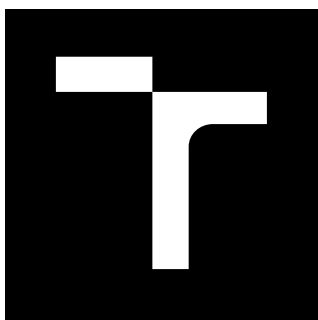


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

SEMESTRÁLNÍ PRÁCE



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV RADIOELEKTRONIKY

DEPARTMENT OF RADIO ELECTRONICS

TELEMETRICKÝ ARCHIV DRUŽIC

SATELLITE TELEMETRY ARCHIVE

SEMESTRÁLNÍ PRÁCE

SEMESTRAL THESIS

AUTOR PRÁCE

AUTHOR

Bc. Péter Tóth

VEDOUCÍ PRÁCE

SUPERVISOR

Ing. Tomáš Urbanec, Ph.D.

BRNO 2017

Semestrální práce

magisterský navazující studijní obor **Elektronika a sdělovací technika**

Ústav radioelektroniky

Student: Bc. Péter Tóth

ID: 100291

Ročník: 2

Akademický rok: 2017/18

NÁZEV TÉMATU:

Telemetrický archiv družic

POKYNY PRO VYPRACOVÁNÍ:

MM2E: Seznamte se se strukturou vysílaných telemetrických dat PSK31 družic PSAT, BRICsat, PSAT-2. Realizujte odstranění dopplerova jevu a proveďte demodulaci signálu z SDR IQ záznamu, vytvořte kostru archivu telemetrických dat a ověřte jeho funkčnost. K archivu vytvořte referenční údaje o poloze družice vůči Zemi a Slunci.

MMSE: Zpracujte celý archiv signálů družic PSAT, BRICsat, PSAT-2 a získaná data synchronizujte s referenčními údaji. Vytvořte vhodný nástroj pro zobrazení údajů z telemetrického archivu. Proveďte analýzu získaných dat všech dostupných veličin a jejich vzájemnou vazbu. Porovnejte získaná data z volně dostupných telemetrických archivů. Získejte maximum dat z obsahu transpondéru družic, doplňte o ně archiv a vyhodnoťte souvislosti s telemetrickými údaji. Vyhodnoťte získané údaje vzhledem k budoucím misím.

DOPORUČENÁ LITERATURA:

[1] Urbanec, T., Vágner, P., Kasal, M. P-sat Transponder WEB Specification [online]. 2015 [cit. 2017-10-5]. Dostupné z:

<http://www.urel.feec.vutbr.cz/esl/files/Projects/PSAT/P%20sat%20transponder%20WEB%20spec02.htm>

[2] Bruninga, B. PSAT - APRS plus a new PSK31 Approach [online]. 2017 [cit. 2017-10-5]. Dostupné z:

<http://aprs.org/psat.html>

Termín zadání: 18.9.2017

Termín odevzdání: 13.12.2017

Vedoucí práce: Ing. Tomáš Urbanec, Ph.D.

Konzultant:

prof. Ing. Tomáš Kratochvíl, Ph.D.
předseda oborové rady

UPOZORNĚNÍ:

Autor semestrální práce nesmí při vytváření semestrální práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

ABSTRAKT

Náplní této práce je vytvoření archivu telemetrických údajů amaterských družic NO-83 BRICsat a NO-84 PSat.

KLÍČOVÁ SLOVA

korekce dopplerova posuvu, NO-83, BRICsat, NO-84, PSat

ABSTRACT

The aim of this thesis is to create an archive of telemetric data of satellites NO-83 BRICsat and NO-84 PSat.

KEYWORDS

doppler shift correction, NO-83, BRICsat, NO-84, PSat

TÓTH, Péter. *Název studentské práce*. Brno, 2018, 37 s. Semestrální projekt. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav radioelektroniky. Vedoucí práce: Ing. Tomáš Urbanec , Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svůj semestrální projekt na téma „Název studentské práce“ jsem vypracoval(a) samostatně pod vedením vedoucího semestrálního projektu a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedeného semestrálního projektu dále prohlašuji, že v souvislosti s vytvořením tohoto semestrálního projektu jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno

.....

podpis autora(-ky)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Tomášu Urbanci, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno

.....

podpis autora(-ky)

OBSAH

Úvod	10
1 Struční přehled amaterských družic	11
2 Sledování a predikce pohybu těles na oběžné dráze Země	12
3 Výpočet rychlosti těles na oběžní dráze Země	13
4 Určení dopplerovského posuvu	14
5 Korekce dopplerovského posuvu	15
6 Demodulace a dekódování signálu	16
7 vytvoření archivu dat TLE	17
8 Korekce dopplerovského posuvu	19
9 Závěr	21
Literatura	22
Seznam symbolů, veličin a zkratk	23
Seznam příloh	24
A Zdrojové kódy	25
A.1 Skript TLE.py	25
A.2 Skript get_undopplered.py	31
A.3 Skript wav2spectrogram.py	35

SEZNAM OBRÁZKŮ

7.1	Vývojový diagram extrahování dat TLE	18
8.1	Náčrt fungování modulu pro korekci dopplerovho posuvu	20

SEZNAM TABULEK

SEZNAM VÝPISŮ

/home/ptoth/Documents/Projekt/github/projekt17/Python3/TLE.py	25
/home/ptoth/Documents/Projekt/github/projekt17/Python3/get_undopplered.py	31
/home/ptoth/Documents/Projekt/github/projekt17/Python3/wav2spectrogram.py	35

ÚVOD

Umělé družice obíhající kolem Země jsou systémy, s kterými poslední fyzické kontakty lidí vznikají těsně před její vypuštěním. Prakticky to znamená praktickou nemožnost provádění jakékoliv činnosti na zařízení v místě působení činnosti umělé družice. Proto hraje důležitou roli při provozování umělých družic dálkový sběr naměřených údajů senzorů, tzv. telemetrických dat, umělých družic za účelem vyhodnocení jejího stavu.

V této práci je vytvářena snaha o automatické zpracování telemetrických dat amatérských umělých družic Parkinson NO-83 a NO-84. Protože tyto družice mají svou dráhu na nízké oběžné dráze, je jejich signál silně postižen dopplerovským posuvem. Pro správnou demodulaci signálu se tento posuv kmitočtu signálu musí korigovat, a kvůli číslíkové povaze modulovaného signálu musí být tato korekce prováděna bez vzniku fázové diskontinuity.

Dělení dokumentu do kapitol a podkapitol je tvořeno ze záměrem porozumění a sledování vzniku řešení zadaného a problému. Ke kompenzaci dopplerovského posuvu kmitočtu signálu musíme znát relativní rychlost družice vzhledem ke pozemní stanici. Pro výpočet této rychlosti musíme znát naši polohu, predikovat pohyb umělé družice a tvar její dráhy. Tyto témata jsou blíže popsány v kapitolách <a neveik, számaik, vagy referencia>. V další kapitole <ref> se věnujeme výpočtu relativní rychlosti umělé družice vzhledem k pozemní stanici a v následující určíme velikost a možnosti korekce dopplerovského posuvu. Poslední kapitolou spadajícího do myšlené skupiny teorie patří demodulaci a dekodování signálu.

V následujících kapitolách se prezentuje konkrétní řešení, včetně vývojových diagramů a popisu funkčnosti jednotlivých skript v jazyce Python 3. Radiokomunikace prošla ohromným vývojem v průběhu druhé polovice 20. století. Vypuštění první umělé družice Sputnik-1 v roce 1957 datujeme začátek vesmírného věku. Od té doby se otevřeli nové možnosti radioamatérů věnovat se svému koníčku, nebo výzkumu.

1 STRUČNÍ PŘEHLED

AMATERSKÝCH DRUŽIC

Radiokomunikace prošla ohromným vývojem v průběhu druhé polovice 20. století. Vypuštění první umělé družice Sputnik-1 v roce 1957 datujeme začátek vesmírného věku. Od té doby se otevřeli nové možnosti radioamatérův věnovat se svému koníčku, nebo výzkumu.

Necelé čtyři roky po vypuštění první umělé družice Země se svět dočkal první amatérské umělé družice sestrojeného v rámci projektu Orbiting Satellite Carrying Amateur Radio (OSCAR) [1], kterého nástupnickou organizací se stal Radio Amateur Satellite Corporation (AMSAT-NA) [2]. Družice OSCAR I byla vypuštěná na nízkou oběžnou dráhu jako druhotný náklad, který vyžíval rezervy nosnosti rakety Thor DM-21 Agena-B. Tento způsob dopravy byl zvolen z ekonomických důvodů a je dodnes používán.

Obecně se amatérské družice nasazují na nízkou oběžnou dráhu (Low Earth Orbit – LEO) [3]. Výhodou dráhy tohoto typu je jejich finanční nenáročnost, co je zčásti způsobená vlastností plynoucího z názvu oběžné dráhy, výškou orbitu, který se pohybuje od 300 km do 2000 km. Nedostatkem nízké oběžné dráhy je vysoká relativní rychlost družice vůči pozemní stanici, která přibližně $7,8 \text{ km/s}$ [4] jehož důsledkem je rádiový signál značně postižen dopplerovským posuvem kmitočtu, kterého velikost dosahuje i 26 ppm.

2 SLEDOVÁNÍ A PREDIKCE POHYBU TĚLES NA OBĚŽNÉ DRÁZE ZEMĚ

Objekty, které se pohybují vesmírem kolem Země jsou sledovány organizacemi [5]:

- United States Strategic Command (USSTRATCOM)(součást Department of Defense (DoD))
- European Space Agency (ESA)
- Fraunhofer-Institut für Hochfrequenzphysik und Radartechnik (Fraunhofer-FHR)
- Jet Propulsion Laboratory (JPL)(součást National Aeronautics and Space Administration (NASA))
- Massachusetts Institute of Technology (MIT)
- European Incoherent Scatter Scientific Association (EISCAT)
- United States Air Force (USAF)

Ke sledování se používají pozemní radary, lidary, pozemní a vesmírné teleskopy. Mezi sledované objekty patří umělé družice, pozůstatky raket a jiný vesmírný odpad. Nejrozsáhlejší katalog stavu družic udržuje Ministerstvo obrany Spojených států (DoD) s názvem Space Object Catalog. Civilní varianta této databáze je provozována organizací NASA. Tyto databáze se udržují pomocí různých modelů orbitální mechaniky. Pohyby družic jsou analyticky vypočteny pomocí teorie všeobecných perturbací. Prvky dráhy této teorie jsou publikovány ve formátu NASA/NORAD two-line elements (TLE). [6].

Proč se zabývat přesnou polohou družice? Aby jsme byly schopní provést korekci dopplerovského posuvu, musíme splnit několik požadavek výpočtu:

- polohu pozemní stanice ¹
- polohu a rychlost umělé družice

¹pozemní stanici považujeme za stacionární vzhledem ke povrchu Země

3 VÝPOČET RYCHLOSTI TĚLES NA OBĚŽNÍ DRÁZE ZEMĚ

TBD...

4 URČENÍ DOPPLEROVSKÉHO POSUVU

TBD...

5 KOREKCE DOPPLEROVSKÉHO POSUVU

TBD...

6 DEMODULACE A DEKÓDOVÁNÍ SIGNÁLU

TBD...

7 VYTVOŘENÍ ARCHIVU DAT TLE

Vytvoření lokální databázi dat TLE je nezbytné pro správnou korekci dopplerovského posuvu. Pro umělé družice NO-83 a NO-84 jsou veřejně přístupné data TLE na stránce organizace AMSAT-NA: <<http://amsat.org/pipermail/keps/>>. Tyto data jsou distribuované formou elektronického mailing list, kterého archiv se nachází na výše zmíněném URL v jednom souboru. Aby bylo možné použít údaje TLE obsáhnuty v tomto archivním souboru, je nutné provést extrakci dat TLE dle datu jejího vzniku.

Na obrázku 7.1 je uveden vývojový diagram skriptu pro extrahování dat TLE. Jako povinný vstupní parametr je název souboru staženého z webové stránky organizace AMSAT-NA. Data TLE jsou posílány v jednom emailu. Skript identifikuje začátek i konec balíků dat TLE. Na začátku každého balíku se hledí vzory:

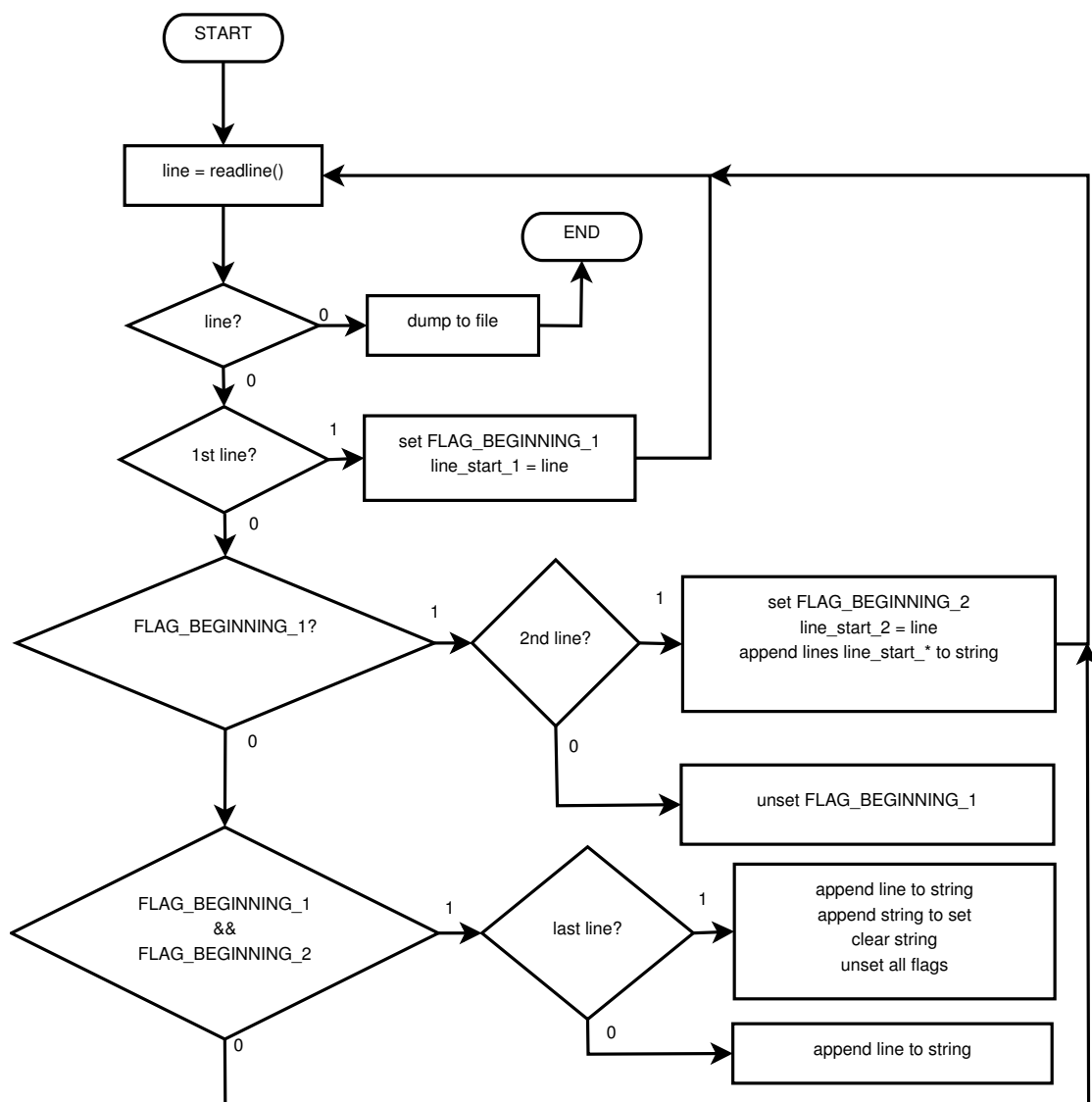
```
SB\s+KEPS\s+@\s+AMSAT\s+$ORB\d5\.[A-Z]
```

```
^2Line
```

pomocí nástroje na vyhledávání regulárních výrazů implementovaného modulem re ze standardní knihovny jazyka Python. Konec balíka je značen řetězcem `\EX`.

V případě, že skript narazí na hledaný výraz, nastaví se příslušný příznak. Tyto příznaky zaručí, aby řádky jednoho balíku se připojili k jednomu řetězci. Když skript identifikuje poslední řádek, pospojovaný řetězec řádků se připojí k množině všech balíků TLE dat. Návrátovou hodnotou je právě tato množina. Množina v jazyce Python 3 zaručuje jedinečnost všech prvků, obdobně jako množiny z teorie množin.

K výpisu souboru patří funkce `dump_to_file` skriptu `TLE.py`. Tato funkce má jeden povinný parametr, množinu dat TLE. Z této množiny se načte každý jeden prvek. V těchto prvcích se prohledává datum vzniku, což se použije jako název souboru. Následně se dle potřeby vytvoří adresář, kterého název je rok vzniku TLE dat. Před vytvářením adresáře se kontroluje přítomnost adresáře, nebo souboru stejného jména, jaký se chystá být vytvořit. Jestli adresář se stejným jménem existuje, začne se de ní zapisovat. V opačném případě předpokládáme, že v adresáři je soubor se stejným jménem musíme název složky, kterou se chystáme vytvořit změnit z důvodu nemožnosti koexistence souboru a adresáře stejného jména v rodičovském adresáři. K názvu složky se přidává znak podtržítka a číslo, které se iteruje inkrementací do doby, kdy v rodičovském se nebude nacházet soubor se stejným jménem. V případě adresáře se stejným jménem nalezeného po přidání dodatečných znaků k názvu souboru, se adresář začne používat k uložení souborů. Čtení z množiny je realizováno pomocí iterací nad její prvky.



Obr. 7.1: Vývojový diagram extrahování dat TLE

8 KOREKCE DOPPLEROVSKÉHO POSUVU

Po úspěšném získání nutných souborů s daty TLE pro umělé družice našeho zájmu, můžeme přistoupit ke korekci dopplerovského posuvu signálu, kterou provedeme pomocí programu `doppler` volně dostupného z repozitáře GIT týmu `codehub` dostupného na URL `<https://github.com/codehub>`.

Program `doppler` je utilitou příkazového řádku, který ze standardního vstupu čte IQ (in-phase, quadrature) data a zpracovává dle zadaných parametrů. Rozlišujeme dva režimy korekce frekvenčního posuvu. V režimu *const* se kompenzuje konstantní posuv kmitočtu, kým v režimu *track* se provádí korekce sledováním pohybu umělé družice a to i dodatečně v případě zpracování předem zaznamenaných dat IQ. V pozdním případě se programu `doppler` musí předat argument datu a času záznamu ve formátu ISO 8601 [8] [9] bez udání časového posuvu v čase UTC.

Korekce dopplerovského posuvu se provádí programem `doppler` za pomoci volně dostupné knihovny `libgpredict`, který je založen na predikčním kódu programu `Gpredict`[?]

Aby se zjednodušilo zpracování velkého množství souborů, byl vytvořen skript v jazyce Python 3 pro automatické zpracování záznamů s IQ daty. Modul `get_undopplred` má definovanou funkci `undoppler_it`, který jako vstupní parametry má:

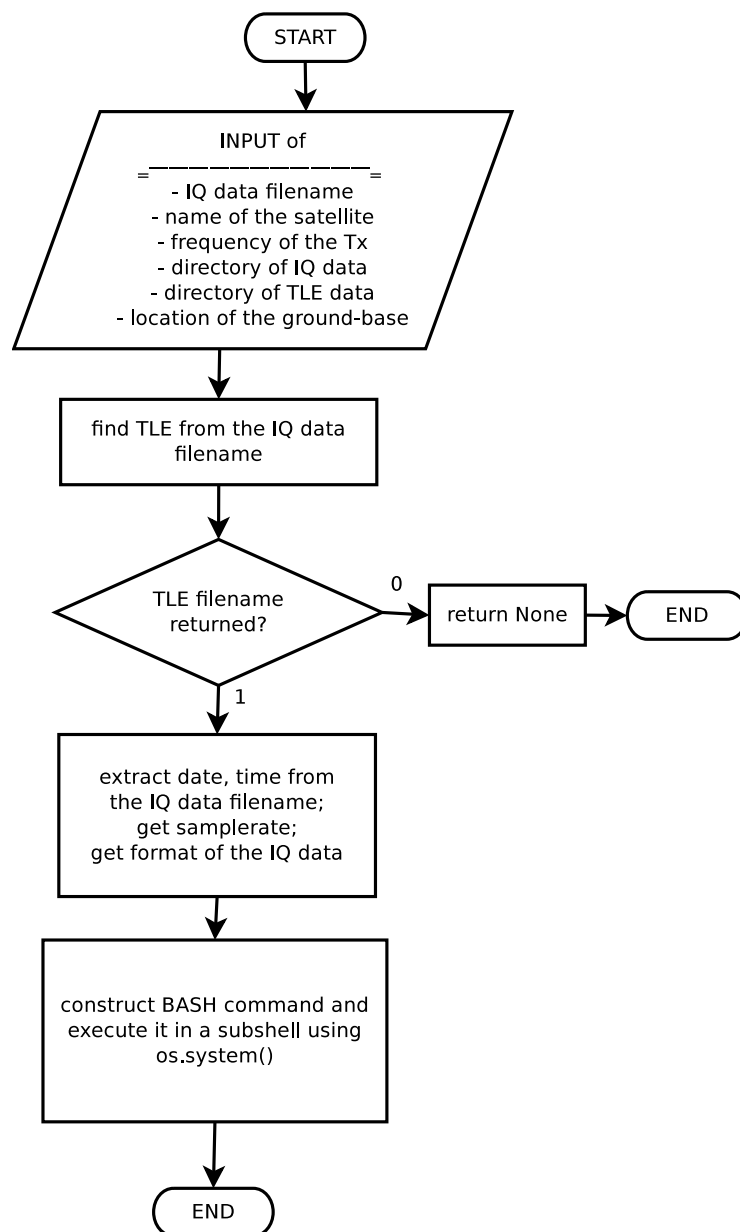
- název souboru IQ dat
- název družice v notaci OSCAR¹
- kmitočet na kterém je z družice vysíláno
- adresář s IQ daty
- adresář s TLE daty
- lokace pozemní stanice

Pomocí těchto údajů se sestojí řetězec obsahující příkaz shellu BASH, kde jednotlivé příkazy jsou řazeny do tzv. kolony. Jde o zřetězení příkazů oddělených meta-znakem svislá čára `']`. Standardní výstup příkazu se předává standardnímu příkazu následujícímu. [11]

Parametry záznamu IQ dat se zjišťují pomocí modulu `pymediainfo`, který je tzv. wrapper function knihovny `Mediainfo` [12]. Pomocí tohoto modelu lze zjistit kmitočet vzorkování, bitovou hloubku, kódování, počet kanálů, kodek.

Aby jsme mohli soubory formátu Waveform Audio File Format (WAVE) použít jako vstupní data pro program `doppler`, je nutné provést změnu formátu dle očekávání programu. K této úloze se použije volný program `Sound eXchange (SoX)`. Podobně na výstupu lze použít `SoX` pro převod z RAW Audio formátu na WAVE.

¹v našem případě se jedná o NO-83, NO-84



Obr. 8.1: Náčrt fungování modulu pro korekci dopplerovho posuvu

9 ZÁVĚR

TBD...

LITERATURA

- [1] *Amateur radio satellite*. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2017-12-10]. Dostupné z URL: https://en.wikipedia.org/wiki/Amateur_radio_satellite.
- [2] *Amateur radio satellite*. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2017-12-10]. Dostupné z URL: <https://en.wikipedia.org/wiki/AMSAT>.
- [3] PUBLISHED BY AMERICAN RADIO RELAY LEAGUE. *The ARRL handbook for radio communications 2011*. 88th ed. Newington, CT: American Radio Relay League, 2010. ISBN 9780872590953.
- [4] *Low Earth orbit*. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2017-12-10]. Dostupné z URL: https://en.wikipedia.org/wiki/Low_Earth_orbit.
- [5] *Space derbis*. In: Wikipedia: the free encyclopedia [online]. Dostupné z URL: https://en.wikipedia.org/wiki/Space_debris.
- [6] *United States Space Surveillance Network*. In: Wikipedia: the free encyclopedia [online]. Dostupné z URL: https://en.wikipedia.org/wiki/United_States_Space_Surveillance_Network.
- [7] *Two-line element set*. In: Wikipedia: the free encyclopedia [online]. Dostupné z URL: https://en.wikipedia.org/wiki/Two-line_element_set.
- [8] *ISO 8601*. In: Wikipedia: the free encyclopedia [online]. Dostupné z URL: https://en.wikipedia.org/wiki/ISO_8601.
- [9] *Doppler documentation*. [online]. Dostupné z URL: <https://github.com/cubehub/doppler>.
- [10] *Libgpredict documentation*. [online]. Dostupné z URL: <https://github.com/cubehub/libgpredict>.
- [11] BRANDEJS, Michal. *UNIX - Linux: praktický průvodce*. Praha: Grada, 1996. ISBN 80-7169-170-4.
- [12] *Pymediainfo github repository*. [online]. Dostupné z URL: <https://github.com/sbraz/pymediainfo>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

AMSAT-NA	Radio Amateur Satellite Corporation
OSCAR	Orbiting Satellite Carrying Amateur Radio
USSTRATCOM	United States Strategic Command
DoD	Department of Defense
ESA	European Space Agency
Fraunhofer-FHR	Fraunhofer-Institut für Hochfrequenzphysik und Radartechnik
TIRA	Tracking & Imaging Radar
NASA	National Aeronautics and Space Administration
JPL	Jet Propulsion Laboratory
GDSCC	Goldstone Deep Space Communications Complex
MIT	Massachusetts Institute of Technology
EISCAT	European Incoherent Scatter Scientific Association
USAF	United States Air Force
TLE	two-line elements
SGP4	Simplified perturbations models
UTC	Coordinated Universal Time
WAVE	Waveform Audio File Format
SoX	Sound eXchange

SEZNAM PŘÍLOH

A	Zdrojové kódy	25
A.1	Skript TLE.py	25
A.2	Skript get_undopplered.py	31
A.3	Skript wav2spectrogram.py	35

A ZDROJOVÉ KÓDY

A.1 Skript TLE.py

Python 3 modul pro extrahování TLE dat.

```
1 #!/bin/python3
2
3 import re
4 import datetime
5 import os
6
7 #SB KEPS @ AMSAT $ORB06243.N
8 #2Line Orbital Elements 06243.AMSAT
9 PATTERN_START_1 = re.compile(r'SB\s+KEPS\s+@\s+AMSAT\s+\
   $ORB\d{5}\.[A-Z]')
10 PATTERN_START_2 = re.compile(r'^2Line')
11 #1 28897U 05043H 06249.26374724 .00000138 00000-0
   38823-4 0 1152
12 #2 28897 098.1525 146.2174 0016539 282.7486 077.1854
   14.59597696 37959
13 #/EX
14 PATTERN_END = re.compile(r'^\s+/EX')
15 #FROM WA5QGD FORT WORTH,TX August 31, 2006
16 PATTERN_DATE = re.compile(r'(January|February|March|
   April|May|June|July|August|September|October|November|
   December)\s+(\d{1}|\d{2})\s,\s+(\d{4})') # Patter to
   search the date inside the extracted TLE chunk. The
   regular expresion is split to three groups as the
   date is writen in the US format: group 1: Month;
   group 2: day (one or two digits); group 3: year (four
   digits)
17 FLAG_BEGINNING_1 = 1 << 0
18 FLAG_BEGINNING_2 = 1 << 1
19 FLAG_ENDING = 1 << 2
20
21
22 # Setting flags on a register passed
23 # register is the register to modify
```

```

24 # multiple flags can be passed
25 def set_flag(register, *flag):
26     """set_flag(register, flag_1, flag_2, ...)
27
28     Setting one or more flags in a register passed as an
        argument.
29     If no flag is passed the register is not going to be
        modified.
30     """
31     if flag:
32         for f in flag:
33             register |= f
34         return register
35     else:
36         # An empty list has been passed
37         return register
38
39 # Unsetting flags on a register passed
40 # register is the register to modify
41 # multiple flags can be passed
42 def unset_flag(register, *flag):
43     """unset_flag(register, flag_1, flag_2, ...)
44
45     Unsetting one or more flags in a register passed as an
        argument.
46     For multiple flags passed they all need to be set in
        the registry
47     in order to achive True to be returned.
48     If no flag is passed the register is not going to be
        modified and
49     it is being returned as it was passed.
50     """
51     if flag:
52         for f in flag:
53             register &= ~f
54         return register
55     else:
56         # An empty list has been passed.
57         return register

```

```

58
59 # Checking flags on a register passed
60 # register is the register to be checked.
61 # Multiple flags can be passed
62 def check_flag(register, *flag):
63     """check_flag(register, _flag_1, _flag_2, ...)
64
65     Checking one or more flags in a register passed as an
        argument.
66     If no flag is passed the register 0 is being returned
        .
67     """
68     flags = 0
69
70     if flag:
71         for f in flag:
72             flags |= f
73         return bool((register & flags) == (register))
74     else:
75         return 0
76
77 # Checking if the passed line matches the passed pattern
        .
78 def check_line(pattern, line_to_check):
79     """check_line(pattern, _line_to_check)
80
81     Checking if the passed line matched the REGEX pattern.
82     """
83     return bool(re.search(pattern, line_to_check))
84
85 # Appending the line to a string.
86 def append_line(string, *lines_to_append):
87     """append_line(string, _*lines_to_append)
88
89     Appending lines passed to the function to the string
90     """
91     for line in lines_to_append:
92         string += line
93

```

```

94     return string
95
96     # Dumping a set of TLEs files
97     def dump_to_file(set_of_TLEs, directory='../keps/'):
98         '''dump_to_file(set_of_TLEs, [directory='../keps/'])
99
100         The function takes a set of TLEs extracted from the
101         AMSAT e-mail list.
102         Each member of the set is written to a file, which name
103         contains the
104         date of the TLE file. In case of multiple files per day
105         , the last one
106         will be written out overwriting the files written
107         before.
108         The function checks the directory if it is available at
109         the CWD. If it
110         exists, then the file is written there. If a file named
111         that exists,
112         a new folder is being created.
113         Probably the function will be able to handle not just
114         set as parameter,
115         but another iterable types. It was not tested due to
116         shortage of available
117         time.
118         Optional directory parameter can be passed, which can
119         be either
120         relative or absolute.
121
122         '''
123         for element in set_of_TLEs:
124             date_string = re.search(PATTERN_DATE, element)
125             if date_string == None:
126                 return None
127             else:
128                 date = datetime.datetime.strptime(date_string.group
129                     (0), '%B%d,%Y')
130                 filename = date.strftime('%Y-%m-%d') + '.amsat.tle
131                     ,
132                 directoryname = date.strftime('%Y')

```

```

122
123     i = 0          # iteration for the directory name
124
125     while True:
126         if os.path.isdir(directoryname):
127             full_path = os.path.join(directory, directoryname
128                                     , filename)
129             break
130         else:
131             if os.path.exists(directoryname):
132                 if directoryname.find('_') == -1:
133                     directoryname += '_' + '%02d' %(i)
134                 else:
135                     directoryname[:directoryname.rfind('_')] += '_' + '%02d' %(i)
136                 i += 1
137             else:
138                 full_path = os.path.join(directory,
139                                         directoryname, filename)
140
141             os.makedirs(os.path.join(directory, directoryname),
142                         exist_ok=True)
143
144             tle_file = open(full_path, 'w')
145             tle_file.writelines(element)
146             tle_file.close()
147
148 # Function to look up TLE in the AMSAT mailing list
149 # archive.
150 #
151 def extract_amsat_TLE(AMSAT_maillist_filename, directory=
152     '../keps/'):
153     '''extract_amsat_TLE(AMSAT_maillist_filename, [
154         directory='../keps/'])
155     The mail list file is from http://amsat.org/pipermail/
156     keps/.
157     The filename is just the year represented by four
158     digits,
159     a period and string "txt" (i.e. 2006.txt).

```

```

152  Optional directory parameter can be passed, which can
      be either
153  relative or absolute.
154  '''
155  f = open(os.path.join(directory,
      AMSAT_maillist_filename), 'r')
156  f_register = 0
157  string_TLE = ''
158  set_of_string_TLE = set()
159  line = f.readline()
160
161
162  while line:
163      if check_line(PATTERN_START_1, line):
164          f_register = set_flag(f_register, FLAG_BEGINNING_1)
165          line_start_1 = line
166          line = f.readline()
167          continue
168      elif check_flag(f_register, FLAG_BEGINNING_1):
169          if check_line(PATTERN_START_2, line):
170              f_register = set_flag(f_register,
                  FLAG_BEGINNING_2)
171              line_start_2 = line
172              line = f.readline()
173              string_TLE = append_line(string_TLE, line_start_1
                  , line_start_2)
174              continue
175          else:
176              f_register = unset_flag(f_register,
                  FLAG_BEGINNING_1)
177              line = f.readline()
178              continue
179      else:
180          if check_flag(f_register, FLAG_BEGINNING_1,
              FLAG_BEGINNING_2):
181              if check_line(PATTERN_END, line):
182                  string_TLE = append_line(string_TLE, line)
183
184                  set_of_string_TLE.add(string_TLE)

```

```

185
186         string_TLE = ''
187         f_register = unset_flag(f_register,
                                FLAG_BEGINNING_1, FLAG_BEGINNING_2,
                                FLAG_ENDING)
188         line = f.readline()
189     else:
190         string_TLE = append_line(string_TLE, line)
191         line = f.readline()
192
193     return set_of_string_TLE

```

A.2 Skript get_undopplered.py

Python 3 modul pro korekci dopplerovského posuvu.

```

1 import re
2 import datetime, pytz
3 import os
4 import pymediainfo
5
6 # Function to get some information from the IQ sample
   filename.
7 def extract_IQ_filename(filename, directory='../NO-84'):
8     '''extract_IQ_filename(filename, [directory='../NO
   -84'])
9
10    This function is extracting the filename of which
   pattern
11    will be the same in the future, or the REGEX needs to
   be changed.
12    The function will return a dictionary conaning:
13    *date//YYYYMMDD//
14    *time//HHMMSS//
15    *frequency//CCMM//[kHz]
16    *extension//re.(\w{3})//e.g. wav
17    If the file is not ending with 'wav', then instead of a
   dictionary
18    None is returned.

```



```

19  '''
20  # How the regex is built -- hint from a filename.
21  #
22      HDSDR_      20160126 _
23      205400  Z _      435320 kHz_ RF .
24      wav
25
26  pattern_named = re.compile(r'HDSDR_(?P<date>\d{8})_(?P<
27      time>\d{6})_*_(?P<frequency>\d{6})kHz_(RF)\.(?P<
28      extension>\w{3})$') # REGEX pattern with named
29      subgroups date, time, frequency, extension
30
31  filename_regexed_named = re.search(pattern_named,
32      filename)
33
34  file_parameters = filename_regexed_named.groupdict()
35
36  if file_parameters['extension'] == 'wav': # filter
37      out non-wav files
38
39  return file_parameters
40
41  else:
42
43  return None
44
45
46  # A little 'key' function for sorting TLE files with
47  sorted()
48
49  def key_function_TLE(date_IQ, date_TLEs):
50
51  return abs(date_IQ - date_TLEs)
52
53
54  def find_TLE_for_IQ_file(filename_IQ, directory_TLE='../
55  keps/'):
56
57  '''find_TLE_for_IQ_file(filename_IQ, [directory_TLE
58  = '../keps/'])
59
60
61  This function is returning the best matching TLEs file
62  based on the IQ filename given and the date encoded in
63  it.
64
65  The best match means the least difference between the
66  dates
67
68  indicated by the names of the TLEs and IQ files.
69
70  '''
71
72  parameters_IQ = extract_IQ_filename(filename_IQ)
73
74  if parameters_IQ != None:
75

```

```

46     date_IQ = datetime.datetime.strptime(parameters_IQ.
        get('date'), '%Y%m%d')
47     directory_path_TLEs = os.path.join(directory_TLE, str
        (date_IQ.year))
48     pattern_TLE_file = re.compile('%d-%02d-' % (date_IQ.
        year, date_IQ.month))
49     TLEs = list()          # Empty list for TLEs file
50
51     for i in os.listdir(directory_path_TLEs):
52         if bool(re.search(pattern_TLE_file, i)):
53             TLEs.append(i)
54     TLEs = sorted(TLEs, key=lambda TLE: abs(date_IQ -
        datetime.datetime.strptime(TLE[:10], '%Y-%m-%d')))
55
56     return TLEs[0]
57 else:
58     return None
59
60 # Function used for undoing the doppler frequency shift
using the doppler application called by os.system
61 def undoppler_it(filename_IQ, satellite_name='NO-84',
    satellite_frequency='435350000', directory_IQ='../NO
    -84', directory_TLE='../keps', location_SDR='lat
    =49.173238,lon=16.961292,alt=263.73'):
62     '''undoppler_it(filename_IQ,[satellite_name='NO-84',[
        satellite_frequency='435350000',[directory_IQ='../NO
        -84',[directory_TLE='../keps',[location_SDR='lat
        =49.173238,lon=16.961292,alt=263.73']]):
63
64     Only one parameter is required not to get an error, but
        not all time
65     will it make sense. It is due to the fact that the
        optional parameters
66     are used only for the purpose of the testing.
67     '''
68     filename_TLE = find_TLE_for_IQ_file(filename_IQ,
        directory_TLE)
69
70     if filename_TLE != None:

```

```

71     parameters_IQ = extract_IQ_filename(filename_IQ)
72     mediainfo_IQ = pymediainfo.MediaInfo.parse(os.path.
        join(directory_IQ, filename_IQ))
73     mediainfo_IQ_tracks = mediainfo_IQ.tracks
74     audio = mediainfo_IQ_tracks[1]
75     audio_dict = audio.to_data()
76
77     local_TZ = pytz.timezone('Europe/Prague')
78     naive_datetime_IQ = datetime.datetime.strptime(
        parameters_IQ.get('date') + parameters_IQ.get('
        time'), '%Y%m%d%H%M%S')
79     local_datetime_IQ = local_TZ.localize(
        naive_datetime_IQ, is_dst=None)
80     utc_datetime_IQ = local_datetime_IQ.astimezone(pytz.
        utc)
81
82     samplerate = '--samplerate_' + str(audio_dict.get('
        sampling_rate')) + '_'
83     if (int(audio_dict.get('resolution')) == 16) & (
        audio_dict.get('format_settings') == 'Little_/
        Signed'):
84         intype = '--intype_i16_'
85         outtype = '--outtype_i16_'
86     else:
87         #in case of strange Wave format, e.g
88         . float samples
89         print('Format_ yet_ not_ tested: ')
90         print('_*_resolution: ', audio_dict.get('
            resolution'))
91         print('_*_format_settings:', audio_dict.get('
            format_settings'))
92         return 1
93     tlefile = '--tlefile_' + os.path.join(
        directory_TLE, filename_TLE[:4], filename_TLE) + '
        _'
94     tlename = '--tlename_' + satellite_name
95         + '_'
96     location = '--location_' + location_SDR
97         + '_'

```

```

94     frequency = '--frequency_' + satellite_frequency
           + '_'
95     time      = '--time_' + local_datetime_IQ.
           isoformat()[:-6] + '_'
96     infile    = os.path.join(directory_IQ, filename_IQ)
           + '_'
97     output    = '_>_'
98     for i in filename_IQ.split('.')[:-1]:
99         output += i
100    output     += '.UD.' + filename_IQ.split('.')[-1]
101
102    sox_cmd_rbche = '--rate_' + str(audio_dict.get('
           sampling_rate')) + '_--bits_16_--channels_2_--
           encoding_signed-integer_'
103    sox_cmd_out = 'sox_' + sox_cmd_rbche + '_--type_raw_-
           ' + sox_cmd_rbche + '_--type_wav_'
104    sox_cmd_inp = 'sox_' + sox_cmd_rbche + '_--type_wav_'
           + infile + sox_cmd_rbche + '_--type_raw_'
105
106    doppler_cmd = 'doppler_track_' + samplerate + intype +
           outtype + tlefile + tlename + location +
           frequency + time
107
108    full_command_doppler = sox_cmd_inp + '_|_' +
           doppler_cmd + '_|_' + sox_cmd_out + output
109
110    os.system(full_command_doppler)
111    print(full_command_doppler)
112
113    else:
114        return None

```

A.3 Skript wav2spectrogram.py

Python 3 modul pro tvorbu spectrogramů.

```

1 import matplotlib
2 matplotlib.use('Agg')
3 import numpy as np

```

```

4 import scipy.signal as signal
5 import numpy as np
6 import scipy.signal as signal
7 from scipy.io import wavfile
8 import matplotlib.pyplot as plt
9 from matplotlib.ticker import FuncFormatter
10
11 # Formatting function used for the purpos of plotting
   with MHz
12 def format_mega(x, pos):
13     if x >= 1e6:
14         return '%3.2f' % (x/1e6)
15     # return x/1e6
16     else:
17         return x
18 # Little function to join a list of strings into one
   string
19 def join(inp_list):
20     out_string = ''
21     for i in inp_list:
22         out_string += i
23     return out_string
24
25 def IQ_to_spectrogram(filename_IQ, Fc=435320000):
26     rate, data = wavfile.read('
        HSDR_20160121_003353Z_435320kHz_RF.UD.local.wav',
        mmap=False)
27     data_cmpl = data.view(np.int16).astype(np.float32).view
        (np.complex64) # Changing the data type to complex
        and
28     data_cmpl = data_cmpl.reshape(data_cmpl.shape[0] *
        data_cmpl.shape[1]) # Reshaping to fit the specgram
29
30     cmap = plt.get_cmap('spectral')
31     function_formatter = FuncFormatter(format_mega)
32     vmin = 10 * np.log10(np.max(np.abs(data_cmpl))) - 40
33
34     color_legend, spectrogram = plt.subplots()

```

```

35     Sxx, f, t, cb = spectrogram.specgram(data_cmpl, NFFT
      =2048, Fs=rate, Fc, vmin=-18, sides='onesided')
36     spectrogram.set_ylabel('f [MHz]')
37     spectrogram.yaxis.set_major_formatter(
      function_formatter)
38     spectrogram.set_xlabel('t [s]')
39 #     ax.ylim(-rate/2, rate/2)
40     color_legend.colorbar(cb)
41     plt.savefig(join(filename.split('.')[0])+'.png', dpi
      =600, bbox_inches='tight', pad_inches=0.5)
42     plt.close()

```