TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Informaatikainstituut

IT40LT
Karl Hõbessalu 163913IAPB

AMATÖÖRSATELLIITIDE JÄLGIMISE RAKENDUS ANDROIDILE

bakalaureusetöö

Juhendaja: Evelin Halling
PhD

Tallinn 2019

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: Karl Hõbessalu

10.05.2019

Annotatsioon

Töö eesmärgiks oli arendada rakendus mobiilseadmetele amatöörsatelliitide jälgimiseks. Rakendus peaks abistama seadistada *yagi*-antenne amatöörsatelliitidelt tulevate signaalide püüdmiseks. Antud ülesande täitmiseks peaks rakendus suutma kuvada satelliidi positsiooni seadme suhtes ja seda võimalikult täpselt ja lihtsalt.

Satelliidi liikumise informatsiooni saamiseks kasutab rakendus veebipäringut. Satelliidi asukoha informatsioon töödeldakse graafiliselt presenteeritavale kujule trigonomeetriliste arvutuste ning teisenduste abil.

Käesoleva töö tulemusena valmis Android rakendus mis annab võimaluse jälgida igat satelliiti, millele vastav TLE on serveris olemas. Satelliidi asukoht kuvatakse ekraanile kollase ringina, mis uuendab enda asukohta vastavalt ajale ning seadme positsioonile kasutaja käes.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 27 leheküljel, 5 peatükki, 11 joonist, 1 tabelit.

Abstract

Amateur satellite tracking application for Android

The following thesis describes the development process for an Android application made for tracking amateur satellites. The purpose of the application is to set up *yagi*-antennas with more precision. For set purpose the application should be able to present the position of the satellite on the screen of the device and provide visual aid to configure the antenna in the direction of the satellite with best possible precision.

To optain orbital information the application uses a web request. Orbital information is then used to present a graphical element on the device by transforming the data via trigonometrical computations.

The result of the thesis was an application which is capable of tracking any satellite the server has proper information about. The application is able to show the position of the satellite as a yellow ring on the screen. The position of the ring on the screen is determined by the time and the orientation which the device is being held in.

The thesis is in Estonian and contains 27 pages of text, 5 chapters, 11 figures, 1 table.

Lühendite ja mõistete sõnastik

API	*Application programming interface*. Kirjeldab kahe või enam komponendi vahelist suhtlemist.
TLE	*Two-line element set*. Kirjeldab objekti liikumist maa suhtes.
AR	*Augmented reality*. Reaalse keskkonna peale genereeritud virtuaalsed objektid.
URI	*Uniform Resource Identifier*. Ühtne ressursi-indikaator on internetis leiduvate tarkvaraliste võrguressursside unikaalne aadress.
GET	*GET request.* Pöördumine veebiserveri poole andmete pärimise eesmärgil.
JSON	*JavaScript Object Notation.* Andmevahetusformaat

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Lõputöö eesmärgiks on hõlbustada käsitsi juhitavate yagi-antennide. Yagi-antennide efektiivsemaks suunamiseks otsustas autor arendada rakenduse, mis kuvaks jahitava satelliidi positsiooni android seadme ekraanile ja liigutaks seda vastavalt seadme asendile käes. Dokkides seadme füüsilise liidese abil antenni külge ja korrigeerides satelliiti tähistava punkti seadme ekraani keskele, peaks antenn olema suunatud satelliidi poole.

Taust ja probleem

Antud töö põhineb tudengisatelliidi projekti algatatud soovile tutvustada satelliitraadiotega seotud tehnoloogiat Eesti koolides. Satelliitidelt signaali kättesaamiseks tuleb aga osata antenne õigesti suunata. Kuna yagi-antennide suunamiseks kasutatav motoorika võib olla kallis või kättesaamatu, aitaks mobiilseadmetes olevate sensorite kaudu satelliitide positsioneerimine alandada õppetöö kulutusi.

Ülesande püstitus

Rakenduse eesmärk on kuvada otsitava satelliidi relatiivne asukoht ekraanile. Selleks on vaja rakendusel järgmisi andmeid ja funktsionaalsusi:

- Satelliidi hetke asukohta kirjeldavate andmete saamine.
- Seadme positsiooni ja asendit kirjeldavate andmete töötlemine.
- Satelliidi asukoha ja seadme positsiooni erinevuste arvutamine.
- Erinevustele vastava info kuvamine ekraanile praktiliselt kasutataval kujul.

Metoodika

Arendatud rakendus valmis arenduskeskkonnas Android Studio [1]. Rakendust katsetati seadmetel Huawei VNS-L21 ja Samsung Galaxy S7. Validatsiooni jaoks kasutati programmi SDR console v3.

Ülevaade tööst

Töö on jaotatud 3 etapiks:

- Olemasolevate rakenduste võimaluste analüüs. Autor analüüsib analoogsete kasutatavate rakenduste võimalusi.
- Autori poolt loodud rakenduse analüüs ja teostus.
- Validatsioon. Autor proovib püüda signaali satelliidilt suunates antenni loodud aplikatsiooni abil.

2 Sarnaste rakenduste võrdlus

Antud peatükis analüüsib autor Google Play poest saadavaid rakendusi ja võrdleb neid enda arendatud rakendusega. Satelliidi antenni või taldriku suunamist abistavaid rakendusi leidub palju ning autor otsustas analüüsimiseks valida kaks rakendust mis vastavad järgmistele tingimustele:

- Rakendusel on hinnanguks antud vähemalt 4 tärni
- Rakendusel on vähemalt 100 000 allalaadimist
- Rakendus on saadav tasuta

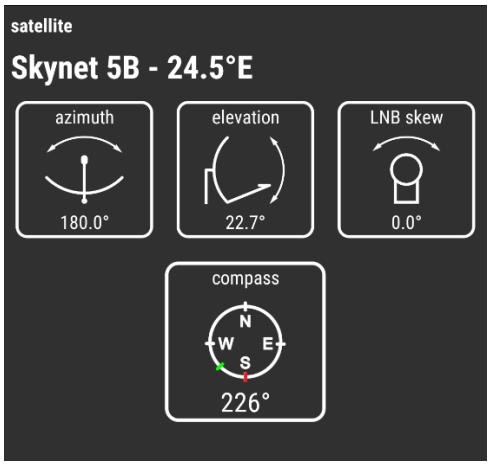
Analüüsimiseks valis autor rakendused Satellite Finder PRO ja SatFinder.

2.1 SatFinder

Rakendus on arendatud Maciej Grzegorczyk poolt. Tegemist on rakendusega, mille eesmärgiks on aidata paigaldada satelliidi taldrikuid, kuid sobib ka yagi-antennide suunamiseks. Rakendus annab asimuudi, kõrgusnurga ja LNB kalde antenni või taldriku suunamiseks vastavalt seadme GPS asukohale ja valitud satelliidile. Rakendus kasutab seadme asendi määramiseks kompassi. Rakenduse võimaldab ka kasutada *augmented reality* vaadet mis kuvab satelliidi asukoha kaameral [8]. Rakenduse suurus on 23M [9].

Rakendus vajab töötamiseks ligipääsu järgmistele andmetele [8]:

- Asukoht
- Internet
- Magnetomeeter ja kiirendusandur
- Kaamera (vabatahtlik)
- Mälu (vabatahtlik)



Pilt 1. näide SatFinder rakenduse vaatest

Rakendus võimaldab jälgida 182 satelliidi positsiooni. Rakendus arvutab vaid satelliidi positsiooni vastavalt staatilistele positsiooni kirjeldavatele väärtustele .csv tüüpi failis. Jälgitavad koordinaadid arvutatakse vaid rakenduse avamisel ning need tuginevad staatilistele väärtustele rakenduse alla laadimisel kaasa tulevas .csv failis.

Rakendus kuvab jälgitava satelliidi asimuudi ja kõrgusnurga vaid kirjalikul kujul ning ainukeseks abivahendiks on kompassi element. AR vaade kuvab kasutajale infot seadme tagakülje kaamera suuna asimuudi ja kõrgusnurga kohta ning muudab ekraani keskel oleva indikaatori roheliseks kui seade on lubatud veaga satelliidi poole suunatud. Rakenduse poolt kuvatud seadme asimuut oli pidevas kõikumises -3/+3 kraadi vahel isegi peale kalibreerimist.

Rakendus võimaldab lisada juurde jälgitavaid satelliite, kuid ei toeta TLE standardile vastavaid kirjeldusi ja ei oska arvutada satelliidi liikumise trajektoori.

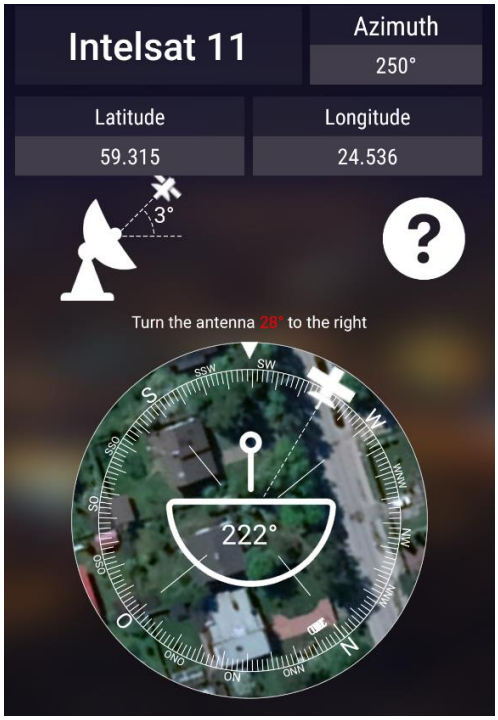
2.2 Satellite Finder PRO

Rakendus on arendatud firma Comptech poolt. Rakendus on loodud paraboolantennide paigaldamiseks ja kuvab jälgitava satelliidi asimuudi ja kõrgusnurga seadme suhtes. Satellite Finder PRO võimaldab jälgida 133 satelliiti, kuid ei võimalda lisada satelliitide andmeid. Rakenduse suuruseks on 4,8 MB [10].

Rakendus vajab töötamiseks ligipääsu järgmistele andmetele:

- Asukoht

- Kaamera
- Magnetomeeter ja kiirendusandur
- Kaamera



Pilt 2. näide Satellite Finder PRO vaatest

Kindla satelliidi jälgimisel kuvatakse kasutajale kompassi moodi element, mis näitab satelliidi asimuuti ning vibreerib kui kasutaja on suunanud seadme satelliidi asukoha poole. Rakenduse poolt näidatud seadme asimuut kõikus rakenduse avamisel kuni 14 kraadises vahemikus, kuid oskas soovitada manuaalset kompassi kalibreerimist. Manuaalse kalibreerimise järel sai asimuudi kõikumise 6 kraadini.

Satellite Finder PRO pakub võimaluse kuvada kõik vaateväljas olevat satelliidid AR tüüpi vaates seadme ekraanile, kuid on äärmiselt ebastabiilne ja hoiab ekraanil liiga palju infot. AR vaatel puudub võimalus jälgida vaid ühte satelliiti.

Dokumentatsioonis ei ole kirjeldatud millele tuginedes arvutatakse satelliidi koordinaadid seadme suhtes. Lisaks ei uuenda aplikatsioon satelliidi asukohta.

2.3 Järeldus

Mõlemad rakendused on loodud paraboolantennide pika-ajaliseks paigalduseks. Rakendused suudavad jälgida geo sünkroonseid satelliite, mis on alati maa suhtes samas kohas.

Mõlema rakenduse puhul oli seadme asimuudi määramine manuaalse kalibreerimise järel täpsusega -3/+3. Tuginedes faktile, et mõlemate rakenduste puhul esines nii suur kõikumine kompassil võib järeldada, et mõlemad rakendused piirdusid seadme asendi määramisel seadmesse sisse ehitatud akselomeetriga.

Autori sooviks on jälgida liikuvaid amatöörsatelliite, mille koordinaadid muutuvad lühikese aja jooksul palju ja on vaateväljas vaid lühikese aja (~10 minutit). Amatöörsatelliitide puhul võib olla ka signaal nõrk ning seetõttu on signaali püüdva antenni suund kriitiline.

Eelnimetatud põhjustel on analüüsitud rakendused autori jaoks puudulikud ning neid ei saa kasutada autori eesmärgi täitmiseks.

3 Rakenduse teostus

Järgnevas peatükis on kirjeldatud rakenduse nõuded, arhitektuur ja vajalike komponentide ehitus.

3.1 Arendatava süsteemi nõuded

Keel	Rakendus peab olema inglise keeles
Kasutajaliides	• Kasutajaliides on soetatud Androidil jooksvatele telefonidele või tahvelarvutitele. • Peavaate taustaks on kaamera. • Näidatakse aega jälgitava satelliidi ülelennu sessiooni alguseni. • Satelliiti kujutav punkt on ekraanil vaid siis, kui satelliit on reaalselt seadme kaamera vaateväljas. • Seadme asendi korrigeerimiseks peavad olema tagatud abivahend
Täpsus	Rakendus näitab satelliidi asukohta kuni 15° veaga.
Veakäsitlus	Rakendus peab teavitama kasutajat ebaõnnestunud veebipäringute puhul
Kiirus	• Satelliidi koordinaatide päring ei tohiks aega võtta kauem kui 5 sekundit. • Jälgitavate koordinaatide vahetus peab toimuma piisavalt tihti, et see ei mõjutaks satelliidilt tulevate signaalide püüdmist.

Ligipääsetavus	Rakendus peaks olema saadav tasuta kõigile Android 5.0 ja kõrgema versiooni kasutajatele Google Play poes või mõnes alternatiivses keskkonnas.
Testitavus	Rakenduses peavad olema testitud komponendid: • Satelliidi positsiooni kujutava punkti arvutused • Satelliidi liikumise info päring • Ülelennu sessiooni alustava alarmi komponent

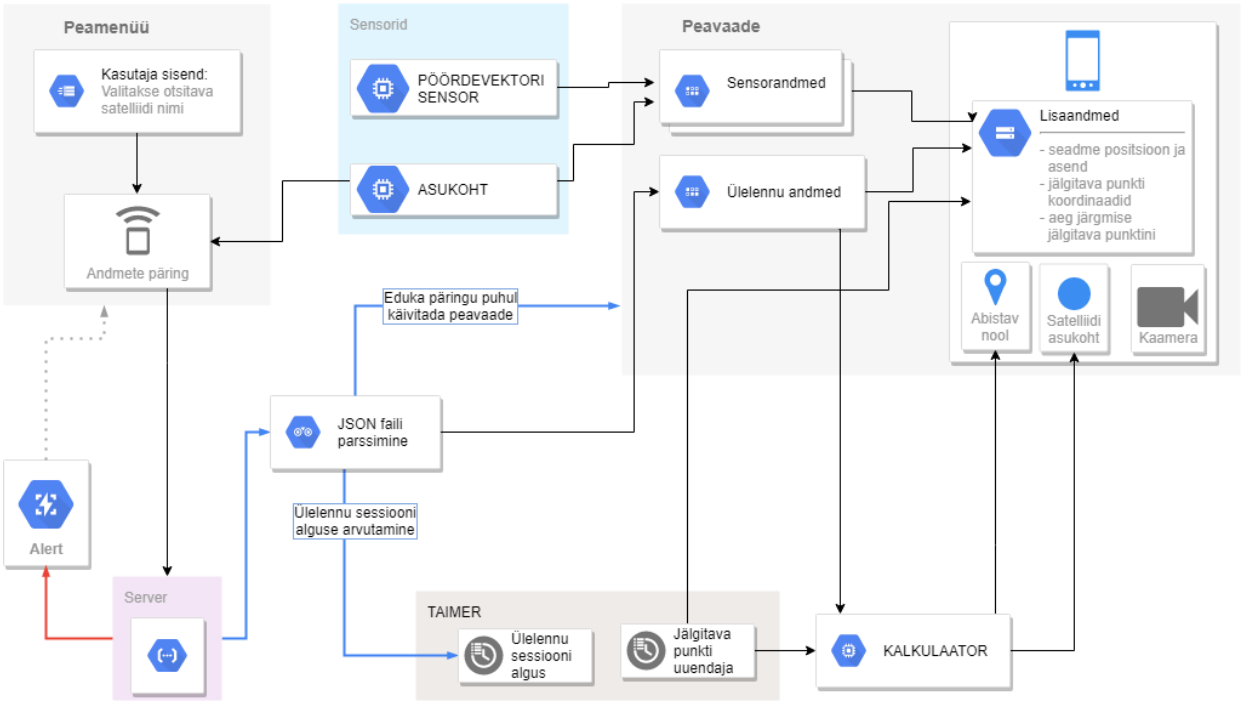
Tabel 1 Süsteemi nõuded

3.2 Kasutatavad seadmed

Rakendus vajab töötamiseks ligipääsu seadmes olevatele sensoritele. Kuna veebirakendused ei võimalda ligipääsu märkimisväärsele osale sensoritest, otsustas autor tehniliste võlgade vältimiseks arendada *native* rakendus. Kuna nutitelefonide seas on Androidi kasutus märkimisväärselt suurem kui IOS kasutus [3] otsustas autor arendada rakenduse Androidile. Rakendus töötab kõikidel 5.0 ja uuemate versioonide Androididel. 5.0 või kõrgemat Androidi versioni omab praegu 85% seadmetest [2].

3.3 Rakenduse arhitektuur

Rakendus koosneb 2 vaatest ja 5 suuremast loogilisest komponendist.



Pilt 3. Rakenduse arhitektuuri diagramm

Rakenduse töötamiseks on rakendusel vaja ligipääsu seadme kaamerale, asukoha haldurile, positsiooni sensoritele ning internetile. Kõik eelnimetatud ligipääsud on rakenduse käivitamiseks kohustuslikud.

Rakenduse esialgsel käivitamisel esitatakse seadmes peamenüü vaade. Peamenüü peab võimaldama koguda infot satelliidi positsiooni päringu jaoks. Päringu jaoks on vaja seadme laius- ja pikkuskraadi ning otsitava satelliidi nime. Satelliidi nime sisestab kasutaja kasutajaliideses olevasse lahtrisse, seadme pikkus- ja laiuskraadi küsib seade Androidi asukoha halduri (*LocationManager*) API käest. Sõltuvalt päringu õnnestumisest kuvatakse kasutajale veateade või viiakse kasutaja edasi Peavaatesse. Peavaate avamise eel töödeldakse päringu vastuseks saadud andmed ja arvutatakse aeg ülelennu sessiooni alguseni.

Ajast sõltuvate tegevuste jaoks on loodud eraldi lõimes (*thread*) taimeri komponent. Antud rakenduses kannab komponent nime PositionUpdateAlarm.java. Komponent töötab kui alarm, mis käivitub API päringust saadud ülelennu sessiooni algul. Alarm käivitab funktsiooni, mis uuendab iga 30 sekundi järel peavaate kalkulaatori objekti

määrates sellele uued jälgitavad koordinaadid. Komponent lõpetab tegevuse kui on jõutud viimaste koordinaatideni.

Peavaates on kasutajaliidesel kolm peamist muudetavat komponenti: Satelliidi asukohta kujutav punkt, punkti korrigeerimiseks loodud abistav nool ja sensori andmed. Iga eelneva komponendi andmeid uuendatakse sensori halduris määratud aja tagant.

3.3.1 API päring

Rakendus pärib kuvatava satelliidi liikumise kohta käivad andmed üle võrgu. Päringu tegemisel tuleb arvestada vahelduva info koguste suurusi – suurte failide pärimisel tuleks kasutada *DownloadManager* teeki ning väiksemate puhul alternatiivseid teeke nagu Volley või Retrofit. Kuna antud rakenduse poolt päritav info on väike (~10KB) kasutab autor rakenduses REST päringute tegemisel Volley't. Volley on ametlik Google poolt pakutav teek HTTP päringute tegemiseks, mille eesmärgiks on teha veebipäringud Androidi aplikatsioonides lihtsamaks ja kiiremaks [6].

Näide GET päringu URI'st:

http://1.2.3.4:8080/endpoint?satname=horyu%202&latitude=59.3148795&longitude=24.5363427&mobiletracker=true

- `satname` – Otsitava satelliidi nimi. Kasutaja sisestatud nimi peab esinema serveris olevas TLE failis. Juhul kui nimi ei ühti andmetega serveris ei ole võimalik arvutada vastust ja tagastatakse veateade.

- `latitude` - Seadme asukoha laiuskraad. Antud väärtuse määrab seade viimase asukoha info järgi.

- `longitude` – Seadme asukoha pikkuskraad. Antud väärtuse määrab seade viimase asukoha info järgi.

- `mobiletracker` – Tõeväärtus kirjeldamaks, et päring on esitatud antud mobiilrakenduse kasutamise eesmärgil.

Näide vastusest seisuga 30.04.2019:


```
[
    {
        "id": 83,
        "paramsId": 11,
        "riseTime": "2019/4/30 07:55:27",
        "riseAzimuth": "29:15:30.3",
        "maximumAltitudeTime": "2019/4/30 08:00:21",
        "maximumAltitude": "7:25:01.2",
        "setTime": "2019/4/30 08:05:12",
        "setAzimuth": "121:35:35.5",
        "rollAngles": [
            {
                "id": 1589,
                "flyOversId": 83,
                "subLat": 75.4296107405999976,
                "subLong": 81.9248884147999945,
                "elevation": 659705.8125,
                "altitudeAngle": -0.0129076793204999998,
                "azimuthAngle": 29.2152940614999999,
                "currentDateTime": "2019/4/30 07:55:27",
                "rollAngle": "23:59:35.64",
                "correct": true
            },
            . . .
            ]
    }
]
```


Päringuga kaasneb palju infot, mida rakendus ei kasuta. Rakendus kasutab vastusest järgmist infot:

- `riseTime` – Satelliidi horisondilt tõusmise aeg. Kirjeldab satelliidi ülelennu sessiooni algust.
- `rollAngles` – List satellidi positsiooni infost kokku lepitud vältuste järel. Antud töö valmimisel tagastatakse satelliidi positsiooni andmeid iga 30 sekundi järel.
    - `altitudeAngle` – Kirjeldab satelliidi kõrgusnurka seadme suhtes
    - `azimuthAngle` – Kirjeldab satelliidi asimuuti seadme suhtes.

Päringu ebaõnnestumisel kuvatakse kasutajale teade päringu ebaõnnestumisest ja antakse juhiseid järgmise päringu parandamiseks.

3.3.2 Ülelennu andmed

Satelliidi orbitaalse liikumise kirjeldamise standardiks on TLE (*two-line element*). TLE koosneb satelliidi nimest ja kahest eraldi reast numbritest. Esimene rida sisaldab enamasti kontroll-andmeid ning ajalisi parameetreid ja teine rida sisaldab satelliidi liikumist kirjeldavaid parameetreid [5]. Näide ühe satelliidi TLE'st võetud tle.info [4] kodulehelt:

```
RADIO ROSTO
1 23439U 94085A   19136.78932077 -.00000014 +00000-0 +80611-3 0 9995
2 23439 064.8136 169.3335 0146644 119.6429 241.9190 11.27567904004189
```

Antud andmetest on võimalik ennustada satelliidi asukohta igal antud ajahetkel. Üheks võimaluseks arvutada satelliitide asukoht oleks teha seda lokaalselt. Lokaalselt arvutamine aga nõuaks seda, et kohalik TLE fail sisaldaks alati kõige viimasemaid andmeid. Lokaalselt arvutamise eeliseks oleks väiksem interneti resursside kasutus ja võimalus kasutada rakendust ka ilma võrguühenduseta. Antud rakendus aga pärib vajalikud arvutusted üle võrgu. Vajalikud arvutused tehakse TTÜ tudengisatelliidi projekti maajaama tiimi poolt hallatud serveris ja edastatakse avaliku API kaudu.

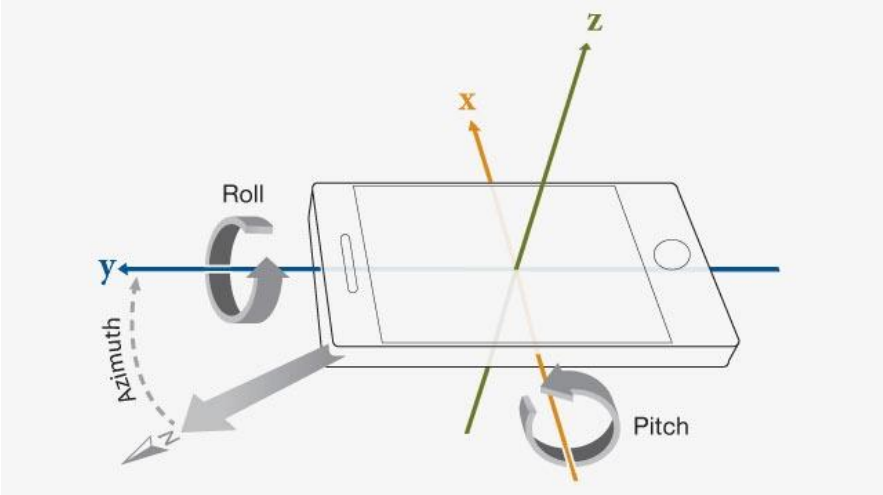
Päritud info on JSON formaadis (näide vastusest peatükis 3.3.1) ja sisaldab töö valmimise hetkel autori jaoks järgmist vajalikku infot: "risetime" ja "rollAngles".

"riseTime" kirjeldab ülelennu sessiooni algust, ehk seda millal jälgitav satelliit horisondi tagant vaatajale paistma hakkab. Antud väärtust kasutatakse taimeri komponendile alarmi seadmisel – infot hakatakse kasutajale kuvama alates sellest hetkest.

"rollAngles" sisaldab listi satelliidi andmetest iga kokku lepitud aja tagant. Antud rakenduse puhul kuvatakse uut infot iga 30 sekundi järel. Autori jaoks on antud andmete puhul olulised väärtused "altitudeAngle" ja "azimuthAngle", mis kirjeldavad satelliidi koordinaate antud ajahetkel. Ülelennu andmeid hoiustatakse parsimise järel nendele ettenähtud listides.

3.3.3 Sensorite kasutus

Antud rakenduse jaoks on vaja seadme tagakülje kaamera asimuuti, kõrgusnurka maapinna suhtes ja pöördenurka. Seadme asendi lugemiseks tuleb lugeda seadmesse sisse ehitatud sensorite infot. Selleks on Androidi operatsiooni süsteemil sensorite halduri API, mille kaudu on võimalik lugeda kõikide kättesaadavate sensorite infot. Vajaliku info saab lugeda TYPE_ROTATION_VECTOR sensorist ehk pöördevektori sensorist liikumissensorite komplektist [12]. Pöördevektor kirjeldab seadme asendit maa suhtes.



Pilt 4. Nutitelefoni asendit kirjeldavad parameetrid [13]

Antud vektor aga tagastab seadme asimuudiks pildil 2 näidatud suuna seadme suhtes. Kuna rakenduse jaoks on vaja asimuudiks seadme tagakülje kaamera suunda, tuleb antud vektor teisendada.

Teisendades teljed sobivale kujule saab kasutada seadme asendi parameetreid graafiliste elementide positsiooni arvutamiseks kasutajaliidesel. Selleks tuleb Android rakenduses defineerida *onSensorChanged()* funktsioon, mis kirjeldab tegevusi uute andmete tuvastamise järel. Antud rakendus arvutab kõikide uute vektorandmete põhjal graafiliste elementide uue positsiooni kasutades DifferenceCalculator komponendi funktsiooni *getDifferenceMatrix()* (funktsiooni kirjeldus on punktis 3.3.4.2).

Sensori andmete küsimise perioodiks on standard vahemik *SENSOR_DELAY_NORMAL*, mis ametliku dokumentatsiooni järgi on kõige optimaalsem kiirus graafiliste elementide uuendamiseks [7]. Antud periood on dünaamiline ja muutub vastavalt seadme võimekusele. Kuna arvutused on ressursimahukad, siis võib kaasneb tihedama andmete küsimisega kaadrite kadumisi ja soovimatuid viivitusi.

3.3.4 Positsiooni arvutamine

Satelliidi punkti positsiooni kujutamiseks vaja minevate arvutuste tegemiseks on loodud komputatsioonide objekt, mis projektis kannab klassi nime DifferenceCalculator. Klassi konstruktoris antakse rakendusele ette jälgitava satelliidi praegused koordinaadid. Kui on aeg uuendada jälgitava punkti koordinaate loob PositionUpdateAlarm objektis jooksev taimer peavaate klassile uute parameetritega kalkulaatori objekti.

Peamiseks kalkulaatori liideseks on funktsioon getDifferenceMatrix. Funktsioon getDifferenceMatrix kutsutakse välja kõikide uute pöördesensorite andmete tuvastamise järel. Funktsioon arvutab algselt otsitava punkti muudu keskpunktist nii x- kui ka y-koordinaadi suhtes arvestades seadme kaamera vaatenurka ja ekraani suurust. Kuna algsed arvutused on tehtud eeldades, et seadet hoitakse -90 kraadi pöördenurga all, siis järgnevalt muudetakse nende väärtusi funktsiooni rotationTransformation abil. Antud funktsioon teisendab jälgitava punkti asukohta x- ja y-teljel sõltuvalt ekraani pöördenurgast ning lisab abistava noole elemendi pöördenurga ja jälgitava punkti kauguse keskpunktist. Saadud andmete põhjal kuvatakse kasutajaliidesel uus punkti asukoht ning muutused abistaval noolel.

3.3.4.1 Konstruktor

Meetod: `DifferenceCalculator(float azimuth, float elevation)`

Eeltingimused: Ülelennu algusest teavitav alarmi komponent on oodanud 30 sekundit ja API käest saadud JSON failis leidub veel läbimata satelliidi punkte.

Kirjeldus: Luuakse uus komputatsiooni objekt. Määratakse jälgitava satelliidi viimase seisu asimuut, kõrgus ja teavitatakse peavaadet vajadusest uuendada kasutajaliidese andmeid.

3.3.4.2 Funktsioon getDifferenceMatrix

Meetod: `int[] getDifferenceMatrix(float currentAzimuth, float currentElevation, float currentRotation)`

Eeltingimused: SatelliteTrackerApplication klassis kutsutakse välja onSensorChanged() funktsioon.

Kirjeldus: Funktsioon arvutab nelja liikmelise listi võttes parameetriteks sensorandmed:

currentAzimuth – kaamera suuna asimuut

currentElevation – kaamera suuna kõrgusnurk maapinna suhtes

currentRotation – kaamera pööre

Tagastatav objekt: Nelja numbrilist väärtust sisaldav list mis kirjeldab:

1) satelliidi asukohta kujutava punkti asukoht ekraani x-teljel

2) satelliidi asukohta kujutava punkti asukoht ekraani y-teljel

3) suunava noole pöördenurk

4) satelliidi asukohta kujutava punkti kaugus pikslites ekraani keskpunkti suhtes

3.3.4.3 Funktsioon getHorizontalPlacement

Meetod: `float getHorizontalPlacement(float currentAzimuth)`

Eeltingimused: DifferenceCalculator klassis kutsutakse välja getDifferenceMatrix funktsioon.

Kirjeldus: Arvutab satelliidi positsiooni muudu pikslites ekraani keskpunkti suhtes y-teljel. Punkt arvutatakse seadme -90° pöördenurga suhtes. *verticalFov* kirjeldab seadme kaamera vaatenurka mis päritakse kaamera API käest *deviceWidth* kirjeldab ekraani laiust ja päritakse ekraani halduri käest.

```
float difference = ((currentAzimuth - targetAzimuth + 180) % 360) - 180;
float u = (difference / 180);
float a = (180f / horizontalFov) * deviceHeight;
float shiftInPixels = u * a;
if (currentAzimuth < 0) {
    return Math.round(shiftInPixels);
} else {
    return - Math.round(shiftInPixels);
}
```

Muutuja *difference* kirjeldab erinevust seadme suuna ja jälgitava punkti vahel kraadides. Kraadide kujutamiseks ekraani suhtes tuleb aga kraadid konverteerida piksliteks. Selleks tuleb võtta arvesse seadme ekraani dimensioone ning kaamera vaatenurka. *shiftInPixels* kujutab endast jälgitava objekti positsiooni erinevust pikslites ekraani keskpunkti suhtes.

Tagastatav objekt: float tüüpi numberväärtus kirjeldamaks satelliidi positsiooni y-telje suhtes.

3.3.4.4 Funktsioon getVerticalPlacement

Meetod: `float getVerticalPlacement(float currentElevation)`

Eeltingimused: DifferenceCalculator klassis kutsutakse välja getDifferenceMatrix funktsioon.

Kirjeldus: Arvutab satelliidi positsiooni muudu pikslites ekraani keskpunkti suhtes x-teljel.

```
float difference = targetElevation - currentElevation;
        float u = (difference/180);
        float a = (180f/verticalFov) * deviceWidth;
        float shiftInPixels = u * a;
        return Math.round(shiftInPixels);
```

Tagastatav objekt: float tüüpi numberväärtus kirjeldamaks satelliidi positsiooni x-telje suhtes.

3.3.4.5 Funktsioon rotationTransformation

Meetod: `int[] rotationTransformation(int[] values, float rotation)`

Eeltingimused: DifferenceCalculator klassis kutsutakse välja getDifferenceMatrix funktsioon.

Kirjeldus: Funktsioon tegeleb seadme pöördenurgast sõltuvate arvutustega:

- Muudab varasemalt arvutatud punkti asukohta x-telje ja y-telje suhtes. Otsitava punkti arvutamiseks x-teljel ning y-teljel kasutan pöördmaatriksist tuletatud valemeid:

$$X = \cos(d) * x + \sin(d) * y$$

$$Y = -\sin(d) * x + \cos(d) * y$$

Kus *x* on jälgitava punkti asukoha erinevus ekraani keskpunkti suhtes x-teljel ning *y* on jälgitava punkti asukoha erinevus y-teljel -90° pöördenurga puhul. *d* on pöördenurk -90° suhtes. Antud valem töötab vaid siis, kui *x* ja *y* on eelnevalt pikslitesse teisendatud.

Teostus koodi kujul:

```
int[] transformed = new int[4];
float rotationDifference;
if (rotation < -90f) {
     rotationDifference = 450f + rotation;
} else {
     rotationDifference = rotation +90f;
}
double difference = Math.toRadians(rotationDifference);
double x = Math.cos(difference) * values[0] +
Math.sin(difference) * values[1];
double y = -Math.sin(difference) * values[0] +
Math.cos(difference) * values[1];
transformed[0] = centerCoordinates[0] + (int) x;
transformed[1] = centerCoordinates[1] + (int) y;
```

- Arvutab punkti suunda kirjeldava nurga keskpunkti suhtes. Kasutades eelnevalt arvutatud X ja Y väärtust, saab arvutada jälgitava punkti poole suunava noole pöördenurga. Selleks kasutan valemit:

$$\alpha = \tan^{-1}(\frac{X}{-Y})$$

- Lisab kaasa satelliiti kujutava punkti kauguse keskpunktist pikslites.

Tagastatav objekt: int[] tüüpi list numbrilistest väärtustest kirjeldamaks satelliiti kujutava punkti asukohta ekraanil, abistava noole suunda ja noole tausta värvi tugevust.

3.4 Kasutajaliides

Kasutajaliidesel on kaks peamist vaadet: peamenüü ning satelliidi jälgimise vaade.

3.4.1 Peamenüü

Peamenüü eesmärgiks on anda kasutajale võimalus kirjutada jälgitava satelliidi nimi ja selle kaudu esitada päring serverile satelliidi asukohaandmete saamiseks. Vastuse ootamise ajaks deaktiveeritakse päringu tegemise nupp. Vastavalt sellele, kas serveripoolne vastus on positiivne või negatiivne kantakse kasutaja kas satelliidi jälgimise vaatesse või tagasi peamenüüsse. Negatiivse vastuse puhul kuvatakse kasutajale märkus võimalikest päringu ebaõnnestumise põhjustest.

Näide peamenüü vaatest ja veateatest on leitavad lisades 1 ja 2.

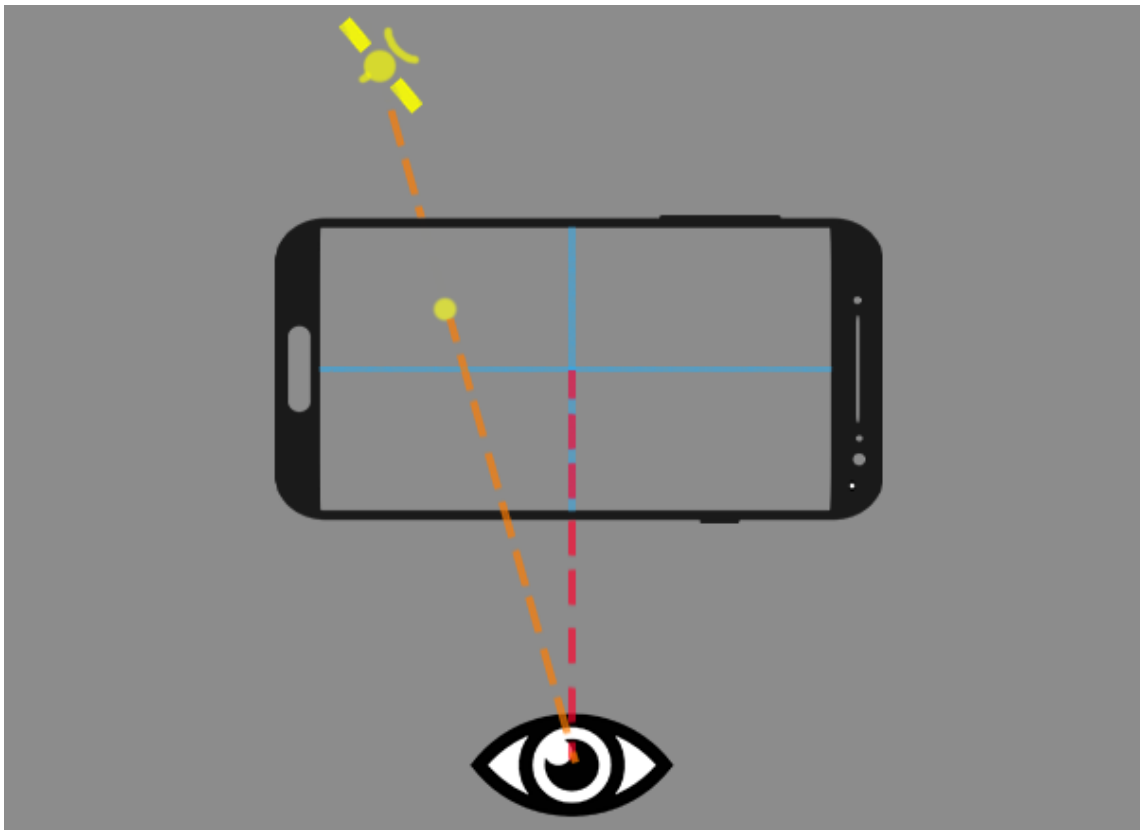
3.4.2 Satelliidi jälgimise vaade

Antud vaates on neli peamist elementi: Satelliiti kujutav punkt, satelliidi asukoha korrigeerimist abistav nool, sensorite infot kirjeldav nurk ja vaadeldava satelliidi kirjalikku informatsiooni kirjeldav nurk. Esmalt aga käivitatakse ekraanil alla loendur mille lõppedes peaks jälgitav satelliit horisondi tagant otsesesse vaatevälja jõudma. Näide loenduri vaatest on leitav lisas 3. Satelliidi jälgimise vaateks on ekraani taustaks määratud seadme tagakülje kaamera. Kasutajal on võimalik lõpetada sessioon ja naaseda peamenüüsse kasutades nuppu ’CLOSE SESSION’. Sessiooni lõppedes, ehk jälgitava satelliidi horisondi taha kadumisel, sulgetakse vaade ja viiakse kasutaja tagasi peamenüüsse.

Pilt 5. Näide rakenduse peavaatest

3.4.2.1 Satelliiti kujutav punkt

Satelliiti kujutav punkt kirjeldab satelliidi hetke asukohta seadme suhtes. Juhul kui satelliit ei peaks seadme kaamera vaateväljas olema ei kuvata ka punkti ekraanil. Punkti arvutamisel kasutatakse kaamera enda vaatevälja nurkade infot ning seega kujutab punkt satelliidi asukohta vaatamata seadme asendile.



Pilt 6. Satelliidi kujutise abijoonis

3.4.2.2 Abistav nool

Abistava noole eesmärk on aidata seadet korrigeerida. Nool näitab kuhu poole tuleks seadet keerata, et satelliiti kujutav punkt ekraani keskele saada. Kuna nool on kogu aeg suunavas seisundis, siis on raske öelda millal punkt täpselt ekraani keskel on. Punkti täpselt ekraani keskele korrigeerimine on mobiilseadmete sensorite ebatäpsuse tõttu praktiliselt võimatu ning selle jaoks on suunava noole elemendi taustaks lisatud roheline toon. Suunava elemendi taust muudab oma läbipaistvust vastavalt sellele kui lähedal on satelliidi punkt ekraani keskmele.

Pilt 7. Noole taust kui satelliit pole ekraani keskel

Pilt 8. Noole taust kui satelliit on ekraani keskmele lähedal

Pilt 9. Noole taust kui satelliit on ekraani keskel

3.4.2.3 Lisainfo

Kasutajale kuvatakse ka info järgmiste andmete kohta:

- *Observed Sat.* – Jälgitava satelliidi nimi.
- *Targ. azimuth* - Jälgitava satelliidi asimuut.
- *Targ. Elevation* – Jälgitava satelliidi kõrgusnurk maapinna suhtes.
- *Next target in* – Alla loendur järgmise jälgitava punktini.
- *az* – Seadme tagakülje kaamera vaatesuuna asimuut
- *el* – Seadme tagakülje kaamera vaatesuuna kõrgusnurk maapinna suhtes.
- *rot* – Seadme kaldenurk portree vaate suhtes.
- *Latitude* – Seadme hetke laiuskraad
- *Longitude* – Seadme hetke pikkuskraad

4 Validatsioon

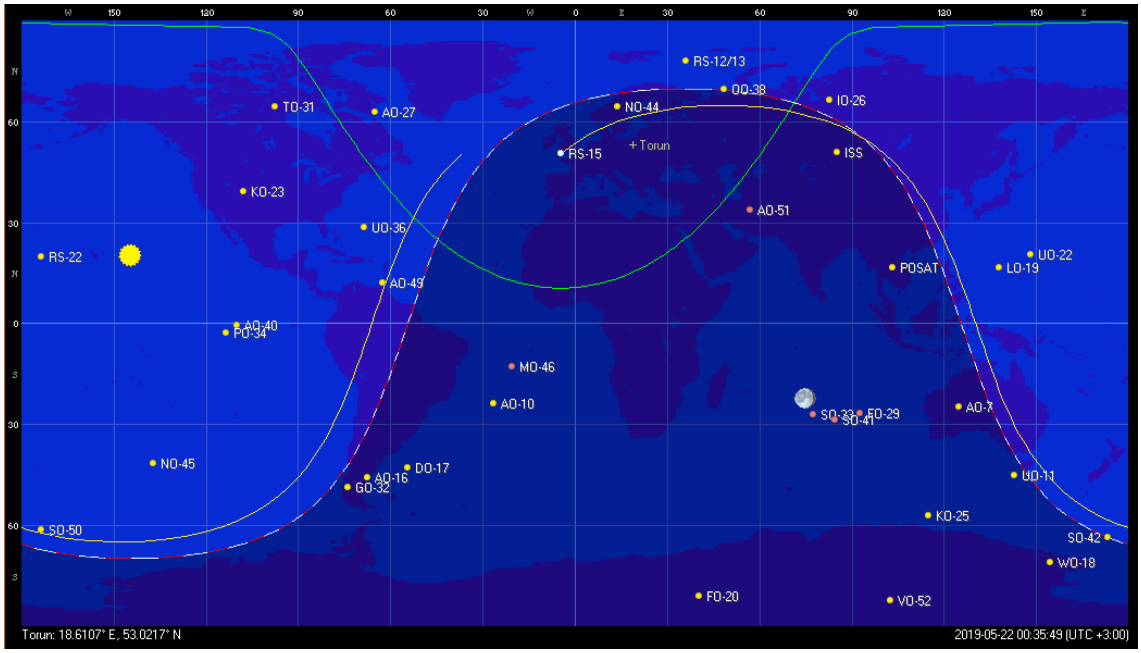
Antud peatükis käivitab autor rakenduse ja kontrollib kas rakenduses satelliiti kujutava punkti poole suunates saab satelliidilt signaali kätte. Selle jaoks suunatakse *yagi*-antenn rakenduse poolt ette näidatud punkti suunas. Satelliidilt tuleva signaali jälgimiseks kasutatakse programmi SDR Console v3 [11]. Raadio vastuvõtjaks on kasutatud DX Patrol vastuvõtjat [16].

4.1 Sobiliku satelliidi leidmine

Enne satelliidilt signaali püüdmist tuleb leida jälgimiseks sobiv satelliit. Tingimused sobiva satelliidi valikul:

- Satelliit pole veel jõudnud vaatevälja, kuid tõuseb horisondilt mõistliku aja järel.
- Satelliit edastab signaali sobilikul sagedusel mida vastuvõtja on võimeline vastu võtma. Antud vastuvõtja puhul 100 KHz kuni 2 GHz

Satelliitide liikumise jälgimiseks kasutati rakendust Orbitron, mis arvutab satelliitide positsiooni koos nende ulatusega tuginedes TLE failile [14].



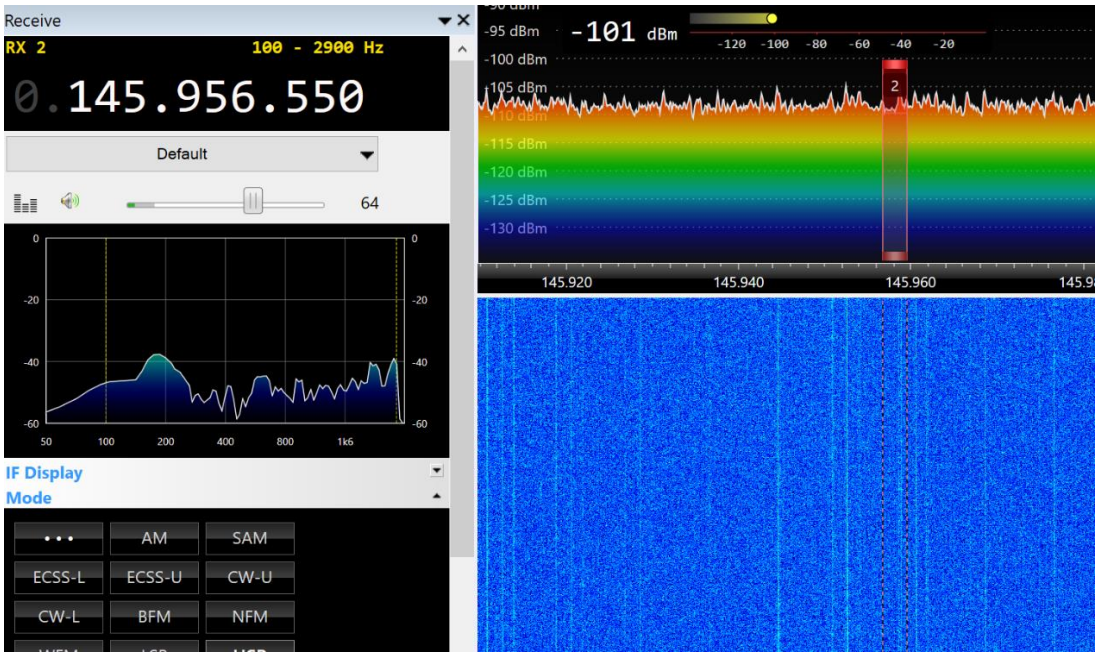
Pilt 10 näide Orbitroni tarkvarast

Satelliidi poolt edastatava sagedus ei ole aga kõikide satelliitide puhul avalikult kätte saadav. Selle tõttu tuleb iga satelliidi jaoks eraldi juurde otsida informatsiooni nende poolt edastatud signaali ja sageduste kohta. Lisaks sellele tuleb kontrollida kas satelliit on veel töökorras. Olukorra teeb keerulisemaks ka see, et satelliidid ei pruugi edastada infot mida on testimise eesmärgil kasutatud rakenduse konsoolist võimalik jälgida või kuulata.

4.2 Katse üles seadmine

Antud katse puhul prooviti jälgida satelliiti nimega NUSAT-1. Antud satelliidi poolt edastatud signaali laiusriba on 30 kHZ ja kuulamiseks on sagedus 145.965 ~145.935 [15]. Katse lihtsustamise eesmärgil ei tegeleta signaali dekodeerimisega, vaid proovitakse antud sagedusel saada õigel ajal kätte satelliidi signaal.

Signaali jälgimiseks käivitatakse programm SDR Console v3 ja ühendatakse vastuvõtja. Vastuvõtja VHF sisendi külge kruvitakse *yagi*-antenn signaali püüdmiseks. Järgnevalt konfigureeritakse vaateaken vastava satelliidi jälgimiseks.

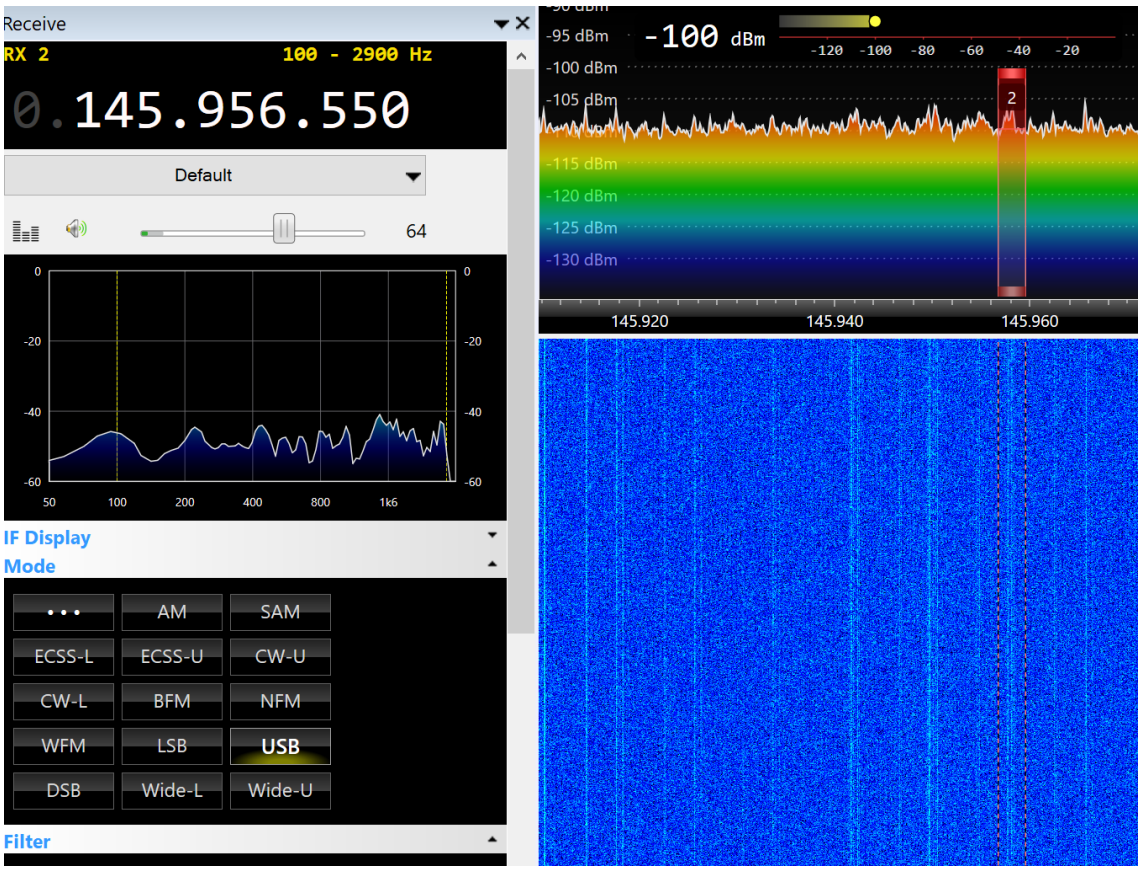


Pilt 10. SDR Console v3 seadistused vaatluseks

Akna vasakul pool on näha jälgitav sagedus, kuulamise tugevus, jälgitava vahemikku spektrogramm ning signaali vastuvõtmise tüüp. Antud testi jaoks on vajalik ainult sageduse muutmine, kuna eesmärgiks on näha spektri ajaloost signaali intensiivsust. Paremal pool on näha vastuvõtja sagedusriba ning spektri ajalugu.

Vaatluse eel käivitatakse autori mobiilrakendus ning luuakse päring NUSAT 1 satelliidi jälgimiseks.

4.3 Katse tulemused



Pilt 11. Katse tulemused rakenduses SDR Console v3

Katse tulemusena ei saa kindlalt öelda kas soovitud eesmärk saavutati. Spektri ajaloost ei ole võimalik lugeda märkimisväärset erinevust jälgitaval sagedusel. Signaal võis siiski eksisteerida, kuid oli liiga nõrk, et spektri ajaloost visuaalsele kontrastile tuginedes hinnata. Jälgitava sageduse ja taustamüra vahel esineb pisut suurem kontrast kui vaatluse ülesseadmisel nähtaval spektri ajalool, kuid see võib olla tingitud katse keskkonna väiksemast taustamürast.

Autori loodud mobiilrakenduses satelliiti kujutav täpp kõikus pidevalt 6°, kuid tagas siiski punkti mis visuaalselt hoidis sarnast positsiooni nutiseadme asendi suhtes.

4.4 Järeldus

Katse tulemustest ei saa järeldada kas rakendus töötab plaanipäraselt või mitte. Katse plaanipäraste tulemuste erinevuse põhjused on teadmata. Võimalikud põhjused soovimatute tulemuste jaoks võivad olla:

- Satelliidi signaali tugevus oli liiga nõrk
- Signaali vastuvõtmiseks tehtud kalibratsioon tarkvaras oli vigane
- Antenni ebasoodne paigutus. Eesolevad füüsilised kehad võisid häirida signaali kandumist.
- Satelliidilt ei esitatudki antud hetkel signaali
- Rakenduse poolt valesti ette antud suund
- Taustamüra ja muud segavad signaalid keskkonnas
- Aegunud TLE informatsioon satelliidi kohta. Arvutused satelliidi positsiooni jaoks tuginevad aegunud andmetele.

Järgnevate katsete puhul tuleks teha parem valik jälgitava satelliidi osas ning tutvuda hoolikalt selle signaali mustritega, vajalike vastuvõtu konfiguratsioonidega ning edastusaegadega. Katset peaks kordama kui ollakse veendutud, et osatakse mingi kindla satelliidi infovoogu jälgida. Seejärel saaks hinnata autori loodud rakenduse poolt antud suunamisjuhiste täpsust.

5 Kokkuvõte

Käesoleva töö raames on arendatud Androidi platvormile satelliitide jälgimise rakendus. Rakendus on ennekõike mõeldud inimestele kelle sooviks on jälgida maakeraga asünkroonselt liikuvaid satelliite. Aplikatsioon võimaldab kasutajal pärida satelliidi asukohta sisestades selle nime. Rakendus kasutab satelliidi liikumise info saamiseks veebipäringut. Jälgitavate satelliitide kogus on piiratud vaid serveri poolse TLE faili poolt.

Töös realiseeriti graafiline kasutajaliides satelliidi jälgimiseks, mis kujutab satelliidi asukohta dünaamiliselt Android seadme ekraanil ja näitab kuhu poole on seadet vaja keerata, et satelliit ekraani keskele saada. Graafilise kasutajaliidese komponentide paigutamisel kasutatakse autori enda algoritmi trigonomeetriliste arvutuste ja teisenduste jaoks.

Rakenduse testimisel kasutati rakendust satelliidi NUSAT-1 jälgimiseks. Katse raames prooviti püüda satelliidi signaali suunates *yagi*-antenni rakenduse poolt juhendatud suunas. Katse ei taganud soovitud tulemusi kuid ei pruugi olla tingitud rakenduse funktsionaalsusest.

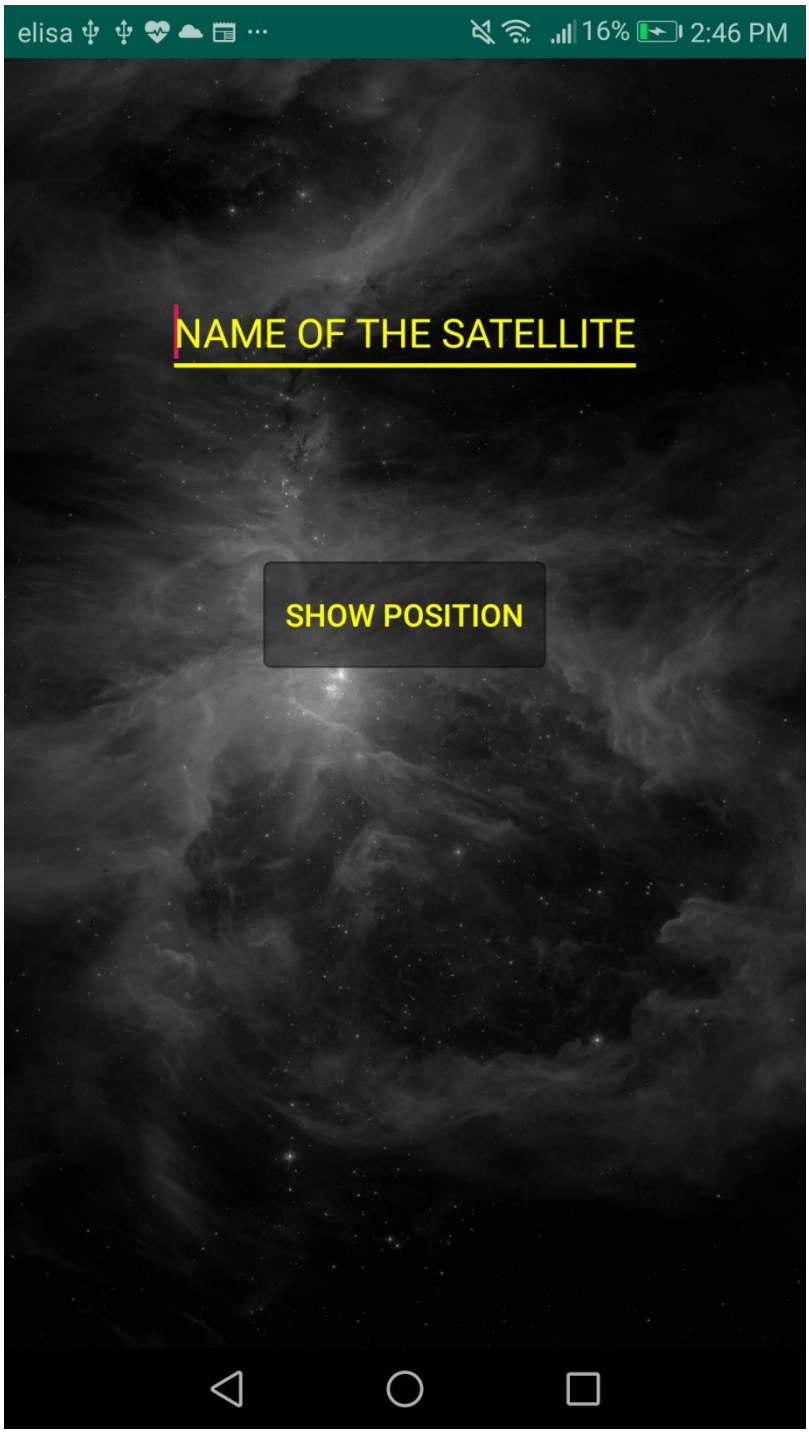
Töö raames leiti, et testitud Android seadmete kompassi andmed on väga ebastabiilsed täppisarvutuste tegemiseks. Seadme asimuudi lugemisel peab arvestama pideva 6° vaheldumisega. Asimuudi kõikumine aga ei mõjuta rakenduse funktsionaalsust suuremal määral ning täpsus jääb ikka nõuetes ette antud vahemikku.

Rakendust saaks edasi arendada lisades funktsionaalsust, mis suudaks arvutada satelliidi trajektoori lokaalselt seadmes ja eemaldades vajaduse suhelda serveriga. Lisaks saaks arendada matemaatilist funktsionaalsust hetkel toimuvate ülelendude arvutamiseks. Hetkel algab ülelennu info hetkest, mil satelliit ilmub horisondi tagant nähtavale ning server ei oska tagastada satelliidi positsiooni kui see on juba nähtaval.

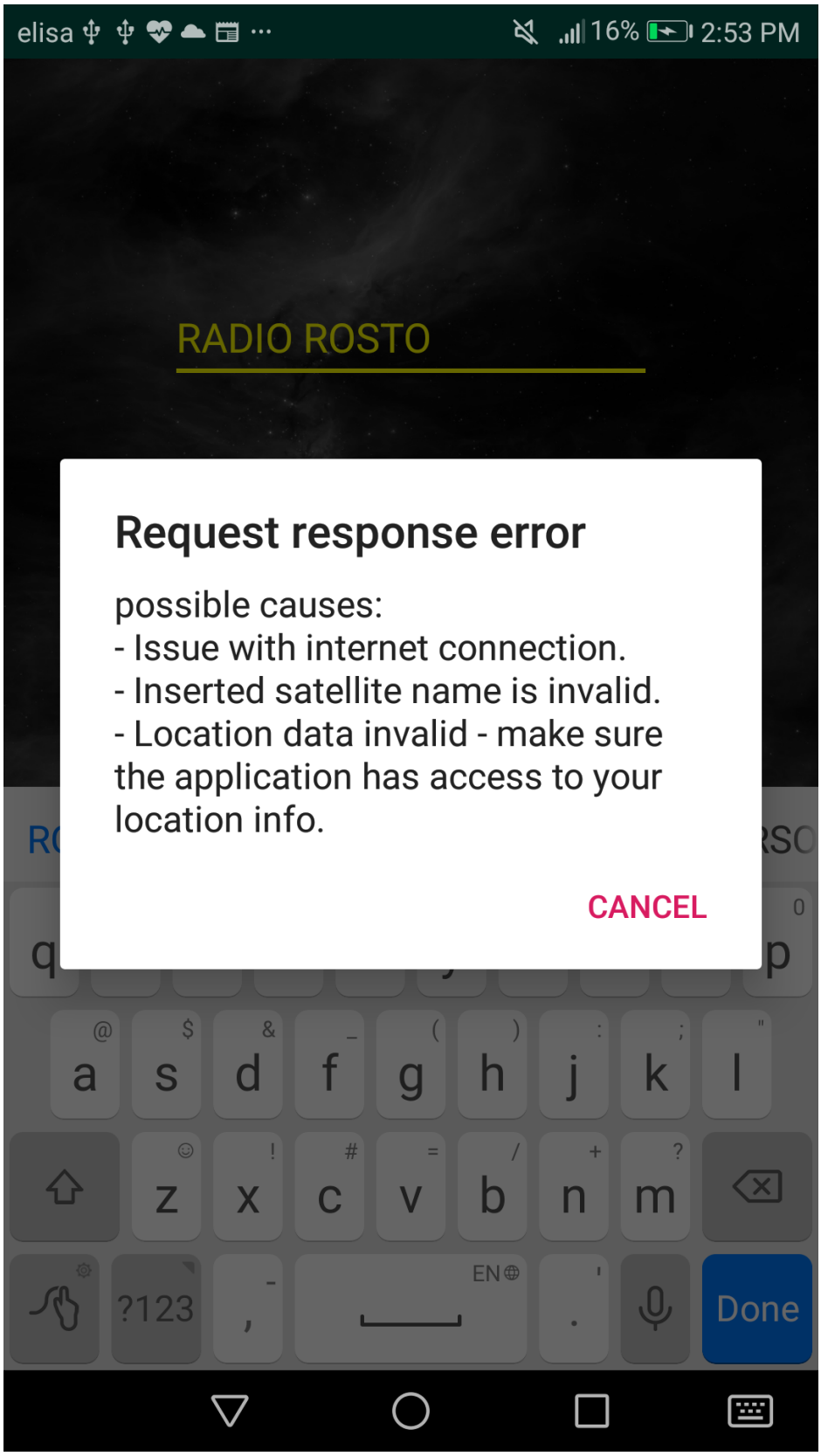
Kasutatud kirjandus

1. Android developer portal: Android Studio, https://developer.android.com/ (22.05.2019)
2. Android developer portal: Distribution dashboard, https://developer.android.com/about/dashboards (22.05.2019)
3. Statcounter: Mobile Operating System Market Share Worldwide – April 2019: http://gs.statcounter.com/os-market-share/mobile/worldwide (22.05.2019)
4. TLE.info: TLE Element Groups for Download, https://tle.info/tlegroups (22.05.2019)
5. TLE standardi kirjeldus: https://www.celestrak.com/NORAD/documentation/tle-fmt.php (22.05.2019)
6. Android developer portal: Volley overview, https://developer.android.com/training/volley (22.05.2019)
7. Android developer portal: SensorManager, https://developer.android.com/reference/android/hardware/SensorManager (22.05.2019)
8. Esys: SatFinder, http://www.esys.com.pl/satfinder/ (22.05.2019)
9. Google Play: SatFinder, https://play.google.com/store/apps/details?id=sil.satorbit (22.05.2019)
10. Google Play: Satellite Finder PRO, https://play.google.com/store/apps/details?id=satellite.finder.comptech (22.05.2019)
11. SDR-Radio: Version3, https://www.sdr-radio.com/Software/Version3 (22.05.2019)
12. Android developer portal: Motion sensors, https://developer.android.com/guide/topics/sensors/sensors_motion (22.05.2019)
13. MathWorks: iPhone and iPad Sensor Support from MATLAB, https://www.mathworks.com/hardware-support/iphone-sensor.html (22.05.2019)
14. Orbitron – Satellite Tracking System: Homepage, http://www.stoff.pl/ (22.05.2019)
15. AMSAT-UK: NUSAT-1 SSB/CW Transponder Satellite, https://amsat-uk.org/tag/nusat-1/ (22.05.2019)
16. Vibroplex: DXPatro, http://www.vibroplex.com/contents/en-us/d9154.html (22.05.2019)

Lisa 1 – Peamenüü vaade



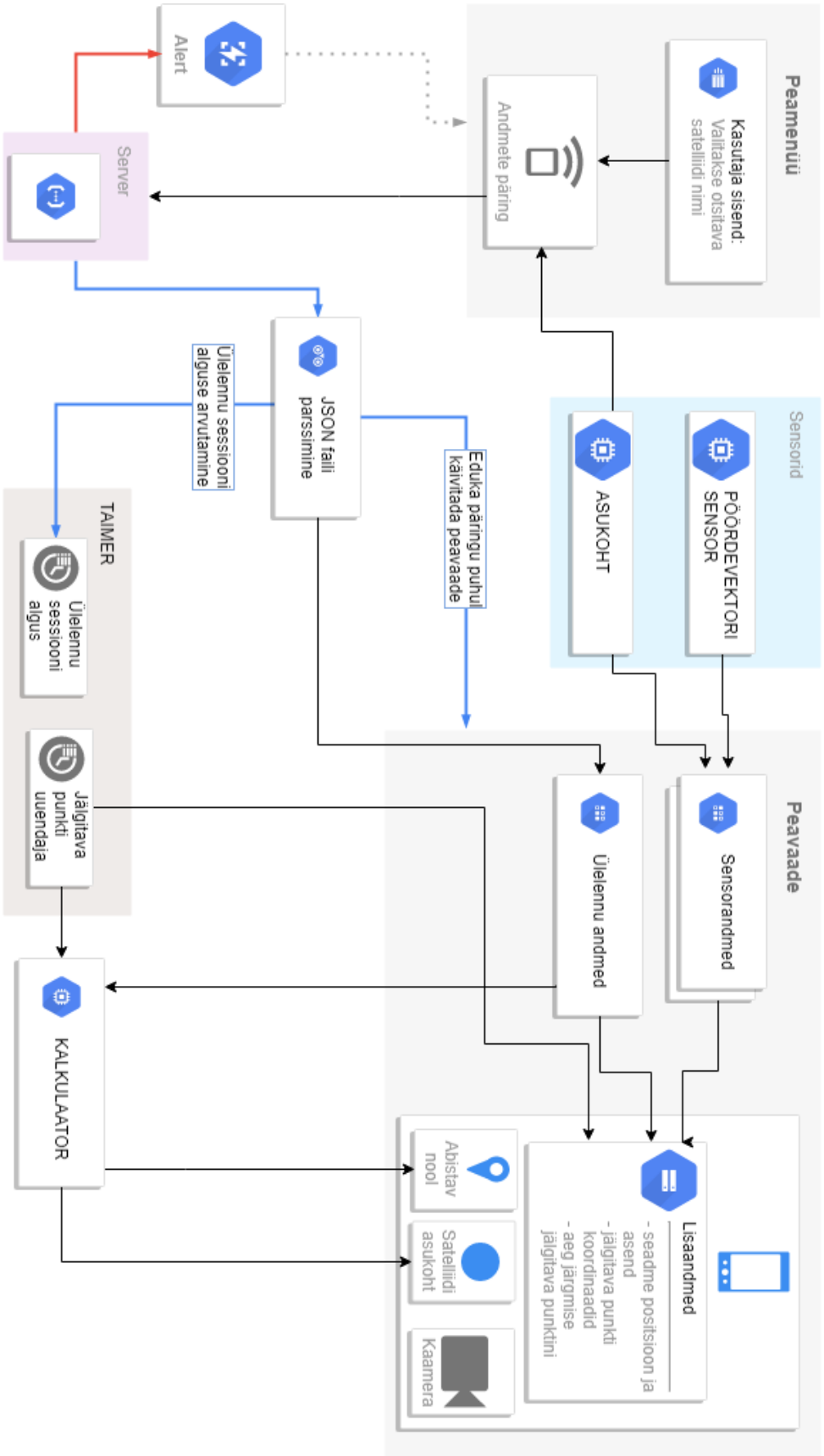
Lisa 2 – Peamenüü veateade



Lisa 3 – Satelliidi jälgimise vaate alla loendur

Lisa 4 – Rakenduse peavaade

Lisa 5 – Arhitektuuri diagramm



Lisa 6 – Rakenduse avalik repositoorium

https://github.com/Giergione/SatelliteTracker