamiseks sisestati asukoht, kust saab ligi *LinuxPPS* poolt loodud ajatemplitele. Antud juhul oli see „/dev/pps0“. „poll“ parameeter määrab kella uuendamisintervalli kahe astendajana nagu NTP korral. Kuna töös kasutati 1 Hz sünkroniseerimissignaali, siis see väärtus sai olla ainult positiivne. Kui ühte intervalli mahub mitu PPS signaali, siis Chrony kasutab sünkroniseerimiseks nende ajatemplite mediaanväärtust. [33]

```
refclock SHM 0 refid GPS offset 0.85 noselect
refclock PPS /dev/pps0 lock GPS poll x
```

Joonis 5. Chrony GPS konfiguratsioon.

8.3 GPS testimine

Kõikide testide käigus oli aktiveeritud GPS mooduli RTC ehk *Real Time Clock* kell, mis on patareiga ka voolu puudumisel aktiivne. See võimaldab käivitamisel kiiremat satelliitidega sünkronisatsiooni saavutamist. *LinuxPPS* ja *gpsd* olid konfigureeritud nii, et nende andmed oleksid kättesaadavad *Chrony* programmile. [3], [5], [24]

8.4 Andmete esitamine

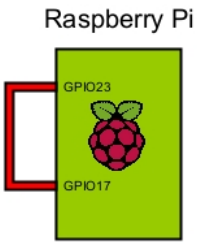
Andmete analüüsiks ja graafikute tegemiseks kasutati *Matplotlib Python* teeki, sest see pakub kõiki antud töö analüüsiks vaja minevaid funktsioone ja graafikuid lisaks *Pythoni* enda võimekusele [35]. Valikut mõjutas ka autori eelne kokkupuude *Pythoni* programmeerimiskeelega. Kõik algandmed olid kogutud kas logi– või tekstifailidesse ning andmete analüüsiks ei kasutatud Raspberry seadmeid vaid eraldi arvutit.

9 Katsed ja tulemused

Katsed jaotuvad peamiselt kolme kategooriasse. Esimesed katsed käsitlevad operatsioonisüsteemi hilistuste mõju süsteemi täpsusele ning lisaks mõõdetakse ka süsteemikella enda stabiilsust. Edasi võrreldakse GPS ja NTP sünkronisatsioonitäpsust erinevates tingimustes. Viimaseks tuuakse välja mõlema teenuse sünkronisatsioonikiirused. Kõik katsed viidi läbi toatemperatuuril.

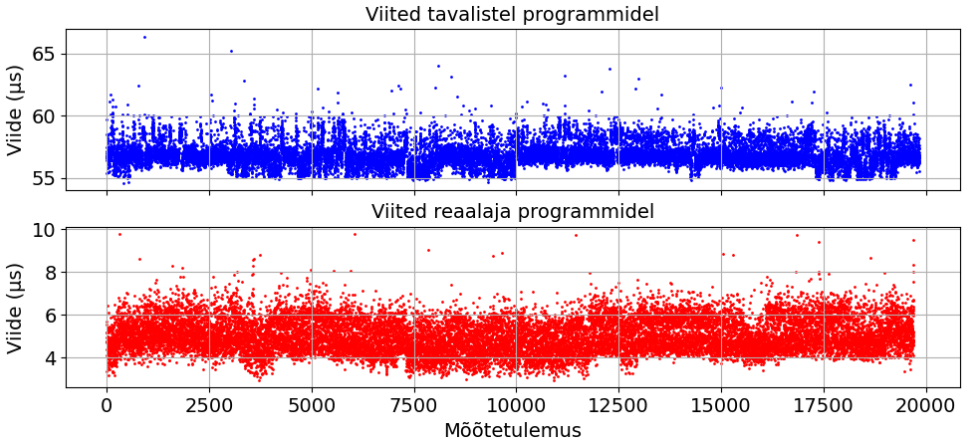
9.1 Reaalaja ja tavalise Linuxi võrdlus

Katse eesmärk oli võrrelda reaalaja ja tavalise *Linuxi* stabiilsust, kasutades viigult viigule signaali saatmise ja vastuvõtu ajavahemike mõõtmist (Joonis 6). Mõlemad viigud olid kokku ühendatud juhtmega ehk reaalne viide oli nullilähedane. Seadme süsteemikella katse ajal ei uuendatud. Tavalõimedele seati kõrgeim prioriteet ehk -20 *nice* programmiga [36].



Joonis 6. Reaalaja- ja tavalõimede võrdluse katse illustratsioon.

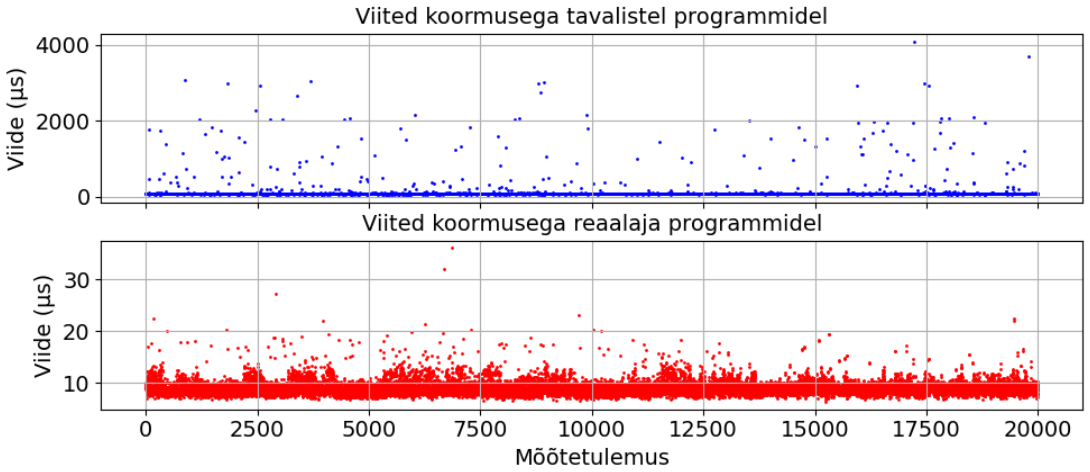
Katse jaoks tehti mõlema lõimega üle 19000 mõõtmise, kust selgus (Joonis 7), et reaalaja võimekusega lõim on stabiilsem kui tavaline lõim. Sellele viitab mõõtmiste standardhälbed, mis olid vastavalt 0,78 µs ja 0,95 µs. Lisaks peab märkima, et tavalise lõime hilistus oli peaaegu 10 korda suurem kui reaalaja oma, kuigi standardhälbed olid lähedasemad.



Joonis 7. Reaalaja- ja tavalõimede võrdlus ilma koormuseta.

Testiti ka viiteid olukorras, kus seadme protsessor on koormatud. See võib juhtuda, kui seadet soovitakse kasutada samal ajal ka muudeks tegevusteks peale mõõtmiste. Katseks kasutati programmi *hackbench*, mis koormab süsteemi lõimede loomise ja mahavõtmisega ning soklite vahelise suhtlusega [37]. Antud programm valiti kuna see tuli kaasa reaalaja *Linuxi* modifikatsiooniga. Katsete käigus olid kõik Raspberry tuumad üle 90% koormatud.

Tulemustest on näha (Joonis 8), et koormuse all on nii programmide hilistus ja stabiilsus oluliselt halvem. Tavalõime kasutades on standardhälve umbes 130 korda kõrgem võrreldes mitte koormuse all olekuga ehk 124 µs ning reaalajalõimedega on standardhälve ainult umbes poole suurem ehk 1,3 µs.



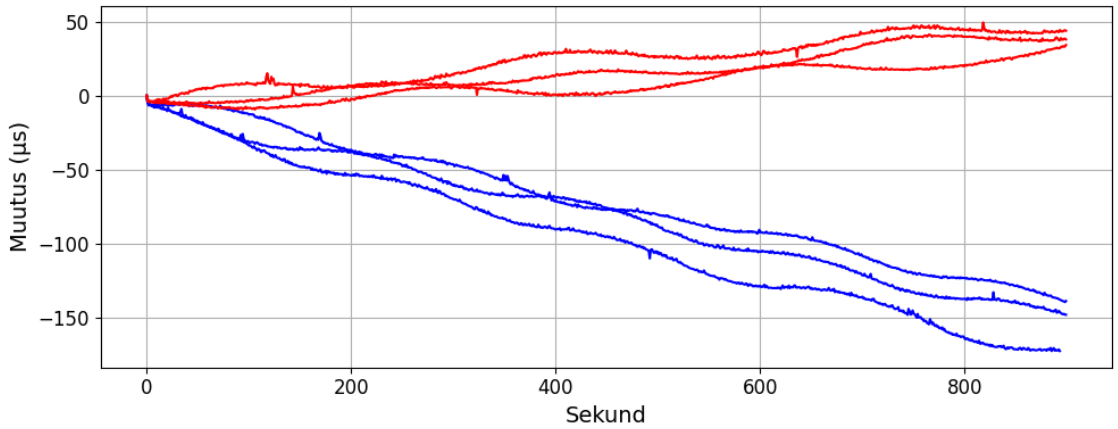
Joonis 8. Reaalaja- ja tavalõimede võrdlus koormusega.

Kokkuvõtteks on reaalajalõim parem nii hilistuselt kui stabiilsuselt, eriti kui protsessor on muude protsessidega oluliselt koormatud. Edasi minnes kasutatakse signaali saatmiseks ja vastuvõtmiseks ainult reaalajavõimekusega lõime.

9.2 Kella stabiilsus ilma sünkronisatsioonita

Katse eesmärk oli määrata Raspberry vabalt jooksva kella stabiilsus ehk olukorras, kus sünkroniseerimist ei toimu. Mõõtetulemuste saamiseks kasutati *ppstest* funktsiooni *LinuxPPS* programmist, mis väljastab PPS signaalide ajatemplid süsteemi kella järgi. Kuna PPS signaal jäljendab õiget aega, siis selle ajatemplid näitavad süsteemikella ujumist. Kõiki mõõtmisi alustati peale seadme taaskäivitust, et eemaldada kõik varem potentsiaalselt peale jäänud *Chrony* kellasättimised.

Katse tulemustes (Joonis 9) on seadmed eraldatud sinise ja punase värviga. Tulemused näitavad, et Raspberry kellad nihkuvad seadmeti ajas erinevate tempodega. See näitab, et Raspberry kella stabiilsus oleneb individuaalsest seadmest. Üleüldiselt on näha, et seadmed eralduvad üksteisest juba mõnikümmend mikrosekundit ainult paari minuti jooksul. Seda tehakse küll ühtlasel tempol ja seadmeti samas suunas ja enamvähem samal kiirusel, mis annab võimaluse potentsiaalselt sünkronisatsiooni katkedes ujumist mingil määral ennetada.



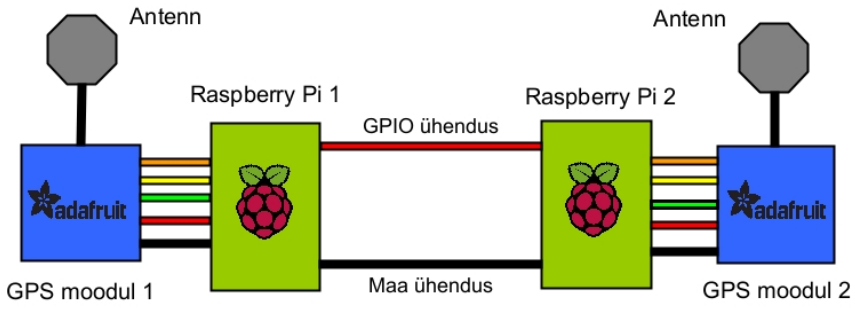
Joonis 9. Kella nihkumine ilma sünkronisatsioonita.

9.3 GPS sünkroniseerimisintervalli võrdlus

PPS signaali täpsus võib operatsioonisüsteemi pärast olla ikkagi ebastabiilne ehk signaalile peale pandud ajatempel võib hilistuste pärast muutuda ebatäpseks. Selle ära

hoidmiseks lubab *Chrony* kasutada mediaanfiltrit, kus kogutakse kokku mitu PPS signaali ning leitakse nende mediaan, mida siis kindla intervalli tagant kasutatakse süsteemikella sättimiseks [33]. Liiga suure intervalliga võib aga süsteemikell ise liiga palju nihkuda. Liiga lühikese intervalliga võib loetud PPS signaal olla aga ebatäpne erinevate häirete ja hilistuste tõttu. Katse eesmärk oli leida optimaalne punkt ehk parima täpsusega intervall.

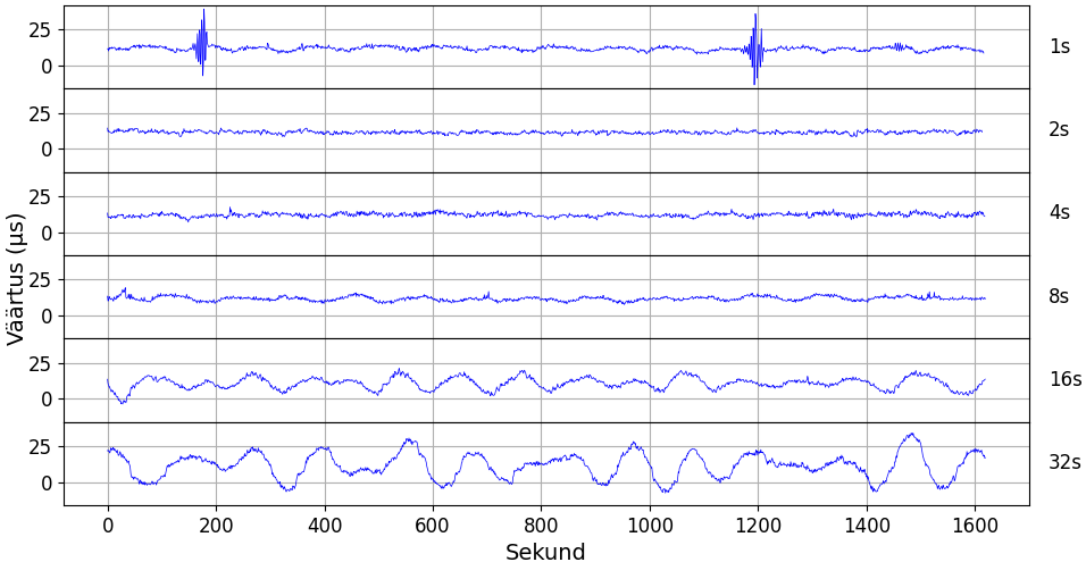
Joonisel 10 on kujutatud katse ülesehitus. Mõlemad Raspberry-d on ühendatud GPS seadmega, mis omakorda on ühendatud antennidega parema GPS signaali saamiseks. Raspberry-d on omavahel ühendatud maa ja GPIO viikudega, millelt hakatakse signaaliimpulsse saatma.



Joonis 10. GPS sünkronisatsioon kahe seadme vahel illustratsioon.

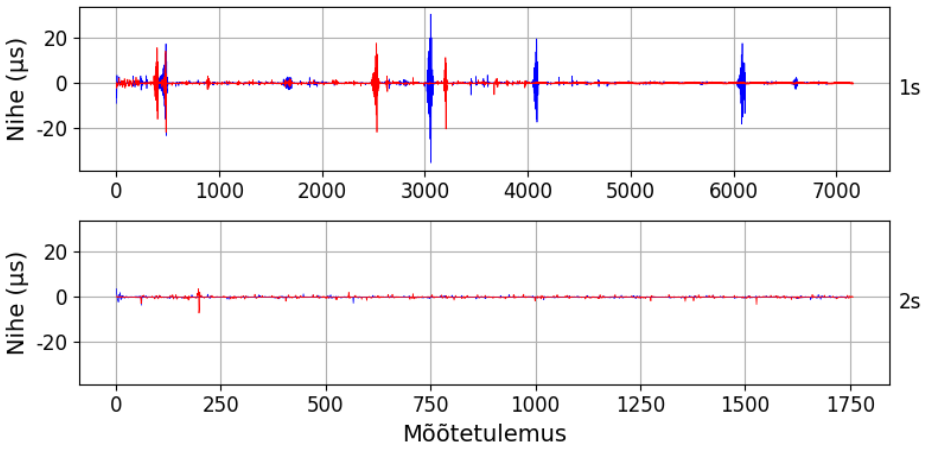
Katse läbiviimiseks kirjutati *Bash* skript, mis enne igat katse alustamist lasi mõlema seadme PPS sünkronisatsioonil stabiliseeruda 5 min. Peale seda jooksutati ühte testi 55 min mille järel katsetulemused salvestati, muudeti uuendamisintervalli *Chrony* tarkvaras ning alustati järgmise testi läbiviimist samuti stabiliseerimiseks vajaliku ajaga.

Tulemustest (Joonis 11) on näha, et pikemate uuendamisintervallidega läheb sünkronisatsiooni täpsus madalamaks. Katse viidi läbi küsimine intervallidega graafikult ülevalt alustades 1, 2, 4, 8, 16 ja 32 sekundit. Ühe sekundilisel intervallil on näha signaalis häireid, mida võivad potentsiaalselt põhjustada katse tegemise ajal Eesti õhuruumis esinenud GPS signaali häired ja/või *Linuxi* hilistused [38].



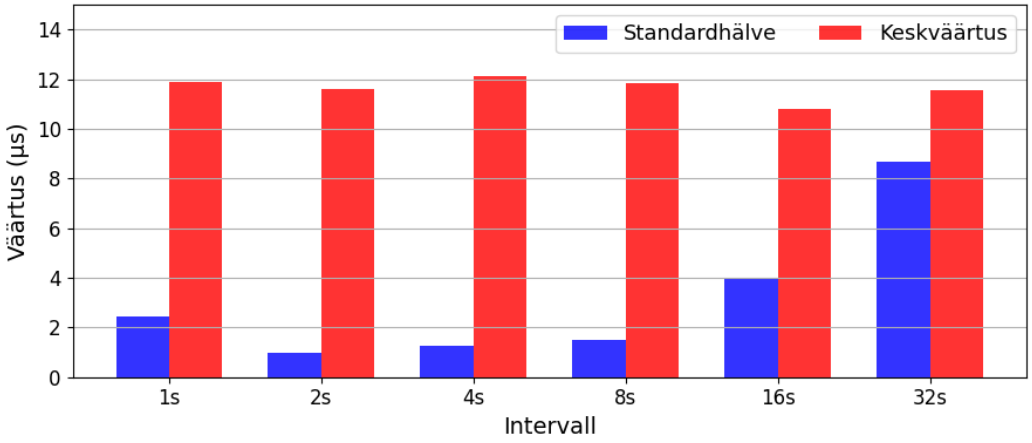
Joonis 11. Viited erinevatel GPS uuendamisintervallidel.

Ühe sekundilise intervalli signaalihäired kajastuvad ka *Chrony* süsteemikella sättimise logides. Joonis 12 esimesel graafikul on kujutatud signaali saatja süsteemikella sättimised sinisega ja vastuvõtja seadme sättimised punasega. On näha, et iga individuaalse PPS signaali järgi süsteemikella uuendades tekkivad vead, mis küündivad mõnekümne mikrosekundini. Teisel graafikul on kujutatud iga kahe sekundi tagant ehk kahe PPS signaali mediaani kasutades tehtud sättimised, kus nii suuri häireid enam ei tekki.



Joonis 12. 1 s ja 2 s intervallide kella sättimise Chrony logid.

Vaadates saadud katsetulemuste kõikide intervallide keskväärtusi ja standardhälbeid (Joonis 13), on näha, et kahesekundiline intervall annab madalaima standardhälbe ehk parima stabiilsuse. Punasega on tähistatud keskväärtused ja sinisega standardhälbed.



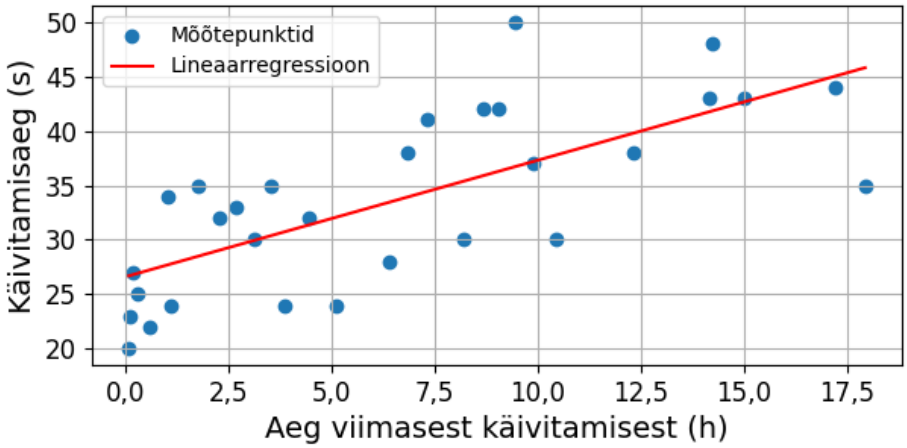
Joonis 13. Keskväärtused ja standardhälbed erinevatel GPS intervallidel.

Parima tulemuse ehk kahesekundilise intervalli standardhälve on 0,97 µs ehk kolme sigma reegli järgi, mis ütleb, et 99,7% väärtustest langeb kolme standardhälbe vahemikku, saame vahemiku ±2,92 µs [39]. See tähendab, et praktiliselt kõik mõõteväärtused on ligikaudu kolme mikrosekundi vahemikus keskväärtusest. Keskväärtuse on praktikas võimalik kõikidest mõõtetulemustest aga ka maha arvata, millepärast on standardhälve mõõtetäpsuse määramisel olulisem.

9.4 GPS sünkroniseerimiskiirus

Katse eesmärk oli uurida, kui kiiresti suudab GPS peale süsteemi käivitamist kellaaega sünkroniseerida. Esimesel katsel mõõdeti aega, mis kulub GPS ühenduse saamiseks Raspberry ja ka GPS mooduli külmkäivitamisest alustades. Katse jaoks mõõdeti aega Raspberry ja GPS mooduli vooluga ühendamisest kuni esimese PPS signaali sünkroonisatsioonini *Chrony* programmis.

Tulemustest on näha (Joonis 14), et käivitamisajal ja enne seda ilma vooluta oleku ajal on seos, aga see võib laialdaselt varieeruda. Sinisena on märgitud individuaalsed katsetulemused ja punane joon tähistab lineaarregressiooni ehk potentsiaalset lineaarfunktsiooni kuhu katsetulemused võiksid langeda [40]. Peab lisama, et katseid tehti ainult kuni umbes 18 tundi vooluta oleku ajani, ning on tõenäoline, et pikemate ajavahemike, nagu mitme päeva või nädalaga, ei jätka käivitamisaeg samasugust tõusu.

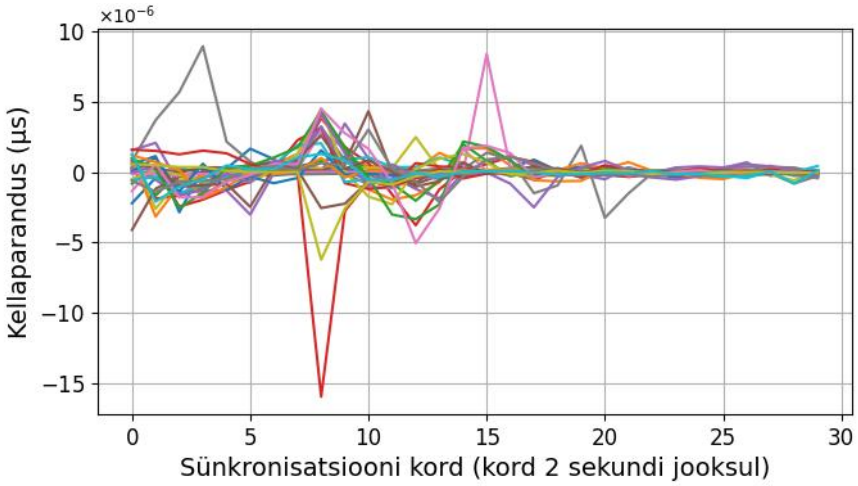


Joonis 14. GPS ühenduse saamise kiirus võrreldes vooluta oleku ajaga.

Tulemustele lisaks on autori üldistest vaatlustest olnud läbi katsete näha, et enamus katsetest on GPS moodul saanud kätte GPS signaali juba enne kui Raspberry on täielikult käivitunud. Seda on näidanud mooduli peal asetsev staatuse valgusdiood. Ainult osadel pikematel käivitamisaegadel, mis jäävad 40 kuni 50 sekundi juurde, on GPS signaali leidmine võtnud kauem.

Teiseks katseks salvestati *Chrony* poolt tehtud sünkronisatsioonide logid, et mõõta sünkronisatsiooni stabiliseerimiseks kuluvat aega (Joonis 15). Graafikult on välja jäetud kõikide mõõtmiste esimene sünkronisatsioon, mis on kella esimese sättimise pärast mitu suurusjärku järgnevatest sättimistest erinevad ning pole antud katse jaoks ka tähtsad. Iga värv joonisel tähistab ühte katset.

Kokku tehti 30 katset, kust on näha, et kokkuvõttes stabiliseeruvad kõik katsed umbes kahekümnekolmandaks korraks, mis on 48 sekundit, arvestades ka väljajäetud esimest sättimist.

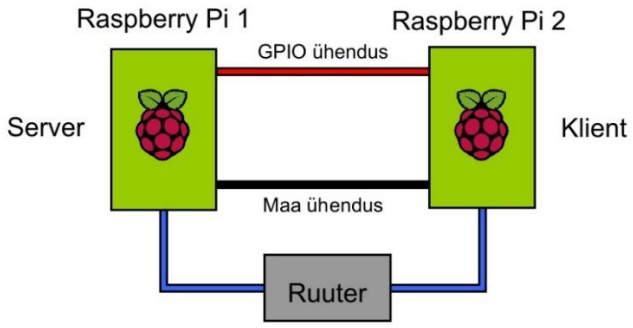


Joonis 15. Sünkronisatsioonikiirus peale GPS ühenduse saamist.

Liites kokku nii käivitamisaja kui ka sünkronisatsiooni stabiliseerumise aja, saab öelda, et kogu protsessi aeg jääb kõikides antud katsetes alla 2 minuti. See võib potentsiaalselt aga pikemate vooluta oleku aegadega suureneda. Enne süsteemi kasutamist on tähtis oodata antud aeg, et väljastatud viited oleksid õiged. Lisa sünkronisatsiooninäitaja on GPS mooduli sisseehitatud punane valgusdiood, mis hakkab pikema intervalliga vilkuma, kui moodul on sünkronisatsiooni saavutanud [23].

9.5 NTP päringusageduste võrdlus kohtvõrgus

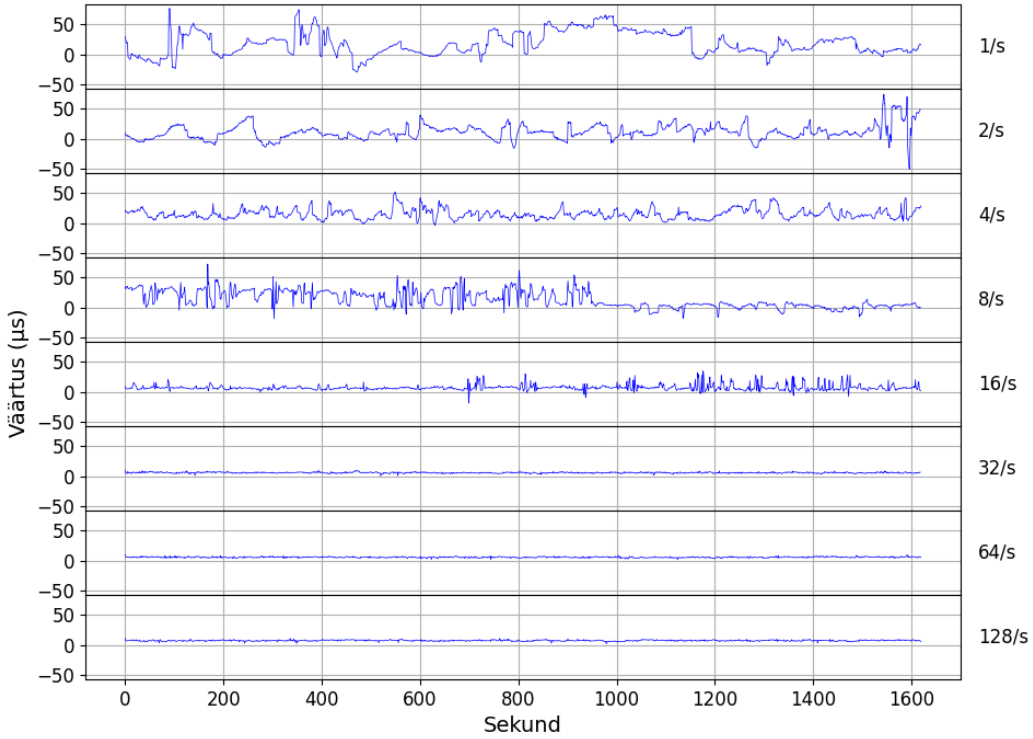
Antud katse oli sarnane katsele, mida käsitleti peatükis 9.3. Peamine vahe on selles, et võrreldakse erinevaid NTP päringusagedusi. Antud katse jaoks muudeti üks Raspberry NTP serveriks ning teine Raspberry kliendiks (Joonis 16) ning võrreldi erinevaid päringusagedusi. Sarnaselt eelpool mainitud katsega, on ka siin kokku ühendatud Raspberry maa ja ühed GPIO viigud, mille pealt hakatakse signaali saatma.



Joonis 16. Kohtvõrgu NTP katse illustratsioon.

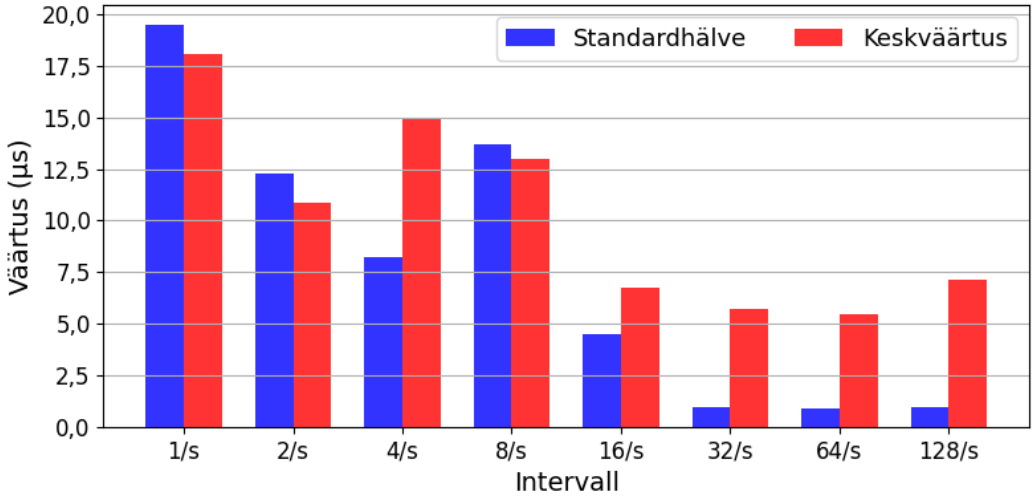
Katse jaoks kasutati sagedusi ühest päringust kuni 128 päringuni sekundis, mis on *Chrony* maksimaalne võimekus. Igal katsel uuendati kella 1 sekundi tagant ehk „filter“ suurus on kõikidel katsetel päringute arv sekundis. Katse läbiviimiseks kasutati sama põhimõttega *Bash* skripti, mida käsitleti peatükis 9.3.

Tulemustest (Joonis 17) on näha, et viited on aeglasematel intervallidel võrreldes GPS pikemate intervallidega oluliselt halvemad. Sünkronisatsioon paraneb oluliselt alates 32-st päringust sekundist. *Chrony* hinnangul on keskmine võrguhilistus serveri ja kliendi vahel umbes 100 µs, mis võimaldab pakkuda juba peaaegu parima võimaliku stabiilsuse 32 päringut sekundi juures.



Joonis 17. Viited erinevate NTP päringuintervallidel.

Tulemused kokkuvõttes (Joonis 18) on näha, et stabiilseimaks pärimissageduseks osutus 64 päringut sekundis, kus standardhälve on 0,87 µs. See on juba väga lähedane peatükis 9.1 tehtud katsetega, kus standardhälve oli 0,78 µs. Nii 32 ja 128 pärimissagedused olid ainult umbes pool sajandikku mikrosekundit ebastabiilsemad ehk peale 32 päringut pole enam otsest suurt kasutegurit ja tulemus võib oleneda juba individuaalsest katsest.

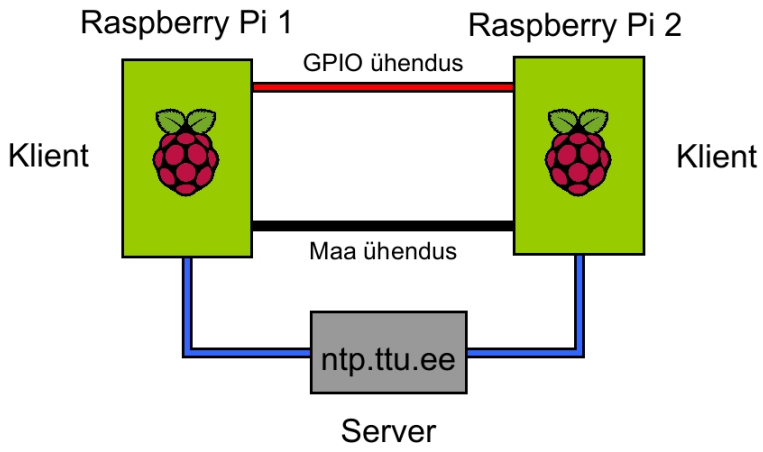


Joonis 18. Keskväärtused ja standardhälbed erinevate NTP päringute intervallidel.

9.6 Sünkronisatsioonitäpsus kolmandast seadmest sünkroniseerimisel

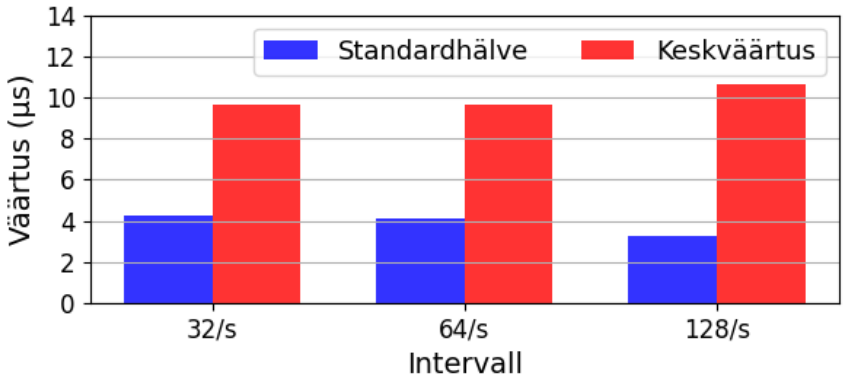
Peatükis 9.5 käsitleti situatsiooni, kus üks Raspberry on NTP server ja teine klient. Teistsugune ja realistlikum võib olla situatsioon, kus mõlemad Raspberry-d küsivad aega ühelt NTP serverilt. Selle jaoks kasutati läbi TalTechi sisevõrgu „ntp.ttu.ee“ aadressil asetsevat serverit, millelt mõlemad seadmed aega pärisid (Joonis 19). Katseks kasutati kolme parimat sünkroniseerimisintervalli ehk 32, 64 ja 128 päringut sekundis, mis leiti peatükis 9.5.

Katses kasutatud ülikooli server jookseb ülikooli IT meeskonna sõnul *Debian* 12 serveri operatsioonisüsteemil ning kasutab samuti *Chrony* tarkvara. Serveri aega ise uuendatakse läbi EENEti NTP serverite. *Stratum* klassifikatsioonis on tegu *stratum* 3 serveriga.



Joonis 19. TalTech NTP serveri katse illustratsioon.

Tulemustest on näha (Joonis 20), et üleüldiselt on nii keskväärtused kui standardhälbed oluliselt suuremad kui peatükis 9.5. See tuleneb asjaolust, et ühendus ülikooli NTP serveri ja Raspberry-de vahel oli oluliselt ebastabiilsem ja pikema hilistusega. *Chrony* sõnul jäid katsetel võrguhilistused umbes kolme millisekundi juurde, mis on umbes 30 korda kõrgem kui eelmises peatükis. See annab aga eelise kõige suuremale pärimissagedusele, sest lisapäringud saavad veel mõõtmisi parandada ning enam ei olda nii lähedal signaali saatmise ja vastuvõtmise programmide stabiilsusele, mis oleks teoreetiline maksimum. 128 päringut sekundis saavutas standardhälbe 3,2 µs, mis kolme sigma reegli järgi annab stabiilsuseks ±9,6 µs ehk umbes 10 mikrosekundit.



Joonis 20. Keskväärtused ja standardhälbed TalTech NTP serveriga.

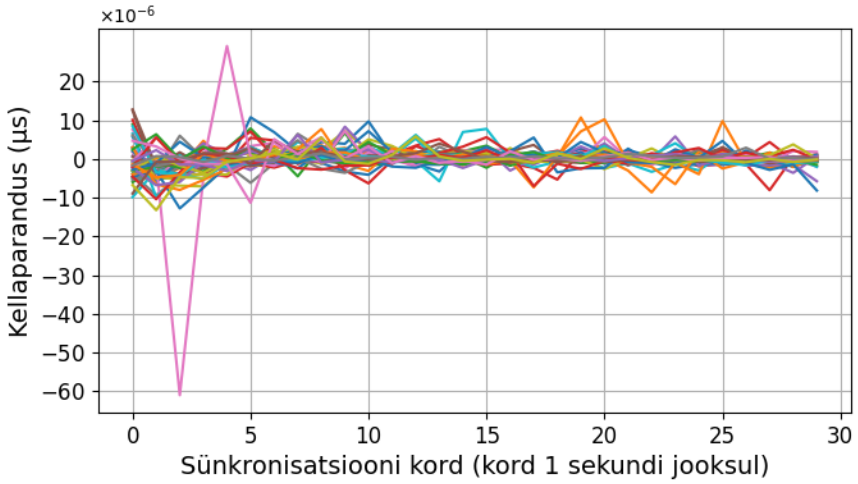
Antud katse vastab rohkem reaalelulistele võrguhilistustele, kus NTP pakett peab minema läbi mitme kohtvõrgu kommutaatori. See, et kiireim pärimissagedus on antud katses ka

kõige parem, näitab, et seda on praktiliselt kõige mõistlikum kasutada. Seda ainult juhul, kui suurem koormus on antud kohtvõrgus ja serveris aktsepteeritav.

9.7 NTP sünkronisatsioonikiirus

Sarnaselt peatükis 9.4 käsitletud GPS sünkronisatsioonikiirusele, on selle katse eesmärk mõõta NTP sünkronisatsioonikiirust. Antud kiirus ei olene enam GPS moodulist ehk kiirust mõjutavad ainult Raspberry käivitamise kiirus ja sünkronisatsiooni stabiliseerimiseks kuluv aeg. Raspberry enda käivitamise kiirus jäi stabiilselt 11 ja 13 sekundi vahele ning ei olenenud ilma vooluta oleku ajapikkusest.

Katse jaoks kasutati parimat intervalli, mis leiti peatükist 9.6 ehk 128 päringut sekundis. Graafikul (Joonis 21) tähistab iga värv ühte katset. Välja on jäetud esimene mõõtmine samal põhjusel, mis peatükis 9.4 ehk see on liiga suur ja summutab graafikult järgmised sättimised. Kokku tehti 50 katset ja tulemustest on näha, et kõik kellasättimised jäävad 10 mikrosekundi vahemikku umbes kuue sekundiga, liites juurde esimese graafikult välja jäetud mõõtmised. Jättes välja ühe katse, mis on tähistatud graafikul roosa värviga, on süsteem sünkroniseeritud juba paari sekundiga.



Joonis 21. NTP sünkronisatsioonikiirus TalTech NTP serveriga.

Kogu süsteemi sünkronisatsiooniaeg jääb kokkuvõttes alla 20 sekundi. See on üle poole kiirem võrreldes umbes 48 sekundiga GPS puhul. NTP annab küll võrguhilistustest tulenevalt palju halvema täpsusega ühe aja päringu kohta kui otse viigule tulev GPS signaal, aga NTP võimaldab saata palju rohkem päringuid kui GPS, mis on limiteeritud ühele päringule sekundis.

10 Soovitused

Katsete tulemustest on näha, et GPS ja NTP on oma eelised ja puudused ning ühe või teise kasutamine sõltub kasutusvajadustest. Katsetest ilmuvad välja soovitused ühe või teise võimaluse kasutamiseks.

10.1 Operatsioonisüsteemi soovitused

Kuna *Linux* operatsioonisüsteemil on arvestatav mõju antud sünkronisatsioonilahenduse kasutamisel, siis on tähtis anda ka selle osas soovitusi. Peamiselt on soovitatav kasutada modifitseeritud *Linuxi kernelit*, millel on võimekus käitada reaalajalõime. Katsetest järeldus, et reaalajalõimed on stabiilsemad ja kiiremad, eriti suurema protsessori koormuse juures.

10.2 GPS soovitused

NTP on ideaalsetes tingimustes GPS-ga samal tasemel, kuid muutub kiiresti halvemaks situatsioonides, kus võrguühendus on NTP sünkrooniks halb. Sellistes tingimustes on soovitatav kasutada GPS ühendust. Internetiühenduse puudumisel on GPS kasutamine aga ainuke saadaolev viis.

On soovitatav kasutada 2 s sünkronisatsiooni intervalli, mis saavutas katsetes parima stabiilsuse. Kahesekundiline intervall laseb kasutada igaks süsteemikella sättimiseks kahe ajatempli mediaani, mis vähendab GPS häirete või operatsioonisüsteemi viivituste mõju.

GPS peamine puudus on vajadus hea GPS signaali järgi, mida saab ainult olukordades, kus vastuvõtja antennil on hea vaade taevale. Kinnistes siseruumides on GPS kasutamine raskendatud või ei ole üldse võimalik. Lisaks peab arvestama, et GPS sünkronisatsioon võib võtta signaali saamiseks kauem aega. Kui need omadused pole süsteemis aktsepteeritavad, siis on soovitatav kasutada NTP lahendusi.

10.3 NTP soovitused

Peamised NTP eelised on selle kasutamise lihtsus. Kui on kohtvõrgus olemas NTP server, siis seadme ülesseadmiseks ei lähe vaja peale kohtvõrgu ühenduse rohkem riistvaralisi komponente. Lisaks toimub algne sünkronisatsioon kiiremini, sest ei pea otsima GPS signaali ja päringusagedus on kõrgem.

Soovitatav on kasutada kiireimat pärimissagedust, mis *Chrony* programmis on 128 päringut sekundis, sest see annab parima stabiilsuse pikemate võrguhilistuste puhul ja võrdväärse stabiilsuse ideaalsemates oludes.

Peamine puudus võrreldes GPS süsteemiga on stabiilsuse kiire vähenemine võrguühenduse halvenemisel: heaks sünkronisatsiooniks on vajalik stabiilne ja võimalikult minimaalse viite kaugusel olev NTP server. Kui sünkronisatsioonitäpsus on süsteemis kõige tähtsam ning ollakse valmis soetama lisa riistvarakomponente, siis on mõttekam kasutada GPS võimalusi.

11 Töö järeldused ja arenguvõimalused

Lõputöö saab lugeda õnnestunuks, sest leiti nii eeliseid ja puuduseid mõlemal süsteemil, mis hõlbustab ühe või teise kasutamist erinevates situatsioonides. Õnnestus optimeerida operatsioonisüsteemi stabiilsemaks ning kasutada registreid signaali saatmiseks ja vastuvõtmiseks.

Üleüldiselt sujus töö hästi. Peamine probleem oli Raspberry Pi 5 kohta info raskesti leitavus: tihti leiti pärast osade katsete tegemist uut informatsiooni, mida taheti ka kasutada. See aga tekitas vajaduse teha katsed uuesti, sest konfiguratsiooni oli muudetud. Näiteks kasutati enne otse mäluregistritele üleminekut eraldi teeki, mida lõpuks vaja ei läinud. Registrite kasutamine tundus alguses liiga keeruline seadme uudsuse tõttu, mis tegi vastava informatsiooni leidmise raskemaks ja registritele üleminek toimus hiljem.

Tulemused olid enamjaolt ootuspärased. Üllatas asjaolu, et NTP saab ajasünkronisatsioonis olla teatud tingimustes sama hea kui GPS. Lisaks ka, et operatsioonisüsteem saab tekitada märgatavaid viiteid isegi olematu koormusega, mis potentsiaalselt jäi ka osade sünkronisatsioonitäpsuse tulemuste piiriks. Seda saab oletada asjaolust, et nii NTP kui ka GPS jõudsid väga lähedale impulssi saatmise ja vastuvõtmise programmide stabiilsusele.

Tööd saab edasi arendada mitmel viisil. Esiteks aga oleneb edasiarendus kasutaja vajadustest. Antud töö kasutusvaldkonnas mõõta viidet kaamera ja ekraani vahel seab mõõtetäpsusele piiri kaamera ja ekraani kaadrisagedus, mis võib edasised mõõtetäpsuse parandused ebavajalikuks muuta. Kui aga nähakse kasu veelgi täpsematel mõõtetulemustel, siis esiteks on võimalus leida veelgi kohti, kus optimeerida operatsioonisüsteemi stabiilsust erinevate komponentide väljalülitamisega või lisa *kerneli* modifikatsioonidega.

Lisaks saab veel võrrelda näiteks erinevaid GPS seadmeid, võrgukaarte, arvuteid ja võrguühendusi. Töös saadud tulemusi saab ka võrrelda PTP ehk *Precision Time Protocol*

sünkronisatsiooniga, mis on kohtvõrkudes potentsiaalselt veel täpsem kui NTP [41]. Selles lõputöös taheti keskenduda ainult NTP ja GPS osale, millepärast seda ei käsitletud.

12 Kokkuvõte

Töö eesmärk oli uurida GPS ja NTP erinevusi ajasünkronisatsioonis. Peamiselt uuriti mõlema teenuse puuduseid ja eeliseid. Kõik töö eesmärgid said täidetud: uurimisküsimused, mis töö alguses püstitati, on vastuse saanud.

NTP sünkronisatsiooni viis läbi *Chrony* programm, mis soosis valima seadmeteks Raspberry Pi 5 väikearvutid, sest need kasutavad *Linux* operatsioonisüsteemi. GPS sünkronisatsiooniks kasutati Adafruit GPS moodulit koos eraldi välisantenniga. Aja sättimiseks kasutati GPS mooduli välja antavat PPS signaali.

Kuna seadmed kasutasid *Linuxi* operatsioonisüsteemi, siis katsete läbiviimiseks seda optimeeriti, et protsesside ajastamisest tulenevad viivitused oleksid minimaalsed. Selle jaoks kasutati reaalaja *kerneli* modifikatsiooni, mis vähendas katsetes operatsioonisüsteemist tulenevaid viiteid märgatavalt. Võrreldes impulsside saatmis- ja vastuvõtmisprogrammide jooksmist tava- ja reaalajalõimedel, olid reaalajalõimed stabiilsemad, eriti kui protsessor on koormuse all.

Leiti nii eeliseid kui puuduseid mõlemal teenusel, kuid teatud tingimustes on nii NTP kui ka GPS põhimõtteliselt võrdväärsed. Mõlemad saavutasid stabiilsuse, kus stabiilsus on parem kui ±3 μs. NTP sünkronisatsioon halvenes aga võrguhilistustega märgatavalt. Realistlikumas olukorras, kus server ja klient on eraldatud mitme kommutaatoriga, langes NTP täpsus umbes ±10 μs juurde.

NTP on parem valik olukordades, kus võrguhilistused on serveri ja kliendi vahel minimaalsed ehk nende vahel on näiteks ainult 1 kommutaator. Lisaks on seda soovitatav kasutada, kui seadmele ei ole võimalik tagada vaadet avatud taevale, mida vajab GPS seade. Kui kasutusjuht peab oluliseks kiiret sünkronisatsiooni peale külmkäivitamist, siis on NTP samuti kiirem tänu oma suurtematele pärimissagedustele võrreldes GPS-ga. Seda tõestasid ka katsed, kus NTP algne sünkronisatsioon oli kiirem.

Edasiarenduse vajalikkus oleneb kasutaja vajadustest. Kui kasutaja võib saada kasu edasisest mõõtetäpsuse parandamisest, siis saab süsteemi optimeerida veelgi. Näiteks

uurida teisi seadmeid, operatsioonisüsteeme ja programme. Lisaks on võimalik võrrelda ka PTP protokolli, mis on potentsiaalselt parem kui NTP.

Töö oli autori arvates edukas: kõik uurimisküsimused said vastuse ning katsetulemused olid paljuütlevad. Peamine ajakulu töö jooksul oli Raspberry 5 kohta informatsiooni leidmine, sest tegemist on valdavalt uue seadmega ning vahepeal uue informatsiooni leidmine põhjustas katsete kordamise.

Kasutatud kirjandus

[1] National Institute of Standards and Technology, „NIST Internet Time Service (ITS),“ 10 Veebruar 2010. [Võrgumaterjal]. Available: https://www.nist.gov/pml/time-and-frequency-division/time-distribution/internet-time-service-its. [Kasutatud 13 Mai 2024].

[2] K. P. A. P. A. M. Laura Kõrgmaa, „Irdtorni videoühenduse viite mõõtmine,“ Tallinn, 2023.

[3] D. Schlösser, „GPS disciplined Stratum-1 NTP Server with Raspberry PI - a HowTo,“ 1 Aprill 2024. [Võrgumaterjal]. Available: https://github.com/domschl/RaspberryNtpServer. [Kasutatud 13 Mai 2024].

[4] Gentoo Authors, „Pi4 Stratum 1 Time Server,“ 31 Märts 2024. [Võrgumaterjal]. Available: https://wiki.gentoo.org/wiki/Pi4_Stratum_1_Time_Server. [Kasutatud 13 Mai 2024].

[5] „GPS Raspberry Pi NTP Server,“ 10 Juuli 2023. [Võrgumaterjal]. Available: https://blog.networkprofile.org/gps-backed-local-ntp-server/. [Kasutatud 13 Mai 2024].

[6] Internet Engineering Task Force, „RFC-5905 Network Time Protocol Version 4: Protocol and Algorithms Specification,“ Juuni 2010. [Võrgumaterjal]. Available: https://www.ntp.org/reflib/rfc/rfc5905.txt. [Kasutatud 13 Mai 2024].

[7] Network Time Foundation, „How NTP Works,“ 23 November 2022. [Võrgumaterjal]. Available: https://www.ntp.org/documentation/4.2.8-series/warp/. [Kasutatud 13 Mai 2024].

[8] University Of Delaware, „Executive Summary: Computer Network Time Synchronization,“ 12 Mai 2012. [Võrgumaterjal]. Available: https://www.eecis.udel.edu/~mills/exec.html#intro. [Kasutatud 13 Mai 2024].

[9] ResearchGate, „High-Performance Time Server Core for FPGA System-on-Chip - Scientific Figure on ResearchGate,“ Mai 2019. [Võrgumaterjal]. Available: https://www.researchgate.net/figure/On-wire-network-time-protocol-NTP-operation_fig1_333054240. [Kasutatud 13 Mai 2024].

[10] National Coordination Office for Space-Based Positioning, Navigation, and Timing, „GPS, Space Segment,“ [Võrgumaterjal]. Available: https://www.gps.gov/systems/gps/space/. [Kasutatud 13 Mai 2024].

[11] National Coordination Office for Space-Based Positioning, Navigation, and Timing, „GPS Educational Poster,“ [Võrgumaterjal]. Available: https://www.gps.gov/multimedia/poster/. [Kasutatud 13 Mai 2024].

[12] National Coordination Office for Space-Based Positioning, Navigation, and Timing, „GPS Accuracy,“ [Võrgumaterjal]. Available: https://www.gps.gov/systems/gps/performance/accuracy/#how-accurate. [Kasutatud 13 Mai 2024].

[13] D. L. Mills, „Pulse-Per-Second (PPS) Signal Interfacing,“ [Võrgumaterjal]. Available: https://www.ntp.org/documentation/4.2.8-series/pps/. [Kasutatud 13 Mai 2024].

[14] „gpsd - a GPS service daemon,“ 7 Aprill 2024. [Võrgumaterjal]. Available: https://gpsd.io/index.html#documentation. [Kasutatud 13 Mai 2024].

[15] LinuxPPS, „LinuxPPS wiki,“ 26 November 2020. [Võrgumaterjal]. Available: http://linuxpps.org/doku.php/start. [Kasutatud 13 Mai 2024].

[16] Chrony, „Introduction,“ 16 Detsember 2021. [Võrgumaterjal]. Available: https://chrony-project.org/index.html. [Kasutatud 13 Mai 2024].

[17] Chrony, „Comparison of NTP implementations,“ 10 August 2023. [Võrgumaterjal]. Available: https://chrony-project.org/comparison.html. [Kasutatud 13 Mai 2024].

[18] O. Obleukhov, „Building a more accurate time service at Facebook scale,“ 18 Märts 2020. [Võrgumaterjal]. Available: https://engineering.fb.com/2020/03/18/production-engineering/ntp-service/. [Kasutatud 13 Mai 2024].

[19] Red Hat, „Configuring NTP Using the chrony Suite,“ [Võrgumaterjal]. Available: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system_administrators_guide/ch-configuring_ntp_using_the_chrony_suite. [Kasutatud 13 Mai 2024].

[20] J. Butts, „What Is The Raspbeery Pi,“ 28 September 2021. [Võrgumaterjal]. Available: https://www.howtogeek.com/754492/what-is-raspberry-pi/. [Kasutatud 13 Mai 2024].

[21] K. S. Eben Upton, „https://www.jbs.cam.ac.uk/2023/how-the-raspberry-pi-mini-computer-freed-up-engineers/,“ 1 August 2023. [Võrgumaterjal]. Available: https://www.jbs.cam.ac.uk/2023/how-the-raspberry-pi-mini-computer-freed-up-engineers/. [Kasutatud 17 Mai 2024].

[22] Raspberry Pi Foundation, „Raspberry Pi 5 Tech Specs,“ [Võrgumaterjal]. Available: https://www.raspberrypi.com/products/raspberry-pi-5/. [Kasutatud 13 Mai 2024].

[23] Adafruit, „Adafruit Ultimate GPS HAT for Raspberry Pi,“ [Võrgumaterjal]. Available: https://www.adafruit.com/product/2324#technical-details. [Kasutatud 13 Mai 2024].

[24] CD Technology, „CD-PA1616D GNSS patch antenna module,“ [Võrgumaterjal]. Available: https://cdn-shop.adafruit.com/product-files/1059/CD%20PA1616D%20Datasheet%20v.05.pdf. [Kasutatud 13 Mai 2024].

[25] Raspberry Pi Foundation, „Operating Systems,“ [Võrgumaterjal]. Available: https://www.raspberrypi.com/software/operating-systems/. [Kasutatud 13 Mai 2024].

[26] The Linux Foundation, „Technical details of the real-time preemption,“ 2 November 2023. [Võrgumaterjal]. Available: https://wiki.linuxfoundation.org/realtime/documentation/start. [Kasutatud 13 Mai 2024].

[27] G. M. W. F. Federico Reghenzani, „The Real-Time Linux Kernel: A Survey on PREEMPT_RT,“ Veebruar 2019. [Võrgumaterjal]. Available: https://dl.acm.org/doi/pdf/10.1145/3297714. [Kasutatud 13 Mai 2024].

[28] The Linux Foundation, „HOWTO build a basic cyclic application,“ 20 Juuni 2017. [Võrgumaterjal]. Available: https://wiki.linuxfoundation.org/realtime/documentation/howto/applications/cyclic. [Kasutatud 13 Mai 2024].

[29] 15 Veebruar 2015. [Võrgumaterjal]. Available: https://codeandlife.com/2012/07/03/benchmarking-raspberry-pi-gpio-speed/. [Kasutatud 13 Mai 2024].

[30] H. Fairhead, „Raspberry Pi IoT In C - Pi 5 Memory Mapped GPIO,“ 10 January 2024. [Võrgumaterjal]. Available: https://www.i-programmer.info/programming/148-hardware/16887-raspberry-pi-iot-in-c-pi-5-memory-mapped-gpio.html. [Kasutatud 13 Mai 2024].

[31] The Linux Foundation, „HOWTO build a simple RT application,“ 29 August 2023. [Võrgumaterjal]. Available: https://wiki.linuxfoundation.org/realtime/documentation/howto/applications/application_base. [Kasutatud 13 Mai 2024].

[32] IBM, „Interrupt coalescing,“ 24 Märts 2023. [Võrgumaterjal]. Available: https://www.ibm.com/docs/en/aix/7.3?topic=options-interrupt-coalescing. [Kasutatud 13 Mai 2024].

[33] Chrony, „chrony.conf(5) Manual Page,“ 9 August 2023. [Võrgumaterjal]. Available: https://chrony-project.org/doc/4.4/chrony.conf.html. [Kasutatud 13 Mai 2024].

[34] Chrony, „FAQ: How can I improve the accuracy of the system clock with NTP sources?,“ 14 Märts 2024. [Võrgumaterjal]. Available: https://chrony-project.org/faq.html#_how_can_i_improve_the_accuracy_of_the_system_clock_with_ntp_sources. [Kasutatud 13 Mai 2024].

[35] The Matplotlib development team, „Matplotlib examples,“ [Võrgumaterjal]. Available: https://matplotlib.org/stable/gallery/index.html. [Kasutatud 13 Mai 2024].

[36] M. Kerrisk, „nice(1) — Linux manual page,“ August 2023. [Võrgumaterjal]. Available: https://man7.org/linux/man-pages/man1/nice.1.html#DESCRIPTION. [Kasutatud 13 Mai 2024].

[37] Arch Linux, „Hackbench,“ 19 September 2020. [Võrgumaterjal]. Available: https://man.archlinux.org/man/hackbench.8.en. [Kasutatud 13 Mai 2024].

[38] V. Väino, „GPS-signaalide häirimine Venemaa poolt on Eesti kohal suurenenud,“ 4 Aprill 2024. [Võrgumaterjal]. Available: https://www.err.ee/1609302099/gps-signaalide-hairimine-venemaa-poolt-on-eesti-kohal-suurenenud. [Kasutatud 13 Mai 2024].

[39] A. Sauga, „STATISTIKA JA TÕENÄOSUSTEOORIA,“ 2006. [Võrgumaterjal]. Available: https://www.sauga.pri.ee/audentes/download/stait.pdf. [Kasutatud 13 Mai 2024].

[40] Yale University, „Linear Regression,“ [Võrgumaterjal]. Available: http://www.stat.yale.edu/Courses/1997-98/101/linreg.htm. [Kasutatud 13 Mai 2024].

[41] O. O. Ahmad Byagowi, „PTP: Timing accuracy and precision for the future of computing,“ Meta, 21 November 2022. [Võrgumaterjal]. Available: https://engineering.fb.com/2022/11/21/production-engineering/future-computing-ptp/. [Kasutatud 13 Mai 2024].

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Auris Prääm

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Mikroarvutite aja sünkroniseerimine GPS ja NTP abil“, mille juhendaja on Priit Roosipuu
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

13.05.2024

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

Lisa 2 – Programmikoodide repositoorium

Kõikide programmikoodide haldus käis GitHubi keskkonnas. Koodidega tutvumiseks on loodud GitHubi repositoorium:

https://github.com/AurisP/NTP