
# VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

## FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

## ÚSTAV RADIOELEKTRONIKY
DEPARTMENT OF RADIOENGINEERING


# TELEMETRICKÝ ARCHIV DRUŽIC
SATELLITE TELEMETRY ARCHIVE

**DIPLOMOVÁ PRÁCE**
MASTER'S THESIS


**AUTOR PRÁCE** — **Bc. Jan Vorálek**
AUTHOR

**VEDOUCÍ PRÁCE** — **Ing. Tomáš Urbanec, Ph.D.**
SUPERVISOR


**BRNO 2020**

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

# Diplomová práce

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

Ústav radioelektroniky

***Student:*** Bc. Jan Vorálek ***ID:*** 155261
***Ročník:*** 2 ***Akademický rok:*** 2019/20

**NÁZEV TÉMATU:**

## Telemetrický archiv družic

**POKYNY PRO VYPRACOVÁNÍ:**

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.

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:*** 3.2.2020 ***Termín odevzdání:*** 28.5.2020

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

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

# ABSTRAKT

Práce se zabývá návrhem archivu vysílaných telemetrických dat družic PSAT, PSAT-2 a BRICSat, která je třeba získat z SDR IQ záznamů. Součástí práce je teoretický popis Dopplerova jevu a struktury vysílaných telemetrických dat. Dále je zde popsán návrh programu pro odstranění Dopplerova jevu, demodulaci a dekódování těchto záznamů a uložení dat do telemetrického archivu. Práce se dále zabývá analýzou získaných dat.

# KLÍČOVÁ SLOVA

telemetrický archiv, telemetrie, družice, PSAT, PSAT-2, BRICSat, Dopplerův jev, SDR, softwarově definované rádio, GNU Radio, PSK31, demodulace, SSTV

# ABSTRACT

This thesis deals with a design of telemetry archive of PSAT, PSAT-2 and BRICSat sattelites. This telemetry data need to be extracted from SDR IQ records. The thesis contains a Doppler effect theory and description of structure of telemetry data. Then it presents a design of a program for Doppler effect correction, demodulation and decoding of these records and saving the data to telemetry archive. Thesis also deals with analysis of decoded data.

# KEYWORDS

telemetry archive, telemetry, sattelite, PSAT, PSAT-2, BRICSat, Doppler effect, SDR, software defined radio, GNU Radio, PSK31, demodulation, SSTV

VORÁLEK, Jan. Telemetrický archiv družic. Brno, 2019. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/120613. Semestrální práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav radioelektroniky. Vedoucí práce Tomáš Urbanec.

# PROHLÁŠENÍ

Prohlašuji, že svůj semestrální projekt na téma Telemetrický archiv družic jsem vypracoval 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 uvedeného semestrálního dále prohlašuji, že v souvislosti s vytvořením tohoto semestrálního projektu jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících 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.

V Brně dne ..... .....

(podpis autora)

# OBSAH

# ÚVOD

Satelity PSAT, PSAT-2 a BRICsat jsou malé satelity typu Cubesat o velikosti 1,5U. Byly vyvinuty studenty Námořní akademie Spojených států amerických, na vývoji BRICsat se podílela i Univerzita George Washingtona [1],[2],[3]. Jejich účelem je mj. provoz PSK31 transpondérů, které byly vyvinuty Laboratoří experimentálních družic Ústavu Radioelektroniky FEKT VUT v Brně [4],[5],[6]. Transpondéry vysílají vlastní telemetrický PSK31 signál obsahující informace o provozních podmínkách satelitů. Cílem práce je zpracovat záznamy rádiového signálu transpondéru pořízené pomocí softwarově definovaného rádia, získat z nich vysílaná telemetrická data, uložit je do archivu a provést jejich analýzu. Z těchto dat se vychází při správě činnosti těchto satelitů [7]. Mohou také poskytnout cenné informace, které mohou být využity při vývoji budoucích satelitů.

# 1 ROZBOR ZADÁNÍ

Cílem práce je vytvořit archiv vysílaných telemetrických dat PSK31 družic PSAT, PSAT-2 a BRICsat. Archiv je třeba doplnit údaji o datu a čase vysílání, poloze družice vůči Zemi a Slunci a případně dalšími údaji užitečnými při následné analýze těchto dat. Telemetrická data je třeba získat ze záznamů rádiového signálu pořízených pomocí softwarově definovaného rádia. Těchto záznamů je velké množství, je tedy třeba proces jejich zpracování co nejvíce automatizovat.

Záznamy mají podobu komplexního IQ signálu. Jsou uloženy ve zvukovém formátu WAV s dvěma kanály, první kanál obsahuje synfázní složku signálu, druhý kanál kvadraturní složku signálu.

Část záznamů byla pořízena na VUT v Brně. Vzorkovací frekvence těchto záznamů je obvykle 96000 nebo 111111 Hz, u některých souborů ale i větší. Soubory byly vytvořeny v programu HDSDR a mají jednotný formát názvu souboru, který obsahuje datum a čas pořízení záznamu a frekvenci přijímače. Toho lze využít k odvození data a času vysílání telemetrie. Název souborů má tvar

HDSDR_RRRRMMDD_hhmmssZ_fkHz_RF.wav,

kde *RRRR* je rok, *MM* je měsíc a *DD* je den pořízení záznamu, *hh* představuje hodiny, *mm* minuty a *ss* vteřiny času pořízení záznamu, *f* je frekvence přijímače v kilohertzích. K příjmu signálu družice PSAT-2 bylo použito dvojí směšování, přičemž první mezifrekvenční kmitočet je vyšší, než přijímaný kmitočet, čímž dochází k inverzi kmitočtového spektra.

Dále je třeba zpracovat vybrané záznamy získané z internetových stránek *satnogs.org*, jejich název obsahuje unikátní identifikační číslo a někdy i datum a čas pořízení záznamu. Vzorkovací frekvence je u těchto souborů obvykle 48kHz.

U signálu dochází k výraznému posuvu frekvence vlivem Dopplerova jevu a je nutné jej odstranit. Mimo to se u signálu projevuje frekvenční offset v řádu jednotek kHz. Ten se výrazně mění v případě družice PSAT-2, kde je způsoben velkým teplotním driftem oscilátoru vysílače.

Telemetrický archiv by měl mít podobu textových, lidsky čitelných souborů. Jako ideální se jeví formát CSV, jelikož je určen pro ukládání tabulkových dat a je podporován mnoha programy.

# 2 STRUKTURA TELEMETRICKÝCH DAT

Satelity PSAT, BRICSat i PSAT-2 jsou vybaveny PSK31 transpondérem, který přijímá v pásmu 28 MHz a vysílá v pásmu 435,350 MHz. Telemetrická data mají podobu krátké textové zprávy, která je jako PSK31 signál přidána k signálu transpondéru [4],[5],[6].

## 2.1 PSK31

PSK31 je digitální mód určený pro přenos psaného textu v reálném čase. Přenosová rychlost je 31,25 Bd, což umožňuje přenos textu rychlostí, která přibližně odpovídá rychlosti psaní na klávesnici [8]. PSK31 signál se vyznačuje velmi malou šířkou pásma (asi 60 Hz pro pokles 26 dB [9]), což umožňuje přenos mnoha nezávislých konverzací v úzkém pásmu [8]. Koncentrace energie signálu v takto úzkém pásmu také zvyšuje odolnost proti rušení a snižuje nároky na výkon vysílače [9].

Jednotlivé znaky jsou reprezentovány sekvencemi bitů různé délky, které se označují jako tzv. varicode abeceda. Každý znak začíná a končí bitem s úrovní 1 a nemůže obsahovat po sobě jdoucí bity s úrovní 0, neboť dvojice bitů s úrovní 0 se používá pro oddělení jednotlivých znaků. Toto řešení umožňuje snadnou synchronizaci na straně přijímače. Jeden znak je vyjádřen jedním až deseti bity. Znakům s velkou četností výskytu v anglickém jazyce jsou přiřazeny sekvence s malým počtem bitů [9].

PSK31 používá diferenciální dvojstavovou modulaci s klíčováním fázovým posuvem (DBPSK). Bit s úrovní 0 je vyjádřen změnou fáze nosné vlny o 180°, u bitu s úrovní 1 ke změně fáze nedochází. Délka jednoho symbolu (bitu) je 32 ms. Na začátku přenosu je tzv. preamble, což je série bitů s úrovní 0 (reprezentovaná změnami fáze o 180°), která umožní synchronizaci obvodu pro obnovu časování na straně přijímače. Po skončení přenosu je vysílána tzv. postamble, což je pouze nemodulovaná nosná vlna. Signál je filtrován cosinovým filtrem [9].

## 2.2 PSAT

Družice PSAT je vybavena PSK31 transpondérem, který přijímá signál v kmitočtovém rozsahu 28,120160 MHz až 28,122560 MHz. Ten je doplněn o telemetrický PSK31 signál na podnosné frekvenci 312,5 Hz. Tímto signálem je poté frekvenčně modulován signál vysílače na frekvenci 435,350 MHz s frekvenčním zdvihem 5 kHz. Transpondér může pracovat ve dvou módech, v módu A je vysílač zapnutý pořád, v módu B pouze v případě, že je detekován příchozí PSK31 signál [4].

Formát telemetrických dat je [4]:

CALL beacon MODE NOF DET AGC VC IC TMP

Význam jednotlivých částí je v tabulce 2.1.

Tabulka 2.1 Význam jednotlivých polí telemetrie PSAT

| | |
|---|---|
| CALL | Volací značka (W3ADO) |
| MODE | Mód transpondéru (A nebo B) |
| NOF | Číslo rámce |
| DET | Úroveň detekovaného BPSK31 signálu v procentech (0 – 99 %) |
| AGC | Úroveň AGC v procentech (0 – 99 %) |
| VC | Napětí zdroje (násobky 10 mV) |
| IC | Proud Výkonového zesilovače (mA) |
| TMP | Teplota výkonového tranzistoru (°C) |

## 2.3 BRICSat

Transpondér družice BRICSat přijímá signál v kmitočtovém rozsahu 28,120530 MHz až 28,122930 MHz. Ten je doplněn o telemetrický PSK31 signál na podnosné 375 Hz. Tímto signálem je poté frekvenčně modulován signál vysílače na frekvenci 435,351 MHz s frekvenčním zdvihem 5 kHz [6].

Transpondér může pracovat ve čtyřech módech (A, B, C nebo D). V módu A je vysílač transpondéru stále zapnutý. V módu B je vysílač zapínán v 20s slotech podle přítomnosti PSK31 signálu v pásmu přijímače, v případě že žádný signál není přítomen, je vysílač zapínán každých 120s pro přenos telemetrie. V módu C je vysílač zapínán pouze pro přenos telemetrie každých 180 s. V módu D je přenášena pouze volací značka bez telemetrie v intervalech 180 s [6].

Formát telemetrických dat je [6]:

aaaaaaa b ccddeeffgghhiijjkkllmm

Význam jednotlivých částí je v tabulce 2.2.

Číselné hodnoty jsou vyjádřeny pomocí číselné soustavy o základu 32, což umožňuje vyjádřit číslo o rozsahu 0 a 1023 pomocí pouhých dvou znaků. Pro čísla v rozsahu 0 až 25 jsou použity znaky *a* až *z*, pro čísla 26 až 31 znaky *A* až *F*. K hodnotám teploty je přičtena hodnota 99, čímž se předejde problému s reprezentací záporných čísel [6].

Tabulka 2.2 Význam jednotlivých polí telemetrie BRICSat

| | |
|---|---|
| aaaaaaa | Volací značka (W3ADO-6) |
| b | Mód transpondéru (A, B nebo C) |
| cc | Číslo rámce |
| dd | Úroveň detekovaného BPSK31 signálu v procentech (0 – 99 %) |
| ee | Úroveň AGC v procentech (0 – 99 %) |
| ff | Napětí zdroje (násobky 10 mV) |
| gg | Napětí na nižším článku baterie (násobky 10 mV) |
| hh | Napětí 1 (násobky 10mV) |
| ii | Napětí 2 (násobky 10mV) |
| jj | Napětí 3 (násobky 10mV) |
| kk | Proud koncového zesilovače |
| ll | Teplota přijímače (-98 – 156°C, přičteno 99) |
| mm | Teplota koncového tranzistoru (-98 – 156°C, přičteno 99) |

## 2.4 PSAT-2

Transpondér družice PSAT-2 přijímá signál v pásmu 29,4804 MHz až 29,4826 MHz, vysílá na frekvenci 435,350 MHz s frekvenční modulací. Telemetrický signál je přítomen na podnosné 374 Hz. Transpondér může pracovat ve čtyřech módech, které jsou přepínány podle napětí baterie. Družice umožňuje přenos SSTV obrázků ve formátu Robot36, obvody pro zpracování generují vlastní telemetrická data, která jsou vysílána transpondérem na podnosné 280 Hz [5]. Pro dekódování telemetrických dat je dostupný online dekodér[10], jehož součástí je i archiv dekódovaných dat [11].

Telemetrický signál obsahuje mj. informaci o aktivním módu transpondéru, úrovni AGC, napětí akumulátoru, napětí 5V větve napájení, odebíraném proudu a teplotě[5]. Telemetrická zpráva obsahuje aktuální datový rámec telemetrie a jeden historický rámec, který je pseudonáhodně vybrán z paměti[5].

Formát telemetrických dat je analogický k formátu BRICSat, číselné hodnoty jsou též vyjádřeny pomocí číselné soustavy o základu 32. Telemetrická zpráva má tvar:

aaaaaa b cccc ddeeffgghhiijj klmn b cccc ddeeffgghhiijj

Význam jednotlivých polí je v tabulce 2.3.

Tabulka 2.3 Význam jednotlivých polí telemetrie PSAT-2

| | |
|---|---|
| aaaaaaa | Volací značka (PSAT-2) |
| b | Mód transpondéru (A, B, C nebo D) |
| cccc | Číslo rámce |
| dd | Počet resetů (0 – 1023) |
| ee | Úroveň detekovaného BPSK31 signálu v procentech (0 – 99 %) |
| ff | Napětí AGC (násobky 10 mV) |
| gg | Napětí baterie (násobky 10 mV) |
| hh | Napětí zdroje 5V (násobky 10 mV) |
| ii | Odběr proudu transpondéru a SSTV kamery (mA) |
| jj | Teplota přijímače (°C, k záporným hodnotám přičteno 1024) |
| k | Číslo periody (0 – 11) |
| l | Počet period, po které bude zapnut vysílač kvůli SSTV |
| m | Počet period, po které bude zapnut příjem |
| n | Počet period, po které bude zapnuto vysílání |

# 3 DOPPLERŮV JEV

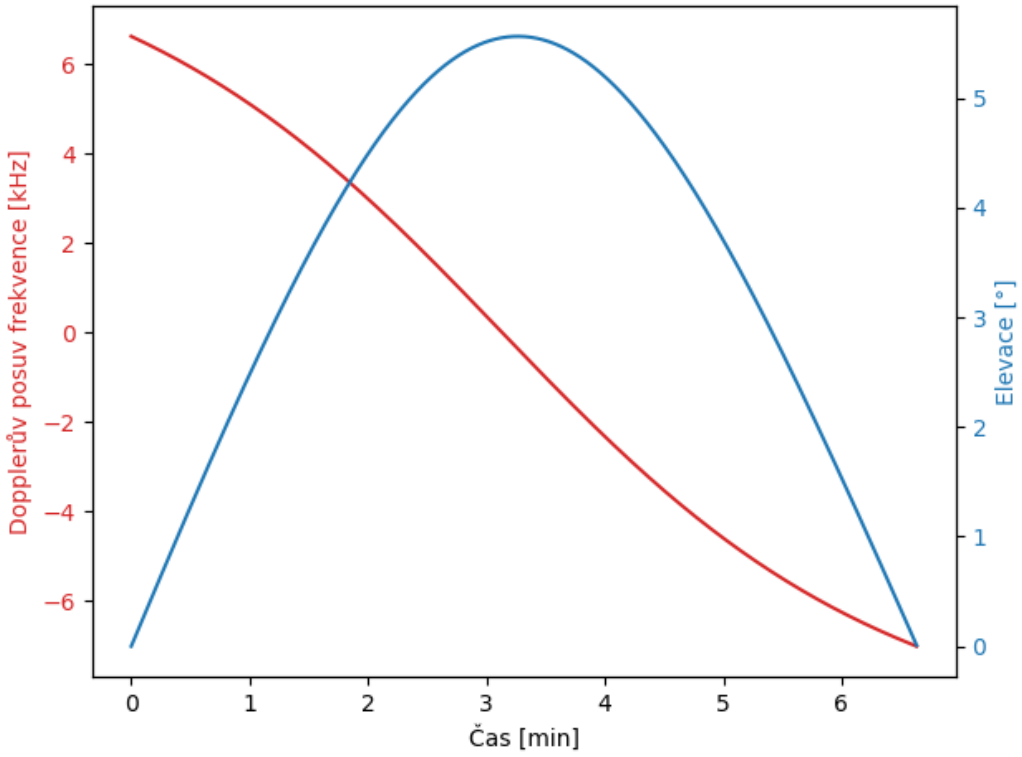
Dopplerův jev je jev, kdy dochází ke změně frekvence vlnění v důsledku vzájemného pohybu pozorovatele a zdroje vlnění. V případě pohybující se družice lze Dopplerův posuv frekvence popsat vztahem[12]:

$$\Delta f = \frac{\mathrm{f} \cdot \mathrm{v} \cdot \cos \Theta}{\mathrm{c}}, \tag{1}$$

kde $\Delta f$ je výsledná změna frekvence, $f$ je frekvence zdroje signálu, $v$ je rychlost pohybu družice a $\Theta$ je úhel mezi vektorem rychlosti družice a směrem k pozorovateli.

Dopplerův jev se nejvíce projevuje u družic s nízkou oběžnou dráhou, což je způsobeno velkou rychlostí družice vůči pozorovateli, naopak u geostacionárních družic je téměř nulový[12].

Příklad Dopplerova posuvu frekvence pro jeden konkrétní přelet družice PSAT je na obr. 3.1.




Obrázek 3.1 Příklad Dopplerova posuvu frekvence a elevace družice PSAT

# 4 ZPRACOVÁNÍ SDR ZÁZNAMŮ

Jako formát telemetrického archivu byl zvolen souborový formát CSV. Pro zpracování SDR záznamů byl vytvořen program v programovacím jazyce Python, který se ovládá pomocí argumentů v příkazové řádce. Kompletní seznam možných argumentů je v příloze A. Program je určen pro operační systém GNU/Linux, a to kvůli použití knihovny GNU Radio. Výstupem programu je jeden nebo více souborů ve formátu CSV, kde jeden záznam (řádek) v souboru odpovídá jedné vysílané telemetrické zprávě. V případě družic PSAT a BRICSat jsou ukládány dekódované hodnoty (podle tab. 2.1 a 2.2), v případě telemetrie PSAT-2 je ukládána textová zpráva s telemetrií a pro dekódování je použit existující online dekodér[10]. Záznamy jsou doplněny o další údaje, jako datum a čas vysílání, název souboru, ze kterého byla telemetrická data získána, poloha družice vůči zemi a Slunci v daný moment, frekvenční offset signálu aj. Formát CSV souborů je popsán v kapitole 4.9.

Program je tvořen několika soubory. Spouští se pomocí souboru *dop_demod_dec.py*, který importuje ostatní soubory jako moduly. Modul *doppler.py* slouží k odstranění Dopplerova posuvu frekvence, konstantnímu posuvu kmitočtového spektra a převedení vstupního souboru ve formátu WAV na surová IQ data. Modul *demod_grc.py* je použit k následné demodulaci FM signálu, demodulaci a dekódování PSK31 signálu a detekci textových sekvencí s telemetrií. Pro samotné dekódování telemetie slouží modul *decode_tlm.py*. Soubor *sat.py* obsahuje parametry družic, *tle.py* potom implementuje jednoduchý archiv kepleriánských elementů, které jsou potřebné pro výpočet dráhy satelitu. Modul *sun.py* poskytuje funkce pro určení polohy družice vůči Slunci. Modul *satnogs.py* je určen pro získání informací o souboru z internetových stránek Satnogs. Dále byly vytvořeny moduly, které tvoří knihovnu bloků pro GNU Radio.

Programu se předá jeden nebo více názvů vstupních souborů ve formátu WAV. Pokud je název souboru ve formátu HDSDR, program z něj určí datum a čas pořízení záznamu a frekvenci přijímače. V opačném případě je třeba datum a čas zadat přímo pomocí argumentu, nebo lze použitím příslušného parametru tuto informaci získat z internetových stránek Satnogs. K tomu je třeba identifikační číslo záznamu Satnogs, které program může automaticky odvodit z názvu souboru (první sekvence číslic v názvu), nebo ho lze opět zadat přímo.

Následně je soubor zpracován pomocí modulu *doppler.py*. Tento modul volitelně provede odstranění Dopplerova jevu, posune frekvenční spektrum signálu o rozdíl mezi frekvencí vysílače a přijímače (pokud byla zjištěna z názvu souboru, nebo byla zadána pomocí argumentu) a takto upravená surová IQ data uloží do dočasného souboru. Odstranění Dopplerova jevu se provede, pokud se nejedná o záznam Satnogs, kde už je Dopplerův jev odstraněn. Jeho odstranění však lze i vynutit nebo zakázat. Také lze volitelně provést inverzi kmitočtového spektra.

Dočasný soubor je následně zpracován pomocí modulu *demod_grc.py*. V ideálním případě by měl být FM signál v dočasném souboru přítomen na nulovém kmitočtu, obvykle však vykazuje určitý kmitočtový offset. Ten může být poměrně velký vzhledem k celkové šířce pásma signálu. Obvykle pohybuje v řádu jednotek kilohertzů (u některých souborů Satnogs je však až 20 kHz) a může se měnit. To představuje problém při demodulaci signálu. Záznam je proto zpracováván opakovaně, přičemž

pokaždé je frekvenční spektrum signálu posunuto. Rozsah posunu spektra a velikost kroku, případně výčet konkrétních hodnot posunu je zadán jako parametr programu.

Program umožňuje nastavit šířku propustného pásma vstupního filtru (viz kap. 4.3), pokud se zadá více hodnot, zpracování záznamů se provede opakovaně pro každou z těchto hodnot. To je užitečné při zpracování záznamů PSAT-2, kde se šířka pásma signálu mění v závislosti na tom, zda je vysílán SSTV signál.

Následně jsou všechna získaná telemetrická data sloučena a na základě času výskytu je určeno, která data s různými kmitočtovými offsety odpovídají stejné telemetrické zprávě. Z každé skupiny telemetrie náležící jedné telemetrické zprávě je vybrána ta, která obsahuje nejméně chyb, v případě více stejných se rozhodne na základě odhadu poměru signál/šum.

Vybraná telemetrická data jsou doplněna o další údaje a uložena do výstupního CSV souboru. Název výstupního souboru se zadá jako argument. Pokud název obsahuje některé ze zástupných řetězců v tab. 4.1, dojde k jejich nahrazení příslušnou hodnotou, pokud je známá. To je výhodné při zpracování více souborů, kdy pro telemetrická data z každého vstupního souboru má být použit samostatný výstupní soubor. Pokud výstupní soubor existuje, telemetrická data jsou připsána na konec, použitím příslušných parametrů lze ale docílit i toho, že se výstupní soubor přepíše nebo se zpracování vstupního souboru přeskočí.

Program dále umožňuje uložit úseky audio signálu získaného FM demodulací, které obsahují zakódované SSTV obrázky vysílané družicí PSAT-2. Pro jejich výstupní název lze též použít zástupné řetězce podle tab. 4.1.

Tabulka 4.1 Zástupné řetězce pro název výstupního souboru

| | |
|---|---|
| %n | Pořadové číslo vstupního souboru |
| %i | Název vstupního souboru |
| %I | Název vstupního souboru bez přípony |
| %d | Datum a čas |
| %N | Satnogs id |
| %t | Čas výskytu SSTV obrázku (pouze pro výstupní soubor SSTV) |

Postup instalace programu je popsán v příloze C.

## 4.1 Použité knihovny

Programy byly napsány pro Python 3.8 [13]. Pro zpracování zaznamenaného signálu byla použita knihovna GNU Radio 3.8.0[14], pro výpočet polohy a rychlosti družice byla použita knihovna Skyfield [15]. Dále byly použity knihovny argparse [16] (zpracování argumentů příkazové řádky), datetime [17] (práce s časem), math [18] (matematické funkce), scipy [19], csv [20] (práce s CSV soubory), re [21] (regulární výrazy), urllib [22] (stažení internetových stránek) a json [23].

### 4.1.1 GNU Radio

GNU Radio [14] je knihovna pro Python poskytující bloky pro digitální zpracování signálu se zaměřením na softwarově definované rádio. Použití knihovny spočívá ve vytvoření instancí potřebných bloků a definování vzájemných propojení. Výsledný řetězec těchto bloků se označuje jako flowgraph [24].

Knihovna pracuje se signálem jako s datovým proudem, hodnoty mohou být uloženy pomocí různých datových typů (*float*, *complex*, *byte*, aj.) [24].

Bloky mohou mít různý počet vstupů a výstupů, které se označují jako porty. Pokud má blok pouze výstupní port, označuje se jako *source*, blok který má pouze vstupní port se označuje *sink*. Dále se bloky dělí podle množství dat, která odebírají ze vstupního portu a která generují na výstupním portu. Pokud blok odebírá a vytváří stejné množství dat, označuje se jako *sync* blok. Pokud generuje více dat, než odebírá, označuje se jako *interpolator*, blok který generuje méně dat, než spotřebovává, se označuje *decimator*. Pokud poměr mezi množstvím odbíraných a generovaných vzorků není pevný, označuje se jako *general* blok [24].

Každému výstupnímu portu je přidělen kruhový buffer, který zároveň slouží jako vstupní buffer připojeného bloku. Plánovač GNU Radia volá funkci *work()* jednotlivých bloků, tato funkce odebírá data ze vstupních bufferů (v libovolném množství), zpracuje je a ukládá do výstupního bufferu. Tato funkce je volána, kdykoliv jsou dostupná vstupní data a nedošlo k úplnému naplnění výstupního bufferu. Rychlost zpracování dat není nijak omezena. Data jsou zpracovávána po různě velkých dávkách a jednotlivé bloky pracují vzájemně asynchronně [25]. Tato koncepce klade určitá omezení na použití knihovny, nelze například bloky propojit do smyčky[26].

Kromě dodávaných bloků lze použít i bloky vlastní, které mohou být naprogramovány v jazyce Python nebo C++ [27], případně mohou být vytvořeny propojením jiných bloků (tzv. hierarchické bloky) [28].

Libovolný vzorek proudu dat lze označit pomocí tzv. tagu, který může mít hodnotu libovolného typu. Tagy prochází většinou bloků beze změn. V případě *sync* bloku jsou tagy vstupního proudu dat přiřazeny odpovídajícím vzorkům na výstupu, u ostatních typů bloků jsou tagy ve výstupním proudu rozmístěny odhadem [29].

S knihovnou je dodáván i program GNU Radio Companion, který umožňuje vytvořit flowgraph grafickou formou a vygenerovat soubor se zdrojovým kódem [30].

### 4.1.2 Skyfield

Skyfield [15] je knihovna pro Python určená k astronomickým výpočtům. Je napsána v čistém Pythonu a nevyžaduje žádnou kompilaci. Jedinou binární závislostí je knihovna NumPy. Umožňuje určit polohu hvězd, planet i satelitů na oběžné dráze Země[15]. Pozici satelitů lze spočítat na základě tzv. kepleriánských elementů dráhy, což je soubor hodnot popisující pohyb družice. Kepleriánské elementy dráhy jsou vyžadovány v tzv. dvouřádkovém formátu [31].

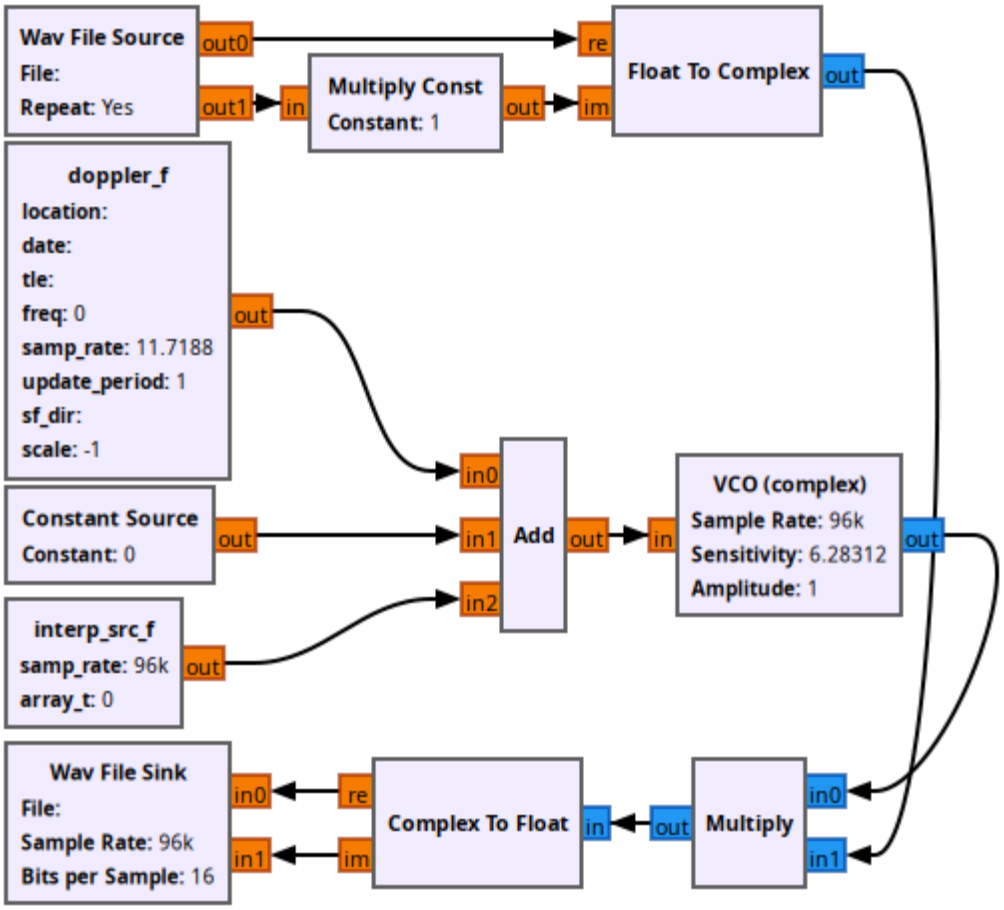
## 4.2 Odstranění Dopplerova jevu a frekvenčního offsetu

Pro odstranění Dopplerova jevu a frekvenčního offsetu byl vytvořen modul *doppler.py*. Modul lze použít i jako samostatný program, potom se ovládá pomocí argumentů příkazové řádky. Přehled možných argumentů je v příloze B. Programu je třeba zadat název vstupního a výstupního souboru, typ družice a úkon, který se má provést. Tím může být odstranění Dopplerova jevu, konstantní posun frekvenční spektra nebo lineární korekce frekvenčního offsetu. Může být provedeno i více z těchto úkonů najednou.

Pro výpočet Dopplerova jevu je třeba znát čas a místo pořízení záznamu a kepleriánské elementy dráhy dané družice. Čas pořízení záznamu je automaticky odvozen z názvu souboru (pokud je to možné), místo pořízení záznamu je uloženo v programu. Kepleriánské elementy dráhy jsou automaticky vyhledány v archivu kepleriánských elementů, který je popsán v kap. 4.4. Tyto parametry je ale možné zadat i pomocí argumentů příkazové řádky.

Pokud je modul použit programem *dop_demod_dec.py*, všechny potřebné parametry jsou předány tímto programem.

Pro zpracování signálu byla použita knihovna GNU Radio. Blokové schéma znázorňující propojení bloků GNU Radia je na obr 4.1.




Obrázek 4.1 Blokové schéma propojení bloků GNU Radia skriptu doppler.py

Na začátku řetězce je blok *Wav File source*, který generuje datový proud typu *float* ze zadaného vstupního souboru. Tento blok má dva výstupy, první výstup odpovídá synfázní složce komplexního signálu, na druhém výstupu je pak kvadraturní složka. Kvadraturní složka signálu je pak blokem *Multiply Const* volitelně násobena konstantou 1 nebo -1. Násobením zápornou hodnotou se dosáhne inverze kmitočtového spektra[32], což je nutné pro některé záznamy družice PSAT-2. Obě složky signálu jsou poté blokem *Float To Complex* sloučeny do jednoho datového proudu typu *complex*. Data jsou následně přivedena na vstup bloku *Multiply*, kde jsou násobena signálem řízeného oscilátoru *VCO*. Tím se dosáhne požadovaného frekvenčního posuvu. Pokud má název výstupního souboru příponu *wav*, datový proud je poté opět rozdělen na dva proudy typu *float* a uložen do výstupního souboru pomocí bloku *Wav File Sink*. Pokud má výstupní soubor jinou příponu, jsou tyto dva bloky nahrazeny blokem *File Sink*, pomocí kterého jsou uložena surová IQ data typu *complex*. To je i případ použití s programem *dop_demod_dec.py*, kdy je použita přípona *dat*. Následné zpracování takového souboru je mnohem rychlejší než v případě souboru *wav*, navíc se předejde znehodnocení signálu při převodu z datového typu s plovoucí desetinou čárkou na celočíselný datový typ, který provádí blok *Wav File Sink*.

Oscilátor VCO generuje komplexní harmonický signál o amplitudě *Amplitude* a okamžité úhlové frekvenci:

$$\omega(t) = \text{sensitivity} \cdot \text{i(t)}, \tag{2}$$

kde *sensitivity* je parametr bloku udávající citlivost na řídící signál a *i(t)* je okamžitá hodnota řídícího signálu. Parametr *sensitivity* byl zvolen $2\pi$, takže hodnoty řídícího signálu přímo odpovídají požadovaně frekvenci v Hertzích. Násobení signálu výstupem oscilátoru je ekvivalentní procesu směšování a způsobí posun frekvenčního spektra signálu o frekvenci oscilátoru.

Řídící signál oscilátoru je součtem signálů bloků *doppler_f*, *Constant Source* a *interp_src_f*. Blok *doppler_f* generuje vzorky odpovídající okamžité hodnotě Dopplerova posuvu frekvence a je blíže popsán v kap. 4.2.1. *Constant Source* vytváří konstantní signál a slouží pro korekci konstantního frekvenčního offsetu. Blok *interp_src_f* vytváří signál lineární korekce a je popsán v kap. 4.2.2. Některé z těchto tří bloků nemusí být použity, což závisí na zadaných parametrech skriptu.

V příloze D je příklad spektrogramů SDR záznamu PSAT před úpravou, po odstranění Dopplerova jevu a po dodatečné lineární korekci kmitočtového offsetu.

### 4.2.1 Výpočet Dopplerova jevu

Pro výpočet Dopplerova posuvu frekvence byl vytvořen blok *doppler_f*, který pro výpočet okamžité polohy a rychlosti satelitu používá knihovnu Skyfield. Tento blok má parametry *location* (zeměpisné souřadnice umístění přijímače), *date_time* (datum a čas pořízení záznamu), *tle* (kepleriánské elementy dráhy satelitu v dvouřádkovém formátu), *freq* (frekvence vysílače), *samp_rate* (vzorkovací frekvence), *update_period* (časový interval v sekundách, s kterým má být výpočet proveden), *sf_dir* (adresář s daty knihovny Skyfield), *scale* (hodnota, kterou je násoben výstup) a *interp_n* (udává, kolik hodnot se má najednou spočítat).

Při inicializaci bloku je vytvořena proměnná *topos*, která obsahuje zeměpisné souřadnice přijímače a proměnná *sat* která obsahuje pozici satelitu. Jejich rozdílem

(proměnná *self.sat_topos*) se získá pozice satelitu vztažená k pozici přijímače. Pozici satelitu pro konkrétní okamžik lze získat voláním metody *at(t)* této proměnné, jejímž argumentem je požadovaný čas. Do proměnné *self.date_time* je uložen počáteční datum a čas.

```
topos = Topos(latitude_degrees=location[0],
            longitude_degrees=location[1],
            elevation_m=int(location[2]))
sat = EarthSatellite(tle[0], tle[1])
self.sat_topos = sat - topos
```

Pro samotný výpočet Dopplerova posuvu frekvence byla vytvořena funkce *doppler()*, které se předá pole s hodnotami času v sekundách, které jsou vztaženy k počátečnímu datu a času. Výpočet se provádí pro všechny hodnoty tohoto pole najednou, což je výhodné z hlediska rychlosti výpočtu. Z vypočtené pozice satelitu pro daný čas (proměnná *sat_top_at*) se určí polohový vektor satelitu (*sat_pos*), vektor rychlosti (*sat_vel*) a absolutní hodnota rychlosti (*sat_sp*). Z těchto hodnot je pak vypočten Dopplerův posuv frekvence podle vztahu (1).

```
def doppler(self, time_offsets):
    t = self.ts.utc(*self.date_time[:-1],
            self.date_time[-1] + time_offsets)
    sat_top_at = self.sat_topos.at(t)
    sat_vel = sat_top_at.velocity.km_per_s
    sat_pos = sat_top_at.position.km
    sat_sp = sat_top_at.speed().km_per_s*1000.0
    angle = self.vec_angle(sat_vel, -sat_pos)
    return (self.scale*self.freq
            *sat_sp*np.cos(angle)/self.light_sp
```

Hodnoty Dopplerova posuvu frekvence jsou vypočítány pro *interp_n* hodnot času a uloženy do proměnné *self.interp_dop*, hodnoty času jsou uloženy v proměnné *self.interp_t*. Funkce *work()*, která je volána plánovačem GNU Radia, potom určí hodnotu pro každý vzorek výstupního bufferu pomocí lineární interpolace hodnot *self.interp_dop*, k čemuž byla použita funkce *numpy.interp()*. Pokud je čas konkrétního vzorku větší než poslední hodnota *self.interp_t*, Dopplerův posuv frekvence se vypočte pro nových *interp_n* hodnot času.

```
def work(self, input_items, output_items):
    out = output_items[0]
    t_offsets = self.t_offset + np.arange(len(out))/self.samp_rate
    if t_offsets[0] > self.interp_t[-1]:
        self.update_interp(self.interp_t[-1])
    if t_offsets[-1] > self.interp_t[-1]:
        items_to_process = np.max(np.where(
            t_offsets < self.interp_t[-1])) + 1
    out[:items_to_process] = np.interp(
        t_offsets[:items_to_process], self.interp_t, self.interp_dop)
    self.update_interp(self.interp_t[-2])
    self.t_offset += items_to_process/self.samp_rate
    return items_to_process)
```

### 4.2.2 Lineární korekce kmitočtového offsetu

Pro lineární korekci kmitočtového offsetu byl vytvořen blok *interp_src_f*. Parametry tohoto bloku jsou *samp_rate* (vzorkovací kmitočet), *array_t* (pole s hodnotami času) a *array_outp* (pole požadovaných hodnot na výstupu). Výstupem tohoto bloku pro časy zadané v *array_t* jsou odpovídající hodnoty v proměnné *array_outp*, pro jiné časy jsou výstupní hodnoty získány interpolací nebo extrapolací lomenou přímkou.

K výpočtu bylo využito třídy *interp1d* z knihovny *scipy.interpolate*, která vrací funkci, jejíž volání používá interpolaci (nebo extrapolaci) k nalezení požadovaných hodnot [33].

```
self.interp_func = interp1d(array_t, array_outp, kind="linear",
   fill_value="extrapolate")
```

Ve funkci *work()* je potom vytvořeno pole s časy jednotlivých vzorků a to je předáno vytvořené interpolační funkci.

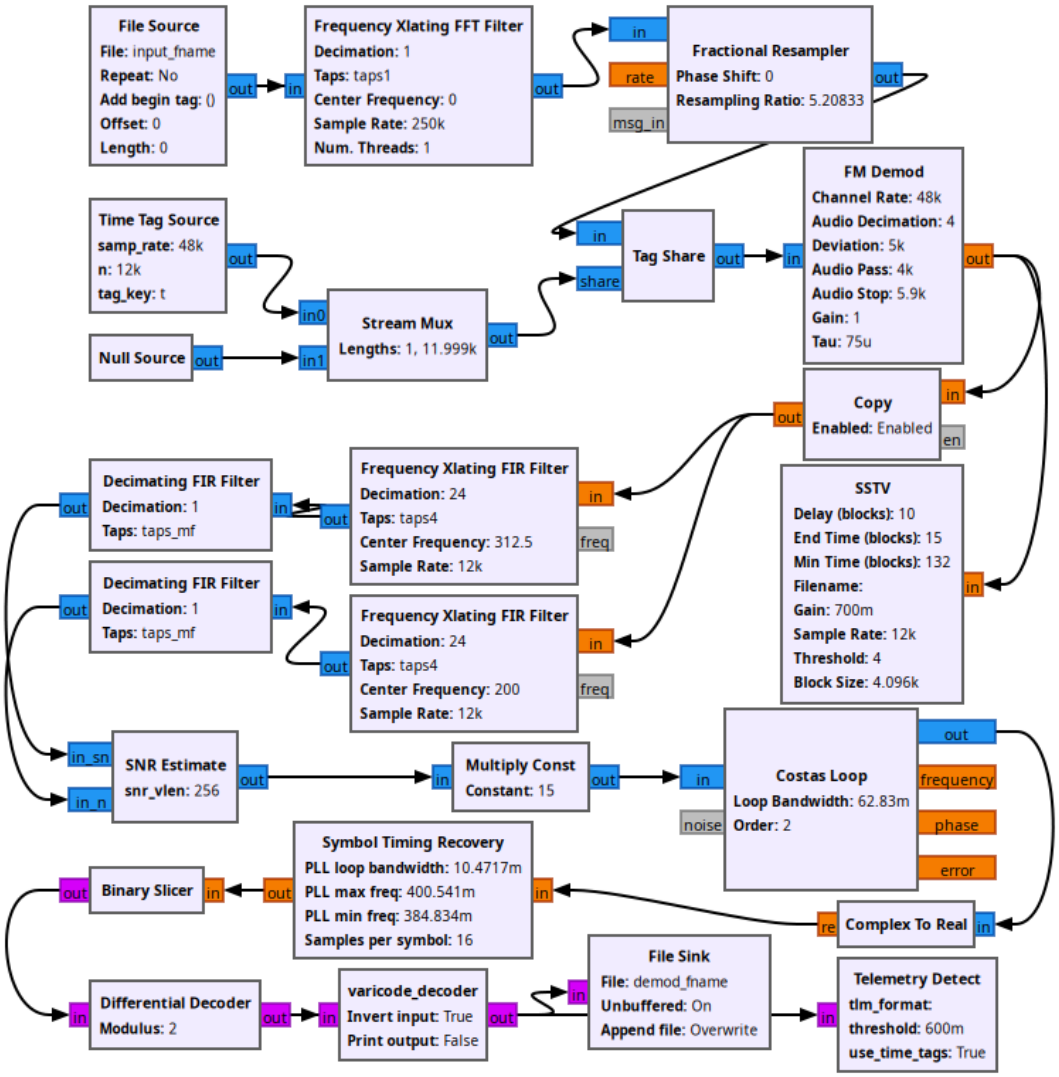
```
vec_t = numpy.linspace(self.t, self.t
   + (num_samples - 1)/self.samp_rate, num=num_samples)
out[:] = self.interp_func(vec_t)
```

## 4.3 Demodulace a dekódování signálu

Demodulaci a dekódování signálu provádí modul *demod_grc.py*, který byl vygenerován v programu GNU Radio Companion. Tento modul je importován skriptem *dop_demod_dec.py* a jsou mu předány potřebné parametry. Blokové schéma znázorňující propojení bloků GNU Radia je na obr. 4.2.

Prvním v řetězci je blok *File Source*, který čte IQ data ze zadaného vstupního souboru (dočasný soubor vytvořený pomocí modulu *doppler.py*) a předává je na vstup bloku *Frequency Xlating FFT filter*, který posune frekvenční spektrum signálu o zadaný frekvenční offset. Tento blok zároveň představuje filtr typu dolní propust, jehož mezní frekvence byla zvolena 2 kHz, lze ji však změnit pomocí argumentu programu. Omezením šířky pásma signálu se sníží výkon šumu a zabrání se zrcadlení spektra při převzorkování signálu následujícím blokem *Fractional Resampler*. Převzorkování je nutné, jelikož vstupní signál může mít různou vzorkovací frekvenci. Výstupní signál o jednotné vzorkovací frekvenci 48 kHz je následně přiveden na vstup bloku *Tag Share*, kde je sloučen s pomocným signálem opatřeným tagy, které nesou informaci o čase vzorků od počátku záznamu. Vytvoření časových tagů je blíže popsáno v kapitole 4.3.1. Následuje blok *FM Demod*, který provádí frekvenční demodulaci. Výstup je decimován čtyřmi na vzorkovací frekvenci 12 kHz.

Demodulovaný reálný signál je připojen na vstupy bloků *SSTV* a *Copy*. Blok *SSTV* slouží k detekci SSTV signálu a jeho uložení do souboru ve formátu WAV. Jeho funkce je popsána v kapitole 4.3.3. Pomocí bloku *Copy* lze odpojit zbývající část řetězce, pokud je cílem pouze uložení SSTV signálu. Signál je poté rozdělen do dvou větví.




Obrázek 4.2 Blokové schéma skriptu demod_dec_grc.py

V první větvi je signál přiveden na vstup bloku *Frequency Xlating FIR Filter*. Tímto blokem je signál frekvenčně posunut o frekvenci podnosné telemetrického signálu, čímž je telemetrický signál přenesen do základního pásma. Zároveň je signál podvzorkován na 500 Hz, což odpovídá 16 vzorkům na jeden symbol PSK31 signálu. Součástí bloku je i filtr typu dolní propust, který je nutný k zabránění zrcadlení spektra při podvzorkování. Signál je následně filtrován blokem *Decimating FIR filter*. Tento filtr typu dolní propust musí být navržen tak, aby se omezily mezisymbolové interference a zároveň se maximalizoval poměr signál/šum. Byl použit filtr podle [34][35].

Zapojení druhé větve je obdobné, pouze je frekvenční posuv o 100Hz větší, čímž se získá signál obsahující pouze šum. Ten je využit v následujícím bloku *SNR Estimate*, který slouží k odhadu poměru signál/šum (SNR). Jeho výstupem je signál první větve opatřený tagy s hodnotou SNR. Popis bloku je v kapitole 4.3.2.

PSK signál je poté zesílen a demodulován pomocí Costasovy smyčky, kterou předsavuje blok *Costas Loop*. Z výstupního signálu je pomocí bloku *Complex To Real* odebrána pouze reálná složka a je přivedena na vstup bloku *Symbol Timing Recovery*. Tento blok slouží k obnově časování symbolů a je popsán v kap. 4.3.4. Jeho výstupem

je signál, kde každý vzorek odpovídá jednomu symbolu. Následně je signál převeden na digitální pomocí bloku *Binary Slicer*, vzorky s kladnou hodnotou jsou převedeny na bity s úrovní 1, vzorky se zápornou hodnotou na bity s úrovní 0. Výstupem je datový proud typu byte, kde každý byte nabývá pouze hodnot 0 nebo 1 a představuje tak jeden bit. Následuje blok *Differential Decoder*, který porovnává dva po sobě jdoucí vzorky a jeho výstupem je úroveň 1 v případě, že se liší. Signál je poté přiveden na vstup bloku *varicode_decoder*, který dekóduje varicode abecedu. Výstupem je datový proud typu *byte*, kde každý vzorek představuje jeden ASCII znak. Tento dekódovaný text je poté uložen do souboru pomocí bloku *File Sink* (jako výchozí soubor je použit /dev/null, takže jsou data zahozena) a zároveň je přiveden na vstup bloku *telemetry_detect_b*, který slouží k detekci a uchování textových sekvencí telemetrie.

### 4.3.1 Použití časových tagů

Součástí telemetrického archivu je i informace o datu a čase vysílání telemetrické zprávy. K tomu je třeba určit čas výskytu telemetrické zprávy v SDR záznamu a přičíst jej k datu a času pořízení tohoto záznamu. Tento čas nelze určit z pozice zprávy v dekódovaném textu, jelikož výstupem varicode dekodéru je text pouze pro platné bitové sekvence varicode abecedy. Problém představuje i proměnná vzorkovací frekvence bloku na obnovu časování symbolů. Z tohoto důvodu bylo k určení času využito tagů.

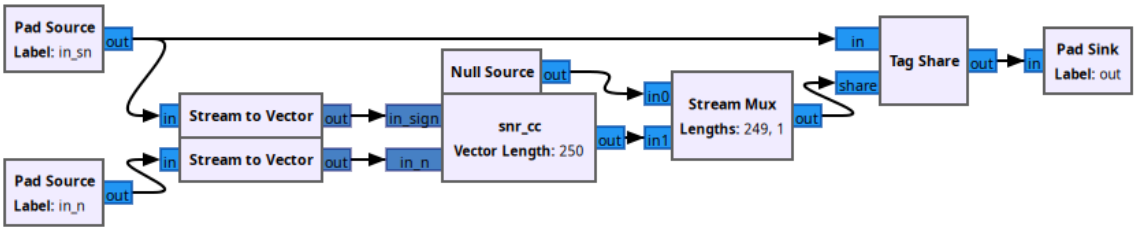
Vzorky datového proudu jsou na začátku řetězce opatřeny tagy, jejichž hodnota odpovídá času v sekundách od počátku záznamu. Vzorky s tagy projdou celým řetězcem a na konci jsou využity v bloku pro detekci telemetrie. Při tvorbě všech vlastních bloků GNU Radia byl kladen důraz na korektní propagaci těchto tagů.

Pro generování tagů byl vytvořen blok *time_tag_src_c*. Parametry tohoto bloku jsou *samp_rate* (vzorkovací frekvence), *n* (udává, že tagem má být označen každý n-tý vzorek) a *tag_key* (název tagu, lze použít pro odlišení tagů z více zdrojů).

Je zbytečné tagem označovat každý vzorek, protože signál je postupně decimován a jednomu vzorku by bylo přiřazeno více tagů. Čas není ani nutné určit s takovou přesností, rozlišení času na vteřiny je dostačující. Vzorky s tagy jsou tedy prokládány vzorky bez tagů, které jsou generovány blokem *Null Source*. Přepínání těchto dvou zdrojů je realizováno pomocí bloku *Stream Mux*. Vzorky bez tagů nejsou generovány přímo v bloku *time_tag_src_c* z důvodu optimalizace, jelikož vlastní bloky vytvořené v Pythonu jsou výrazně pomalejší než standardní bloky GNU Radia implementované v C++ [27].

### 4.3.2 Odhad poměru signál/šum

Pro výpočet odhadu poměru signál/šum byl vytvořen blok *SNR Estimate*. Jedná se o hierarchický blok, je tedy tvořen propojením jiných bloku GNU Radia. Blokové schéma je na obr. 4.3.




Obrázek 4.3 Blokové schéma bloku SNR Estimate

Blok má dva vstupy, na první vstup je přiveden užitečný přijatý signál (který obsahuje aditivní šum), na druhý vstup je přiveden signál o stejné šířce pásma, který obsahuje pouze šum. Odhad SNR je získán porovnáním výkonové úrovně těchto signálu. Oba signály jsou přivedeny na vstupy bloků *Stream to Vector*, které sloučí každých 250 vzorků do jednoho vzorku typu komplexní vektor (pole). Díky tomu lze využít vektorové výpočetní operace knihovny *numpy*, které jsou mnohem rychlejší než práce s jednotlivými vzorky. Pro samotný výpočet SNR byl vytvořen blok *snr_cc*. Zdrojový kód funkce *work()* je uveden níže.

```
def work(self, input_items, output_items):
in_sn = input_items[0]
        in_n = input_items[1]
        out = output_items[0]
        for i, (sn, n) in enumerate(zip(in_sn, in_n)):
            sn_mag_sqrd = np.abs(sn)**2
            n_mag_sqrd = np.abs(n)**2
            sn_mag_sqrd_sum = np.sum(sn_mag_sqrd)
            n_mag_sqrd_sum = np.sum(n_mag_sqrd)

            snr = np.float64(sn_mag_sqrd_sum/n_mag_sqrd_sum)
            out[i] = snr

            self.add_item_tag(0, # Port number
                 self.nitems_written(0) + i, # Offset
                 self.tag_key, # Key
                 pmt.from_double(snr) # Value
            )

        return len(out)
```

U obou vstupních komplexních vektorů je nejdříve získána absolutní hodnota jednotlivých vzorků voláním metody *abs()*, poté jsou umocněny a sečteny, čímž se získá odhad energie v daném úseku signálu. Podíl těchto hodnot potom odpovídá poměru signál/šum podle definice:

$$SNR = \frac{P_s + P_n}{P_n}, \tag{3}$$

kde $P_s$ je výkon signálu a $P_s$ je výkon šumu. Odečtením výkonu šumu od výkonu signálu (s aditivním šumem) by se získala hodnota SNR podle obvyklé definice [36]:

$$SNR = \frac{P_s}{P_n} \tag{4}$$

Jelikož je ale úroveň šumu měřena v jiné části spektra než úroveň signálu, čitatel zlomku (4) by mohl být záporný, což by představovalo problém při určení SNR v decibelech. Byl proto použit vztah (3).

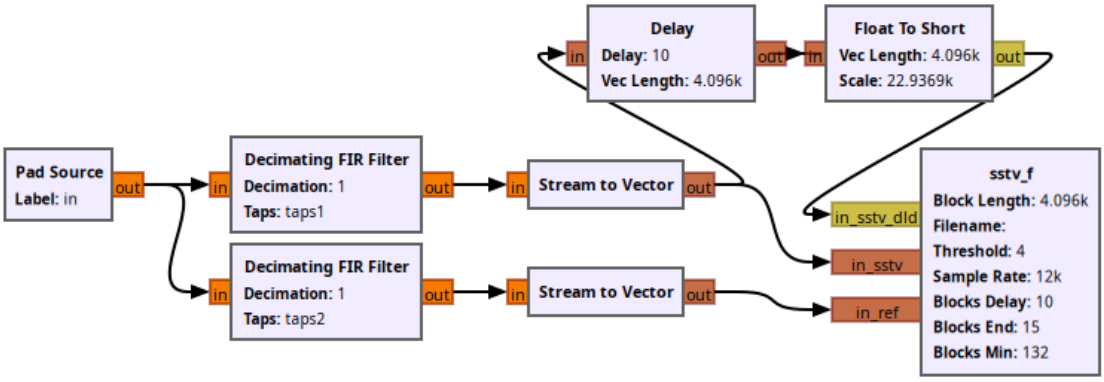
Výstupem bloku *snr_cc* je signál opatřený tagy s hodnotou SNR. Pomocí bloku *Stream Mux* je potom každý vvorek doplněn 249 nulovými vzorky bez tagu, čímž je dosaženo stejné datové rychlosti, jako má vstupní signál. Tagy jsou poté přeneseny na vstupní signál pomocí bloku *Tag share* a signál je přiveden na výstup.

V bloku *telemetry_detect_b* jsou pro každou detekovanou telemetrickou zprávu přečteny tagy v časovém úseku zprávy. Je určena minimální a průměrná hodnota SNR a ty jsou převedeny na decibely podle vztahu[36]:

$$SNR_{dB} = 10 \cdot \log_{10}(SNR) \tag{5}$$

### 4.3.3 Detekce a uložení SSTV signálu

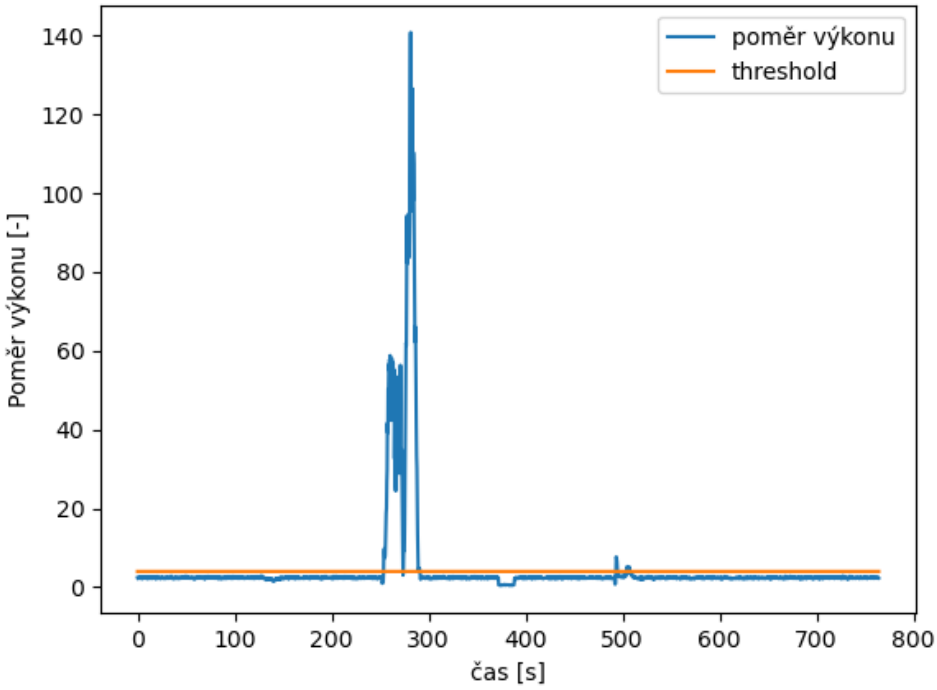
Družice PSAT-2 vysílá SSTV signál ve formátu Robot 36[5]. Přenos jednoho obrázku trvá 36s[37] a signál je obsažen v části frekvenčního spektra přibližně mezi 1200 až 2300 Hz[38]. Část spektra pod touto oblastí je vyhrazena pro provoz PSK31, část spektra nad 2300 Hz by vzhledem k celkové šířce pásma transpondéru neměla obsahovat žádný užitečný signál. Pro detekci a uložení SSTV signálu byl vytvořen hierarchický blok *SSTV*, jehož blokové schéma je na obr. 4.4.




Obrázek 4.4 Blokové schéma bloku SSTV

Vstupem bloku je zvukový signál získaný demodulací přijatého FM signálu. Signál bloku je rozdělen do dvou větví. V první větvi prochází filtrem typu pásmová propust, jehož propustné pásmo je 1100 až 2400 Hz a propustí tedy pouze SSTV signál. V druhé větvi je umístěn filtr typu horní propust, který má mezní frekvenci 2500 Hz. Přítomnost SSTV signálu je zjišťována porovnáním výkonu těchto dvou signálů. Při příjmu SSTV signálu se zvýší výkon signálu první větve a zároveň se sníží výkon signálu druhé větve vlivem poklesu úrovně šumu.

Vzorky obou signálů jsou převedeny na vektory o 4096 vzorcích pomocí bloků *Stream To Vector*, což umožní rychlejší zpracování než operace s jednotlivými vzorky. Signál první větve je poté připojen na vstup *in_sstv* bloku *sstv_f*, který byl vytvořen v Pythonu. Signál druhé větve je připojen na vstup *in_ref* téhož bloku.
Blok *sstv_f* určí poměr výkonu těchto dvou signálů pro každý vstupní vzorek (typu vektor), pokud překročí hodnotu danou jeho parametrem *threshold*, spustí záznam signálu, který je připojen na jeho vstup *in_sstv_dld*. Na tento vstup je připojen signál první větve, který byl zpožděn pomocí bloku *Delay* a převeden na datový typ *short*. Zpoždění je nutné z toho důvodu, aby došlo ke spuštění záznamu s určitým předstihem.
Blok uloží do souboru minimálně 132 vektorových vzorků (parametr *Blocks Min*), což představuje přibližně 45s záznamu. Pokud je po této době poměr výkonů stále větší než *threshold*, záznam pokračuje do té doby, než tento poměr klesne pod tuto hranici. Poté je zapsáno ještě dalších 15 vektorových vzorků (parametr *Blocks End*).
Příklad poměru výkonu na vstupech *sstv_f* v závislosti na čase je na obrázku 4.5. Jsou zde přítomny dva SSTV signály, první v čase 250s, druhý výrazně slabší signál potom v čase 493s.
Signál je ukládán jako dočasný soubor formátu WAV do složky */tmp*. Skript *dop_demod_dec.py* získá informace o uložených SSTV souborech voláním metody *get_info()* bloku *sstv_f*.
Pokud je zpracování signálu provedeno vícekrát pro různé hodnoty frekvenčního offsetu FM signálu, může dojít k uložení stejného SSTV signálu vícekrát. Potom je určeno, které soubory odpovídají stejnému SSTV obrázku (na základě času výskytu) a je z nich uchován pouze ten, u kterého byl průměrný poměr výkonu na vstupech *sstv_f* největší.

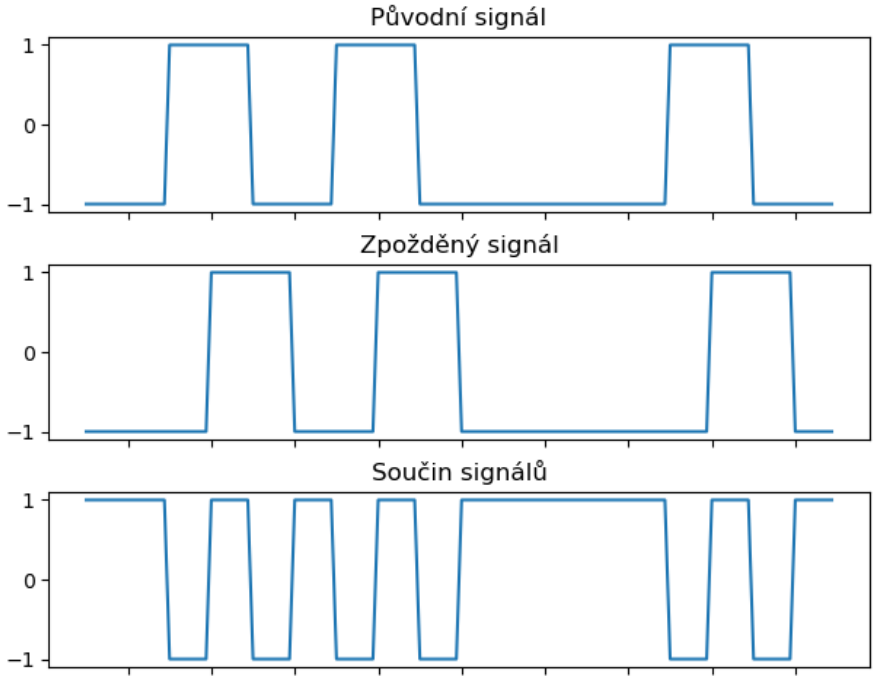



Obrázek 4.5 Poměr výkonu na vstupech *in_sstv* a in_ref bloku *sstv_f*

### 4.3.4 Obnova časování symbolů

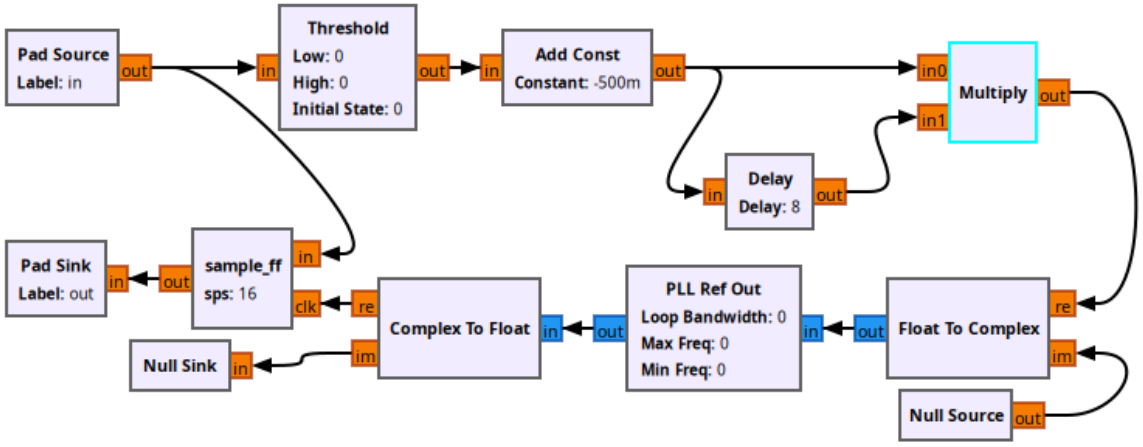
Pro získání původních modulačních dat PSK31 je třeba signál podvzorkovat s periodou rovnou symbolové periodě. Pro dosažení minimální chybovosti by okamžiky vzorkování měly být uprostřed intervalu jednotlivých symbolů. K tomuto účelu byl vytvořen blok *Symbol Timing Recovery*, který využívá metodu popsanou v [39].

Pro určení ideálního času vzorkování je vytvořen pomocný hodinový signál, který vznikne úpravou vstupního signálu. Vstupní signál je nejdříve převeden na symetrický pravoúhlý pomocí oboustranného omezovače. Poté je rozdělen do dvou větví, přičemž v jedné větvi je zpožděn o polovinu symbolové periody. Poté jsou signály obou větví spolu vynásobeny. V případě, že vstupní signál mění úroveň s každým symbolem, vznikne pravoúhlý signál s periodou rovnou polovině symbolové periody. Náběžné hrany tohoto signálu jsou uprostřed symbolového intervalu původního signálu a mohou být použity k určení času vzorkování. Princip vytvoření tohoto pomocného signálu je znázorněn na obr 4.6. Jak je z obrázku zřejmé, tento signál nelze ke vzorkování použít, pokud vstupní signál obsahuje dva nebo více stejných symbolů po sobě. Je proto ještě použita smyčka fázového závěsu, která vytváří harmonický signál o stejné frekvenci a fázi.




Obrázek 4.6 Princip vytvoření pomocného hodinového signálu

Blok *Symbol Timing Recovery* byl vytvořen propojením jiných boků GNU Radia a jedná se tedy o tzv. hierarchický blok. Zdrojový kód byl vygenerován pomocí programu GNU Radio Companion. Blokové schéma je na obr. 4.7.




Obrázek 4.7 Blokové schéma bloku pro obnovu časování symbolů

Na začátku řetězce je blok *Pad Source*, který představuje vstup hierarchického bloku. Vstupní signál je převeden na obdélníkový pomocí bloku *Threshold*, který představuje komparátor s hysterezí, kterou lze nastavit pomocí dvou prahových úrovní. Obě tyto úrovně byly zvoleny 0, výstup tedy závisí na znaménku vstupního signálu. Výstupem je pravoúhlý signál o úrovni 0 nebo 1. Pro získání signálu symetrického kolem nuly je odečtena hodnota 0,5 pomocí bloku *Add Const*. Pro zpoždění signálu byl použit blok *Delay*, zpožděný signál je vynásoben s nezpožděným pomocí bloku *Multiply*. Poté je signál převeden na komplexní blokem *Float To Complex*, aby mohl být přiveden na vstup následujícího bloku. Tímto blokem je *PLL Ref Out*, který představuje smyčku fázového závěsu. Jeho výstupem je harmonický signál, který má stejnou frekvenci a fázi

jako jeho vstupní signál. Tento harmonický signál je použit jako hodinový signál bloku *sample_ff*, který vzorkuje původní signál. Blok *Pad Sink* je potom výstupem hiearchického bloku.

Blok *sample_ff* byl vytvořen v Pythonu. Blok odebírá vzorky z obou vstupů a v případě náběžné hrany na hodinovém vstupu uloží aktuální vzorek na signálovém vstupu do výstupního bufferu. Jako náběžná hrana na hodinovém vstupu je vyhodnocen případ, kdy aktuální vzorek je větší nebo roven nule a předchozí vzorek je záporný.

Při tvorbě bloku byl brán zřetel i na korektní propagaci tagů. Každý vzorek na výstupu je opatřen tagem aktuálního vzorku, pokud je dostupný, jinak se použije nejbližší starší tag.

### 4.3.5 Dekódování varicode

Pro dekódování varicode abecedy byl vytvořen blok *varicode_decoder_bb*. Parametry bloku jsou *invert_input* (udává, zda má být vstupní signál invertován) a *print_output* (umožňuje výpis dekódovaného textu).

Blok umožňuje invertovat vstupní signál z důvodu snazšího propojení s blokem *Differential Decoder*. Jeho výstupem je totiž logická 1 v případě, že se po sobě následující vzorky liší (což je způsobeno změnou fáze modulovaného signálu), což ale při použití varicode znamená logickou úroveň 0.

Část kódu funkce *work()* je uvedena níže.

```
for i in in0:
      inp_consumed += 1
      if self.invert_input: i = (0 if i==1 else 1)

      if not (i == 0 and self.prev_bit == 0):
            self.curr_symbol = (self.curr_symbol << 1) + i

      else: #end of symbol
            self.curr_symbol >>= 1
            if self.curr_symbol in varicode_dict:
                  character = varicode_dict[self.curr_symbol]
                  if self.print_output: print(chr(character), end='')
                  out0[outp_index] = character

                  #propagate tags
                  tags = self.get_tags_in_range(0,
                              self.tag_search_start_pos,
                              self.nitems_read(0) + inp_consumed)
                  for t in tags: self.add_item_tag(0,
                              self.nitems_written(0) + outp_index,
                              t.key,
                              t.value)
                  outp_index += 1
                  self.tag_search_start_pos = self.nitems_read(0)
                        + inp_consumed
            self.curr_symbol = 0
            if outp_index >= len(out0): break

      self.prev_bit = i
```

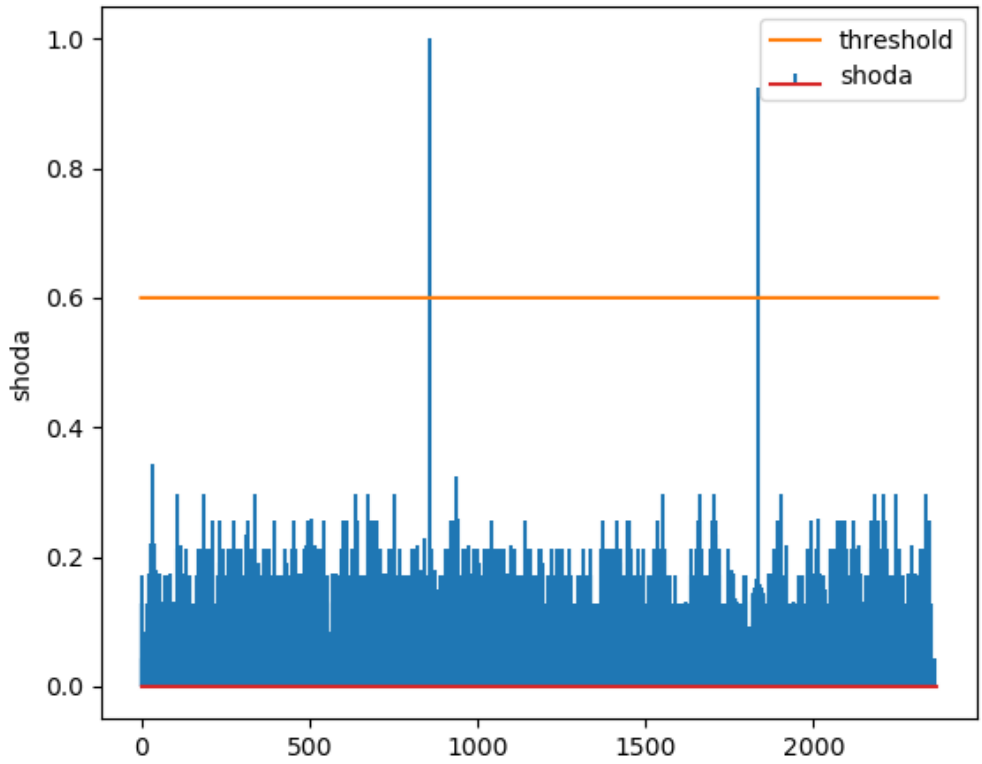
Postupně jsou ze vstupu odebírány vzorky, které představují jednotlivé bity kódovaného textu. Pokud aktuální a předchozí bit nejsou rovny nule (což by značilo konec znaku), jsou přičítány k proměnné *curr_symbol*, která je nejdříve bitově posunuta doleva. Tato proměnná tak obsahuje sekvenci bitů aktuálního znaku. Pokud je rozpoznán konec znaku (aktuální a předchozí bit roven 0), je z proměnné odebrána poslední nula bitovým posunem doprava. Hodnota proměnné se potom použije jako index vyhledávací tabulky s ASCII znaky a tento znak je uložen do výstupního bufferu. Vyhledávací tabulka je realizována pomocí asociativního pole (*dictionary*).

Výstupní vzorky jsou opatřeny tagy všech vstupních vzorků, které odpovídají danému znaku.

### 4.3.6 Dekódování telemetrie

Pro detekci textu telemetrie byl vytvořen blok *telemetry_detect_b*. Jeho parametry jsou *tlm_format* (formát telemetrie – *psat*, *bricsat*, *psat2-psk* nebo *psat2-sstv*), *threshold* (minimální nutná úroveň shody telemetrické zprávy se zadaným vzorem, viz níže) a *use_time_tags* (udává, zda se mají použít časové tagy).

Vstupem bloku je datový proud typu byte, jehož vzorky představují jednotlivé ASCII znaky. V době, kdy přijímaným signálem je pouze šum, tento datový proud obsahuje náhodné znaky. Blok tedy v dekódovaném textu musí telemetrické zprávy najít. Z přijatého textu jsou postupně kopírovány části o délce telemetrické zprávy, přičemž počáteční pozice této části je vždy posunuta o jeden znak. Tento text je poté porovnáván se vzorem telemetrie daného formátu. Výsledkem porovnání je hodnota v rozsahu 0 až 1, přičemž hodnota 1 značí úplnou shodu s daným vzorem a tedy kompletní nepoškozenou zprávu. Pokud je vyhodnocená shoda větší nebo rovna parametru *threshold*, je text vyhodnocen jako platná telemetrická zpráva a je uložen do proměnné pro pozdější dekódování. Zároveň je uložen i čas výskytu a odhad poměru signál/šum. Výchozí hodnota parametru *threshold* byla zvolena 0,6. Tento způsob je výhodný v tom, že umožňuje najít a dekódovat i částečně poškozenou zprávu, kdy jsou některé znaky nahrazeny za jiné. Na obrázku 4.8 je příklad závislosti této shody na pozici v textu pro záznam družice PSAT, který obsahuje dvě telemetrické zprávy, přičemž druhá je částečně poškozena.




Obrázek 4.8 Příklad závislosti hodnoty shody na pozici v textu

Vzor telemetrie má formu pole o délce telemetrické zprávy, jehož prvky jsou textové řetězce s výčtem možných znaků na dané pozici. Je zřejmé, že některé části textu poslouží k identifikaci telemetrie lépe než jiné (ideální je např. volací značka, která je vždy stejná), jednotlivým pozicím v textu tak byla pro posouzení shody přiřazena různá váha. Ta byla zvolena jako převrácená hodnota počtu dovolených znaků na dané pozici. Tyto váhy jsou poté ještě normovány, aby jejich součet byl roven jedné.

Níže je uvedena část zdrojového kódu se vzorem pro telemetrii PSAT. Vzor je uložen v proměnné *pattern*, příslušné váhy potom v poli *weights*.

```
d = "0123456789"
if tlm_format == "psat":
      self.pattern = ["W", "3", "A", "D", "O", "-", "5", " ",
                  "b", "e", "a", "c", "o", "n", " ", "AB", " ",
                  d, d, d, " ", d, d, " ", d, d, " ", d, d, d,
                  " ", d, d, d, " ", "+-", d, d+" "]

self.pattern_len = len(self.pattern)
self.weights = [1/len(s) for s in self.pattern]
self.weights[:] = [x/sum(self.weights) for x in self.weights]
```

Níže je uveden zdrojový kód funkce, která porovnává přijatý text (v proměnné *data*) se vzorem. Funkce pro každý znak zjišťuje, zda je obsažen v textovém řetězci příslušného prvku pole vzoru, pokud ano, k proměnné *match_score* se přičte příslušná váha. Zároveň se do pole *chr_valid* ukládá informace o shodě jednotlivých znaků.

```
def test_pattern_match(self, data):
        chr_valid = [False for i in range(self.pattern_len)]
        match_score = 0
        for i, (p, w) in enumerate(zip(self.pattern, self.weights)):
            if chr(data[i]) in p:
                match_score += w
                chr_valid[i] = True
        return match_score, chr_valid
```

Text s telemetrií, spolu s informací o čase (který je určen pomocí časových tagů), poměru signál/šum, proměnnou *chr_valid*, pozicí v textu a shodě, se z bloku získají voláním metody *get_data()*, poté co je zpracován celý SDR záznam.

Pro dekódování je použit modul *decode_tlm.py*, v případě telemetrie PSAT2 navíc i modul *psat2_dec.py*. Pro každou část textu vyjadřující jednu telemetrickou hodnotu se zjistí, zda neobsahuje chybu (kontrolou hodnot v proměnné *chr_valid*) a pokud ne, je tato hodnota uložena do struktury s telemetrickými daty.

## 4.4 Archiv kepleriánských elementů

Pro predikci dráhy satelitu je třeba znát tzv. kepleriánské elementy dráhy, což je soubor číselných hodnot popisující pohyb daného satelitu. Obvykle jsou udávány v tzv. dvouřádkovém formátu, jsou tedy tvořeny dvěma řádky textu, kde jednotlivé hodnoty mají přesně danou pozici [12].

Jedním z těchto elementů je tzv. epocha, což je čas, ve kterém byly tyto elementy stanoveny[40]. Epocha je udávána ve formátu[41]:

AABBB.CCCCCCCC

kde *AA* jsou poslední dvě číslice roku, *BBB* je pořadové číslo dne v tomto roce, desetinná část *CCCCCCCC* představuje čas vyjádřený jako část dne.

Predikce dráhy satelitu je nejpřesnější pro čas odpovídající epoše, elementy jsou použitelné asi jeden až dva týdny od epochy [31]. Z tohoto důvodu jsou elementy pravidelně aktualizovány.

Pro zpracování SDR záznamů byl vytvořen archiv kepleriánských elementů, který obsahuje elementy s různými epochami. Elementy každé družice jsou uloženy v samostatném souboru. Každé dva řádky těchto souborů představují jeden soubor elementů.

Skript *tle.py* obsahuje funkci *load_tle_from_file()*, pro vyhledání nejvhodnější sady elementů pro daný satelit a čas. Funkce nejdříve převede zadaný čas do formátu epochy, aby mohly být porovnány. Následně otevře soubor archivu pro daný satelit a čte z něj po dvou řádcích. Pokud řádek začíná znaky „1 “ a následující řádek znaky „2 “, jsou tyto řádky považovány za soubor Kepleriánských elementů a je z nich získán hodnota epochy, která je porovnána se zadaným časem. Funkce vrátí textový řetězec s kepleriánskými elementy, pro něž je absolutní hodnota rozdílu epochy a zadaného času nejmenší.

## 4.5 Určení polohy družice vůči Zemi

Součástí archivu je spolu s telemetrickými daty i poloha družice vůči Zemi a vůči pozorovateli v daný čas. Poloha vůči Zemi je určena jako výška družice nad zemí a zeměpisná šířka a délka místa na Zemi, nad kterým se družice nachází. Poloha vůči pozorovateli je určena jako azimut, elevace a vzdálenost. Pro výpočet polohy byla vytvořena funkce *sattelite_position()*, která je součástí skriptu dop_*demod_dec.py*. Funkce využívá knihovnu Skyfield. Zdrojový kód funkce je uveden níže.

```
def sattelite_position(tle, date_time, location, sf_data_dir):
    load = Loader(sf_data_dir)
    ts = load.timescale()
    sat = EarthSatellite(tle[0], tle[1])
    if isinstance(date_time, datetime.datetime):
        t = ts.utc(date_time)
    else:
        t = ts.utc(*date_time)

    subpoint = sat.at(t).subpoint()
    topos = Topos(
        latitude_degrees=location[0],
        longitude_degrees=location[1],
        elevation_m=int(location[2]))
    alt, az, distance = (sat - topos).at(t).altaz()

    return (
         subpoint.latitude.degrees,
         subpoint.longitude.degrees,
         subpoint.elevation.m,
         alt.degrees, az.degrees, distance.km)
```

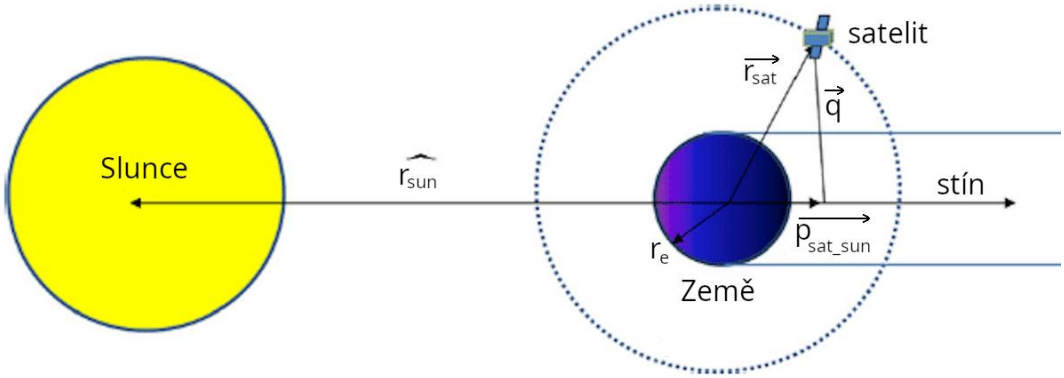
Funkci je předán řetězec s kepleriánskými elementy, datum a čas, pro který má být poloha určena, souřadnice pozorovatele (zeměpisná délka, zeměpisná šířka a nadmořská výška) a adresář s daty knihovny Skyfield. Po inicializaci objektů pro práci s časem je vytvořena proměnná *sat*, která představuje daný satelit. Poloha satelitu vůči středu Země pro konkrétní čas je získána voláním metody *at(t)*. Z této polohy je určeno místo na Zemi nacházející se pod satelitem pomocí metody *subpoint()*. Návratová hodnota této metody je uložena do proměnné *subpoint*. Z ní je následně získána zeměpisná šířka (*subpoint.latitude.degrees*), zeměpisná délka (*subpoint.longitude.degrees*) a výška nad Zemí (*subpoint.elevation.m*).

Pro určení polohy vůči pozorovateli je vytvořena proměnná *topos*, která reprezentuje polohu pozorovatele. Ta je odečtena od proměnné *sat*, čímž se získá jejich vzájemná poloha. Voláním metody *at()* se určí poloha pro daný okamžik a pomocí metody *altaz()* je získán azimut, elevace a vzdálenost satelitu od pozorovatele.

## 4.6 Určení polohy družice vůči Slunci

Pro každý záznam telemetrického archivu je určeno, zda byla družice osvětlena slunečním světlem, nebo byla v zemském stínu a jak dlouho. Tato informace je užitečná např. ve vztahu k teplotě družice nebo napětí akumulátoru. Pro výpočet byl použit

algoritmus pro cylindrický model zemského stínu (ECSM) podle [42], jehož přesnost je pro toto použití dostačující. Tento model předpokládá dokonalý kulový tvar Země a rovnoběžné sluneční paprsky, výsledný zemský stín má proto tvar válce. Model je zobrazen na obr. 4.9.




Obrázek 4.9 Cylindrický model zemského stínu (převzato z [42], upraveno)

Určení stavu osvícení družice sestává z těchto kroků[42]:

1. výpočet polohového vektoru satelitu $\overrightarrow{r_{sat}}$ vztaženého ke středu Země
2. výpočet jednotkového vektoru $\widehat{r_{sun}}$ směřující od středu Země ke Slunci
3. výpočet skalárního součinu $d_{prod}$ mezi $\overrightarrow{r_{sat}}$ a $\widehat{r_{sun}}$
4. výpočet projekce $\overrightarrow{p_{sat\_sun}}$ vektoru $\overrightarrow{r_{sat}}$ na $\widehat{r_{sun}}$
5. výpočet vektoru $\vec{q}$, který začíná na ose stínu v bodě $\overrightarrow{p_{sat\_sun}}$ a končí pozicí satelitu
6. výpočet velikosti vektoru $\vec{q}$
7. pokud je $d_{prod} > 0$, družice je osvícena
8. pokud je $d_{prod} < 0$ a velikost vektoru $\vec{q}$ je menší než poloměr Země $r_e$, družice je osvícena
9. jinak je družice ve stínu

Vlastní implementace algoritmu je uvedena níže.

```
def ecsm_sunlit(r_sat, r_sun):
    r_e = 6378.137
    r_sun_unit = r_sun/la.norm(r_sun)
    d_prod = np.dot(r_sat, r_sun_unit)
    p_sat_sun = (r_sun_unit * np.dot(r_sat, r_sun_unit)
                 / np.dot(r_sun_unit, r_sun_unit))
    q = r_sat - p_sat_sun
    q_mag = la.norm(q)
    if d_prod >= 0:
        return True
    elif q_mag < r_e:
        return False
    else:
        return True
```

Tato funkce je volána funkcí *sat_sunlight_predict()*, která určí polohu Země, Slunce a družice pomocí knihovny Skyfield. Nejdříve je stav osvětlení družice určen pro okamžik měření telemetrie. Poté je hledán okamžik v minulosti, kdy došlo ke změně

tohoto stavu. Funkce postupuje do minulosti s určitým časovým krokem (zvoleno 20s), pokud dojde ke změně stavu, vrátí se o jeden krok zpět a zmenší krok na polovinu. To se opakuje tak dlouho, až je časový krok menší než zadaná konstanta a okamžik změny je tak určen s požadovanou přesností. Doba na světle nebo ve stínu je potom rozdílem počátečního času a času změny v minulosti.

Poté je tato funkce volána znovu, ale jako počáteční okamžik je zadán tento nalezený čas změny a hledá se další okamžik změny osvětlení v minulosti. Rozdíl těchto dvou časů potom odpovídá trvání předchozího stavu osvětlení, pokud tedy např. byla družice v čase telemetrie osvětlena, určí se doba předchozího zastínění.

Následně je tato funkce volána ještě jednou, jako počáteční okamžik je opět použit čas měření telemetrie, ale funkce postupuje v čase do budoucnosti. Z nalezeného času změny v budoucnosti se určí, kolik zbývá času do změny současného stavu osvětlení.

Obě funkce jsou v modulu *sun.py*.

## 4.7 Získání informací o souboru z webu Satnogs.org

Projekt SatNOGS je platforma, jejímž cílem je vytvoření globální sítě pozemních stanic pro příjem satelitního vysílání [43]. Součástí projektu je databáze zaznamenaných pozorování satelitů. Záznamy databáze obsahují pořízený záznam rádiového signálu a informace o podmínkách záznamů. Každý záznam v databázi je opatřen unikátním identifikačním číslem, pomocí kterého lze záznam dohledat.

Pro získání informací o konkrétním záznamu lze využít webového rozhraní, URL adresa je:

https://network.satnogs.org/api/data/?id=ID,

kde *ID* je identifikační číslo záznamu. Data jsou ve formátu JSON.

Pro získání informací o souboru z webu SatNOGS byl vytvořen modul satnogs.py. Data jsou získána pomocí funkce *get_satnogs_obs_info()*, jejíž zdrojový kód je níže.

```
def get_satnogs_obs_info(observation):
    url = f"https://network.satnogs.org/api/data/?id={observation}"
    max_attempts = 3
    for i in range(max_attempts):
        try:
            response = urllib.request.urlopen(url, timeout=20)
        except urllib.error.URLError as e:
            print(f"Error: {e.reason}")
            time.sleep(5)
        except socket.timeout:
            print(f"Error: socket timeout")
            time.sleep(5)
        else:
            resp_body = response.read()
            j = json.loads(resp_body.decode("utf-8"))
            if len(j) == 1: return j[0]
            break
    print(f"Error: Could not get valid Satnogs data")
    return None
```

Pro stažení dat bylo využito knihovny *urllib*, pro dekódování potom knihovny *json*, což jsou standardní knihovny Pythonu. Pokud se nepodaří data získat, funkce se o to po 5 vteřinách pokusí znovu, maximálně třikrát.

Ze stažených dat je získán datum a čas pořízení záznamu, typ satelitu (katalogové číslo NORAD), název pozemní stanice a její poloha. Zdrojový kód funkcí pro získání těchto informací je níže.

```
def satnogs_date_time(satnogs_data):
    dt_str = satnogs_data["start"]
    return [
        int(dt_str[0:4]),   #year
        int(dt_str[5:7]),   #month
        int(dt_str[8:10]),  #day
        int(dt_str[11:13]), #hours
        int(dt_str[14:16]), #minutes
        int(dt_str[17:19])  #seconds
        ]


def satnogs_norad_id(satnogs_data):
    return satnogs_data["norad_cat_id"]

def satnogs_station_name(satnogs_data):
    return satnogs_data["station_name"]

def satnogs_station_location(satnogs_data):
    return [
        satnogs_data["station_lat"],
        satnogs_data["station_lng"],
        satnogs_data["station_alt"]
        ]
```

## 4.8 Dekódování telemetrie PSAT-2 pomocí online dekodéru

Pro dekódování telemetrie družice PSAT-2 byl použit existující dekodér[10], který je propojen s online archivem telemetrických dat[11]. Stránku dekodéru tvoří formulář, který obsahuje dvě pole pro vložení textu – pole pro text telemetrie a pro komentář. Pro odeslání dat je použita HTTP metoda GET, data jsou tedy obsažena v URL adrese. Pokud byl vložen platný text telemetrie, stránka obsahuje pod formulářem dekódovaná telemetrická data. Mohou zde být data aktuálního i historického rámce telemetrie, případně jen jednoho z nich, pokud text telemetrie obsahuje chyby a nelze jej celý dekódovat. V příloze E je ukázka části stránky pod formulářem při dekódování PSK telemetrie, v příloze F potom pro SSTV telemetrii.

Pro dekódování telemetrie pomocí online dekodéru byl vytvořen modul *psat2_dec.py*. Modul obsahuje funkci *get_data()*, jejíž zdrojový kód je uveden níže.

```
def get_data(tlm_str, comment):
    tlm_str = "PSAT-2 " + tlm_str[7:]
    url = ("http://www.urel.feec.vutbr.cz/esl/psat2/psat2tlm.php?tlm="
      + tlm_str.replace(" ", "+") + "&comment="
      + comment.replace(" ", "+"))
    page = download(url)
    data = extract_data(page, comment)
    return data
```

Funkci se předá textový řetězec s telemetrií a text, který je použit jako komentář formuláře. Komentář obsahuje název zdrojového WAV souboru a čas výskytu telemetrického signálu v souboru. Prvních 7 znaků telemetrické zprávy by mělo vždy obsahovat text „PSAT-2 “, pokud obsahuje chybné znaky, dekodér zprávu nezpracuje. Začátek zprávy je proto tímto textem nahrazen. Funkce následně z textu telemetrické zprávy a komentáře vytvoří URL adresu dekodéru a stáhne stránku s dekódovanými daty. Ta je poté zpracována pomocí funkce *extract_data()*, jejíž zdrojový kód je níže.

```
def extract_data(page, comment):
    lines = page.splitlines()
    contains_cur_frame = False
    for i, line in enumerate(lines):
        if "---- Current frame ----" in line:
            contains_cur_frame = True
        if "---- Spreadsheet format ----" in line:
            lines_data = [lines[i+2], lines[i+3]]
            break
    else:
        return None

    data = []
    for line in lines_data:
        d = line[len(comment):].replace(",", ".").split()
        if d: data.append(d[1:])

    if len(data) == 1:
        return ([data[0], None] if contains_cur_frame
                else [None, data[0]])
    else:
        return data
```

Funkce najde řádky, kde jsou telemetrická data uvedena ve formátu pro tabulkový editor (všechny hodnoty na jednom řádku). Nejdříve je odstraněn začátek řádku, který obsahuje komentář, následně jsou nahrazeny desetinné čárky za tečky a řádek je rozdělen na jednotlivá slova. Tím se získá pole obsahující textové řetězce odpovídající jednotlivým telemetrickým údajům. Pokud stránka obsahuje jen jeden dekódovaný rámec telemetrie, funkce musí určit, zda se jedná o aktuální nebo historický. Pokud se na stránce vyskytuje text „---- *Current frame* ----“, jedná se o aktuální rámec, v opačném případě o historický. Chybějící rámec je nahrazen hodnotou *None*.

Data jsou poté dále zpracována pomocí modulu *decode_tlm.py*. U historických rámců telemetrie je třeba určit čas měření telemetrie, který je odlišný od času vysílání telemetrického signálu. U aktuálních telemetrických rámců jsou tyto dva časy považovány za identické. Pro výpočet data a času historického rámce telemetrie byla vytvořena funkce *psat2_hist_frame_datetime()*, jejíž zdrojový kód je níže.

```
def psat2_hist_frame_datetime(dt_cur_frame, cur_frame, hist_frame):
    frame_diff = cur_frame - hist_frame
    if frame_diff < 0:
        frame_diff += 1048576 #32^4
    return dt_cur_frame - datetime.timedelta(seconds=frame_diff*20)
```

Funkce od data a času aktuálního rámce odečte rozdíl čísla aktuálního a historického rámce, který je vynásoben trváním rámce (20 s). Pokud je rozdíl záporný, předpokládá se, že došlo k přetečení počítadla rámců. Číslo rámce je vyjádřeno pomocí čtyř cifer v číselné soustavě o základu 32 (viz. kap. 2.4), v případě přetečení je proto přičtena hodnota $32^4$.

## 4.9 Formát telemetrického archivu

Formát telemetrického archivu je CSV. Jedná se o textový formát používaný pro ukládání tabulkových dat. Každý záznam v souboru začíná na novém řádku a je tvořen položkami oddělenými čárkami. V případě, že text položky obsahuje čárku nebo znak nového řádku, musí být uzavřen do uvozovek. Každý záznam obsahuje stejný počet položek, význam jednotlivých položek tak lze odvodit z pozice v záznamu. První řádek může být použit jako tzv. hlavička souboru, potom obsahuje názvy položek pro jednotlivé sloupce [44].

V případě PSK telemetrie PSAT2 jsou data aktuálního a historického rámce telemetrie uloženy jako dva samostatné záznamy, jinak vždy každý záznam odpovídá jedné telemetrické zprávě. První položkou každého záznamu archivu je název souboru, ze kterého byla telemetrická data získána. Následuje název pozemní stanice použité k příjmu (získán z webu Satnogs nebo zadán jako argument programu) a identifikační číslo Satnogs. Pokud se nejedná o záznam Satnogs, položka je prázdná. Poté je uveden datum a čas vysílání telemetrie ve formátu:

RRRR-MM-DD hh:mm:ss

kde *RRRR* je rok, *MM* je měsíc, *DD* je den, *hh* představuje hodiny, *mm* minuty a *ss* vteřiny. Další položkou je datum a čas vyjádřený v sekundách od vypuštění družice. Poté je uveden čas výskytu telemetrie v souboru, ze kterého byla data získána.

Následně je uvedena pozice satelitu vůči Zemi. Jsou to tři položky - zeměpisná šířka a délka místa na zemi, nad kterým se satelit nachází, a výška satelitu nad zemí. Poté následuje poloha satelitu vůči pozorovateli – elevace, azimut a vzdálenost.

Další položkou je číslo, které udává aktuální stav osvětlení Sluncem – zda je satelit osvětlen (1) nebo je v zemském stínu (0). Následuje čas v sekundách, po který je osvětlen nebo zastíněn, zbývající čas než dojde ke změně stavu osvětlení a doba předchozího stavu osvětlení.

Dalšími údaji jsou průměrný a minimální poměr signál/šum za dobu vysílání telemetrické zprávy a frekvenční offset. Další hodnota udává, zda je záznam kompletní, tedy obsahuje všechny hodnoty telemetrie. Jedná se o číslo 1 nebo 0, číslo 1 značí kompletní záznam. Dále je uveden text telemetrické zprávy.

Poté následují data specifická pro danou družici a typ telemetrie. V případě archivu družic PSAT a BRICSat jsou to dekódovaná telemetrická data, jednotlivé položky odpovídají popořadě hodnotám v tab. 2.1 a tab. 2.2.

V případě archivu PSK PSAT2 je další položkou číslo, které udává, zda se jedná o aktuální (1) nebo historický (0) rámec telemetrie. Následně je uveden datum a čas měření telemetrie, který má stejný formát jako datum a čas vysílání. Poté následují dekódovaná telemetrická data, odpovídající popořadě položkám v tabulce 2.3 (kromě polí *k* až *n*). Archiv SSTV PSAT-2 obsahuje telemetrická data uvedená v příloze F.

Poloha družice vůči Zemi a údaje o osvětlení družice jsou v případě archivu PSAT2 PSK vztaženy vůči datu a času měření telemetrie, poloha vůči přijímací stanici je vztažena k datu a času příjmu signálu.

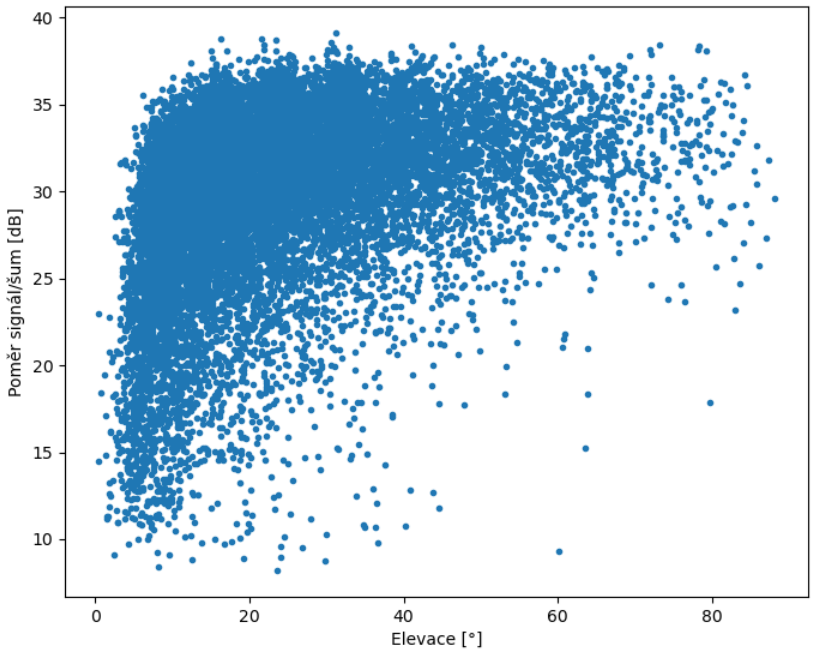
Příklad CSV souboru telemetrie PSAT je uveden v příloze G. Jednotlivé řádky musely být zalomeny, aby se vešly na stránku.

# 5 ANALÝZA ZÍSKANÝCH DAT

Byly zpracovány SDR záznamy družic PSAT a PSAT-2, záznamy družice BricSAT nebyly k dispozici, jelikož krátce po jejím vypuštění došlo k poškození baterie a transpondér družice tak není v provozu. Jelikož telemetrický archiv používá formát CSV, pro analýzu a zobrazení dat lze použít např. program Microsoft Excel.

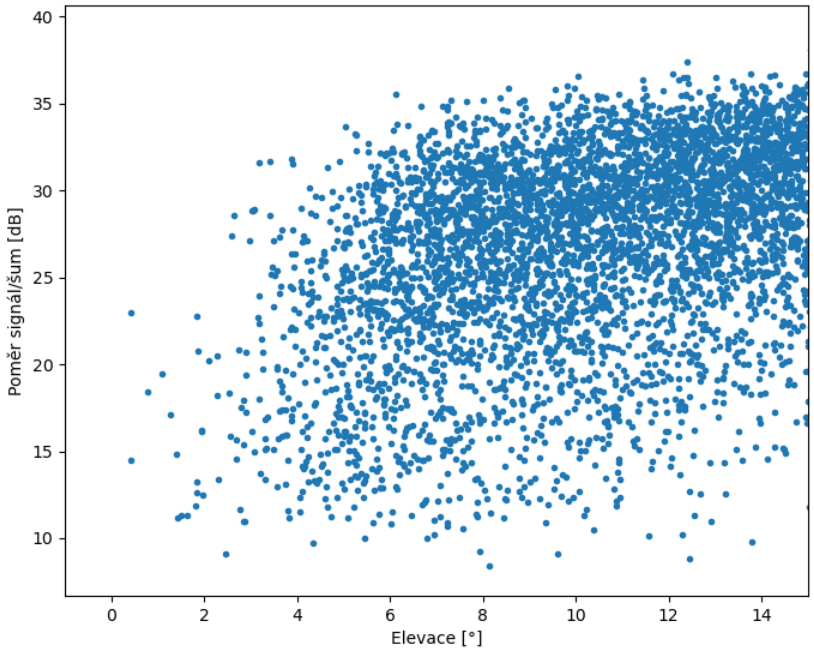
## 5.1 PSAT

Celkem bylo zpracováno 4931 SDR záznamů družice PSAT, z kterých bylo získáno 24992 telemetrických zpráv. Kompletně dekódovat se podařilo 14486 zpráv, ostatní obsahují chyby a byly dekódovány jen částečně. Poškozené zprávy byly většinou na začátku nebo konci záznamu, kdy byla družice nízko nad horizontem. Tomu odpovídá i graf na obr. 5.1, který znázorňuje závislost poměru signál/šum na elevaci družice. Poměr signál/šum byl měřen po FM demodulaci a z grafu je zřejmé, že pro nízké hodnoty elevace byl signál pod šumovým prahem detektoru. Na obr. 5.2 je zobrazena tato závislost pro rozsah elevace 0 – 15°.

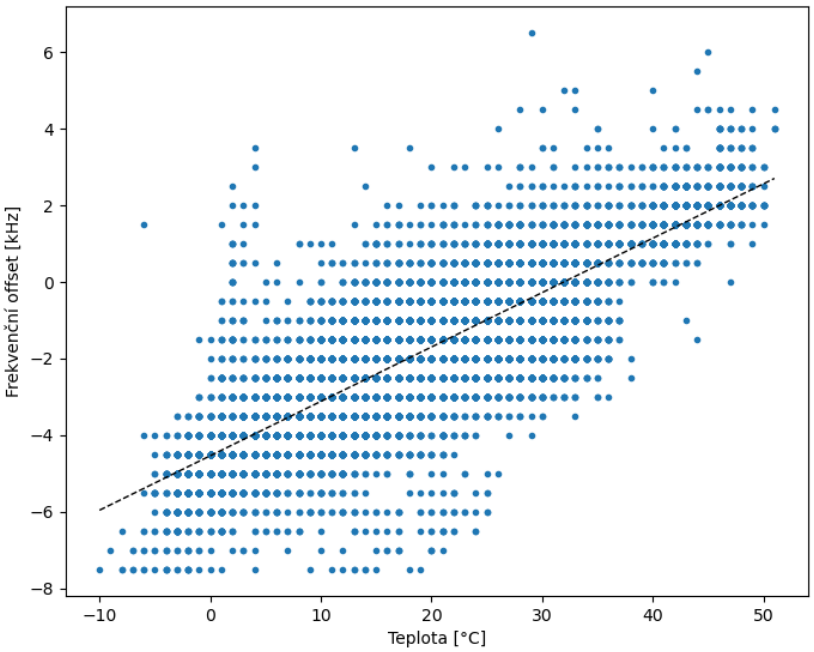



Obrázek 5.1 Závislost poměru signál/šum na elevaci

Na obr. 5.3 je znázorněn frekvenční offset signálu vůči teplotě družice (koncového zesilovače). Při zpracování signálu bylo frekvenční spektrum posouváno po 500Hz, čemuž odpovídá rozlišení zjištěné hodnoty offsetu, přesnost je asi $\pm 1$ kHz. Jak je z grafu patrné, frekvenční offset s teplotou přibližně lineárně roste, avšak vykazuje velký rozptyl hodnot. Změna frekvence je přibližně 140 Hz/°C.



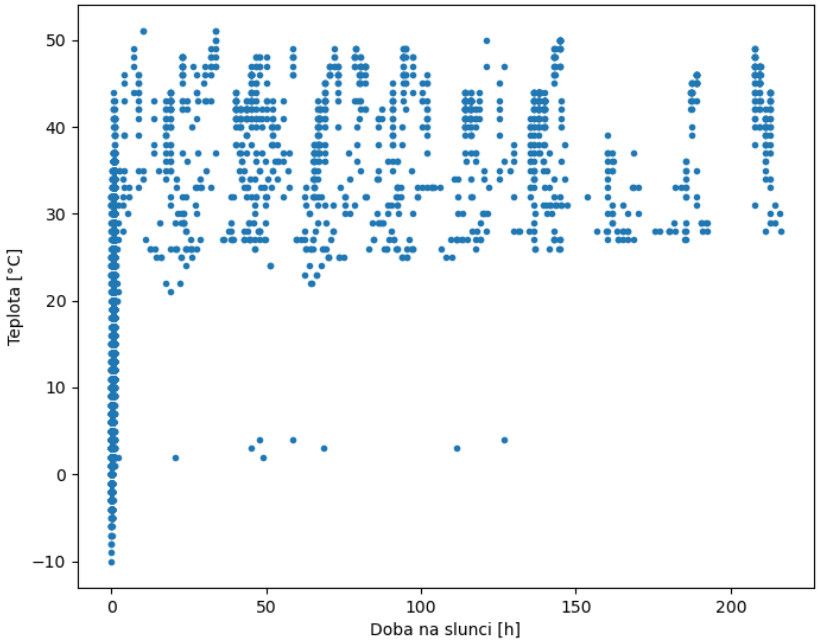

Obrázek 5.2 Závislost poměru signál/šum na elevaci



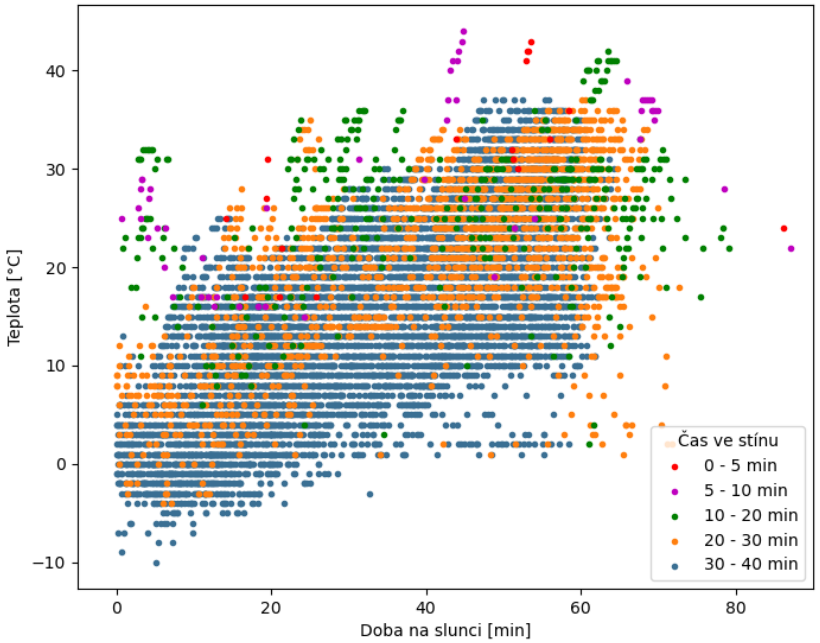

Obrázek 5.3 Závislost frekvenčního offsetu na teplotě

Na obrázku 5.4 je znázorněna závislost teploty družice na době, která uplynula od jejího vylétnutí ze zemského stínu. Doba jednoho obletu družice je přibližně 90 minut, z grafu je tedy patrné, že dochází k případům, kdy je družice nepřetržitě osvětlena několik obletů po sobě. To je způsobeno velkou inklinací dráhy družice. Algoritmus pro výpočet doby osvětlení používá časový krok 20 s, zastínění po kratší dobu tedy nemusí

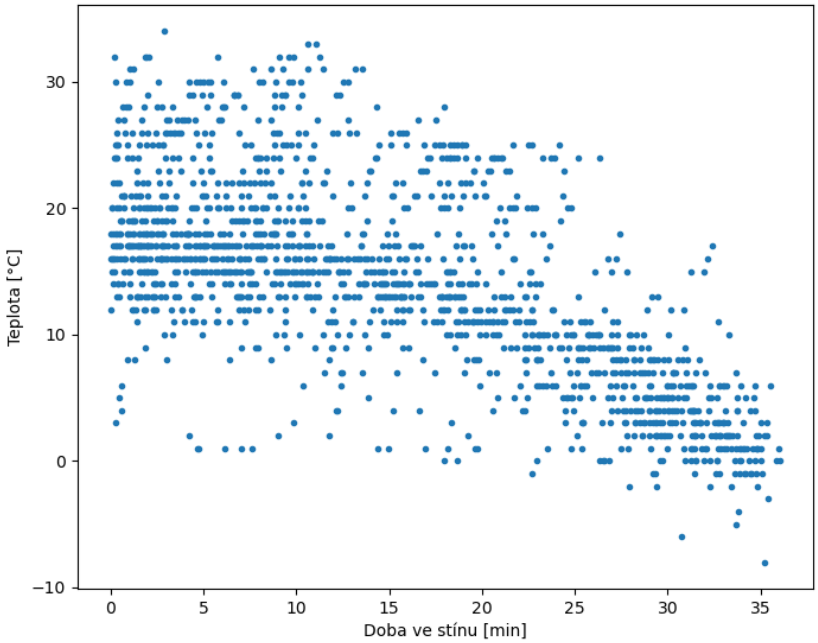
být zaznamenáno. Body o teplotě 0 až 5 °C mezi časy 15 až 130 h jsou pravděpodobně způsobeny chybou příjmu. Na obr. 5.5 je potom tato závislost pro dobu jednoho obletu družice, přičemž data byla rozdělena podle doby, kterou byla předtím ve stínu. Graf na obr. 5.6 zobrazuje závislost teploty na čase pro dobu, kdy je družice ve stínu.




Obrázek 5.4 Závislost teploty družice na době osvětlení Sluncem

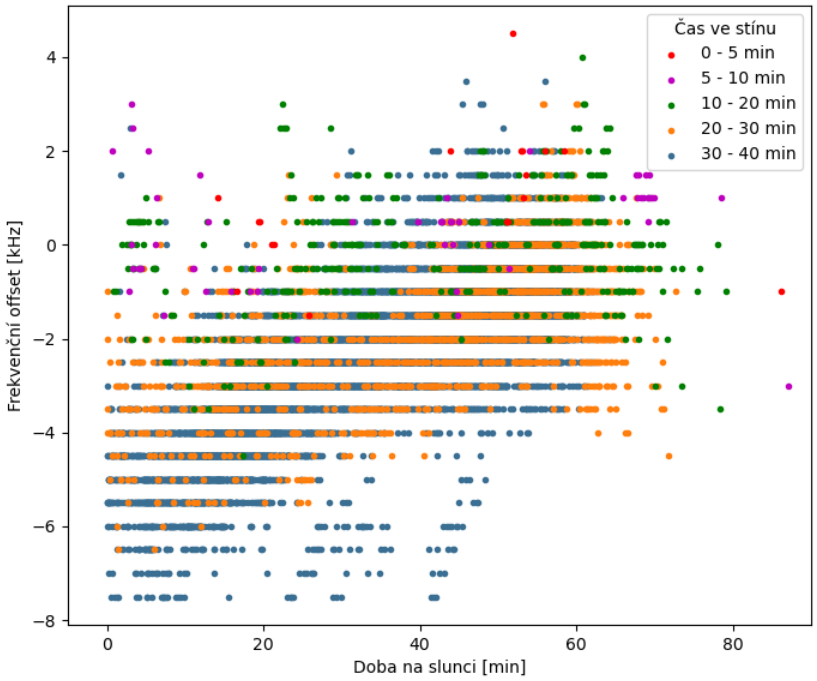



Obrázek 5.5 Závislost teploty družice na době osvětlení Sluncem



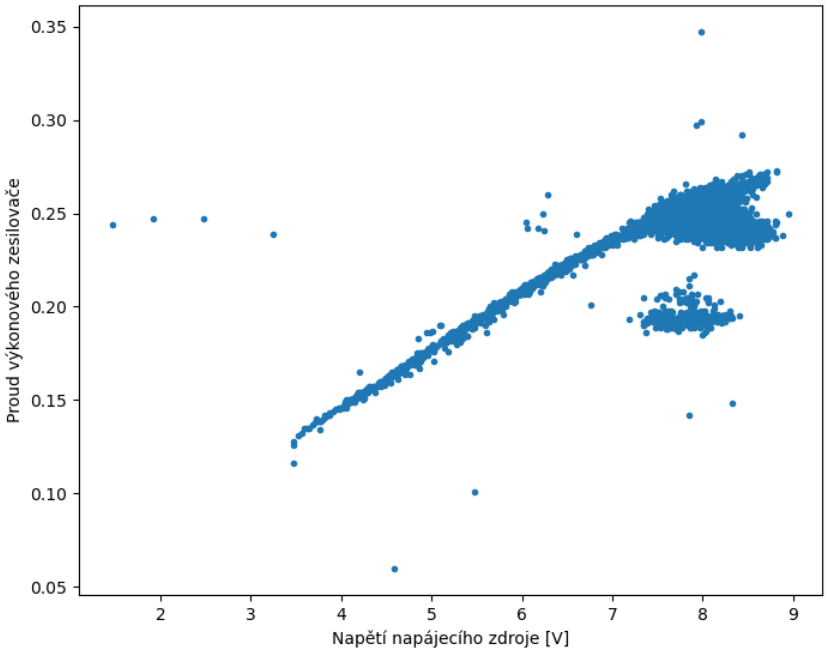

Obrázek 5.6 Závislost teploty družice na době ve stínu

Na obrázku 5.7 je potom zobrazena závislost frekvenčního offsetu vysílače transpondéru na době osvětlení družice, data byla rozdělena podle trvání předchozího stínu.



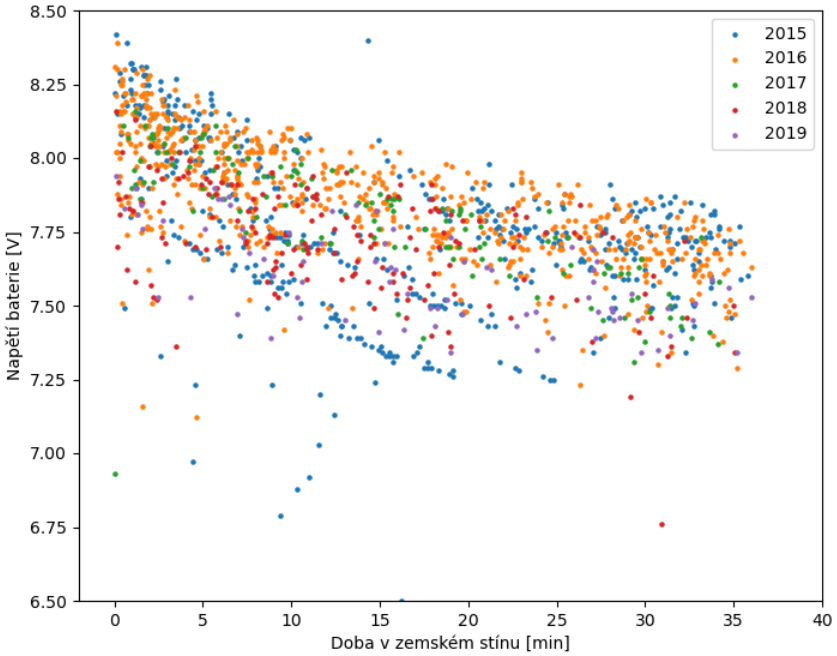

Obrázek 5.7 Závislost frekvenčního offsetu na době osvětlení Sluncem

Závislost proudu koncového zesilovače transpondéru na napětí napájecího zdroje je na obrázku 5.8. Pro napětí v rozmezí 3,5 až 7 V je tato závislost lineární. Změna proudu na 1V je asi 3,15mA, což odpovídá dynamickému odporu 317Ω.




Obrázek 5.8 Závislost proudu koncového zesilovače transpondéru na napětí zdroje

Graf na obr. 5.9 zobrazuje závislost napětí napájecího zdroje na čase v zemském stínu.




Obrázek 5.9 Závislost napětí napájecího zdroje na době ve stínu

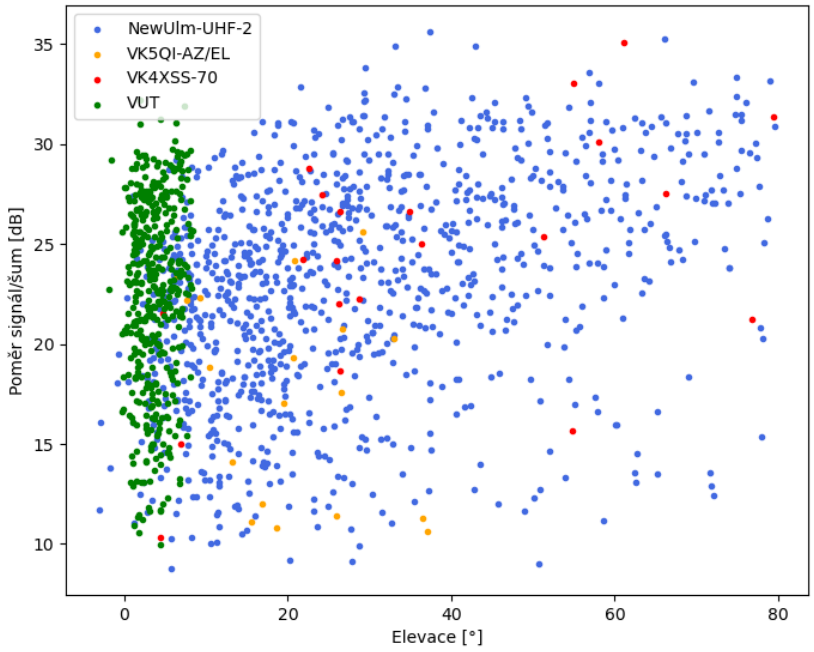
Data za rok 2015 vykazují velký rozptyl hodnot. U dat z následujících let dochází postupně k výraznějšímu poklesu napětí, což lze přisoudit poklesu kapacity akumulátoru vlivem stárnutí.

Transpondér pracoval v 506 případech (2,19%) v módu A a v 22570 případech (97,81%) v módu B, kdy je vysílač transpondéru zapnut jen při zjištění příchozího PSK31 signálu.

## 5.2 PSAT-2

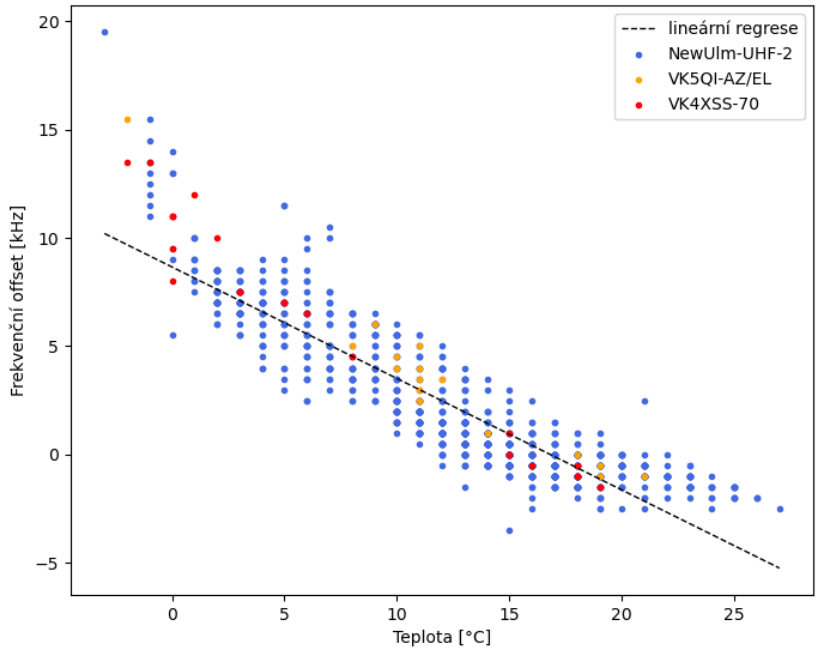
Celkem bylo zpracováno 420 SDR záznamů družice PSAT-2, 160 pořízených na VUT v Brně a 260 získaných z internetových stránek Satnogs. Z nich bylo získáno 2220 telemetrických zpráv PSK (1715 nepoškozených) a 1609 telemetrických zpráv SSTV (1458 nepoškozených).

Na obr. 5.10 je závislost poměru signál/šum na elevaci. Tato závislost není tak výrazná, jako je tomu v případě záznamů PSAT. Z grafu je patrné, že záznamy z VUT byly pořízeny pro přelet družice nízko nad horizontem, což je dáno dráhou družice.



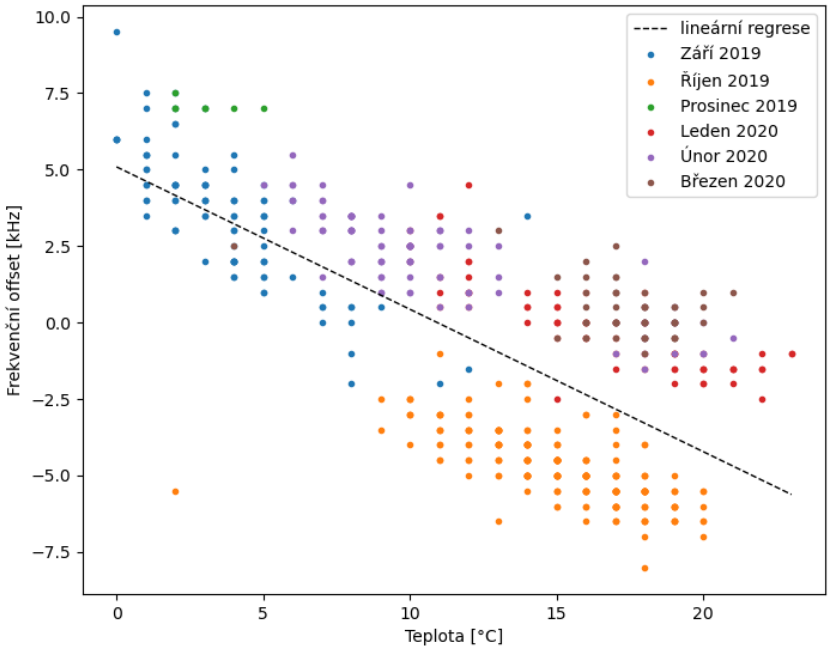

Obrázek 5.10 Závislost poměru signál/šum na elevaci

Na obr. 5.11 je zjištěná závislost frekvenčního offsetu vysílače transpondéru na teplotě pro záznamy Satnogs. Data byla rozdělena podle pozemních stanic, pomocí kterých byly záznamy pořízeny, neboť frekvenční offset může být ovlivněn i přijímačem. Přesnost určení frekvence je asi $\pm 1$ kHz. Na rozdíl od družice PSAT frekvence s teplotou klesá, a to mnohem výrazněji. Pro teploty v rozmezí 0°C – 20°C je tato závislost lineární, změna frekvence je přibližně -523 Hz/°C.



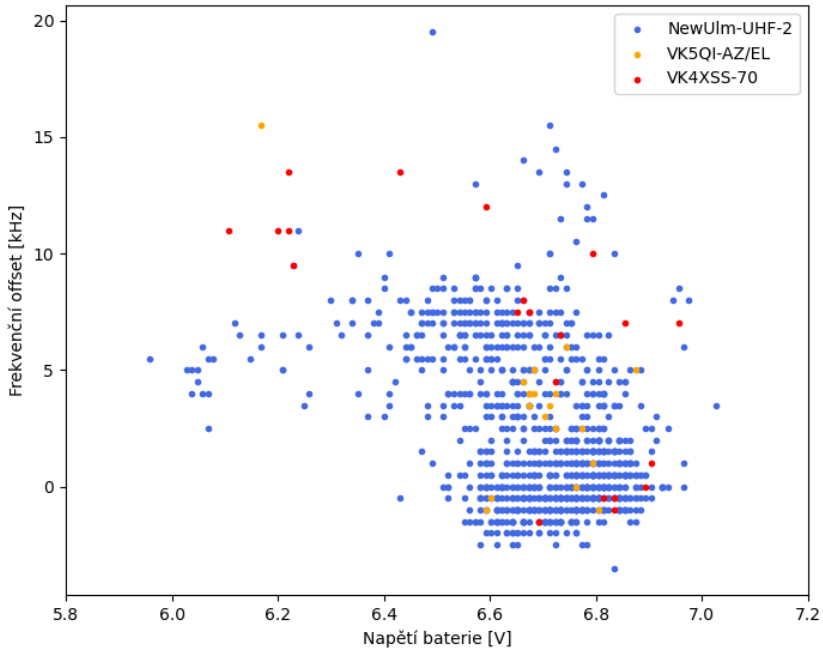

Obrázek 5.11 Závislost frekvenčního offsetu na teplotě, záznamy Satnogs

Na obrázku 5.12 je znázorněna stejná závislost pro záznamy pořízené na VUT v Brně. Z grafu je patrné, že frekvenční offset závisí na datu pořízení záznamu, což je pravděpodobně způsobeno rozdílným nastavením přijímače



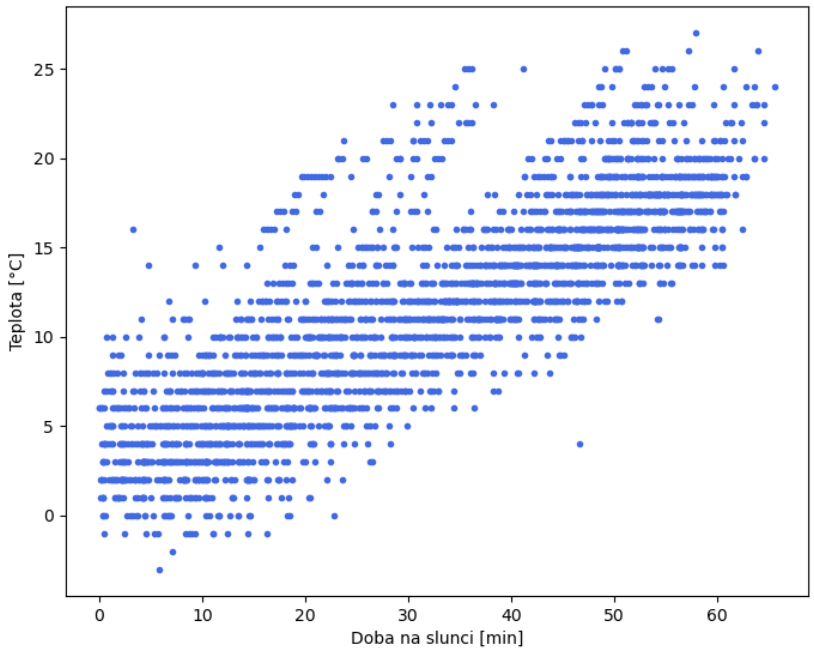

Obrázek 5.12 Závislost frekvenčního offsetu na teplotě, záznamy z VUT

Graf na obrázku 5.13 znázorňuje závislost frekvenčního offsetu vysílače transpondéru na napětí baterie. Není zřejmé, že by mělo napětí na frekvenci vysílače vliv.



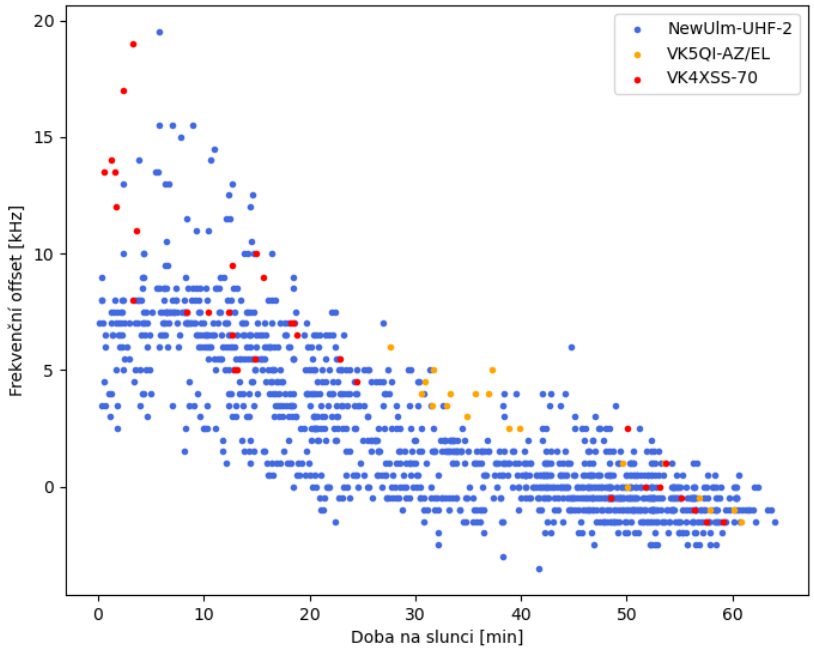

Obrázek 5.13 Závislost frekvenčního offsetu na na napětí baterie

Závislost teploty na době osvětlení družice Sluncem je na obr. 5.14. Teplota je značně závislá na provozu transpondéru.



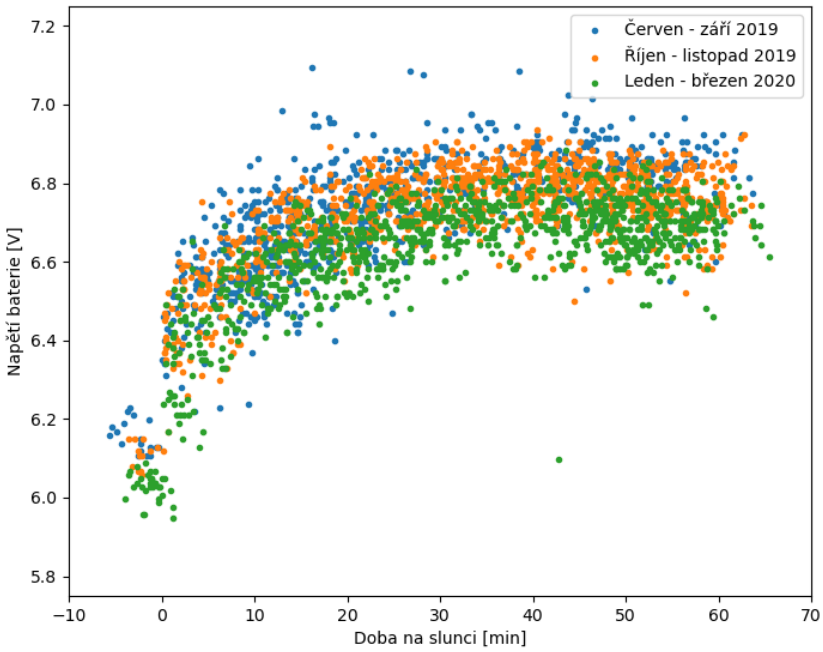

Obrázek 5.14 Závislost teploty na době osvětlení družice Sluncem

Graf na obr. 5.15 znázorňuje závislost frekvenčního offsetu vysílače transpondéru na době osvětlení sluncem. K ustálení frekvence vysílače dojde asi po 40 minutách.




Obrázek 5.15 Závislost frekvenčního offsetu na době osvětlení Sluncem

Závislost napětí baterie na době osvětlení družice je na obr. 5.16. Data byla rozdělena podle data pořízení záznamu. Z grafu je zřejmé, že postupem času došlo k poklesu napětí, kterého baterie dosahuje.
Odebíraný proud se ve většině případů pohybuje v rozmezí 305 mA až 335 mA a není zde zjevná závislost na napětí baterie, jako je tomu u družice PSAT.



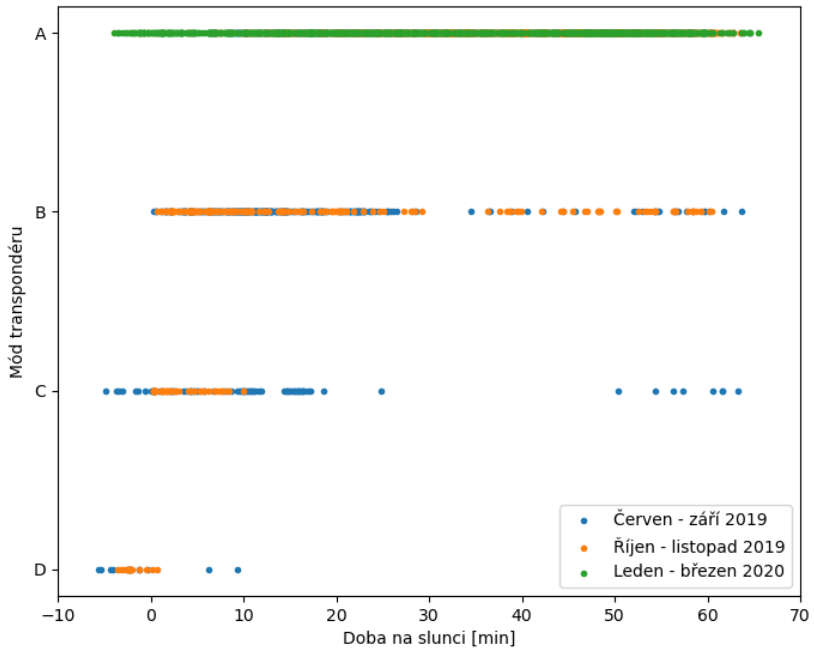

Obrázek 5.16 Závislost napětí napájecího zdroje na době osvětlení Sluncem

Jak vyplývá z tabulky 5.1, transpondér družice pracoval ve většině případů v módu A nebo B, družice tedy měla dostatek energie pro plnohodnotný provoz transpondéru. V módu B je pouze omezeno vysílání SSTV signálu, pokud není detekován žádný příchozí PSK31 signál[5].

Tabulka 5.1 Četnosti módů transpondéru PSAT-2

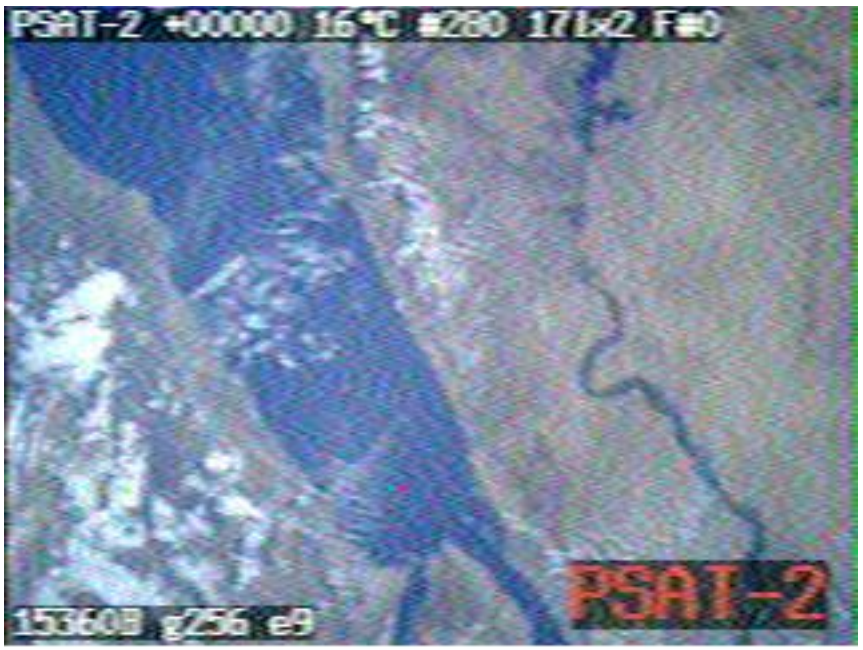
| Mód | Četnost |
|---|---|
| A | 3305 (79,79%) |
| B | 645 (15,57%) |
| C | 158 (3,81%) |
| D | 34 (0,82%) |

Na obrázku 5.17 je závislost módu transpondéru na čase po vylétnutí družice ze stínu. Data byla rozdělena podle data pořízení záznamů. Z grafu je patrné, že postupem času transpondér méně často pracoval v úsporných módech. V případě zpracovaných dat za leden až březen 2020 byl použit pouze mód A, při kterém je odebírán největší proud. Tím lze vysvětlit pokles maximálního napětí baterie. Pravděpodobně je to způsobeno tím, že v průběhu času byly měněny hodnoty napětí, při kterých dochází k přepínání módů.

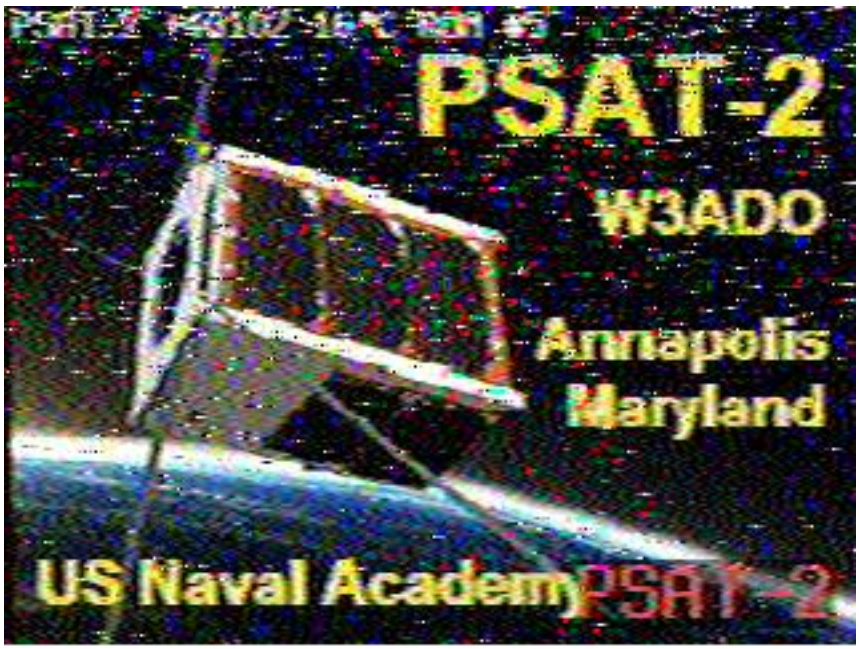



Obrázek 5.17 Závislost módu transpondéru na době osvětlení Sluncem

Vytvořený program byl použit i k uložení částí WAV souborů záznamů, které obsahují SSTV signál. Na obr. 5.18 až 5.20 je porovnání obrazové kvality dekódovaných obrázků, a to v závislosti na poměru signál/šum telemetrického signálu, který byl vysílán zároveň s SSTV signálem. K dekódování byl použit program MMSSTV [45].




Obrázek 5.18 SSTV obrázek, poměr signál/šum 29 dB




Obrázek 5.19 SSTV obrázek, poměr signál/šum 18 dB

Obrázek 5.20 SSTV obrázek, poměr signál/šum 15 dB

# 6 ZÁVĚR

V práci byla popsána struktura vysílaných telemetrický dat družic PSAT, PSAT-2 a BRICSat. Dále byl teoreticky popsán Dopplerův jev a jeho vliv na rádiový signál vysílaný družicí. V další části práce je představen počítačový program, který byl vytvořen pro zpracování SDR záznamů signálu družic. Program nejdříve u záznamů odstraní Dopplerův jev. Následně je signál demodulován a dekódován a telemetrická jsou uložena do vytvořeného archivu ve formátu CSV. Spolu s telemetrickými daty se ukládá mj. i název souborů SDR záznamů, datum a čas vysílání telemetrie, poměr signál/šum po FM demodulaci a poloha družice vůči Zemi. Záznamy jsou také doplněny o informaci, zda byla družice osvícena slunečním světlem, nebo byla v zemském stínu, a jak dlouho. Program umožňuje i uložení demodulovaných úseků záznamů s SSTV signálem pro zpracování jinými programy.
V následující části je provedena analýza dat, která byla získána zpracování záznamů družic PSAT a PSAT-2. Záznamy družice BRICsat nebyly k dispozici.
Zajímavá je především závislost frekvence vysílače transpondéru na teplotě a na době, po jakou je družice vystavena slunečnímu světlu. Tato závislost je značná hlavně u družice PSAT-2, kde může představovat problém při příjmu signálu družice. V případě budoucích návrhů transpondéru by proto měla být snaha o dosažení lepší teplotní stability frekvence oscilátoru vysílače.

# LITERATURA


[1] PSAT (NO-84). AMSAT [online]. [cit. 2019-12-07]. Dostupné z: https://www.amsat.org/two-way-satellites/psat-no-84/

[2] BRICSAT2 and PSAT2 Designated Navy-OSCAR 103 (NO-103) and Navy-OSCAR 104 (NO-104). AMSAT [online]. [cit. 2019-12-07]. Dostupné z: https://www.amsat.org/bricsat2-and-psat2-designated-navy-oscar-103-no-103-and-navy-oscar-104-no-104/

[3] BRICSat-P (NO-83, PSAT-B). AMSAT [online]. [cit. 2019-12-07]. Dostupné z: https://www.amsat.org/two-way-satellites/bricsat-p-no-83-psat-b/

[4] URBANEC, T., P. VÁGNER a M. KASAL. P-sat Transponder WEB Specification. EXPERIMENTAL SATELLITES LABORATORY [online]. 2015 [cit. 2019-12-07]. Dostupné z: http://www.urel.feec.vutbr.cz/esl/files/Projects/PSAT/P%20sat%20transponder%20WEB%20spec02.htm

[5] POVALAČ, Aleš. PSAT-2 PSK/SSTV transponder. GitHub [online]. 2019 [cit. 2019-12-07]. Dostupné z: https://github.com/alpov/PSAT-2/blob/master/README.md

[6] URBANEC, T., P. VÁGNER a M. KASAL. BricSat Transponder WEB Specification. EXPERIMENTAL SATELLITES LABORATORY [online]. 2015 [cit. 2019-12-07]. Dostupné z:

http://www.urel.feec.vutbr.cz/esl/files/Projects/BRICsat/Bricsat%20transponder%20WEB%20spec02.htm

[7] Czech Space Office. Transpondéry z VUT v Brně pracují na dvou družicích Vojenské námořní akademie US. GitHub [online]. 2015 [cit. 2019-12-07]. Dostupné z: https://www.technickytydenik.cz/rubriky/denni-zpravodajstvi/transpondery-z-vut-v-brne-pracuji-na-dvou-druzicich-vojenske-namorni-akademie-us_30856.html

[8] PSK31. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation [cit. 2019-12-07]. Dostupné z: https://en.wikipedia.org/wiki/PSK31

[9] KARTY, Steven L. PSK31 Spec. ARRL [online]. [cit. 2019-12-07]. Dostupné z: http://www.arrl.org/psk31-spec

[10] PSAT-2 TLM decoder. EXPERIMENTAL SATELLITES LABORATORY [online]. [cit. 2019-12-07]. Dostupné z: http://www.urel.feec.vutbr.cz/esl/psat2/psat2tlm.php

[11] PSAT2 telemetry. Tabulky Google [online]. [cit. 2019-12-07]. Dostupné z: https://docs.google.com/spreadsheets/d/1X_4X0AgCshNDFIrDecOif41eRRwNzOKZwRn9tjI5_hs/edit?usp=sharing

[12] KASAL, M. Směrové a družicové spoje. Skripta FEKT VUT, 2003. ISBN 80-214-2496-6.

[13] Python [online]. [cit. 2019-12-07]. Dostupné z: https://www.python.org/

[14] GNU Radio [online]. [cit. 2019-12-07]. Dostupné z: https://www.gnuradio.org/

[15] Skyfield [online]. [cit. 2019-12-07]. Dostupné z: https://rhodesmill.org/skyfield/

[16] Argparse — Parser for command-line options, arguments and sub-commands. Python [online]. [cit. 2019-12-07]. Dostupné z: https://docs.python.org/3/library/argparse.html

[17] Datetime — Basic date and time types. Python [online]. [cit. 2019-12-07]. Dostupné z: https://docs.python.org/3/library/datetime.html

[18] Math — Mathematical functions. Python [online]. [cit. 2019-12-07]. Dostupné z: https://docs.python.org/3/library/math.html

[19] SciPy [online]. [cit. 2019-12-07]. Dostupné z: https://www.scipy.org/

[20] Csv — CSV File Reading and Writing. Python [online]. [cit. 2019-12-07]. Dostupné z: https://docs.python.org/3/library/csv.html

[21] Re — Regular expression operations. Python [online]. [cit. 2019-12-07]. Dostupné z: https://docs.python.org/3/library/re.html

[22] Urllib — URL handling modules [online]. [cit. 2020-04-24]. Dostupné z: https://docs.python.org/3/library/urllib.html

[23] Json — JSON encoder and decoder [online]. [cit. 2020-04-24]. Dostupné z: https://docs.python.org/3/library/json.html

[24] TutorialsCoreConcepts. GNU Radio [online]. [cit. 2019-12-07]. Dostupné z: https://wiki.gnuradio.org/index.php/TutorialsCoreConcepts

[25] MÜLLER, Marcus. Behind the Veil: A Peek at GNU Radio's Buffer Architecture. GNU Radio [online]. [cit. 2019-12-07]. Dostupné z: https://www.gnuradio.org/blog/2017-01-05-buffers/

[26] Why can't we do loops? GNU Radio [online]. [cit. 2019-12-07]. Dostupné z: https://wiki.gnuradio.org/index.php/FAQ#Why_can.27t_we_do_loops.3F

[27] OutOfTreeModules. GNU Radio [online]. [cit. 2019-12-07]. Dostupné z: https://wiki.gnuradio.org/index.php/OutOfTreeModules

[28] Hierarchical Blocks. GNU Radio [online]. [cit. 2019-12-07]. Dostupné z: https://wiki.gnuradio.org/index.php/GNURadioCompanion#Hierarchical_Blocks

[29] Guided Tutorial Programming Topics. GNU Radio [online]. [cit. 2019-12-07]. Dostupné z: https://wiki.gnuradio.org/index.php/Guided_Tutorial_Programming_Topics

[30] GNURadioCompanion. GNU Radio [online]. [cit. 2019-12-097]. Dostupné z: https://wiki.gnuradio.org/index.php/GNURadioCompanion

[31] Earth Satellites. Skyfield [online]. [cit. 2019-12-07]. Dostupné z: https://rhodesmill.org/skyfield/earth-satellites.html

[32] JACOBSEN, Eric. Handling Spectral Inversion in Baseband Processing. DSPRelated.com [online]. February 11, 2008 [cit. 2019-12-07]. Dostupné z: https://www.dsprelated.com/showarticle/51.php

[33] Scipy.interpolate.interp1d. SciPy [online]. [cit. 2019-12-07]. Dostupné z: https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.interp1d.html

[34] Filtering for PSK31 demodulation. StackExchange [online]. [cit. 2019-12-07]. Dostupné z: https://ham.stackexchange.com/questions/7703/filtering-for-psk31-demodulation

[35] WHEATLE, Moe. PSKCore.DLL Software Specification and Technical Guide. StackExchange [online]. September 24, 2008 [cit. 2019-12-07]. Dostupné z: https://www.moetronix.com/ae4jy/files/pskcoredll141.pdf

[36] Signal-to-noise ratio. In: Wikipedia: the free encyclopedia [_online]. San Francisco (CA): Wikimedia Foundation [cit. 2020-04-25]. Dostupné z: https://en.wikipedia.org/wiki/Signal-to-noise_ratio

[37] BRUCHANOV, Martin. Formats of slow-scan TV transmission [online]. 2019 [cit. 2020-04-27]. Dostupné z: https://www.sstv-handbook.com/download/sstv_04.pdf

[38] PSK31 SSTV spectrum [online]. [cit. 2020-04-26]. Dostupné z: http://aprs.org/PSAT2/SSTV/PSK31-SSTV-spectrum-cr.png

[39] Obnova časování symbolů STR. ŽALUD, Václav. Moderní radioelektronika. Praha: BEN - technická literatura, 2000, s. 251-252. ISBN 80-86056-47-3.

[40] Keplerian Elements Tutorial. AMSAT [online]. [cit. 2019-12-07]. Dostupné z: https://www.amsat.org/keplerian-elements-tutorial/

[41] NORAD Two-Line Element Set Format. CelesTrak [online]. [cit. 2019-12-07]. Dostupné z: https://www.celestrak.com/NORAD/documentation/tle-fmt.php

[42] SRIVASTAVA, Vineet K., ASHUTOSH, M. PITCHAIMANI a B.S. CHANDRASEKHAR. Eclipse prediction methods for LEO satellites with cylindrical and cone geometries: A comparative study of ECSM and ESCM to IRS satellites. Astronomy and Computing. 2013, 2, 11-17. DOI: 10.1016/j.ascom.2013.06.001. ISSN 22131337. Dostupné také z: https://linkinghub.elsevier.com/retrieve/pii/S2213133713000206

[43] About SatNOGS [online]. [cit. 2020-04-27]. Dostupné z: https://satnogs.org/about/

[44] Comma-separated values. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation [cit. 2019-12-09]. Dostupné z: https://en.wikipedia.org/wiki/Comma-separated_values

[45] MMSSTV - HamSoft. HamSoft [online]. [cit. 2020-05-19]. Dostupné z: https://hamsoft.ca/pages/mmsstv.php

# SEZNAM PŘÍLOH

# A PŘEHLED ARGUMENTŮ PROGRAMU DOP_DEMOD_DEC.PY

| Argument | Význam |
|---|---|
| -h, --help | Nápověda (výpis možných argumentů) |
| -i INPUT_FNAME | Názvy vstupních souborů |
| -o OUTPUT_FNAME | Název výstupního souboru |
| -w, --owerwrite | Výstupní CSV soubor se přepíše, místo připsání dat na konec |
| -W, --skip-if-exist | Pokud výstupní soubor existuje, zpracování záznamu se přeskočí |
| --output-text | Výstupní soubor pro PSK31 text |
| -t DATE_TIME | Datum a čas, čísla oddělená mezerou (rok, měsíc, den, hodiny, minuty, vteřiny) |
| -s {psat,psat2,bricsat} | Satelit |
| -H, --add-csv-header | Do výstupního CSV souboru se zapíše hlavička s názvy jednotlivých polí |
| -T {psk,sstv} | Typ telemetrie PSAT-2 |
| --telemetry-decoder-threshold | Minimální nutná shoda telemetrie s příslušným vzorem |
| -F RECEIVER_FREQ, --receiver-freq RECEIVER_FREQ | Frekvence, na kterou byl nastaven přijímač |
| -I, --invert-spectrum | Provede se inverze kmitočtového spektra |
| -B FILT_CUTOFF [_FILT_CUTOFF ...], --filt-cutoff FILT_CUTOFF [_FILT_CUTOFF ...] | Mezní frekvence vstupního filtru |
| --location LOCATION | Poloha přijímače (zeměpisná šířka, délka a nadmořská výška) |
| -S SSTV_FNAME, --sstv SSTV_FNAME | Název výstupního souboru pro uložení SSTV signálu |
| -d, --doppler | Provede se odstranění Dopplerova jevu |
| -D, --no-doppler | Neprovede se odstranění Dopplerova jevu |
| -n UPDATE_PERIOD, --doppler-update-period UPDATE_PERIOD | Udává, po kolika vteřinách má být přepočítán Dopplerův posun frekvence |
| --tle-file TLE_FILE | Název souboru s kepleriánskými elementy |
| -c OFFSET [_OFFSET ...], --const-offset OFFSET [_OFFSET ...] | Výčet hodnot frekvenčního offsetu, o které mát být posunuto frekvenční spektrum signálu |
| -C F_MIN F_MAX F_STEP, --const-offset-range F_MIN F_MAX F_STEP | Rozsah a krok frekvenčního offsetu, o které mát být posunuto frekvenční spektrum signálu |
| -N [_SATNOGS], --satnogs [_SATNOGS] | Informace o vstupním souboru mají být zjištěny z satnogs.org |
| -x TEMP_FNAME, --temp TEMP_FNAME | Název dočasného souboru pro uložení signálu s provedenou korekcí Dopplerova jevu |
| -K, --keep-temp | Po skončení běhu programu se nesmaže dočasný soubor s provedenou korekcí Dopplerova jevu |

# B PŘEHLED ARGUMENTŮ PROGRAMU DOPPLER.PY

| Argument | Význam |
|---|---|
| -h, --help | Nápověda (výpis možných argumentů) |
| -i INPUT_FNAME | Název vstupního souboru |
| -o OUTPUT_FNAME | Název výstupního souboru |
| -O OUTPUT_FNAME_T | Začátek názvu výstupního souboru, k němuž je připojen datum a čas |
| -d, --doppler | Provede se odstranění Dopplerova jevu |
| -I, --invert-spectrum | Provede se inverze frekvenčního spektra |
| -c CONST_OFFSET, --const-offset CONST_OFFSET | Provede se posun frekvenčního spektra o zadanou hodnotu |
| -l LINEAR_CORRECTION , --linear-correction LINEAR_CORRECTION | Provede se lineární korekce frekvenčního offsetu, zadají se dvojice hodnot představující čas a frekvenci |
| -t DATE_TIME | Datum a čas, číselné hodnoty oddělené mezerou |
| -f FREQ | Frekvence vysílače transpondéru |
| -s {psat,psat2,bricsat}, --sattelite {psat,psat2,bricsat} | Satelit |
| --location LOCATION | Poloha přijímače |
| --tle-file TLE_FILE | Soubor s kepleriánskými elementy |
| -n DOPPLER_UPDATE_PERIOD | Udává, po kolika vteřinách má být přepočítán Dopplerův posun frekvence |
| -q, --quiet | Omezí se výpis textu do konzole |

# C INSTALACE PROGRAMU

Program vyžaduje k běhu operační system UNIXového typu (testováno s operačním systémem Arch linux) s balíčky *gnuradio* verze 3.8, *python3*, *python-numpy*, *python-scipy* a jejich závislostmi. Dále je potřeba knihovna *Skyfield* pro Python, pro instalaci je potřeba balíček *python3-pip*. Instalace se provede pomocí příkazu:

```
sudo pip install skyfield
```

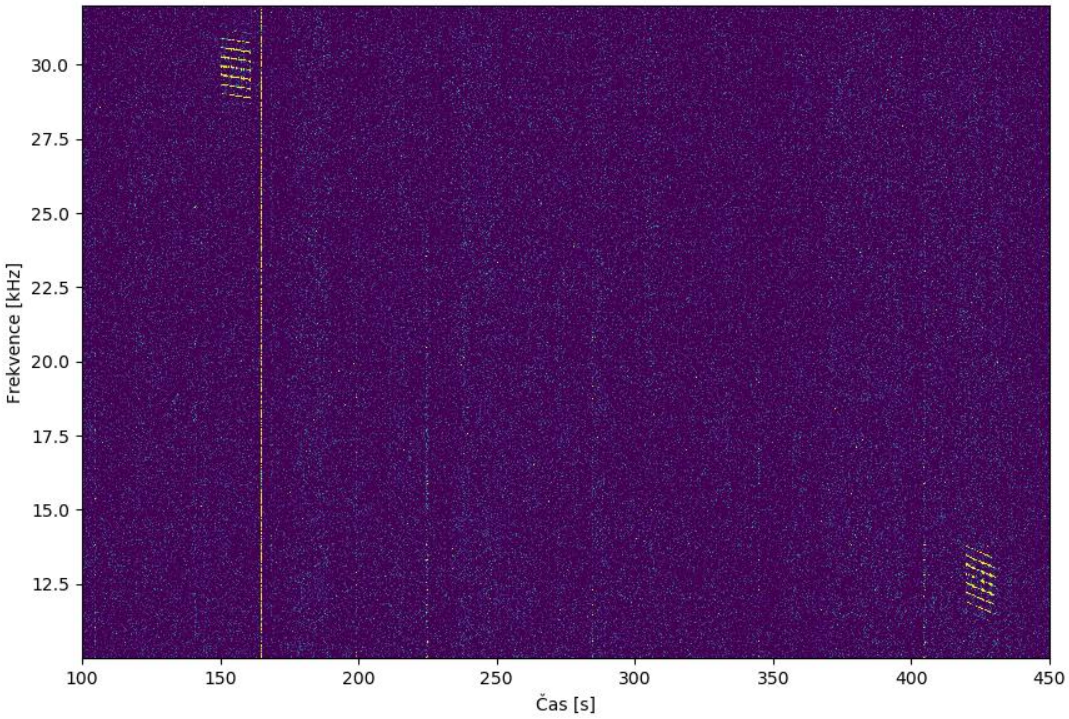
Program využívá instalační skript, který byl vygenerován knihovnou GNU Radio (programem *gr_modtool*). Při instalaci jsou potřeba balíčky *make*, *cmake*, *gcc* a *boost*. Po instalaci programu je lze odstranit. Pro instalaci je třeba v adresáři *gr-gr_telemetry_archive/build* spustit příkazy:

```
cmake ..
sudo make install
```

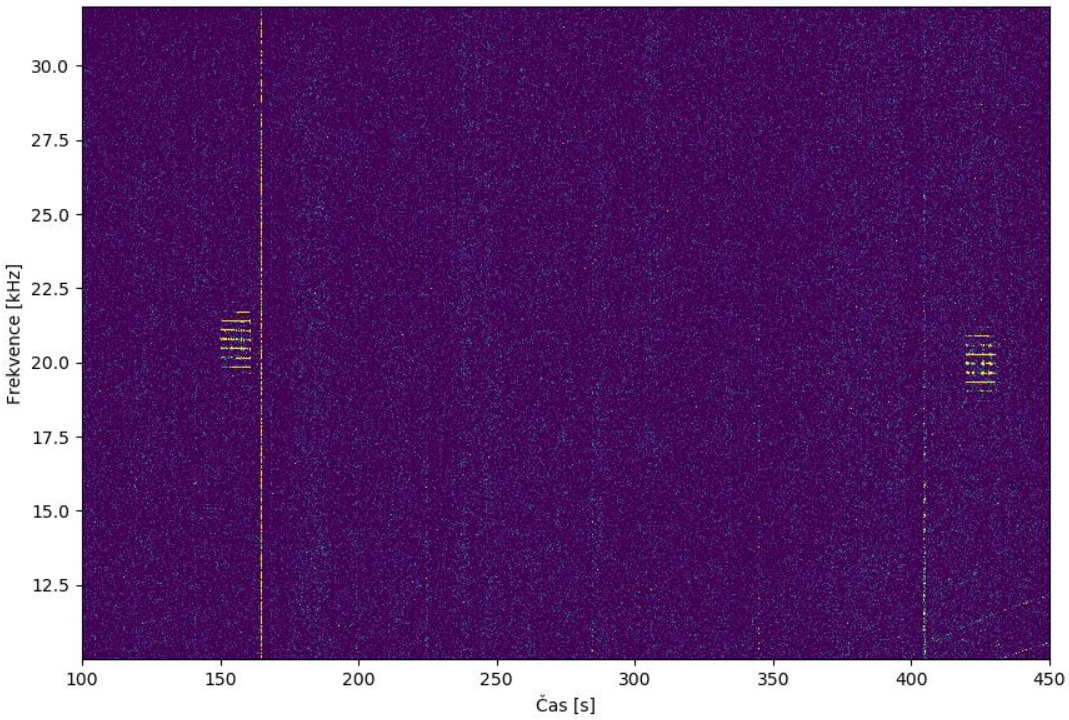
Adresář *.grc_gnuradio* obsahuje zdrojové kód hierarchických bloků GNU Radia vytvořených v programu GNU Radio Companion. Tento adresář je třeba zkopírovat do domovského adresáře uživatele.
Složka se samotným programem může být umístěna libovolně, ale je vhodné přidat cestu do systémové proměnné PATH, aby šel program spouštět z libovolného umístění. Při použití shellu BASH to lze udělat např. v souboru *~/.bash_profile*, kam se přidá řádek:

```
export PATH="$PATH:cesta_k_programu"
```

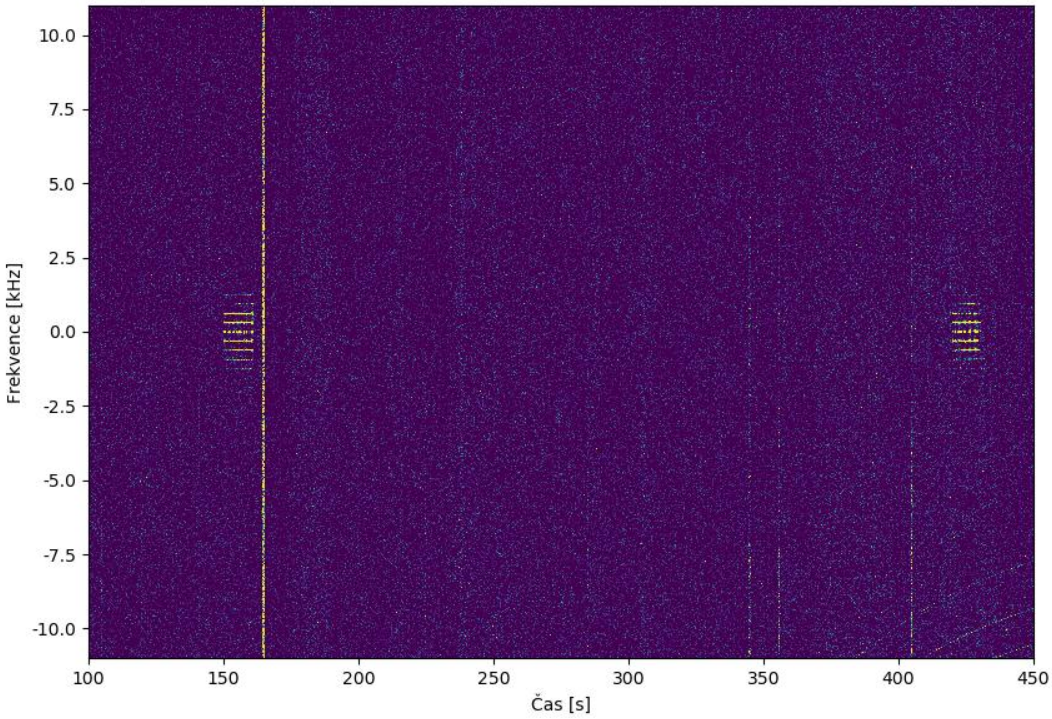
# D PŘÍKLAD SPEKTROGRAMŮ UPRAVENÉHO SDR ZÁZNAMU




Obrázek E.1 Spektrogramu SDR záznamu družice PSAT před úpravou




Obrázek E.2 Spektrogram SDR záznamu po odstranění Dopplerova jevu




Obrázek E.3 Spektrogram SDR záznamu po lineární korekci offsetu

# E UKÁZKA STRÁNKY ONLINE DEKODÉRU – TELEMETRIE PSK

```
PSK TLM

---- Current frame ----

Mode: C

ClockTimer = 266146 ticks = 1478:35:20

RebootCnt  = 59 times
val_PSK    = 27 %
val_AGC    = 330
val_Vbat   = 667 = 6,723 V
val_5V     = 495 = 4,943 V
val_Ic     = 331 mA
val_T_RX   = 23 deg C

status.PeriodNr       = 5
status.PeriodsSSTV_RX = 0
status.PeriodsRX      = 5
status.PeriodsTX      = 0


---- History frame ----

Mode: B

ClockTimer = 264290 ticks = 1468:16:40

RebootCnt  = 128 times
val_PSK    = 0 %
val_AGC    = 0
val_Vbat   = 0 = 0 V
val_5V     = 0 = 0 V
val_Ic     = 0 mA
val_T_RX   = 0 deg C


---- Spreadsheet format ----

satnogs 929896 2020-04-12          266146      1478:35:20  C     59
      27    330   6,723 4,943 331   23
satnogs 929896 2020-04-12          264290      1468:16:40  B     128
      0     0     0     0     0     0
```

# F UKÁZKA STRÁNKY ONLINE DEKODÉRU – TELEMETRIE SSTV

```
SSTV TLM
---- Current frame ----

Mode: S

Tick = 1009597 sec = 280:26:37

ADC_Temperature = 1 deg C
ADC_Light       = 330 lux
Plan_Auth       = 0
Plan_*_Count    = 0

cnt_Boot        = 0 times
cnt_*_Error     = 0 times
cnt_AudioStart  = 0 times
cnt_CamSnapshot = 0 times
cnt_CmdHandled  = 0 times
cnt_CmdIgnored  = 0 times
cnt_AuthError   = 0 times


---- Spreadsheet format ----

test 2020-05-15         1009597     280:26:37   S     1     330   0
      0     0     0     0     0     0     0     0
```

# G PŘÍKLAD TELEMETRICKÉHO ARCHIVU PSAT

```
fname,station,satnogs_id,date_time,t_sec,t_offset,ssp_lat,ssp_lon,elev,alt,az,dist,sunlit,sun_dur,sun_remains,p
rev_sun_dur,snr_db_avg,snr_db_min,f_offset,complete,tlm_str,mode,frame,bpsk31_det,agc_perc,supply_v,pa_current,
pa_temp
HDSDR_20150521_151236Z_435487kHz_RF.wav,VUT,,2015-05-21 15:14:42,86982,126,49.74,-7.23,480146,7.27,280.99,1840,
1,2471,1260,1985,32.23,28.34,-6500,1,W3ADO-5 beacon B 057 34 27 823 244 +10,B,57,34,27,8.23,0.244,10
HDSDR_20150521_151236Z_435487kHz_RF.wav,VUT,,2015-05-21 15:14:54,86994,138,49.38,-6.24,477878,8.07,279.28,1774,
1,2483,1248,1985,28.98,-1.45,-7500,1,W3ADO-5 beacon B 058 37 36 823 250 +14,B,58,37,36,8.23,0.250,14
HDSDR_20150521_151236Z_435487kHz_RF.wav,VUT,,2015-05-21 15:15:06,87006,150,49.01,-5.26,475617,8.89,277.43,1710,
1,2495,1236,1985,31.78,25.58,-7000,1,W3ADO-5 beacon B 059 53 50 808 248 +16,B,59,53,50,8.08,0.248,16
HDSDR_20150521_151236Z_435487kHz_RF.wav,VUT,,2015-05-21 15:15:18,87018,162,48.63,-4.30,473364,9.73,275.43,1647,
1,2507,1224,1985,30.81,10.53,-7500,1,W3ADO-5 beacon B 060 46 37 810 248 +18,B,60,46,37,8.10,0.248,18
HDSDR_20150521_151236Z_435487kHz_RF.wav,VUT,,2015-05-21 15:16:17,87077,221,46.62,0.22,462410,13.91,262.85,1375,
1,2566,1165,1985,31.62,5.23,-7000,0,W3ADO-5 beacon B 064 40 4saj07 249 +20,B,64,40,,,0.249,20
HDSDR_20150521_151236Z_435487kHz_RF.wav,VUT,,2015-05-21 15:16:29,87089,233,46.19,1.10,460210,14.73,259.63,1330,
1,2578,1153,1985,34.95,32.09,-6000,1,W3ADO-5 beacon B 065 21 43 804 248 +21,B,65,21,43,8.04,0.248,21
```