TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Thomas Johann Seebecki elektroonikainstituut

Sander Mikkov 142942IALB

Lennukite ADS-B signaalide vastuvõtt ja dekodeerimine tarkvaralise raadioga

Bakalaureusetöö

Juhendajad: Ivo Müürsepp
dotsent
Julia Berdnikova
teadur

Tallinn 2017

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: Sander Mikkov

22.05.2017

Annotatsioon

Käesoleva töö eesmärgiks on uurida, kuidas on võimalik konstrueerida lennukite ADS-B (*Automatic Dependent Surveillance-Broadcast*) signaalide vastuvõtjat ja dekodeerijat, kasutades tarkvaralist raadiot ehk SDRi (*Software Defined Radio*).

Töö teises peatükis antakse lähemalt ülevaade ADS-B süsteemist, kirjeldatakse, kuidas standard töötab ning millised on tema komponendid. Lisaks võrreldakse ADS-B süsteemi hetkel samuti kasutusel oleva primaar- ja sekundaarradari süsteemiga ning tuuakse välja ADS-B süsteemi eelised teiste leviedastusstandardite ees.

Kolmandas peatükis vaadeldakse detailselt ADS-B paketi struktuuri ning kirjeldatakse lühidalt ka signaali modulatsiooni. Järgmisena on kasutusele võetud tarkvaraline raadio ning seadistatud ta selliselt, et ta võtaks vastu meid huvitavad ADS-B signaalid. Seejärel on informatsiooni dekodeerimiseks koostatud programm C keeles ning kokkuvõttena esitatud hinnang ja järeldused töö tulemustele.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 29 leheküljel, 5 peatükki, 22 joonist, 3 tabelit.

Abstract

Capturing and Decoding ADS-B Messages Using Software-Defined Radio

The purpose of this thesis is to examine the possibility of constructing a receiver designed to capture and decode ADS-B messages broadcasted by aircrafts using software defined radio.

The second chapter gives an overview of the ADS-B system, describes how the standard works and which components ADS-B consists of. Additionally, the ADS-B is also compared to the primary and secondary surveillance radar system and the benefits of ADS-B are listed as well.

The next chapter covers the structure of an ADS-B packet in great detail, specifies what kind of information can be found inside said packet and briefly describes the modulation and preamble of an ADS-B packet.

The fourth chapter covers the basic principle of how Software-Defined Radio works and is the beginning of capturing ADS-B messages broadcasted by aircrafts in sufficient proximity. GNU Radio is the software used to process and save ADS-B messages. The author presents several flowcharts, which are designed to meet the specifications of the hardware and the frequency of the transmitted signals to properly capture and process ADS-B signals. A demonstration of how to decode an ADS-B packet manually is presented in the following chapter. Finally, a program written in C language decodes packets previously saved to a text file.

A summary of the work constructed and an assessment to the security of using the ADS-B standard is presented in the conclusion. Taking into consideration the effort required to capture ADS-B messages, and the fact that the messages are not encrypted nor authenticated, the author is of belief that the standard can be exploited.

The thesis is in Estonian and contains 29 pages of text, 5 chapters, 22 figures, 3 tables.

Lühendite ja mõistete sõnastik

SDR	*Software Defined Radio.* Tarkvaraline raadio
ADS-B	*Automatic Dependent Surveillance-Broadcast*
MTOW	*Maximum Takeoff Weight.* Maksimaalne raskus, mille juures on lubatud õhusõidukil õhku tõusta
ICAO	*International Civil Aviation Organization*
ICAO aadress	Õhusõiduki 24-bitine aadress
USRP	*Universal Software Radio Peripheral.* Ettus Research-i poolt arendatavad tarkvaralised raadiod
FAA	*Federal Aviation Administration*
UAT	*Universal Access Transceiver,* leviedastusstandard
USA	*United States of America,* Ameerika Ühendriigid
FIS-B	*Flight Information Services-Broadcast.* ADS-B komponent, mis edastab informatsiooni ilma ning lennukeeldude kohta
GPS	*Global Positioning System.* Satelliitnavigatsioonisüsteem
Mode-S Extended Squitter	Mode-S transponderi poolt edastatavad lühiajalised impulsid
CRC	*Cyclic Redundancy Check*. Veaavastuseks mõeldud kontrollsumma

Sisukord

1 Sissejuhatus 9
2 ADS-B 10
2.1 ADS-B Out ja ADS-B in 10
2.2 ADS-B kasulikkus 11
3 ADS-B pakett 12
3.1 ADS-B modulatsioon ja päis 13
4 Signaalide vastuvõtt tarkvaralise raadioga 16
4.1 GNU Radio 18
4.2 Signaalide demoduleerimine ja dekodeerimine 21
5 ADS-B pakettide dekodeerimine 26
5.1 Horisontaalne kiirus 29
5.2 Kurss 31
5.3 Vertikaalne kiirus 31
5.4 Pakettide dekodeerimine C keskkonnas 32
6 Kokkuvõte 36
Kasutatud kirjandus 38
Lisa 1 – Programmikood 39
Lisa 2 – Rakenduste ekraanitõmmised 51

Jooniste loetelu

Joonis 1. ADS-B paketi osad [6]. 12
Joonis 2. Manchesteri kood. 14
Joonis 3. On-off Keying. 14
Joonis 4. ADS-B päis. 15
Joonis 5. USRP B210. 16
Joonis 6. SDR ülesseadistus. 17
Joonis 7. ADS-B signaalid programmis HDSDR. 18
Joonis 8. ADS-B signaalide faili salvestamine. 19
Joonis 9. Signaalide taasesitamine failist. 20
Joonis 10. Vastu võetud ADS-B signaali spektrogramm. 20
Joonis 11. Vastu võetud ADS-B signaali spekter. 21
Joonis 12. ADS-B demoduleerija ja dekodeerija. 21
Joonis 13. Sisendsignaali ajaline kuju. 22
Joonis 14. Demodulaatori väljund. 22
Joonis 15. ADS-B pakett. 23
Joonis 16. Külge pandud silt lähemalt. 23
Joonis 17. Voodiagrammi väljundfail. 25
Joonis 18. Modifitseeritud voodiagramm. 26
Joonis 19. Voodiagrammi väljundfail. 27
Joonis 20. Programmi algoritm. 32
Joonis 21. Programmi väljundfail. 33
Joonis 22. Programmi väljundfail jätkub. 34

Tabelite loetelu

Tabel 1. ADS-B paketi struktuur [6]..... 12
Tabel 2. Tüübikoodi tähendused [6]. 13
Tabel 3. Andmevälja struktuur [6]..... 29

1 Sissejuhatus

Tehnoloogia kiire kasv toob alatasa kaasa uusi ja paremaid võimalusi kommunikatsiooniks. Ka lennunduse valdkond ei jää muudatustest puutumata. Sammhaaval eemaldutakse klassikalisest asukoha edastuse viisist ning liigutakse kaasaegsema standardi, ehk ADS-B (*Automatic Dependent Surveillance-Broadcast*) poole. ADS-B rakendamist alustati juba eelmisel kümnendil, ning aastaks 2020 plaanib FAA (*Federal Aviation Administration*) viia kogu kommunikatsiooni üle ADS-B peale. ADS-B süsteemil on mitmeid eeliseid võrreldes varasemalt kasutuses olnud primaarse ja sekundaarse radariga. Tänu ADS-B kasutuselevõtule on tekkinud erinevaid teenuseid ja veebisaite, mis võimaldavad reaalajas kaardi peal kuvada õhus olevaid lennukeid, nende lähte- ja sihtkohta ning muud lennuga seonduvat informatsiooni. Sellised veebileheküljed on näiteks flightradar24.com ja radarbox24.com. Sellised teenused eksisteerivad selle tõttu, et entusiastid üle maailma kasutavad enda koostatud ADS-B signaalide vastuvõtjaid ning laevad kätte saadud info eelnevalt mainitud veebilehekülgedele üles. Lisaks on juba alustatud ka droonide varustamist ADS-B tehnoloogiaga, et aidata parandada turvalisust droonide lennutamisel. Hetkel tegeleb ka TTÜ-s startup firma nimega Avionica droonidele teisaldatava transponderi varustamisega, mis muudab droonid õhuliikluses nähtavaks.

Käesoleva bakalaureusetöö eesmärgiks on uurida, kuidas luua ADS-B signaalide vastuvõtja ja dekodeerija kasutades USRP (*Universal Software Radio Peripheral*) B210 tarkvaralist raadiot. Ülesanneteks on USRP sobivate kasutatavate parameetrite välja selgitamine, realiseerida ADS-B signaalide vastuvõtja ja dekodeerija, kasutades selleks GNU Radio tarkvara ja/või teisi keskkondi, ning hinnata ka ADS-B süsteemi turvalisust.

2 ADS-B

ADS-B tähendab *Automatic Dependent Surveillance-Broadcast*-i. Tegemist on lennukite seiresüsteemiga, kus lennuk määrab oma asukohta, kasutades satelliitnavigatsiooni ehk GPSi ning perioodiliselt edastab oma asukoha infot, mida saavad vastu võtta lennujuhtimisüksused. Informatsiooni on võimalik vastu võtta ka teistel lennukitel, omades vastavat ADS-B In süsteemi. Sellisel juhul näevad infot vastu võtvate lennukite piloodid sama radaripilti, mis lennujuhid. ADS-B liike on kaks tüüpi:

1. *Mode-S Extended Squitter*, sagedusel 1090 MHz edastatud sõnumid
2. *UAT (Universal Access Transceiver)*, sagedusel 978 MHz, mis on kasutusel USA-s.

ADS-B on:
Automaatne (*Automatic*) – ta ei nõua piloodipoolset või muud välist sisendit
Sõltuv (*Dependent*) – sõltub õhusõiduki navigatsioonisüsteemi andmetest
Seiresüsteem (*Surveillance*) – ta on seiresüsteem, mis võimaldab lennujuhiüksustel jälgida antud lennukit
Leviedastus (*Broadcast*) – õhusõiduk saadab oma asukohainfot kõikidele teistele lennukitele ja lennujuhiüksustele [1].

2.1 ADS-B Out ja ADS-B in

ADS-B koosneb kahest osast: ADS-B Out, mis edastab õhusõiduki asukoha infot lennujuhiüksustele ja teistele lennukitele, ning ADS-B In, mis võimaldab teistel õhusõidukitel seda infot vastu võtta.

ADS-B Out on õhusõidukite jälgimiseks välja töötatud seiresüsteem, millel on määrav roll lennujuhiüksustele. Ta määrab antud õhusõiduki asukoha, kiiruse ja kõrguse. See informatsioon jõuab lennujuhiüksusteni ja teistele lähedal olevatele lennukitele ning moodustab selle põhjal passiivradaripildi. Euroopas peavad olema aastaks 2020 õhusõidukid, mille MTOW (*Maximum Takeoff Weight*) ületab 12566 naela (5700 kg) või

mille maksimaalne lennukiirus ületab 250 sõlme, olema varustatud ADS-B tehnoloogiaga [2].

ADS-B In võimaldab õhusõidukitel vastu võtta teiste lennukite ja ka maapealsete jaamade ADS-B sõnumeid. Tänu sellele jõuab pilootideni teiste lennukite asukoha- ja ka hetke ilmainfo, mis parandab oluliselt pilootide ülevaadet olukorrast. ADS-B In ei ole kohustuslik, kuid on rangelt soovituslik.

2.2 ADS-B kasulikkus

ADS-B süsteemil on mitu eelist pilootidele ja lennujuhtimisüksustele võrreldes primaar- ja sekundaarradari süsteemiga mis parandavad nii lennuohutust, kui ka -efektiivsust [3] [4]:

1. Lennuliiklus – õhusõidukid, mis on varustatud ADS-B In-ga, saavad vastu võtta teiste ADS-B Out-ga varustatud õhusõidukite sõnumeid. Need sõnumid sisaldavad informatsiooni kõrguse, kursi, kiiruse ja kauguse õhusõidukist kohta.
2. Ilm – õhusõidukid, mis kasutavad UAT ADS-B In tehnoloogiat, on võimelised vastu võtma ilmaraporteid.
3. Lennuinformatsioon – FIS-B (*Flight Information Services-Broadcast*) on võimeline edastama UAT-ga varustatud õhusõidukitele informatsiooni ajutiste keelualade ja piirangute kohta.
4. Kulu – maapealsete ADS-B jaamade paigaldamine ja juhtimine on märkimisväärselt odavam võrreldes primaar- ja sekundaarradari süsteemiga.

3 ADS-B pakett

Tänapäeval on juba enamus õhusõidukeid varustatud ADS-B tehnoloogiaga, mis pidevalt edastavad enda asukohainfot.

ADS-B sõnumeid on kahe erineva pikkusega: pikem, mis on 112 bitti ja lühem, 56 bitti, kestvusega vastavalt 112 või 56 mikrosekundit [5]. Ühe biti kestus on üks mikrosekund ja andmeedastuskiirus on 1Mbit/s.
Järgmisena on võetud vaatluse alla 112 bitine sõnum, mis koosneb järgnevatest osadest, mis on kujutatud Joonisel 1.

Joonis 1. ADS-B paketi osad [6].

Tabel 1 annab ülevaate iga osa tähendusest.

Tabel 1. ADS-B paketi struktuur [6].

Bittide arv	**Bitid**	**Lühend**	**Nimi**
5	1 - 5	DF	*Downlink Format* (Allalingi formaat)
3	6 - 8	CA	Võimekus
24	9 - 32	ICAO	ICAO õhusõiduki aadress
56	33 - 88	Data	Andmed
24	89 - 112	PI	Paarsuskontroll

Ükskõik, milline ADS-B sõnum peab algama *Downlink Format* 17-ga (binaarkoodis 10001). Bitte 6-8 kasutatakse lisaidentifikaatorina, millel on eri tähendused sõltuvalt ADS-B sõnumi tüübist. Bittidesse 9-32 on kodeeritud lennuki ICAO aadress, mida

väljendatakse kuueteistkümnendsüsteemis. Bitid 33-37 näitavad, millist informatsiooni sisaldab andmeväli, bitid 38-88 on andmed ise ning viimased 24 bitti on paarsusbitid [6]. Et teada saada, millist infot pakett sisaldab, tuleb vaadata lähemalt Tüübikoodi (*Type code*), ehk bitte 33-37.
Tabel 2 kirjeldab suhet Tüübikoodi ja paketis sisalduva info vahel.

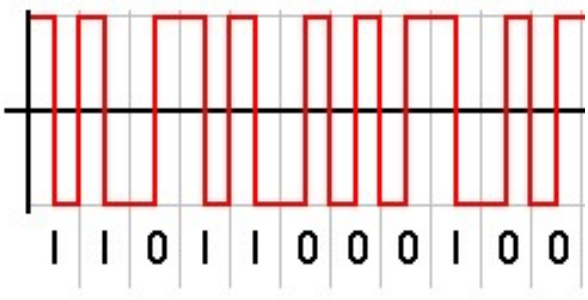
Tabel 2. Tüübikoodi tähendused [6].

TC	**Info**
1 - 4	Õhusõiduki tuvastus
5 - 8	*Surface Position*
9 - 18	Õhupositsioon (koos baromeetri kõrgusega)
19	Lennukiirus
20 - 22	Õhupositsioon (koos GNSS kõrgusega)
23 - 31	Reserveeritud teisteks juhtudeks

3.1 ADS-B modulatsioon ja päis

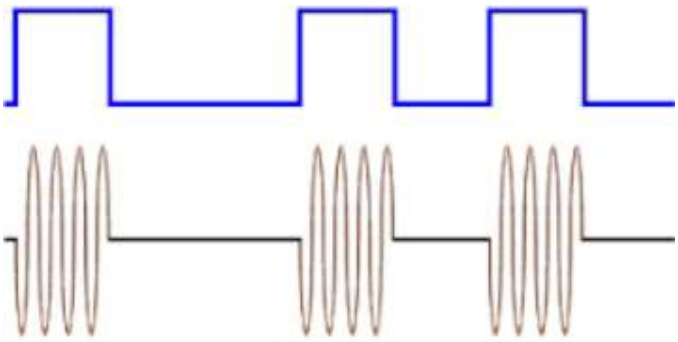
ADS-B kasutab modulatsiooniks *Pulse Position Modulation*it (PPM) ehk impulss-positsioonmodulatsiooni või teise nimetusega Manchesteri koodi [7].

Manchesteri koodi põhimõte seisneb selles, et iga biti kodeerimiseks on signaali väärtus kas kõrge ja seejärel madal, või madal ja seejärel kõrge. Teisisõnu läheb ühe biti kodeerimiseks vaja kaht signaali väärtust. Biti väärtust 1 edastatakse nii, et signaali väärtus läheb poole biti kestel kõrgelt nivoolt madalaks, nulli korral aga madalast kõrgeks [8]. Joonis 2 kujutab Manchesteri koodi.



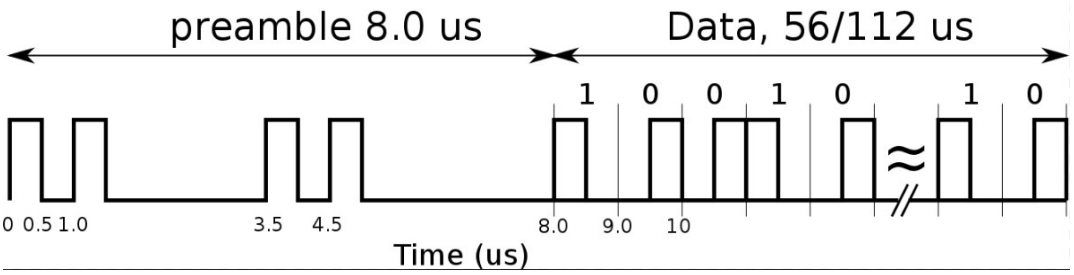
Joonis 2. Manchesteri kood.

Põhiribas kasutatakse bittide kirjeldamiseks Manchesteri koodi. Sagedusel 1090 MHz on tegemist digitaalse amplituudmodulatsiooniga ASK – ehk siis kui biti väärtus on parasjagu kõrge, edastatakse mingi mittenullise amplituudiga võnkumist, kui biti väärtus on 0, siis signaali ei edastata (amplituud 0). Sellist ASK moodust nimetatakse ka OOK (*On-Off Keying*). Joonisel 3 on kujutatud OOK.



Joonis 3. On-off Keying.

Iga ADS-B sõnum algab unikaalse päisega (*preamble*) pikkusega 8 mikrosekundit. Ta koosneb neljast impulsist kestvusega 0,5 mikrosekundit impulsi kohta. Kuna ülejäänud sõnumis selliseid pause ei ole, siis on päis tänu sellele unikaalselt tuvastatav. Joonisel 4 on kujutatud ADS-B päis.



Joonis 4. ADS-B päis.

Kuna iga biti kodeerimiseks läheb vaja kaht väärtust, siis võib ADS-B päise bittide kujul üles märkida järgnevalt: kmkmmmmkmkmmmmmm (1010000101000000).

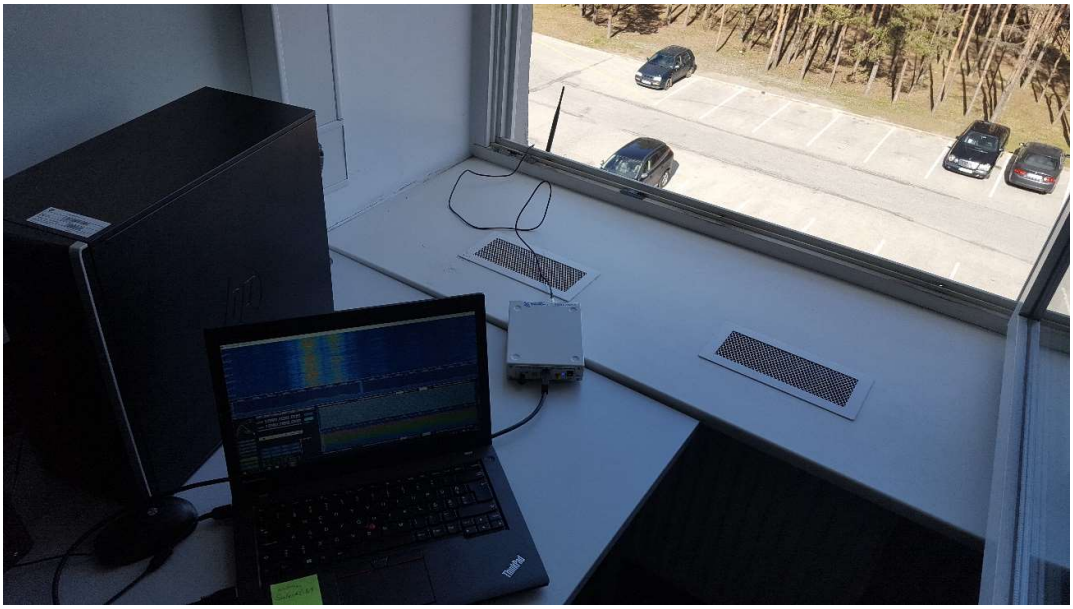
4 Signaalide vastuvõtt tarkvaralise raadioga

Tarkvaraline raadio (SDR) on raadioside süsteem, kus komponendid, mis on tavaliselt teostatud riistvaraliselt, näiteks mikserid, filtrid, võimendid, modulaatorid, demodulaatorid, detektorid, jne, on selle asemel realiseeritud tarkvaraliselt kas personaalarvutis või sardsüsteemis (*embedded system)*. Ideaalses vastuvõtja süsteemis on antenn ühendatud analoog-digitaalmuunduriga. Signaalid suunduvad muundurist seejärel protsessorini, mille tarkvara muudab andmevoo selliseks, mida parasjagu vaja läheb.

ADS-B signaalide vastuvõtmiseks olen käesolevas töös kasutanud USRP B210 SDR tarkvaralist raadiot. Antud seadmel on kaks kanalit, sagedusvahemik on 70 MHz – 6 GHz ning suudab nii vastu võtta kui ka saata. Joonisel 5 on kujutatud eelmainitud seade.

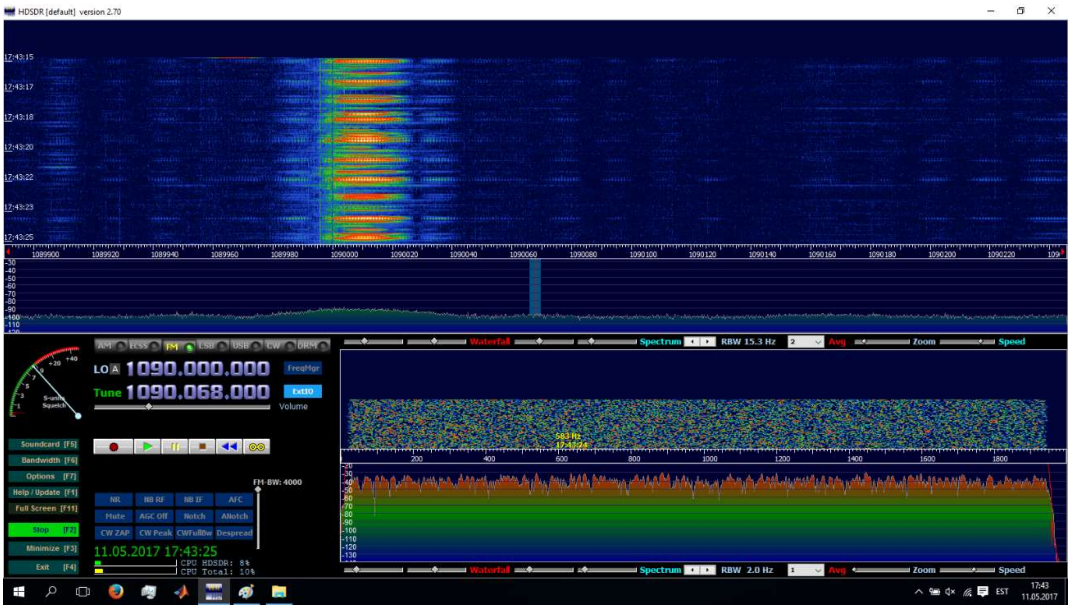
Joonis 5. USRP B210.

Seadme sisendisse on ühendatud varrasantenn TP-LINK TL-ANT2405C, mis tõsteti välja akna taha, ning seade ise ühendati arvuti külge USB 3.0 kaabliga. SDR ülesseadistus on kujutatud Joonisel 6.



Joonis 6. SDR ülesseadistus.

Esimese sammuna on kasutatud HDSDR tarkvara, et näha, kas kasutada oleva riistvaraga õnnestub signaalide vastuvõtmine reaalajas. Selleks on USRP sageduseks seadistatud 1090 MHz, proovitud erinevaid võimenduse ja diskreetimissageduse väärtusi, kuni tulemuseks oli pilt, kus on selgelt näha ADS-B signaale. Joonisel 7 on näha ADS-B signaale HDSDR tarkvaras.



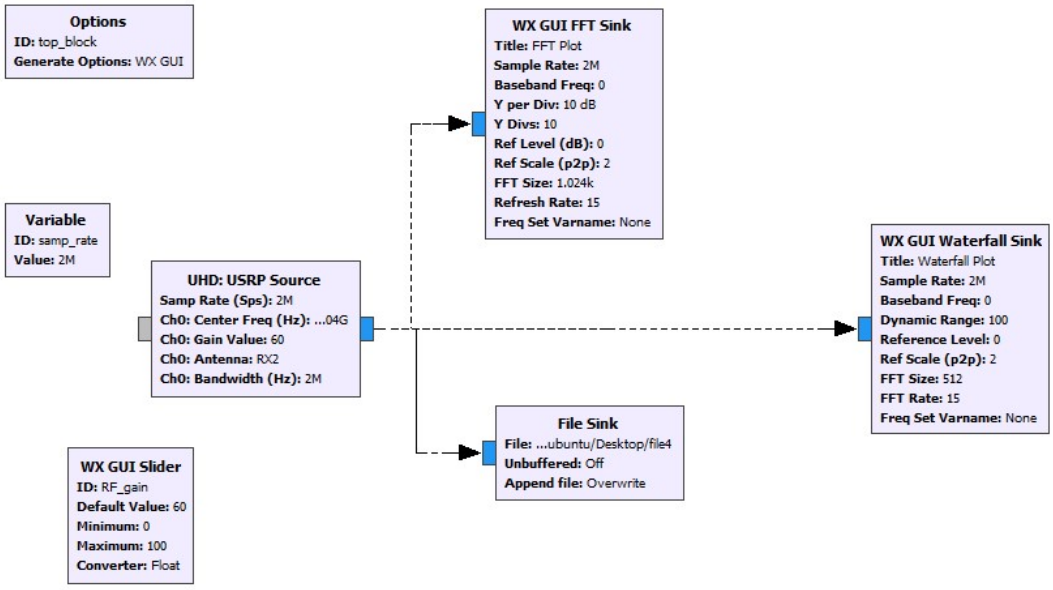
Joonis 7. ADS-B signaalid programmis HDSDR.

4.1 GNU Radio

Olles kindlaks teinud, et antud seadmega on võimalik ADS-B signaale vastu võtta, suunduti järgmise sammuna ehitama vastuvõtjat GNU Radios.

GNU Radio on tasuta avatud lähtekoodiga tarkvaraarenduse vahend, mis sisaldab erinevaid signaalitöötluseks mõeldud plokke, mida saab kasutada tarkvaralises raadios. Seda võib kasutada koos RF riistvaraga, et luua tarkvaralist raadiot või ilma riistvarata simulatsioonikeskkonnana [9].

Joonisel 9 on kujutatud voodiagramm ADS-B signaalide faili salvestamiseks.



Joonis 8. ADS-B signaalide faili salvestamine.

USRP parameetrid:

Diskreetimissagedus – 2 MHz. Ühe biti kestus on 1 mikrosekund ning ühe sümboli kodeerimiseks läheb tarvis vähemalt kaht väärtust. Diskreetimissagedusel 2 MHz saame kaks miljonit väärtust sekundis, millest piisab edukaks dekodeerimiseks.

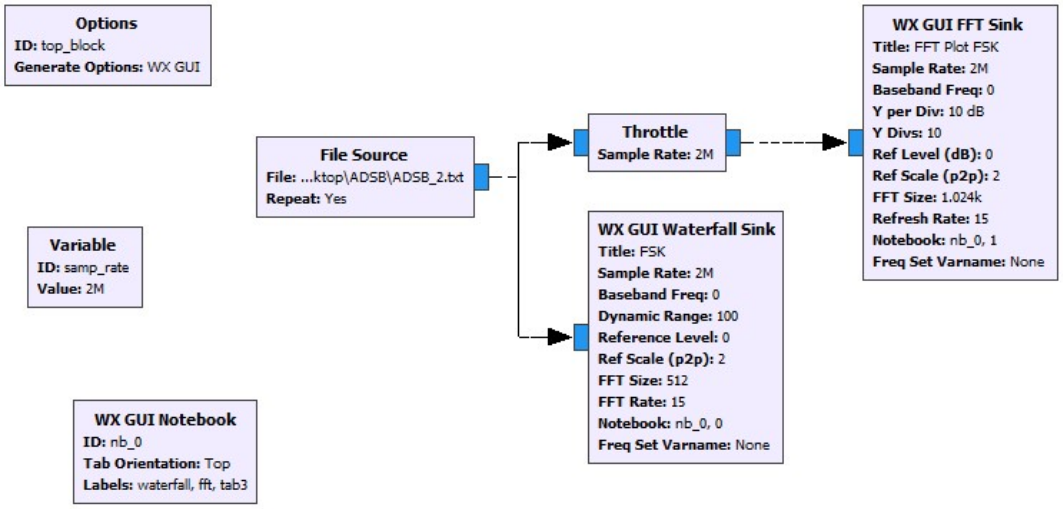
Kesksagedus – 1090,4 MHz. Tegemist on vastuvõetava signaali sagedusega. Kuna SDR viib signaali diskreetimise käigus põhiribasse nullise kandesageduse juurde, kuhu aga satub ka vastuvõtja mittelineaarsusest tingitud alaliskomponent, siis nihutatakse signaali sagedus nullist veidi kõrvale, seepärast 0,4 MHz-ne kõrvalekalle. Lisaks peab igasuguste liikuvate objektide puhul arvestama Doppleri efektiga.

Võimendus – 60 dB. Võimendada on vaja, sest signaal nõrgeneb levides ning demoduleerimiseks on vaja teatavat signaalitugevust.

Ribalaius – 2 MHz. Ribalaius signaalitöötluses tähendab ülemise ja alumise lõikesageduse vahet. Alumiseks lõikesageduseks on töös seega 1090,4 MHz – 1 MHz = 1089,4 MHz ja ülemiseks sageduseks 1090,4 MHz + 1 MHz = 1091,4 MHz.

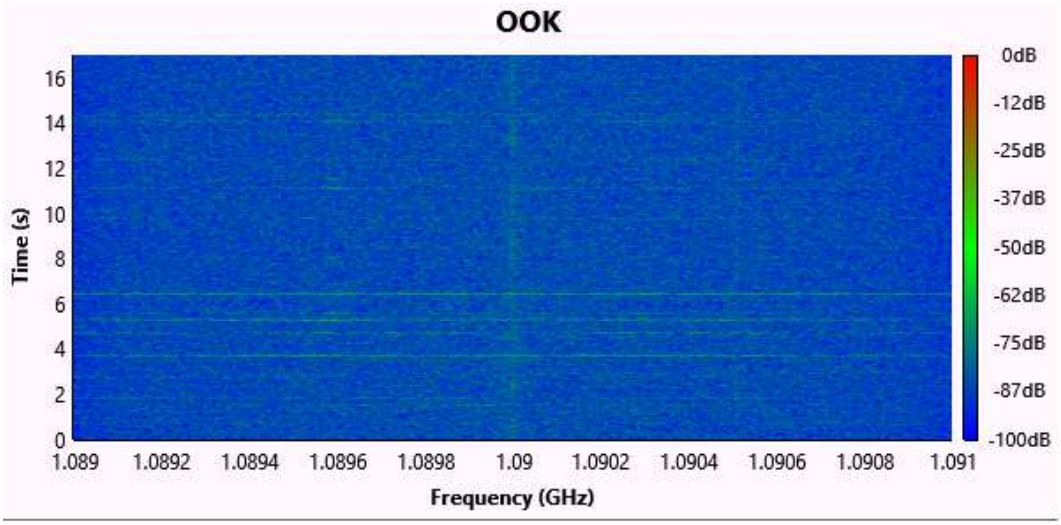
Plokke „*WX GUI FFT Sink*“ ja „*WX GUI Waterfall Sink*“ saab kasutada, et kuvada vastuvõetud signaali spektrit ja spektrogrammi reaalajas. Konkreetne vastuvõtja salvestab lisaks signaalide kuvamisele nad ka binaarfaili, mida teeb plokk File Sink. Sellisel juhul

on võimalik salvestada signaalid faili ning töödelda neid hiljem ilma, et peaks neid uuesti samal ajal vastu võtma ja ilma, et oleks vajadust riistvara ja lennukite järele. Joonisel 9 on kujutatud voodiagramm signaalide taasesitamiseks failist.



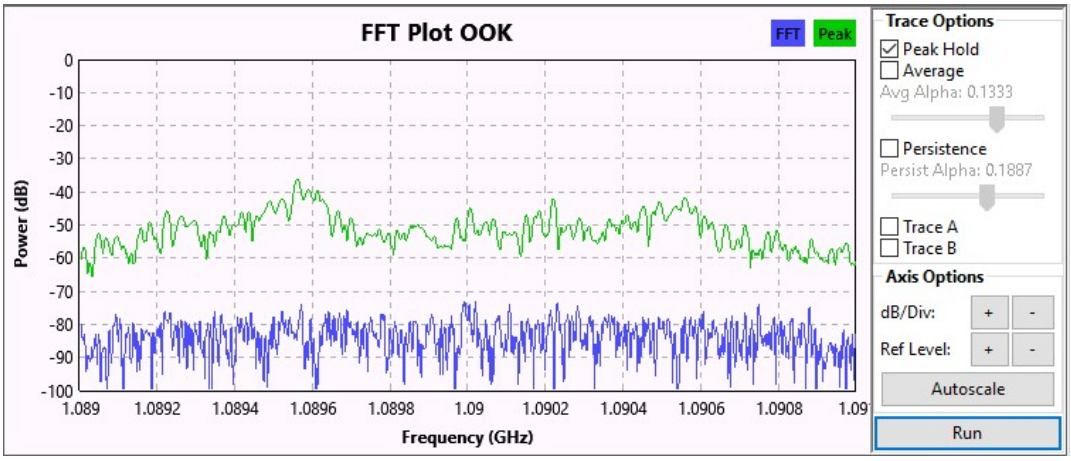
Joonis 9. Signaalide taasesitamine failist.

Signaaliallikaks on salvestatud fail, millele tagab juurdepääsu plokk File Source. Lisaks on kasutusel veel ka plokk nimega „Throttle“, mis on vajalik siis, kui skeemi pole ühendatud riistvara. Ta tagab selle, et arvuti ei kasutaks voodiagrammi käitamisel liigselt arvutusressurssi. Vastasel juhul võib juhtuda, et arvuti muutub aeglaseks või isegi jookseb kokku. Joonis 10 kujutab ADS-B signaali spektrogrammi.



Joonis 10. Vastu võetud ADS-B signaali spektrogramm.

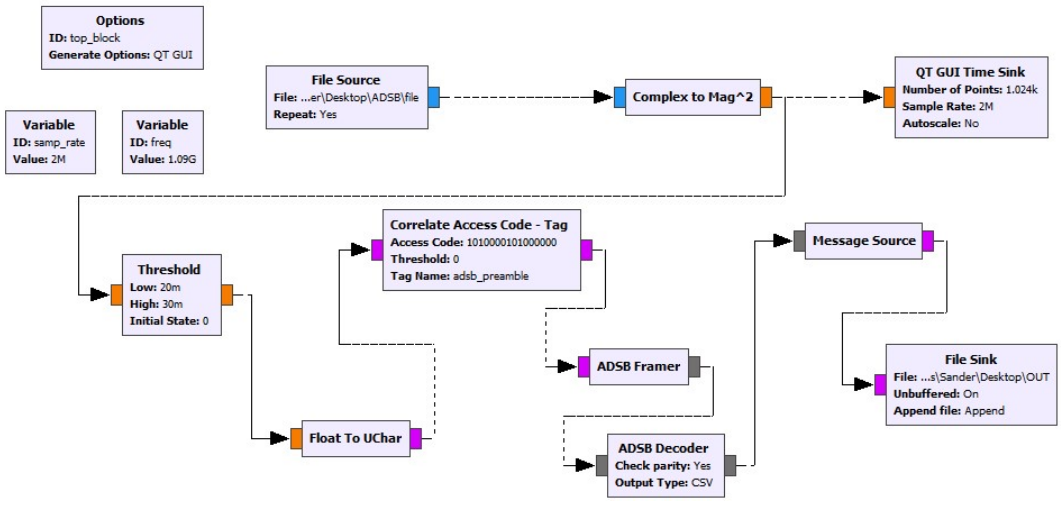
Joonisel 11 on kujutatud ADS-B signaali spetker.



Joonis 11. Vastu võetud ADS-B signaali spekter.

4.2 Signaalide demoduleerimine ja dekodeerimine

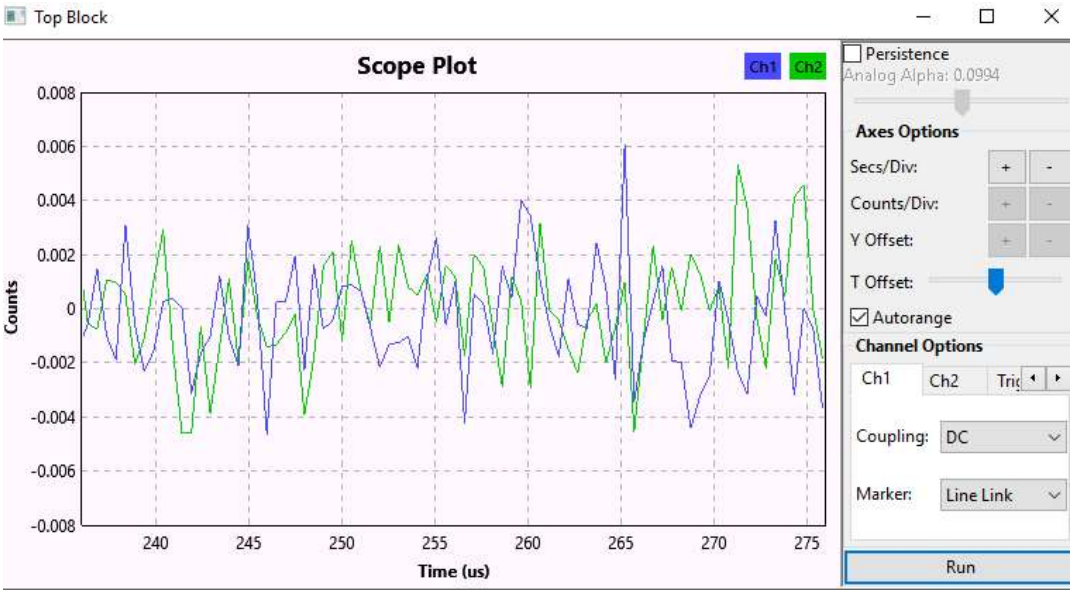
GNU Radios on ADS-B sõnumite dekodeerimiseks valmis ehitatud plokid nimega „*ADSB Framer*“ ja „*ADSB Decoder*“, mis võib leida gr-adsb kaustast [10]. Kasutades neid plokke, on võimalik pärast signaalide demoduleerimist neid dekodeerida.
Järgnevalt on koostatud ADS-B signaalide demoduleerija ja dekodeerija kasutades eelmainitud plokke, mida on kujutatud joonisel 12.



Joonis 12. ADS-B demoduleerija ja dekodeerija.

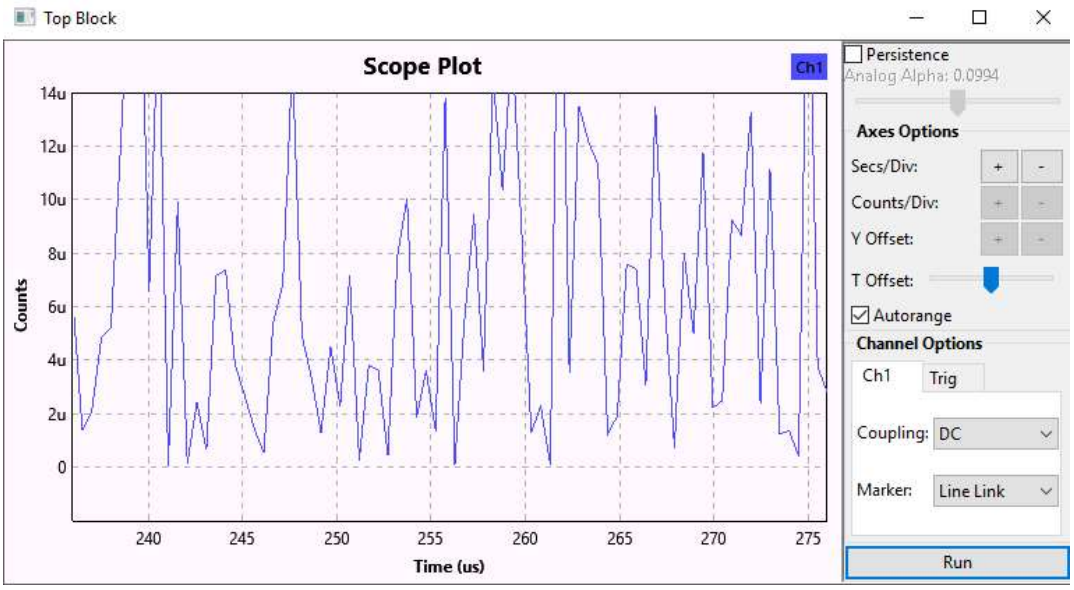
Plokkide kirjeldused:

UHD: USRP Source – tagab juurdepääsu kasutatavale riistvarale. Joonisel 13 on näha vastu võetud signaali ajaline kuju.



Joonis 13. Sisendsignaali ajaline kuju.

Complex to Mag^2 – implementeerib funktsiooni out = Re(in)^2 + Im(in)^2, ehk leiab sisendsignaali mooduli ruudu. Kuna meie signaali edastatakse ASK kujul, siis on tegemist põhimõtteliselt ruutdetektoriga. Antud skeemis on ta signaali demodulaatoriks. Joonis 14 kujutab demodulaatori väljundsignaali.



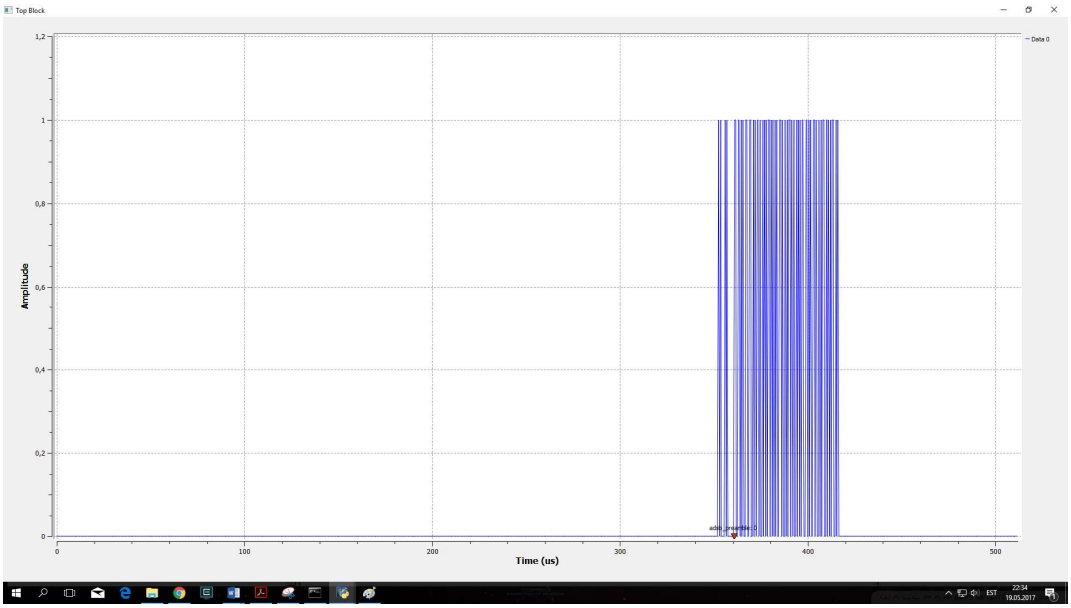
Joonis 14. Demodulaatori väljund.

Threshold – töötab komparaatori põhimõttel. Väljund muutub 0-st 1-ks, kui sisendsignaalis toimub üleminek allpool madalat piiri üle ülemise piiri, ning 1-st 0-ks, kui signaali väärtus muutub üle ülemise seadistatud piiri alla alumise piiri. Ülemiseks piiriks olen seadnud 0,03 ja alumiseks 0,02. Ehk kõik signaali väärtused, mis on üle 0,03, muudetakse 1-ks ja kõik, mis alla 0,02, muudetakse 0-ks.

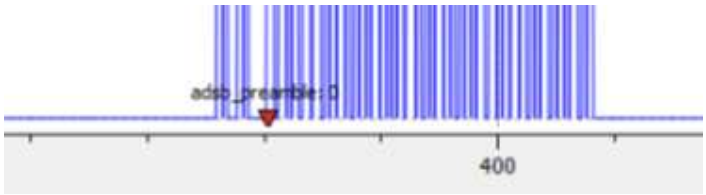
Float to UChar – muudab *float* tüüpi väärtuse *unsigned char*-iks. Muutmine on vajalik edasiseks töötluseks, kuna järgmine plokk nõuab *unsigned char* sisendit.

Correlate Access Code – Tag – otsib signaalist etteantud bitijada ning paneb selle lõppu sildi. Konkreetses skeemis otsib ta ADS-B päist kujul 1010000101000000.

Järgnevatelt joonistelt on näha ADS-B paketti ajalises esituses ja ka eelmainitud ploki poolt külge pandud silti. Jooniselt 15 ja 16 on näha ADS-B paketti temale külge pandud sildiga.



Joonis 15. ADS-B pakett.



Joonis 16. Külge pandud silt lähemalt.

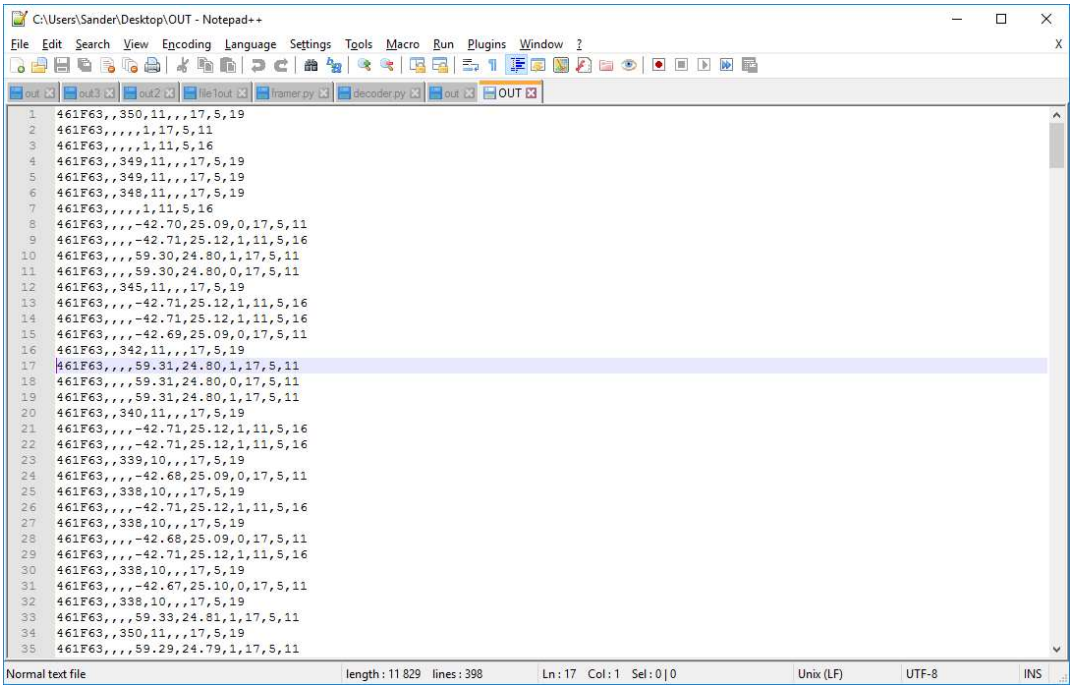
ADSB Framer – otsib eelnevalt seatud silti ning annab järgmisele plokile edasi selle asukoha.

ADSB Decoder – tegeleb sõnumi dekodeerimisega.

Message Source – loob sõnumist andmeteallika. Ta lubab andmevahetust Pythoni ja C++ kihtide vahel.
File Sink – salvestab saadud tulemuse faili.

Esimeseks plokiks on USRP riistvara. Järgmisena tuleks vastu võetud signaal demoduleerida, seda teeb plokk „*Complex to Mag^2*“. Kuna dekodeerida on võimalik ainult diskreetseid väärtusi ja demoduleeritud signaalis on mitu erinevat nivood, siis tuleb kasutada plokki „*Threshold*“, et muuta alla või üle teatud piiri signaali väärtused vastavalt kas nulliks või üheks. „*Float to Uchar*“ plokk muudab ujukoma andmetüübi *unsigned char*-iks, kuna järgmine plokk nõuab *unsigned char* sisendit. „*Correlate Access Code – Tag*“ plokk otsib bitijadast ADS-B päist (1010000101000000), kõikidele sellistele leitud päistele paneb ta juurde spetsiaalse sildi. Järgmine plokk „*ADSB Framer*“ otsib eelneva ploki poolt seadistatud silte ning annab nende asukoha „*ADSB Decoder*“ plokile edasi, mis tegeleb sõnumite dekodeerimisega. Kuna mõned GNU Radio plokid on kirjutatud Pythoni keskkonnas ja teised C++ keskkonnas, siis „*Message Source*“ plokk on vajalik, et tagada andmevahetus eri keskkonnas kirjutatud plokkide vahel. „*File Sink*“ plokk salvestab saadud tulemuse faili.

Faili salvestatud informatsiooni kujutab Joonis 17.



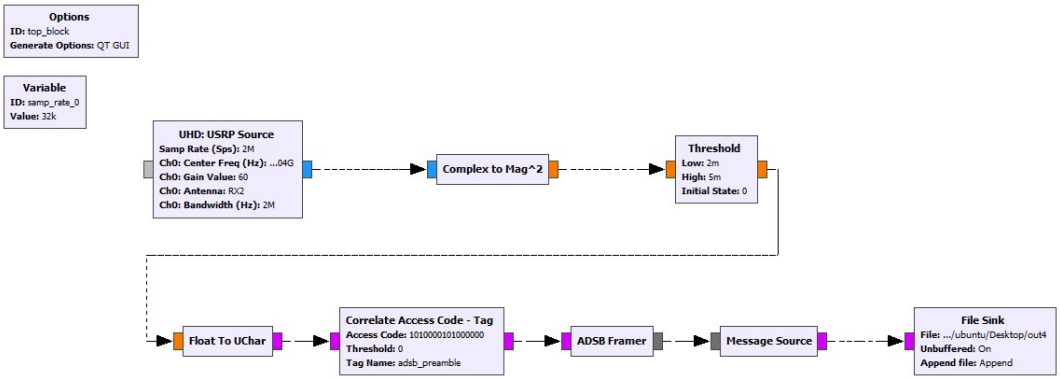
Joonis 17. Voodiagrammi väljundfail.

Tõin näiteks esile rea 17. Esimene tähtede ja numbrite kombinatsioon 461F63 tähistab õhusõiduki Mode-S koodi, järgmised numbrid 59.31 ja 24.80 on hetkekoordinaadid, ehk laius- ja pikkuskraadid.

GNU Radios eksisteerib ka valmis rakendus ADS-B tehnoloogiaga varustatud õhusõidukite jälgimiseks reaalajas [11], mille autoriks on Nick Foster. Lisaks on võimalik õhusõidukeid jälgida ka erinevate veebirakenduste kaudu, näiteks www.flightradar24.com [12]. Ekraanitõmmised mainitud rakendustest võib leida Lisast 2.

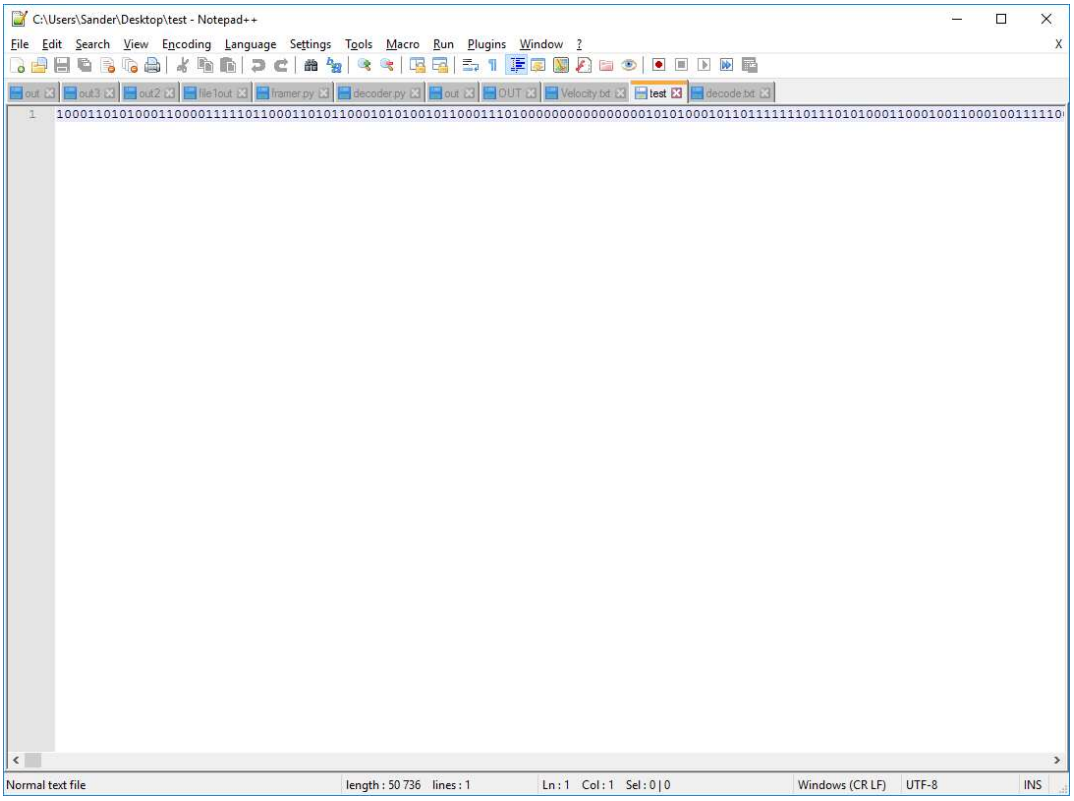
5 ADS-B pakettide dekodeerimine

ADS-B pakettide ise dekodeerimiseks on modifitseeritud eelnevat voodiagrammi nii, et väljundfaili salvestatakse ainult vastu võetud ADS-B paketid bitijada kujul. Selle saavutamiseks on voodiagrammist eemaldatud „*ADSB Decoder*“ plokk, mis tegeleb paketi dekodeerimisega. Joonis 18 esitab modifitseeritud voodiagrammi.



Joonis 18. Modifitseeritud voodiagramm.

Tulemuseks on alljärgnev väljundfail, mida kujutab Joonis 19.



Joonis 19. Voodiagrammi väljundfail.

Esimese asjana paistab silma, et esimesed viis bitti on 10001, mis tähistab *Downlink Formati* 17. Järelikult võib oletada, et esimene vastu võetud ADS-B pakett algab sealtmaalt. Esimese asjana tuleb kontrollida, kas vastu võetud pakett on vigane või mitte. Selleks tuleb kontrollida, kas paketi CRC (*Cyclic Redundancy Check*) koodi väärtuseks on null või miski muu väärtus. Kui kasvõi üks bitt on moonutatud, osutub CRC koodi väärtus mittenulliseks, pakett loetakse vigaseks ning seda arvesse ei võeta. ADS-B pakettide viimased 24 kontrollbitti saadakse 88 andmebiti kombineerimisel generaatorpolünoomiga 1111111111111010000001001. CRC koodi arvutamiseks käsitsi ja ka punktis 5.4 kirjeldatud programmikoodis on kasutatud CRC kalkulaatorit [13]. Selgub, et vastu võetud paketi CRC koodi väärtuseks on null, mis tähendab, et pakett on vigadeta ning seega võib asuda paketti dekodeerima. Paketi dekodeerimisel on lähtutud Junzi Suni avalikust juhendist [6].

Bitid:

10001101010001100001111101100011010110001010100101100011101000000000000000010101000101101111111011101010001100010011000100111100000101010001011011111110111010100011000 1.

Kasutades peatükis 3 välja toodud Tabelit 1, on vastu võetud pakett välja kirjutatud vastavate osadena:

DF - 10001
CA - 101
ICAO - 010001100001111101100011
TC - 10011
DATA - 001 0 1 000 0 0001000100 0 0101011000 0 1 000011011 00 1 0110001
PI -000110111100100101111011

Teisendades ICAO bitijada kuueteistkümnendsüsteemi, saame tulemuseks õhusõiduki Mode-S koodi - 461F63.
Et teada saada, millist infot sisaldab andmevälja osa, tuleb vaadata, mis väärtusega on *TC*. Antud paketis $TC = 10011_2 = 19_{10}$. Jällegi, kasutades eelmainitud tabelit, võib välja lugeda, et õhusõiduk edastab konkreetse paketiga enda lennukiirust (*Airborne Velocity*).

Lennukiiruse paketis on mitu parameetrit. Tabel 3 annab ülevaate iga osa tähendusest:

Tabel 3. Andmevälja struktuur [6].

DATA bitid	**Pikkus**	**Lühend**	**Sisu**
1-5	5	TC	Tüübikood
6 – 8	3	ST	Alatüüp
9	1	IC	*Intent change flag*
10	1	RESV_A	Reserveeritud-A
11 – 13	3	NAC	Kiiruse ebakindlus
14	1	S_ew	Ida-lääs kiiruse märk
15 – 24	10	V_ew	Ida-lääs kiirus
25	1	S_ns	Põhi-lõuna kiiruse märk
26 – 35	10	V_ns	Põhi-lõuna kiirus
36	1	VrSrc	Vertikaalse kiiruse allikas
37	1	S_vr	Vertikaalse kiiruse märk
38 – 46	9	Vr	Vertikaalne kiirus
47 – 48	2	RESV_B	Reserveeritud-B
49	1	S_Dif	Erinevus baromeetri näidust, märk
50 – 56	7	Dif	Erinevus baromeetri näidust

5.1 Horisontaalne kiirus

Horisontaalse kiiruse arvutamiseks läheb vaja nelja väärtust: Ida-lääs kiirus *V_ew*, Ida-lääs kiiruse märk *S_ew*, Põhi-lõuna kiirus *V_ns*, Põhi-lõuna kiiruse märk *S_ns*.

S_ns:

1 – lendab põhjast lõunasse

0 – lendab lõunast põhja

S_ew:

1 – lendab idast läände

0 – lendab läänest itta

Meie paketis *S_ns* = 0 ja *S_ew* = 0.

Järgmisena tuleb teisendada Ida-lääs kiirus *V_ew* ja Põhi-lõuna kiirus *V_ns* kümnendsüsteemi.

$V\_ew = 0001000100_2 = 68_{10}$

$V\_ns = 0101011000_2 = 344_{10}$

Järgnevate valemitega on võimalik välja arvutada lennukiirus [6]:

$$Vwe = \begin{cases} -1 \cdot (Vew - 1), \text{kui } Sew = 1 \\ Vew - 1, \text{kui } Sew = 0 \end{cases}$$

$$Vsn = \begin{cases} -1 \cdot (Vns - 1), \text{kui } Sns = 1 \\ Vns - 1, \quad \text{kui } Sns = 0 \end{cases}$$

$$V = \sqrt{Vwe^2 + Vsn^2}$$

Kuna *S_ew* = 0 ja *S_ns* = 0, siis

$V_{we} = V_{ew} - 1$ ja $V_{sn} = V_{ns} - 1$.

$$V_{we} = 344 - 1 = 343$$

$$V_{sn} = 68 - 1 = 67$$

$$V = \sqrt{343^2 + 67^2} = 349 \; sõlme$$

Üks sõlm on defineeritud kui üks meremiil tunnis, täpsemini 1.852 kilomeetrit tunnis.

Lennuki kiirus on seega 349 · 1,852 = 646 km/h.

5.2 Kurss

Kursi arvutamiseks kasutatakse järgnevat valemit [6]:

$$h = \arctan2(V_{we}, V_{sn}) \cdot 360/2\pi \text{ (kraadi)}$$

arctan2 funktsioon on arkustangens kahe argumendiga ning on defineeritud järgmiselt:

$$\text{arctan2}(y, x) = \begin{cases} \arctan(y/x), \text{kui } x > 0, \\ \arctan(y/x) + \pi, \text{kui } x < 0 \text{ ja } y \geq 0, \\ \arctan(y/x) - \pi, \text{kui } x < 0 \text{ ja } y < 0, \\ +\pi/2, \text{kui } x = 0 \text{ ja } y > 0 \\ -\pi/2, \text{kui } x = 0 \text{ ja } y < 0, \\ \text{määramata, kui } x = 0 \text{ ja } y = 0 \end{cases}$$

Kasutades eelnevalt arvutatud V_{we} = 343 ja V_{sn} = 67, saame tulemuseks:

$$h = \arctan(343/67) = 78{,}9.$$

Kurss on seega 78,9.

5.3 Vertikaalne kiirus

Vertikaalse liikumise suuna määrab 37. andmevälja bitt *S_vr*. Kui biti väärtus on 0, siis õhusõiduki kõrgus parajasti kasvab. Laskumise korral on tema väärtuseks 1. Vertikaalse kiiruse *V* arvutamiseks läheb vaja andmevälja bitte 38 – 46. Kõigepealt tuleb teisendada binaarväärtus kümnendsüsteemi, seejärel lahutada sellest 1 ning korrutada 64-ga. Vastuseks on vertikaalne kiirus ühikutes jalga/minutis.

S_vr = 1 – lennuk laskub

$Vr = 000011011_2 = 27_{10}$

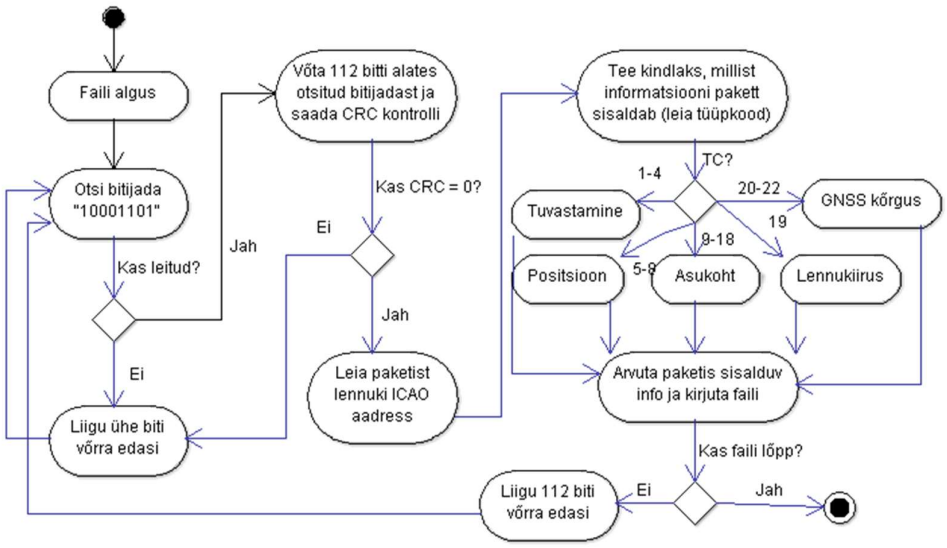
$$V = (Vr - 1) \cdot 64 = (27 - 1) \cdot 64 = 1664 \text{ f/min (507 m/min).}$$

Üks jalg on 0,3048 meetrit, seega saame laskumiskiiruseks 1664 · 0,3048 = 507 meetrit minutis.

5.4 Pakettide dekodeerimine C keskkonnas

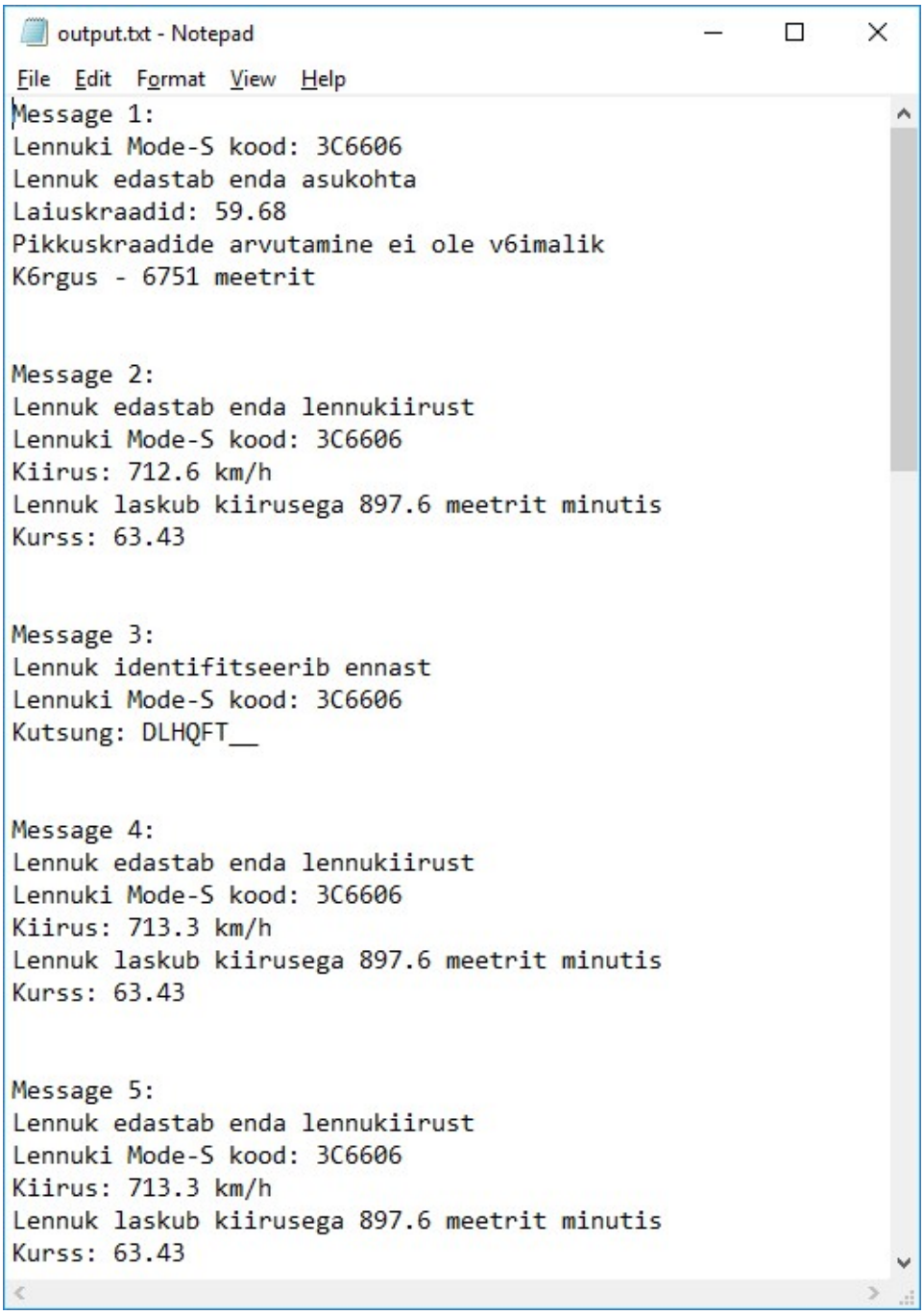
Järgnevalt on kirjutatud autori poolt programm C keeles, mis loeb tekstifaili salvestatud ADS-B paketid sisse, seejärel dekodeerib need ning kirjutab pakettides sisalduva informatsiooni tekstifaili. Programmi sisendiks on peatükis 5 kirjeldatud GNU Radio voodiagrammi väljundfail, mis peab olema käivitatava programmiga samas kaustas, ning mille nimi peab olema „input.txt“. Juhul, kui programmi jooksutajal on olemas C kompilaator ja vastav tekstiredaktor, võib ta programmikoodis avatava failinime muuta selliseks, nagu ta soovib. Programm otsib ADS-B paketi algust, kontrollib paarsusbittide abil CRC koodi, et teha kindlaks, et vastu võetud pakett oleks vigadeta. Seejärel leitakse lennuki ICAO aadress, tehakse kindlaks, millist informatsiooni pakett sisaldab ning dekodeeritakse info. Programmi kood on esitatud Lisas 1.

Joonisel 20 on esitatud programmis ArgoUML koostatud programmi algoritmi tööd kirjeldav diagramm:

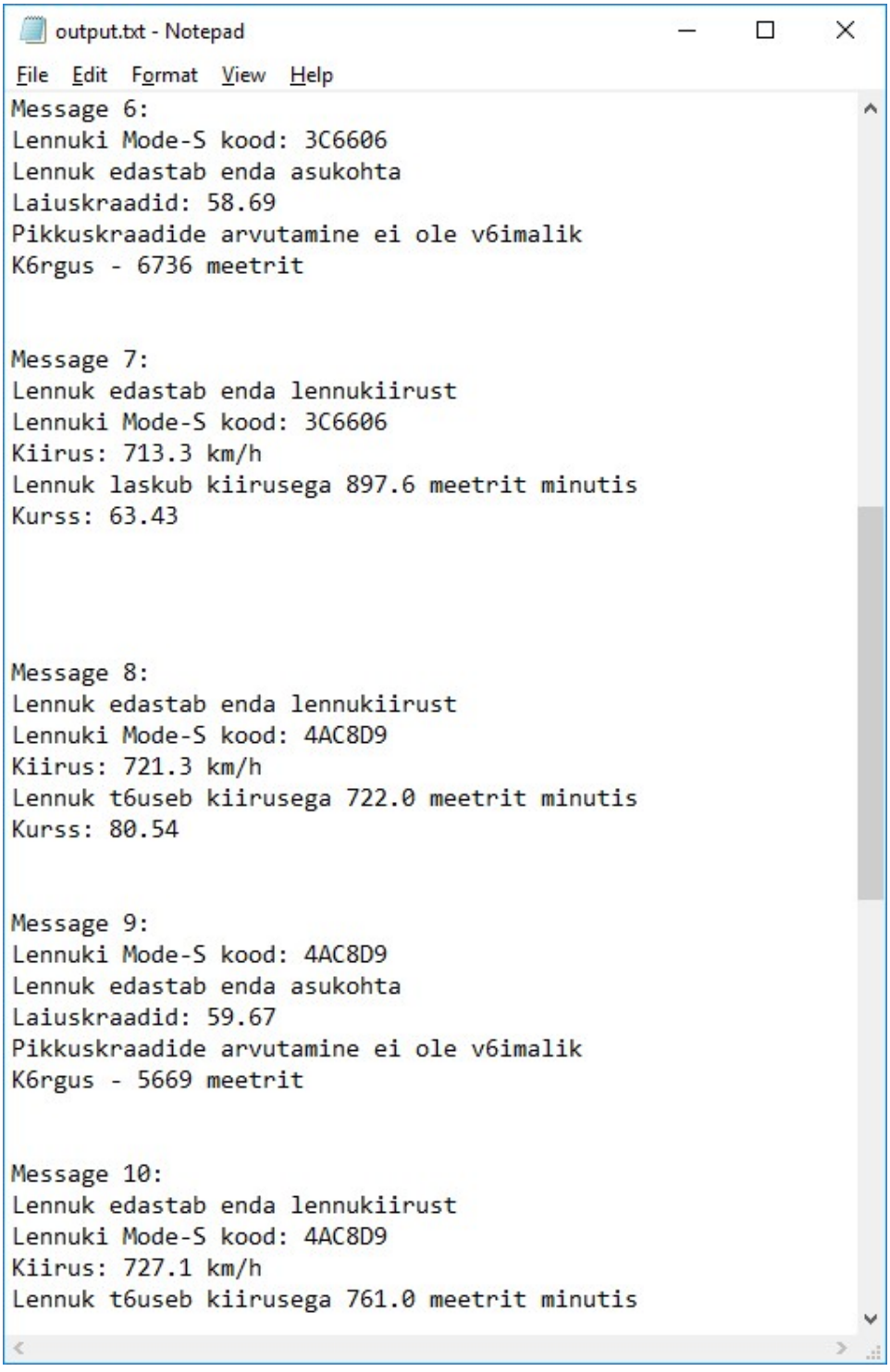


Joonis 20. Programmi algoritm.

Joonised 21 ja 22 kujutavad programmi väljundfaili.



Joonis 21. Programmi väljundfail.



Joonis 22. Programmi väljundfail jätkub.

Failist võib välja lugeda, et signaalide vastuvõtmise ajal olid antennile piisavalt lähedal kaks õhusõidukit *Mode-S* koodiga 3C6606 ja 4AC8D9. Mainitud õhusõidukid edastasid erinevate pakettidega enamjaolt enda lennukiirust ja õhupositsiooni. Lennuk 3C6606 edastas lisaks ka enda kutsungit. Lisaks on näha asukohtade edastamisel ka pikkuskraadide määramatust. Põhjus seisneb asukoha edastuse viisis. Nimelt on vaja õhusõiduki asukoha arvutamiseks kaht järjestikust paketti. Kui õhusõiduki positsioon on nende kahe paketi edastamise vahel liiga palju muutunud, siis asetsevad arvutatud laiuskraadid eri tsoonides ning pikkuskraade ei ole võimalik arvutada. Seega tuleb pikkuskraadide arvutamisest loobuda ning oodata uusi sõnumeid.

6 Kokkuvõte

Käesoleva töö eesmärgiks oli analüüsida õhusõidukite ADS-B signaalide vastuvõtmise ja dekodeerimise võimalikkust ja rakendamist tarkvaralise raadioga ning võimalusel ka konstrueerida selleks otstarbeks vastav vastuvõtja ja dekodeerija. Lisaks seati ka eesmärgiks anda hinnang ADS-B leviedastusstandardi turvalisusele. Eesmärkide saavutamiseks andis autor üldise ülevaate ADS-B leviedastusstandardist, ning täpsemini edastatavate sõnumite tehnilisest kirjeldusest ja edastusviisist, ning kasutada oleva tarkvaralise raadio tööpõhimõttest.

Töö käigus tehti kindlaks ADS-B sõnumite edastatav sagedus ning valiti signaalide vastuvõtuks seade, mille sagedusvahemik katab nõutud sagedust. Seade USRP B210 sobis edukalt selleks otstarbeks. Kasutades mainitud seadet ning valmisrakendust HDSDR, oli võimalik kuvada õhusõidukite poolt edastatavaid signaale ajalises kujus reaalajas. See andis kinnitust, et Eesti õhuruumis lendavad õhusõidukid kasutavad ADS-B standardit ning kasutada olev riistvara suudab nende edastatavaid signaale vastu võtta.

Tarkvaras GNU Radio konstrueeritud ADS-B signaalide vastuvõtja töötas korrektselt ning salvestas vastu võetud paketid tekstifaili. Autori poolt koostatud programm C keskkonnas dekodeeris failis sisalduva informatsiooni ning salvestas tulemused samuti tekstifaili.

Töö tulemusi analüüsides selgub, et omades vastavat riist- ja tarkvara, on võimalik igal soovijal ADS-B signaale vastu võtta ja dekodeerida. See tõstatab turvalisusega seotud probleeme. Omades tarkvaralist raadiot, mis on võimeline ka edastama, on võimalik edastada võltsitud ADS-B signaale, kuna sõnumid pole krüpteeritud ega autenditud. See tähendab, et vastuvõtjal pole võimalik kindlaks teha, kas vastu võetud sõnumid on edastatud õiguspärase allika poolt või mitte ning et kas sõnumite sisu on korrektne ja puutumata. Praeguseks on FAA poolt viidud läbi ADS-B süsteemi turvalisuse parandamise hindamisi ning on võimalik, et standardile lisatakse tulevikus täiendusi [14]. Hetkel kaitstakse end selliste rünnakute vastu primaar- ja sekundaarradari kasutusega lisaks ADS-B süsteemile, välja filtreerimaks võltsitud sõnumid.

Kui arvesse võtta, et aastaks 2020 peab enamus õhuruumis liiklevad õhusõidukid olema varustatud ADS-B tehnoloogiaga [2], on autor seisukohal, et hetkel kasutusel olev ADS-B standard väärib turvalisuse tõstmiseks täiendusi ja/või muudatusi ära hoidmaks potentsiaalseid terroriohte. Hetkel ei ole sellised pahatahtlikud rünnakud levinud, kuid aja möödudes on võimalik, et need muutuvad järjest enam tõenäolisemaks.

Kasutatud kirjandus

[1] Zimmermann, John (2013) [WWW] „ADS-B 101: What It Is And Why You Should Care“. http://airfactsjournal.com/2013/01/ads-b-101-what-it-is-and-why-you-should-care/ (03.05.2017)

[2] Davidson, Jason (2017) [WWW] „2017 Flight Planning Update: ADS-B Mandates and What’s Changed So Far“. http://www.universalweather.com/blog/2017/02/2017-flight-planning-update-ads-b-mandates-and-whats-changed-so-far/ (23.02.2017)

[3] [WWW] https://en.wikipedia.org/wiki/Automatic_dependent_surveillance_%E2%80%93_broadcast (20.01.2017)

[4] Scardina, John (2002) [WWW] „Overview of the FAA ADS-B Link Decision“. https://web.archive.org/web/20070316010309/http://www.faa.gov:80/asd/ads-b/06-07-02_ADS-B-Overview.pdf (12.04.2017)

[5] Wolff, Christian [WWW] http://www.radartutorial.eu/13.ssr/sr24.en.html (23.02.2017)

[6] Sun, Junzi [WWW] https://adsb-decode-guide.readthedocs.io/en/latest/content/introduction.html (23.02.2017)

[7] [WWW] http://www.sigidwiki.com/wiki/Automatic_Dependent_Surveillance-Broadcast_(ADS-B) (23.02.2017)

[8] Fairhurst, Gorry (2007) [WWW] „Manchester Encoding“. http://www.erg.abdn.ac.uk/Users/gorry/course/phy-pages/man.html (23.02.2017)

[9] [WWW] https://wiki.gnuradio.org/index.php/Main_Page (01.03.2017)

[10] [WWW] https://github.com/wnagele/gr-adsb (15.03.2017)

[11] Foster, Nick [WWW] https://github.com/bistromath/gr-air-modes (15.02.2017)

[12] [WWW] www.flightradar24.com (23.02.2017)

[13] [WWW] https://www.ghsi.de/CRC/index.php? (09.05.2017)

[14] Prince, Brian (2012) „Air Traffic Control Systems Vulnerabilities Could Make for Unfriendly Skies [Black Hat]“. [WWW] http://www.securityweek.com/air-traffic-control-systems-vulnerabilities-could-make-unfriendly-skies-black-hat (20.05.2017)

Lisa 1 – Programmikood

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#define PI 3.14159265358979323846


int counter = 1;

char *makeCRC(char *Bitstring); // Kontrollib CRC koodi
int bintodec(int number); // Muudab binaararvu kümnendarvuks
void lennukiirus(char *message, FILE* ft); // Arvutab lennukiiruse
void modeS_TC(char *message, char *buffer, int spot, FILE* ft, int filesize);
// Leiab lennuki Mode-S koodi ja Tüübikoodi
void identify(char *message, FILE* ft); // Leiab lennuki kutsungi
void position(char *message, char *buffer, int spot, int kood, FILE* ft, int
filesize); // Leiab asukoha dekodeerimiseks kaks paketti
void altitude(char *message, FILE* ft); // Leiab lennuki kõrguse
void calc_pos(float even_laius, float even_pikkus, float odd_laius, float
odd_pikkus, char flag, int kood, FILE* ft); // Arvutab lennuki asukoha
int NL(float laius); // Kontrollib, kas arvutatud laiuskraadid langevad
samassse tsooni


int main(void)
{

    FILE* fp = fopen("test.txt", "r");

    if (fp == NULL) {
        printf("Faili ei suudetud avada...\n");
        return 0;
    }

    fseek(fp, 0L, SEEK_END);
    int sz = ftell(fp);
    rewind(fp);


    if (sz < 112) {
        printf("Failis pole piisavalt andmeid\n");
        fclose(fp);
        return 0;
    }

    char *buff;

    buff = malloc(sz + 1);
```

```
    if (!buff) {
        printf("Puhvrile ei suudetud m2lu eraldada\n");
        return 0;
    }

    fgets(buff, sz, fp);


    char *adsb;

    adsb = malloc(113);

    if (!adsb) {
        printf("S6numile ei suudetud m2lu eraldada\n");
        return 0;
    }

    int j = 0;

    int compare;
    char *result;


    // Otsib bitte "10001101"

    FILE* ft = fopen("output.txt", "w");

    while (j <= sz - 112) {
        if (buff[j] == '1' && buff[j+1] == '0' && buff[j+2] == '0' &&
buff[j+3] == '0' && buff[j+4] == '1' && buff[j+5] == '1' && buff[j+6] == '0'
&& buff[j+7] == '1') {   // Kui leiab, eeldab, et on tegemist ADSB paketiga.
            memcpy(adsb, &buff[j], 112);
            adsb[112] = '\0';
            result = makeCRC(adsb); // CRC kontroll
            compare = strcmp(result, "00000000000000000000000");
            if (compare != 0) {
                j++;
            } else {
                j += 112;
                modeS_TC(adsb, buff, j, ft, sz);
            }

        } else {
            j++;
        }
    }

    fseek(ft, 0L, SEEK_END);
    int sz2 = ftell(ft);

    if (sz2 == 0) {
        printf("Failis pole korrektseid pakette\n");
    } else {
        printf("Failis sisalduv info kirjutatud faili 'output.txt'\n");
    }

    free(adsb);
    free(buff);
    fclose(fp);
```

```
    fclose(ft);


    return 0;
}

char *makeCRC(char *BitString) {
   static char Res[25];
   char CRC[24];
   int  i;
   char DoInvert;

   for (i=0; i<24; ++i)  CRC[i] = 0;

   for (i=0; i<(int)strlen(BitString); ++i) {
      DoInvert = ('1'==BitString[i]) ^ CRC[23];

      CRC[23] = CRC[22] ^ DoInvert;
      CRC[22] = CRC[21] ^ DoInvert;
      CRC[21] = CRC[20] ^ DoInvert;
      CRC[20] = CRC[19] ^ DoInvert;
      CRC[19] = CRC[18] ^ DoInvert;
      CRC[18] = CRC[17] ^ DoInvert;
      CRC[17] = CRC[16] ^ DoInvert;
      CRC[16] = CRC[15] ^ DoInvert;
      CRC[15] = CRC[14] ^ DoInvert;
      CRC[14] = CRC[13] ^ DoInvert;
      CRC[13] = CRC[12] ^ DoInvert;
      CRC[12] = CRC[11] ^ DoInvert;
      CRC[11] = CRC[10];
      CRC[10] = CRC[9] ^ DoInvert;
      CRC[9] = CRC[8];
      CRC[8] = CRC[7];
      CRC[7] = CRC[6];
      CRC[6] = CRC[5];
      CRC[5] = CRC[4];
      CRC[4] = CRC[3];
      CRC[3] = CRC[2] ^ DoInvert;
      CRC[2] = CRC[1];
      CRC[1] = CRC[0];
      CRC[0] = DoInvert;
    }

   for (i=0; i<24; ++i)  Res[23-i] = CRC[i] ? '1' : '0';
   Res[24] = 0;

   return(Res);
}

int bintodec(int number) { // Muudab binaararvu kümnendarvuks


    int decimal_val = 0;
    int base = 1;
    int rem;

    while (number > 0) {
        rem = number % 10;
        decimal_val = decimal_val + rem * base;
```

```
        number = number / 10 ;
        base = base * 2;
    }

    return decimal_val;

}


void modeS_TC(char *message, char *buffer, int spot, FILE* ft, int filesize){
// Leiab lennuki Mode-S koodi ja Type Code-i

    char modes[25];
    memcpy(modes, &message[8], 24);
    modes[24] = '\0';

    char *bin = modes;
    char *a = bin;
    int num = 0;

    do {
        int b = *a=='1'?1:0;
        num = (num<<1)|b;
        a++;
    } while (*a);

    char typecode[6];
    memcpy(typecode, &message[32], 5);
    typecode[5] = '\0';


    int val_TC = atoi(typecode);


    int ident = bintodec(val_TC);

    if (ident <= 0 || ident > 22) {
        return;
    }

    switch(ident) {
        case 1 ... 4:
        fprintf(ft, "Message %d:\n", counter);
        counter++;
        fprintf(ft, "Lennuk identifitseerib ennast\n");
        fprintf(ft, "Lennuki Mode-S kood: %X\n", num);
        identify(message, ft);
        break;
        case  5 ... 8:
        fprintf(ft, "Message %d:\n", counter);
        counter++;
        fprintf(ft, "Surface position\n");
        fprintf(ft, "Lennuki Mode-S kood: %X\n", num);
        break;
        case 9 ... 18:
        position(message, buffer, spot, num, ft, filesize);
        break;
        case 19:
        fprintf(ft, "Message %d:\n", counter);
```

```
            counter++;
            fprintf(ft, "Lennuk edastab enda lennukiirust\n");
            fprintf(ft, "Lennuki Mode-S kood: %X\n", num);
            lennukiirus(message, ft);
            break;
            case 20 ... 22:
            fprintf(ft, "Lennuk edastab enda k6rgust (GNSS)\n");
            break;
            default:
            fprintf(ft, "Muu\n");
        }

        fprintf(ft, "\n\n");

    }

    void lennukiirus(char *message, FILE* ft) { // Arvutab lennukiiruse
        char eastwest[11];
        memcpy(eastwest, &message[46], 10);
        eastwest[10] = '\0';

        int val_EW = atoi(eastwest);

        char northsouth[11];
        memcpy(northsouth, &message[57], 10);
        northsouth[10] = '\0';

        int val_NS = atoi(northsouth);

        int speedEW = bintodec(val_EW);
        int speedNS = bintodec(val_NS);

        if (message[46] == 1) {
            speedEW = -1 * (speedEW - 1);
        } else {
            speedEW = speedEW - 1;
        }
        if (message[57] == 1) {
            speedNS = -1 * (speedNS - 1);
        } else {
            speedNS = speedNS - 1;
        }


        float velocity;


        velocity = 1.852 * sqrt(pow(speedNS, 2) + pow(speedEW, 2));

        fprintf(ft, "Kiirus: %0.1f km/h\n", velocity);


        // Vertikaalkiirus

        char vertical[10];

        strncpy(vertical, &message[69], sizeof(vertical) - 1);

        int speedVR = strtol(vertical, NULL, 2);
```

```
    float verticalV = 64 * (speedVR - 1) / 3.28;

    if (message[68] == '0')
    {
        fprintf(ft, "Lennuk t6useb kiirusega %0.1f meetrit minutis\n",
verticalV);
    }

    else
        fprintf(ft, "Lennuk laskub kiirusega %0.1f meetrit minutis\n",
verticalV);
    // Heading


    float arctan;

    if (speedEW > 0)
    {
        arctan = atan(speedNS / speedEW) * (180/PI);
    } else if (speedEW < 0 && speedNS >= 0)
    {
        arctan = (atan(speedNS / speedEW) + PI) * (180/PI);
    } else if (speedEW < 0 && speedNS < 0)
    {
        arctan = (atan(speedNS / speedEW) - PI) * (180/PI);
    } else if (speedEW == 0 && speedNS > 0)
    {
        arctan = 90;
    } else if (speedEW == 0 && speedNS < 0)
    {
        arctan = -90;
    } else
        arctan = 0;

    while (arctan < 0) {
        arctan = arctan + 360;
    }

    fprintf(ft, "Kurss: %0.2f\n", arctan);

}

void identify(char *message, FILE* ft) {

    char callsign[] =
"#ABCDEFGHIJKLMNOPQRSTUVWXYZ#####_###############0123456789######";
    char c1[7];
    char c2[7];
    char c3[7];
    char c4[7];
    char c5[7];
    char c6[7];
    char c7[7];
    char c8[7];

    strncpy(c1, &message[40], 6);
    strncpy(c2, &message[46], 6);
    strncpy(c3, &message[52], 6);
```

```
    strncpy(c4, &message[88], 6);
    strncpy(c5, &message[64], 6);
    strncpy(c6, &message[70], 6);
    strncpy(c7, &message[76], 6);
    strncpy(c8, &message[82], 6);

    int c11 = strtol(c1, NULL, 2);
    int c12 = strtol(c2, NULL, 2);
    int c13 = strtol(c3, NULL, 2);
    int c14 = strtol(c4, NULL, 2);
    int c15 = strtol(c5, NULL, 2);
    int c16 = strtol(c6, NULL, 2);
    int c17 = strtol(c7, NULL, 2);
    int c18 = strtol(c8, NULL, 2);

    fprintf(ft, "Kutsung: %c%c%c%c%c%c%c%c\n", callsign[c11], callsign[c12],
callsign[c13], callsign[c14], callsign[c15], callsign[c16], callsign[c17],
callsign[c18]);

}

void position(char *message, char *buffer, int spot, int kood, FILE* ft, int
filesize) {

    static float cpr_lat_even;
    static float cpr_lon_even;

    static float cpr_lat_odd;
    static float cpr_lon_odd;

    int compare;
    char *result;



    switch(message[53]) {
        case '0': ;
        // Even pakett
        char lat_even[18];
        char lon_even[18];
        char lat_odd[18];
        char lon_odd[18];

        char uus[113];

        strncpy(lat_even, &message[54], sizeof(lat_even) - 1);
        strncpy(lon_even, &message[71], sizeof(lon_even) - 1);

        int cpr_lat_even1 = strtol(lat_even, NULL, 2);
        int cpr_lon_even1 = strtol(lon_even, NULL, 2);


        cpr_lat_even = (float)(cpr_lat_even1 / 131072.0);
        cpr_lon_even = (float)(cpr_lon_even1 / 131072.0);


        char algus1[38] = "10001101";
        char* leia1;
```

```
        strncpy(algus1 + 8, &message[8], 29);

        int i = 0;


        while (spot + i < filesize - 112) {
            leia1 = strstr(buffer + spot + i, algus1);

            if (!leia1) {
                break;
            }

            strncpy(uus, leia1, 112);


            result = makeCRC(uus);
            compare = strcmp(result, "000000000000000000000000");
            if (compare == 0 && uus[53] == '1')
            {
                break;
            } else {
                i++;
            }
        }


        if (!leia1) {
            break;
        }

        strncpy(lat_odd, &uus[54], 17);
        strncpy(lon_odd, &uus[71], 17);


        int cpr_lat_odd1 = strtol(lat_odd, NULL, 2);
        int cpr_lon_odd1 = strtol(lon_odd, NULL, 2);

        cpr_lat_odd = (float)(cpr_lat_odd1 / 131072.0);
        cpr_lon_odd = (float)(cpr_lon_odd1 / 131072.0);

        calc_pos(cpr_lat_even, cpr_lon_even, cpr_lat_odd, cpr_lon_odd,
uus[53], kood, ft);
        altitude(uus, ft);

        break;

        case '1': ;
        // Odd pakett
        char lat_even12[18];
        char lon_even12[18];
        char lat_odd12[18];
        char lon_odd12[18];

        char uus2[113];

        strncpy(lat_odd12, &message[54], sizeof(lat_odd12) - 1);
        strncpy(lon_odd12, &message[71], sizeof(lon_odd12) - 1);

        int cpr_lat_odd12 = strtol(lat_odd12, NULL, 2);
```

```
        int cpr_lon_odd12 = strtol(lon_odd12, NULL, 2);

        cpr_lat_odd = (float)(cpr_lat_odd12 / 131072.0);
        cpr_lon_odd = (float)(cpr_lon_odd12 / 131072.0);

        char algus2[33] = "10001101";
        char *leia2;

        strncpy(algus2 + 8, &message[8], 24);

        int k = 0;

        while (spot + k < filesize - 112) {
            leia2 = strstr(buffer + spot + k, algus2);

            if (!leia2) {
                break;
            }

            strncpy(uus2, leia2, 112);

            result = makeCRC(uus2);
            compare = strcmp(result, "0000000000000000000000000");

            if (compare == 0 && uus2[53] == '0') {
                break;
            } else {
                k++;
            }
        }

        if (!leia2) {
            break;
        }


        strncpy(lat_even12, &uus2[54], 17);
        strncpy(lon_even12, &uus2[71], 17);

        int cpr_lat_even12 = strtol(lat_even12, NULL, 2);
        int cpr_lon_even12 = strtol(lon_even12, NULL, 2);

        cpr_lat_even = (float)(cpr_lat_even12 / 131072.0);
        cpr_lon_even = (float)(cpr_lon_even12 / 131072.0);

        calc_pos(cpr_lat_even, cpr_lon_even, cpr_lat_odd, cpr_lon_odd,
uus2[53], kood, ft);
        altitude(uus2, ft);


        break;
    }

}

void altitude(char *message, FILE* ft) {

    char q_bit;
    char value_char[12];
```

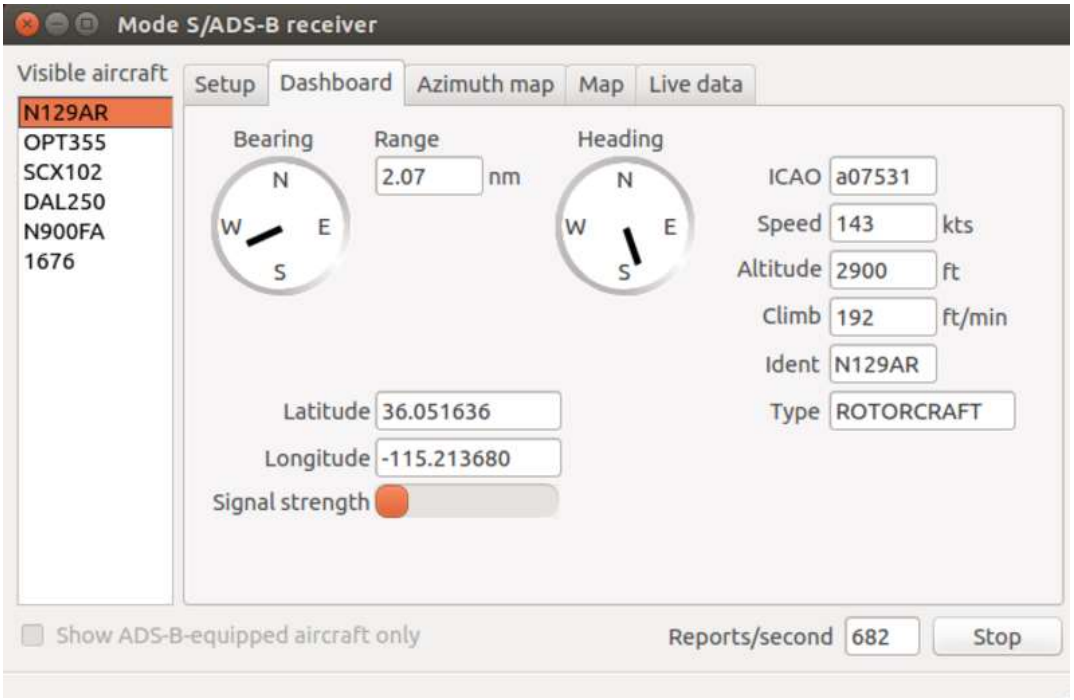
```
    int altitude_value;


    q_bit = message[47];

    strncpy(value_char, &message[40], 7);
    strncpy(value_char + 7, &message[48], 4);

    int value_int = strtol(value_char, NULL, 2);

    if (q_bit == '0')
    {
        altitude_value = (value_int * 100 - 1000) * 0.3048;
    } else {
        altitude_value = (value_int * 25 - 1000) * 0.3048;
    }

    fprintf(ft, "K6rgus - %d meetrit\n", altitude_value);

}

void calc_pos(float even_laius, float even_pikkus, float odd_laius, float
odd_pikkus, char flag, int kood, FILE* ft) {

        int index;


        index = (59 * even_laius - 60 * odd_laius + 0.5);

        int dL_even = 6;
        float dL_odd = 360 / 59;
        float even_lat;
        float odd_lat;

        even_lat = dL_even * ((index % 60) + even_laius);
        odd_lat = dL_odd * ((index % 59) + odd_laius);


        while (even_lat >= 270)
        {
            even_lat = even_lat - 360;
        }

        while (odd_lat >= 270)
        {
            odd_lat = odd_lat - 360;
        }


        int NL_even = NL(even_lat);
        int NL_odd = NL(odd_lat);


        if (abs(NL_even - NL_odd) > 1)
        {
            fprintf(ft, "Message %d:\n", counter);

            counter++;
            fprintf(ft, "Lennuki Mode-S kood: %X\n", kood);
```

```
    fprintf(ft, "Lennuk edastab enda asukohta\n");

    if (flag == '0')
    {
        fprintf(ft, "Laiuskraadid: %0.2f\n", even_lat);
    } else {
        fprintf(ft, "Laiuskraadid: %0.2f\n", odd_lat);
    }

    fprintf(ft, "Pikkuskraadide arvutamine ei ole v6imalik\n");
    return;
}

fprintf(ft, "Message %d:\n", counter);
counter++;
fprintf(ft, "Lennuki Mode-S kood: %X\n", kood);
fprintf(ft, "Lennuk edastab enda asukohta\n");


int ni;
float dLon;
float Lon;
int m;

if (flag == '0')
{
    if (NL_even > 1)
    {
        ni = NL_even;
    } else {
        ni = 1;
    }


dLon = 360 / ni;

m = (even_pikkus * (30 - 1) - odd_pikkus * 30 + 0.5);

Lon = dLon * (abs(m % ni) + even_pikkus);


} else {

    if (NL_odd  > 2)
    {
        ni = NL_odd - 1;
    } else {
        ni = 1;
    }


dLon = 360 / ni;

m = (even_pikkus * (30 - 1) - odd_pikkus * 30 + 0.5);

Lon = dLon * (abs(m % ni) + odd_pikkus);

}
```

```
        fprintf(ft, "Enne - %f\n", Lon);

        if (Lon > 180)
        {
            Lon = 360 - Lon;
        }


    fprintf(ft, "Pikkuskraadid: %f\n", Lon);
}

int NL(float laius) {

    int NL_lat;
    float a, b, c;

    a = pow(cos(PI * laius / 180), 2);
    b = 1 - cos(PI / 30);
    c = acos(1 - b / a);

    NL_lat = 2 * PI / c;

    return NL_lat;

}
```

Joonis: Programmikood.

Lisa 2 – Rakenduste ekraanitõmmised



Tallinn 2017