Fakulta elektrotechnická

Katedra měření

Diplomová práce

Realizace SW prostředků pro avionickou síť

Implementation of SW tools for avionics network

Bc. Pavel Tržil

Vedoucí: Ing. Martin Šipoš, Ph.D.

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Tržil** Jméno: **Pavel** Osobní číslo: **465834**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra měření**

Studijní program: **Kybernetika a robotika**

Studijní obor: **Kybernetika a robotika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Realizace SW prostředků pro avionickou síť

Název diplomové práce anglicky:

Implementation of SW tools for avionics network

Pokyny pro vypracování:

Navrhněte a zrealizujte SW prostředky pro avionickou síť založenou na Raspberry Pi, která využívá komunikační HAT s výstupními rozhraními CAN a ARINC429, dále pak čtyř-kanálový digitalizační HAT. U poskytnutého HW zrealizujte následující dílčí úkoly:
- Zprovozněte obousměrnou komunikaci mezi jednotlivými Raspberry Pi3 po sběrnici ARINC429 a CAN.
- Zprovozněte funkci digitalizace dat z libovolných kanálů z Raspberry s tím, že daná digitalizovaná data bude možné odeslat do jiného avionického systému pomocí sběrnice ARINC429/CAN.
- Vytvořte SW blok, který přijme data z jiného subsystému po sběrnici CAN, data rozparsuje a odešle dále po sběrnici ARINC429.
- Zrealizujte avionickou síť, která bude využívat minimálně tři moduly s Raspberry Pi a bude vzájemně komunikovat přes ARINC429 (1 hlavní jednotka, 2 podřízené). K podřízeným jednotkám bude možné připojit další moduly přes sběrnici CAN.
- Zrealizujte grafické uživatelské rozhraní pro zobrazení a odesílání zpráv přes sběrnice ARINC429 a CAN.

Seznam doporučené literatury:

[1] Mark 33 Digital Information Transfer System (DITS) Part 1, Functional Description, Electrical Interface, Label Assignments and Word Formats, ARINC SPECIFICATION 429 PART 1-17, 2004, Aeronautical Radio, Inc.
[2] Cary R. Spitzer: Digital Avionics Handbook, Second Edition, Avionics: Development and Implementation, CRC Press, 2007, ISBN: 978-0-8493-8441-7.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Martin Šipoš, Ph.D. katedra měření FEL (13138)

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **14.02.2022** Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce:
do konce zimního semestru 2023/2024

Ing. Martin Šipoš, Ph.D.		prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

Čestné prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a s použitím uvedené literatury a pramenů.

Praha, Leden, 2023

. . . . . .

Bc. Pavel Tržil

Poděkování

Rád bych poděkoval vedoucímu mé diplomové práce Ing. Martinu Šipošovi, Ph.D. za odbornou přípravu a metodologickou pomoc při zpracování mé práce.

Abstrakt

Tato diplomová práce se věnuje návrhu a realizaci avionické sítě se sběrnicemi ARINC429 a CAN. Jednotky v avionické síti jsou tvořeny minipočítačem Raspberry Pi, ke kterému jsou připojena zařízení (HAT-hardware attachement on top) rozšiřující jeho funkcionalitu o sběrnice CAN, ARINC429 a A/D převodník.

Práce předkládá dvě možné realizace avionické sítě. První realizace se věnuje možnosti avionické sítě s více jednotkami, které jsou řízené sběrnicí ARINC429 a možnosti překládání CAN a ARINC429 rámců. Druhá realizace se věnuje avionické síti vytvořené za účelem studijní pomůcky pro výuku sběrnic ARINC429 a CAN.

Součástí realizace avionické sítě jsou i dvě verze grafické aplikace pro PC, které umožňují řízení komunikace na těchto avionických sítích.

Klíčová slova: ARINC429, CAN, avionická síť, Raspberry Pi

Abstract

This thesis is devoted to the design and implementation of an avionics network with ARINC429 and CAN buses. The units in the aviation network are made up of a Raspberry Pi minicomputer, to which devices (HAT-hardware attachment on top) are added to expand its functionality with a CAN bus, ARINC429 and an A/D converter.

The work presents two possible realizations of the avionics network. The first implementation is devoted to the possibilities of an avionics network with multiple units, which are controlled by the ARINC429 bus and the possibilities of translation of CAN and ARINC429 frames. The second implementation is dedicated to the avionics network created for the purpose of a study aid for teaching ARINC429 and CAN buses.

Part of the implementation of the avionics network are also two versions of a graphic application for the PC, which enable the control of communication on these avionics networks.

Keywords: ARINC429, CAN, avionics network, Raspberry Pi

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1
Úvod

Tato diplomová práce se zabývá návrhem a realizací avionické sítě. Realizovaná avionická síť je tvořena jednotkami, které spolu komunikují po sběrnici ARINC429 a CAN.

ARINC429, též známý jako ”Mark33 Digital Transformation System” [1], definuje standard pro datovou sběrnici avioniky, která se velmi často používá u komerčních a dopravních letadel.

CAN sběrnice je původně datová sběrnice používaná v automobilovém průmyslu. Tato sběrnice byla vyvinuta společností Robert Bosch, s.r.o. a postupem času se začala používat i v leteckém průmyslu. Sběrnice se nejčastěji využívá pro vnitřní komunikační síť senzorů a funkčních jednotek v zařízení. [2]

Jako jednotka v avionické sběrnici se používá minipočítač Raspberry Pi. Jedná se o malý jednodeskový počítač, který byl vyvinut za účelem podpoření výuky informatiky. Raspberry Pi podporuje připojení dalších zařízení, které rozšiřují jeho funkcionalitu. Tato zařízení jsou označována jako HAT (Hardware Attachement on Top). [3]

Jednotka použitá v avionické síti je rozšířena o dvě tato zařízení (HATy). Tyto HATy byly vyvinuty na Fakultě elektrotechnické ČVUT v Praze v rámci výzkumné skupiny NavLIS, která se zabývá výukou a výzkumem avionických systémů. Připojené HATy rozšiřují Raspberry Pi o sběrnice CAN a ARINC429. Dále poskytují možnost digitalizace analogového signálu.

Vytvořená avionická síť slouží pro demonstraci možností průběhu komunikace mezi avionickými systémy. Definuje a provádí operace pro posílání a příjem ARINC429 a CAN rámců. Dále umožňuje měření napětí a zasílání měřených hodnot uvedenými sběrnicemi. Avionická síť také definuje způsob překladu CAN datového rámce na rámec specifikace ARINC429.

V rámci diplomové práce jsem sestavil avionickou síť se dvěmi jednotkami vhodnou pro využití ve výuce pro demonstraci sběrnic ARINC429 a CAN. Tato ”výuková” avionická síť umožňuje demonstraci převodu měřeného napěťového signálu na letecká data (Baro corrected altitude #1, Computed Airspeed, Exhaust Gas Temperature, Pitch Angle). Letecká data jsou následně zasílána avionickými jednotkami po sběrnicích CAN a ARINC429 do PC. Pro PC jsem vytvořil grafickou aplikaci, která umožňuje komunikaci s jednotkami avionické sběrnice a zobrazuje přijaté datové rámce sběrnic ARINC429 a CAN v jejich binární reprezentaci pro jejich další zpracování. Grafická aplikace dále umožňuje kontrolu správnosti dekódování těchto přijatých datových rámců.

V teoretické části diplomové práce jsou uvedeny specifikace použitých sběrnic, popsán princip dekódování jejich datových rámců a dále je zde vysvětlen princip komunikace HAT s Raspberry Pi.

Kapitola 2
Použité HW moduly, sběrnice a komunikační protokoly

2.1 Raspberry Pi

Raspberry Pi byl vyvinut v roce 2012 britskou nadací Raspberry Pi Foundation za účelem vytvoření pomůcky pro výuku informatiky. Jedná se o malý jednodeskový počítač, který se vyrábí v několika různých variantách. V mé diplomové práci jsem použil model Raspberry Pi 3 Model B+. [3]

Tento model má 64 bitový procesor z rodiny ARM (Cortex-A53), 1GB RAM paměti, plnohodnotný ethernet a 40 GPIO pinů, které umožňují připojení dalších zařízení k Raspberry Pi. [3]

GPIO piny jsou obecné vstupně/výstupní piny. Tyto piny využívají 3,3 V TTL logiku. Kromě toho jsou na určitých pinech GPIO definovány komunikační protokoly UART, I2C a SPI. [3]

V Raspberry Pi je nainstalovaný plnohodnotný operační systém Raspbian. Raspbian je linuxová distribuce založená na systému Debian. Tento operační systém podporuje použití tzv. Device Tree. Jedná se o nástroj, který v Raspberry Pi popisuje HW připojeného zařízení. Díky tomu je možné vyvíjet složitější aplikace bez detailní znalosti fyzického HW. [3]

Použití operačního systému přináší výhodu v podpoře velkého množství programovacích jazyků a knihoven. Při vytváření avionické sítě jsem použil objektově orientovaný programovací jazyk Python 3.6.

2.1.1 Konfigurace ovladačů pro rozšiřující HW v OS Raspbian

Jak bylo uvedeno v předchozí kapitole Raspberry Pi je minipočítač, na kterém běží plnohodnotný operační systém Raspbian OS (linuxová distribuce). Zdrojové kódy v linuxových systémech mohou běžet ve dvou různých prostředích tzv. ”user space” (uživatelský prostor) a ”kernel space” (prostor jádra).

Toto rozdělení slouží k ochraně jádra operačního systému (kernelu) před škodlivým kódem, ať už úmyslným, nebo neúmyslným. Běžný uživatelský/aplikační SW (včetně vyvinutého SW pro jednotku avionické sítě) je spouštěn v user space Linuxu. V kernel space Linuxu jsou spouštěny drivery zařízení připojených k Raspberry Pi. Drivery zařízení zajišťují nízkoúrovňovou komunikaci s připojenými zařízeními a obvody. [4]

Pro to, aby driver v kernel space mohl komunikovat s procesy v uživatelském prostoru je použita tzv. socketová komunikace. Ta umožňuje nastavit ”komunikační cestu” pro dva různé procesy-/drivery. Pro komunikaci se poté používají standardizovaná volání definovaná v [5].

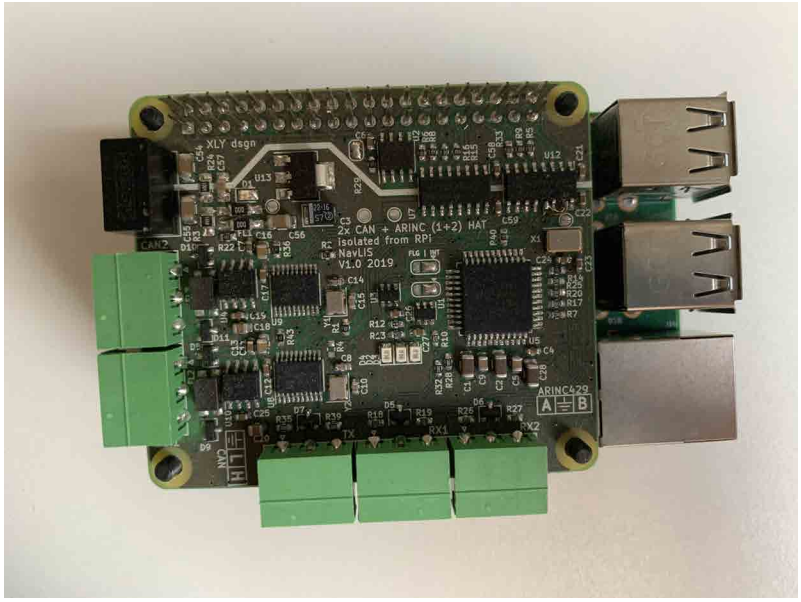
2.2 Rozšiřující moduly HAT pro Raspberry Pi 3 B+

Hardware **A**ttached on **T**op (dále značeno jako HAT) je rozšiřující deska pro Raspberry Pi 3 B+. Provedením se jedná o destičku plošného spoje velikosti 65x56 mm s montážními otvory v rozích, kterou je možno sestavit se základní procesorovou deskou, případně s dalšími HAT moduly do jednoho konstrukčního celku. Datové propojení je zabezpečeno prostřednictvím 40-ti pinového konektoru GPIO, který umožňuje propojit všechny sestavené moduly sběrnicemi SPI.

Dle [6] by po připojení HATu k základní desce a nastartování systému měla mít tato rozšiřující deska (HAT) možnost autokonfigurace systému a automatické nastavení GPIO a driveru v operačním systému. Pro tuto automatickou konfiguraci jsou vyhrazeny piny ID-SD a ID-SC na 40-ti pinovém konektoru, ke kterým je připojena I2C EEPROM. Specifikace pro HAT však umožňuje propojovat HAT moduly i bez této funkcionality. V tomto případě musí být odpovídající drivery a konfigurace GPIO nastaveny individuálně v obslužném SW. [6]

2.2.1 Komunikační HAT s moduly pro sběrnice ARINC429 a CAN

Tento rozšiřující modul HAT umožňuje po propojení s Raspbery Pi 3 B+ komunikovat po dvou sběrnicích CAN a třech sběrnicích ARINC429 (2x jako přijímač, 1x jako vysílač). Rozhraní obou sběrnic je zajištěno integrovanými obvody **HI-3593** a **MCP2515**.



Obrázek 1: Komunikační HAT

HI-3593 [8] Je integrovaný CMOS obvod použitý v komunikačním HATu. Tento obvod umožňuje ”připojit” sběrnici ARINC429 k zařízení se sběrnicí SPI. Obvod v HATu zajišťuje ARINC429 specifikaci (správný formát slova, přenosové charakteristiky). Pro konfiguraci a vyčítání zpráv z obvodu se používají 8-bitové SPI instrukce tzv. ”op-codes”. Obvod má dva ARINC429 přijímače (referované jako RX0 a RX1) a jeden vysílač (referovaný jako TX).

Pro řízení základní komunikace obvodu HI-3593 s procesorovou deskou Raspberry Pi 3 B+ jsem použil driver, který je volně dostupný ze stránek [7].

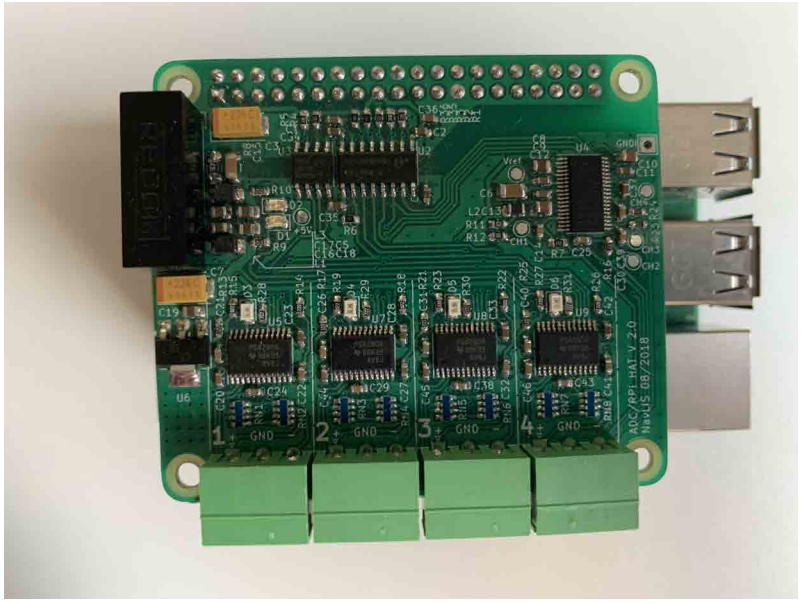
MCP2515 [9] Jedná se o integrovaný obvod, který umožňuje propojení zařízení se sběrnicí SPI se sběrnicí CAN a zajišťuje vysílání i příjem dat dle specifikace pro standardní i rozšířené CAN rámce.

Pro použití v Raspberry Pi nebylo nutné instalovat žádný dodatečný driver, neboť driver pro tento obvod je součástí každé běžné instalace linuxové distribuce.

2.2.2 Digitalizační HAT s převodníkem A/D

Pro převod analogového signálu do digitální podoby je použit integrovaný obvod **ADS8684**. [10] Tento obvod obsahuje čtyři analogové vstupní kanály. Analogový signál na jednotlivých vstupech může mít amplitudu až $\pm 10{,}24$ V. Pro zvýšení rozsahu převáděného analogového signálu je v HATu použita na vstupech do **ADS8684** kombinace rezistorového pole a programovatelného zesilovače signálu **PGA280**. Použité rezistorové pole umožňuje snížit amplitudu vstupního signálu (tím je umožněna konverze signálu s amplitudou až $\pm 38{,}4$ V). **PGA280** [11], který je v obvodu zapojen mezi rezistorovým polem a převodníkem **ADS8684**, pak umožňuje konverzi signálů s velmi malou amplitudou [12]. Integrovaný obvod **ADS8684** umožňuje digitální převod v rozlišení 16 bit s maximálním vzorkovacím kmitočtem 500 ksps [10].

Driver pro tento obvod mi poskytl vedoucí mé diplomové práce jako součást zapůjčené jednotky Raspberry Pi s digitalizačním HATem.



Obrázek 2: Digitalizační HAT

2.3 Sběrnice ARINC429

ARINC429 je standard datové sběrnice, který se používá především v civilním a dopravním letectví. Tento standard definuje signálové úrovně, datový rámec a přenosové charakteristiky pro komunikaci avionických systémů dopravních letadel. Můžeme jej najít také pod označením "Mark33 Digital Information Transfer System (DITS)". [1]

Standard byl vyvinut v roce 1977 společností Aeronautical Radio, Incorporated (ARINC), která byla založena jako soukromá organizace v roce 1929 za účelem vývoje komunikačních standardů a specifikací pro použití v civilním letectví. V současné době jsou standardy pro tuto oblast schvalovány organizací AEEC (mezinárodní organizace pro letecké standardy). [13], [14]

2.3.1 Specifikace sběrnice

Standard **ARINC429** je definován jako jednosměrná sériová asynchronní sběrnice. Jeden vysílač je možno propojit s maximálně dvaceti přijímači prostřednictvím dvou vodičů.

Fyzicky je sběrnice definována stíněnou kroucenou dvoulinkou s impedancí 75 Ω. Přenos signálu je založen na využití rozdílu potenciálu mezi vodiči - diferenciální signál (obdobně jako sběrnice EIA-422/RS-422). Maximální délka vodičů sběrnice není specifikována (sběrnice funguje spolehlivě do 30 m). [13]

2.3.2 Princip komunikace

Jak již bylo uvedeno, sběrnice sestává z jednoho vysílače, ke kterému může být připojeno až 20 přijímačů. Sběrnice umožňuje pouze jednosměrnou komunikaci (tzv. simplex).

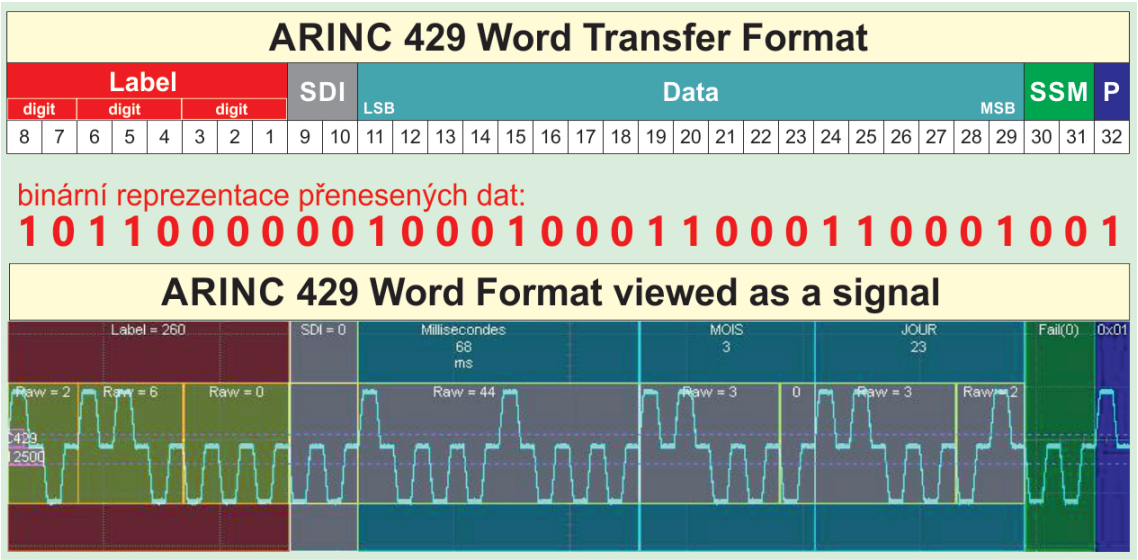
Komunikace tedy probíhá tak, že vysílač vysílá data na sběrnici a všechny připojené přijímače stav sběrnice neustále monitorují. V připojených přijímačích poté probíhá vyhodnocení, jestli jsou data na sběrnici určená pro daný přijímač (zařízení).

Jelikož je sběrnice jednosměrná, nedochází k žádnému potvrzení přijatých dat. V případě nutnosti tohoto potvrzení je možné použít další vysílač + přijímač pro vytvoření druhé sběrnice. Tímto způsobem jsme schopní zajistit obousměrnou komunikaci (tzv. full-duplex). [13], [14]

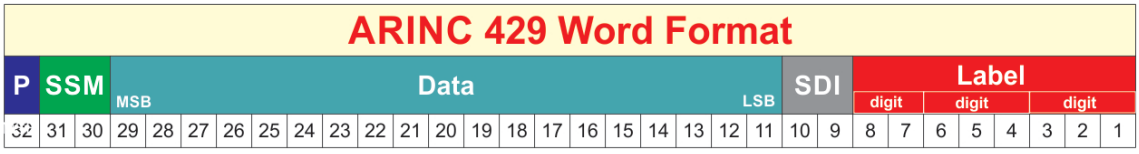
2.3.3 Datový rámec

Datové rámce přenášené po sběrnici mají délku 32 bitů. Datový rámec také označujeme jako WORD. Tento rámec nemá definovaný žádný START a STOP bit. Začátek rámce je tedy definován jako pauza ve vysílání v délce trvání minimálně 4 bity (stav NULL = 0 V mezi datovými vodiči). Přenosová rychlost sběrnice je definovaná pro dvě rychlosti: Low-Speed (14,5 kbps ±10% / 14,5 kHz) a High-Speed (100 kbps ±5% / 100 kHz). Pokud vysílač nemá k dispozici žádná data pro přenos, je hodnota všech bitů rovná úrovni NULL.

Datový rámec je rozdělen na 5 částí (polí): Parity bit (P), Sign/Status Matrix (SSM), Data, Source/Destination Identifiers (SDI), Label - viz obr. 4. Jednotlivá pole datového rámce mají rozličné kódování hodnot do binárního řetězce. Pole SSM, SDI a P jsou kódována binárně. Pole Label je kódováno oktalově a pole Data může být kódováno binárně, nebo jako binary coded decimal (BCD). Způsoby kódování jsou podrobněji popsány v kapitole 2.7. [13], [14]



Obrázek 3: Datový rámec ARINC429 / Word - převod napěťového signálu. [15]



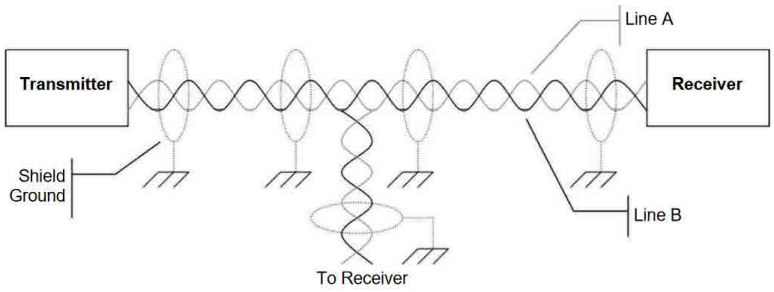
Pole datového rámce ARINC429

Bit	Název pole		Význam, obsah		
32	**Parity bit**		Paritní bit přenášeného rámce - detekce chyb v přenášeném rámci **lichou** (**ODD**) paritou.		
30 - 31	**Sign/Status Matrix**		Dle typu přenášeného rámce určuje znaménko/stav přenášených dat/zařízení		
	bit 31	***bit 30***	***Sign/Status Matrix for BCD Data***	***Status Matrix for BNR Data***	***Status Matrix for Discrete Data***
	0	0	Plus, North, East, Right, To, Above	Failure Warning (FW)	Verified Data, Normal Operation
	0	1	No Computed Data (NCD)	No Computed Data (NCD)	No Computed Data (NCD)
	1	0	Functional Test (FT)	Functional Test (FT)	Functional Test (FT)
	1	1	Minus, South, West, Left, From, Below	Normal Operation (NO)	Failure Warning (FW)
29	**Data** **Sign Matrix for BNR**		Dle typu přenášeného rámce obsahuje datový bit nebo znaménko/stav přenášených dat/zařízení		
11 - 28	**Data**		Dle typu přenášeného rámce obsahuje 18/19 bitů dat, s následujícím kódováním		
	Binary **C**ode **D**ecimal		Každá čtveřice bitů (1/2Byte, Nibble) reprezentuje jednu dekadickou číslici přenášené hodnoty		
	Binary **N**umber **R**epresentation		Přenášená hodnota je vyjádřena dvojkovým číslem		
	Discrete Data		Význam jednotlivých bitů je definován zařízením nebo subsystémem		
9 - 10	**S**ource/**D**estination **I**dentifier		Identifikace adresáta zprávy, nebo vysílajícího subsystému (častěji)		
1 - 8	**Label**		Identifikuje datový typ v oktalovém vyjádření		

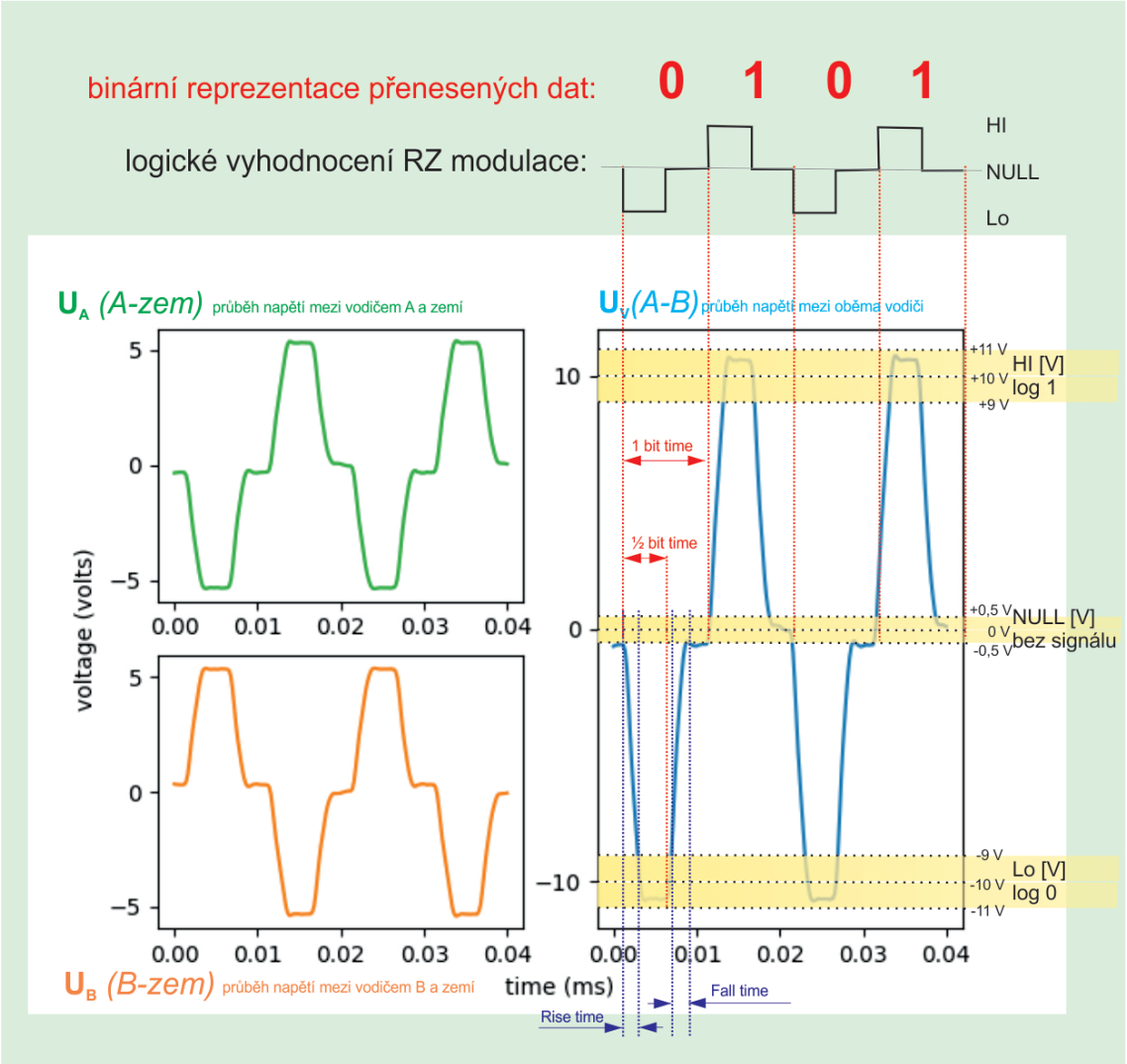
Obrázek 4: Datový rámec ARINC429 / Word - popis jednotlivých polí rámce. [15]

2.3.4 Elektrické charakteristiky

Zařízení ve sběrnici jsou propojeny stíněnou kroucenou dvoulinkou s impedancí 75 Ω. Stínění vodičů musí být u každého připojeného zařízení propojeno se zemnicím potenciálem (viz Obrázek 5).



Obrázek 5: Příklad stínění sběrnice ARINC429 [13].



Obrázek 6: Průběh napětí při modulaci **R**eturn-to-**Z**ero na vodičích sběrnice ARINC429 včetně demonstrace následného převedení přenesených bitů do binární reprezentace [15], [16].

Výstupní impedance připojeného vysílače musí být rovnoměrně rozdělena mezi vodiče a musí mít hodnotu 75±5 Ω Připojené přijímače musí mít vstupní impedanci větší než 8 kΩ.

K přenosu signálu se používají pravoúhlé napěťové impulsy (diferenční napětí mezi vodiči je 10±1 V s kladnou nebo zápornou polaritou). Na jednotlivých vodičích je tedy proti zemi napětí o hodnotách +5 V a -5 V. Pokud je na jednom vodiči napětí +5 V je na druhém napětí -5 V a naopak. Vstupní obvod přijímače řeší diferenční napětí U_V mezi signálovými vodiči (A, B). V závislosti na tomto napětí může sběrnice nabývat tří stavů (viz obr. 6).

Pro komunikaci používá sběrnice bipolární RZ (Return-to-Zero) modulaci. Logická 1 je reprezentována napětím +10 V po první polovinu přenosu jednoho bitu (bit cycle), v druhé polovině klesne úroveň napětí na hodnotu NULL. V případě logické 0 platí obdobný princip, jen je v první polovině bit cycle úroveň napětí rovna -10 V. Každý bit cycle končí signálem s hodnotou napětí 0 V. Tímto je eliminována nutnost použití externího hodinového signálu.

Napěťové impulsy signálu jsou vytvářeny RC obvody, jež jsou součástí ARINC vysílače. V reálném případě nemá signál přesný obdélníkový průběh, přidává se k němu náběžná a sestupná hrana (Rise time / Fall time - viz obr. 6). V tabulce 1 jsou uvedeny definice přípustných hodnot pro průběh signálu. [13], [14]

	High speed	**Low speed**
Bit Rate	100 kbps ±1%	10-14.5 kbps ±1%
1 bit time	10 $\mu sec \pm 2.5\%$	(1/Bit rate) $\mu sec \pm 2.5\%$
1/2 bit time	5 $\mu sec \pm 5\%$	(1 bit time/2) $\pm 5\%$
Rise Time	1.5 $\mu sec \pm 0.5\ \mu sec$	10 $\mu sec \pm 5\ \mu sec$
Fall Time	1.5 $\mu sec \pm 0.5 \mu sec$	10 $\mu sec \pm 5\ \mu sec$

Tabulka 1: Definice přípustných hodnot pro průběh signálu [13]

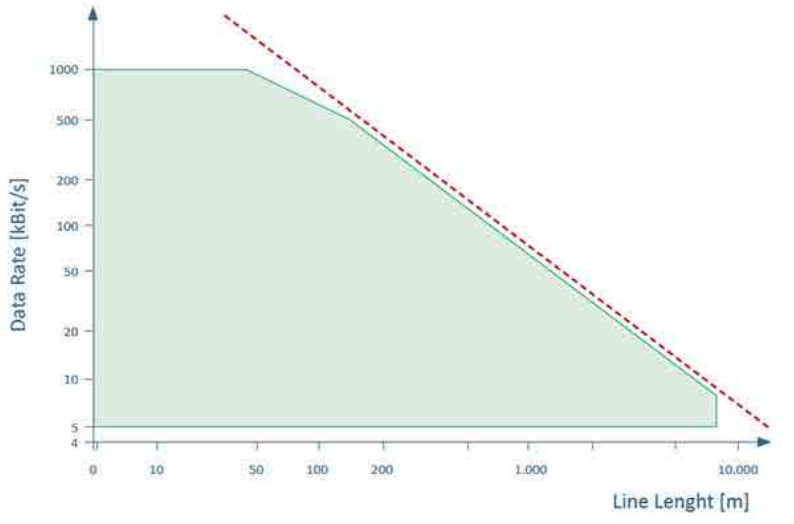
2.4 Sběrnice CAN

Sběrnice CAN je sériový komunikační protokol, který byl vyvinut firmou Bosch pro použití v automobilovém průmyslu. Jeho výhodou je odolnost proti elektrickému rušení, nízká cena a vysoká přenosová rychlost (až 1 Mbps). Protokol je schopen detekce přenesených dat.

Sběrnice slouží k propojení elektronických řídicích jednotek (ECU), které řídí/monitorují jednotlivé části vozidla (motor, protiblokovací brzdový systém/ABS, airbagy, tempomat, elektronické vstřikování paliva, ...). Elektronickou řídicí jednotku (ECU) připojenou ke sběrnici nazýváme také jako **uzel**. [17]

2.4.1 Specifikace sběrnice

Standard **CAN** je definován jako duplexní sériová sběrnice s podporou multi-master provozu. Maximální délka sběrnice je závislá na rychlosti přenosu signálu médiem - do 40 m lze dosáhnout až 1 Mbit/s, při vzdálenosti 500 m dosáhneme rychlost 125 kbit/s a při vzdálenosti 1,2 km je přenosová rychlost pouze 70 kbit/s (viz obr. 7). [18]



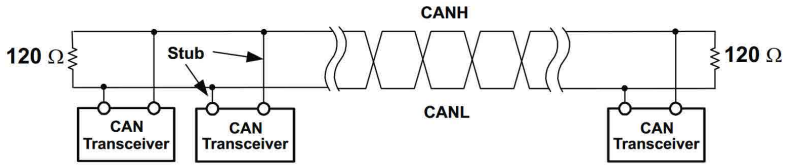
Obrázek 7: Závislost rychlosti přenosu na délce sběrnice [18]

2.4.2 Fyzické přenosové médium

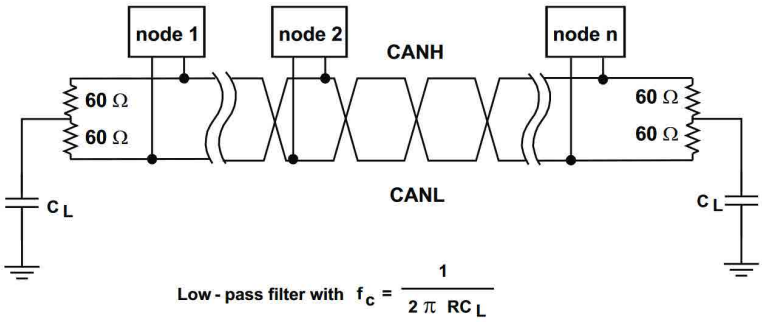
Přenos CAN zpráv zajišťují 2 signálové vodiče značené jako CANH a CANL. Tyto vodiče mohou být realizované například metalickou dvoulinkou. Podle rozdílového napětí mezi CANH a CANL mohou bity na sběrnici nabývat 2 logických úrovní:

- Dominantní - logická 0, rozdílové napětí je 2-5 V
- Recesivní - logická 1, rozdílové napětí je 0 V

Sběrnice má charakteristickou impedanci 120 Ω. Pro její zakončení jsou doporučeny následující 2 možnosti. První možnost spočívá ve standardním zakončení sběrnice rezistorem 120 Ω (viz obrázek 8). Další možností je použít pro zakončení tzv. ”low-pass” filtr (viz obrázek 9). [19]



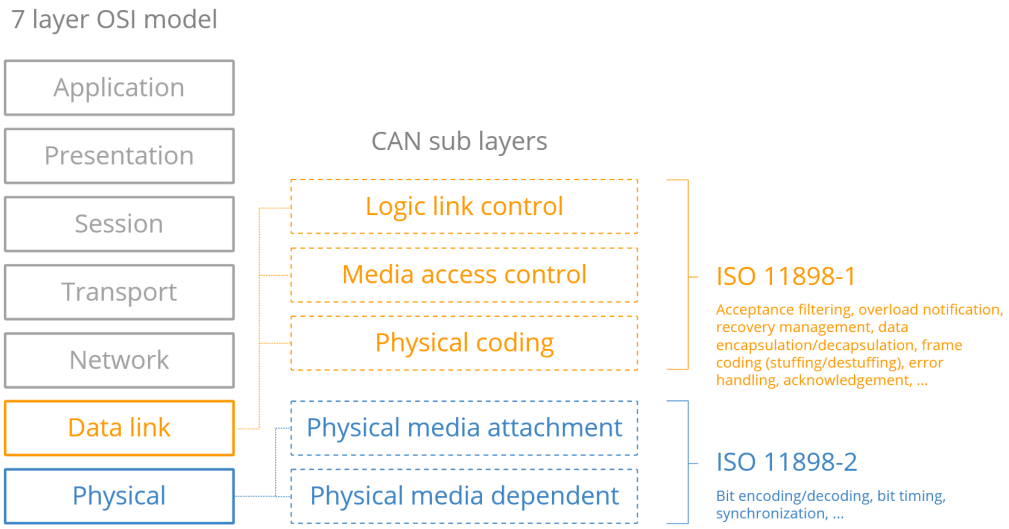
Obrázek 8: Příklad standartního zakončení CAN sběrnice [19]



Obrázek 9: Příklad zakončení CAN sběrnice low-pass filtrem [19]

2.4.3 Vrstvy sběrnice - model OSI/ISO

Protokol CAN je definován standardem ISO 11898 [20]. Tento standard definuje fyzickou a linkovou vrstvu dle modelu OSI/ISO (viz obrázek 10).



Obrázek 10: OSI/ISO model [21]

Fyzická vrstva Definuje elektrickou specifikaci pro CAN sběrnici. Stará se o převod logických úrovní na elektrické pulsy. Tato vrstva je vždy implementována jako součást HW. [19]

Linková vrstva Je zodpovědná za bezchybný přenos zpráv z uzlu do CAN sítě. Řeší kódování zprávy, bit-stuffing/destuffing, detekci a hlášení chyb a potvrzování správně přijatých zpráv. [19]

2.4.4 Typy CAN rámců

Protokol definuje 4 typy datových rámců s různými účely použití.

DATA FRAME Slouží k přenosu dat z vysílačů (např. avionické senzory) do přijímače. Je složen ze sedmi bitových polí (FIELD): START OF FRAME, ARBITRATION FIELD, CONTROL FIELD, DATA FIELD, CRC FIELD, ACK FIELD, END OF FRAME. [2]

- START OF FRAME - Toto pole značí počátek DATA FRAME a skládá se z jednoho dominantního bitu.
- ARBITRATION FIELD - Arbitrační pole stanovuje prioritu zprávy ve sběrnici. Skládá se z identifikátoru a RTR-bitu. Identifikátor může mít délku 11 bitů (standartní rámec, definován standardy CAN 2.0A a CAN 2.0B, viz obrázek 11), nebo 29 bitů (rozšířený rámec, definován standardem CAN 2.0B, viz obrázek 12). RTR-bit je v případě DATA FRAME vždy dominantní. V případě REMOTE FRAME je vždy recesivní.
- CONTROL FIELD - Toto pole se skládá ze šesti bitů. Jedná se o 2 bity rezervované pro budoucí použití a 4 bity (značí se jako DATA LENGHT CODE), které specifikují délku přenášených dat (počet přenášených bajtů).

CAN frame (Base format)		
Field name	**Lenght (bits)**	**Purpose**
Start-of-frame	1	Denotes the start of frame transmission
Identifier A	11	A (unique) identifier which also represents the message priority
Remote transmission request (RTR)	1	Must be dominant (0) for data frames and recessive (1) for remote request frames (see Remote Frame, below)
Identifier extension bit (IDE)	1	Must be dominant (0) for base frame format with 11-bit identifiers
Reserved bit (r0)	1	Reserved bit which must be set dominant (0), but accepted as either dominant or recessive
Data length code (DLC)	4	Number of bytes of data (0–8 bytes)
Data field	0-64 (0-8 Bytes)	Data to be transmitted (length dictated by DLC field)
CRC	15	Cyclic redundancy check
CRC delimiter	1	Must be recessive (1)
ACK slot	1	Transmitter sends recessive (1) and any receiver can assert a dominant (0)
ACK delimiter	1	Must be recessive (1)
End-of-frame (EOF)	7	Must be recessive (1)
Inter-frame spacing (IFS)	3	Must be recessive (1)

Obrázek 11: Pole základního datového rámce sběrnice CAN [2]

CAN frame (Extedend frame format)		
Field name	**Lenght (bits)**	**Purpose**
Start-of-frame	1	Denotes the start of frame transmission
Identifier A	11	First part of the (unique) identifier which also represents the message priority
Substitute remote request (SRR)	1	Must be recessive (1)
Identifier extension bit (IDE)	1	Must be recessive (1) for extended frame format with 29-bit identifiers
Identifier B	18	Second part of the (unique) identifier which also represents the message priority
Remote transmission request (RTR)	1	Must be dominant (0) for data frames and recessive (1) for remote request frames (see Remote Frame, below)
Reserved bits (r1, r0)	2	Reserved bits which must be set dominant (0), but accepted as either dominant or recessive
Data length code (DLC)	4	Number of bytes of data (0–8 bytes)
Data field	0-64 (0-8 Bytes)	Data to be transmitted (length dictated by DLC field)
CRC	15	Cyclic redundancy check
CRC delimiter	1	Must be recessive (1)
ACK slot	1	Transmitter sends recessive (1) and any receiver can assert a dominant (0)
ACK delimiter	1	Must be recessive (1)
End-of-frame (EOF)	7	Must be recessive (1)
Inter-framespacing(IFS)	3	Must be recessive (1)

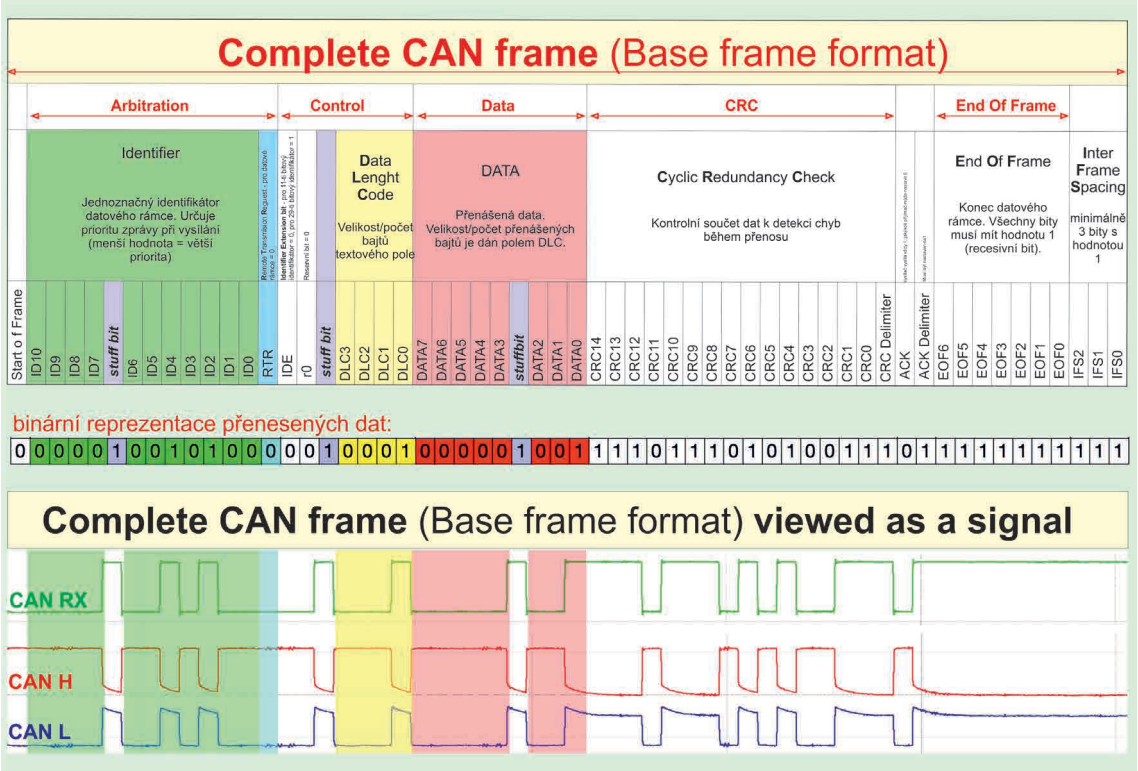
Obrázek 12: Pole rozšířeného datového rámce sběrnice CAN [2]

- DATA FIELD - Toto pole se může skládat z až 8 bajtů, které obsahují datové informace rámce. Bity v jednotlivých bajtech jsou přenášeny v pořadí první bit je MSB (nejvýznamější bit).
- CRC FIELD - Obsahuje CRC kód rámce (15 bitů) za kterým následuje 1 recesivní bit (CRC DELIMITER). Jedná se o kontrolní součet rámce pro ověření případných přenosových chyb. Tento kód je vypočítán z částí ARBITRATION FIELD, CONTROL FIELD a DATA FIELD.
- ACK FIELD - Pole je složeno ze 2 bitů. První bit (ACK) slouží ke kontrole správně přijatého rámce (dominantní stav v případě správného rámce, recesivní stav případě špatného rámce). Za tímto bitem následuje recesivní bit (ACK DELIMITER).
- END OF FRAME - Pole END OF FRAME se skládá ze 7 recesivních bitů.

REMOTE FRAME Jedná se o zprávu, která slouží jako požadavek k získání dat z jiné jednotky/sensorů. Je složen ze šesti bitových polí: START OF FRAME, ARBITRATION FIELD, CONTROL FIELD, CRC FIELD, ACK FIELD a END OF FRAME. Jednotlivá pole mají podobnou strukturu jako pole v případě DATA FRAME. [2]

ERROR FRAME Zpráva vysílaná uzlem, který detekoval chybu přenosu. Tato zpráva zastaví přijímání chybového rámce ostatními jednotkami. [2]

OVERLOAD FRAME Vysláním tohoto rámce se zvětší časové zpoždění při vysílání datových nebo vzdálených rámců. [2]



Obrázek 13: Datový rámec CAN - převod napěťového signálu, rozdělení rámce na bitová pole. [18]

2.4.5 Zabezpečení CAN rámce + detekce chyb

Monitoring Vysílač porovnává hodnotu právě vysílaného bitu s úrovní detekovanou na sběrnici. Pokud jsou tyto hodnoty odlišné, dojde k přeřušení vysílání a vyslání ARBITRATION FIELD. Novou zprávu pak vysílá uzel s nejvyšší prioritou. [17], [22]

CRC kód Chyba cyklické redundance je detekována za použití patnáctibitového CRC kódu, který je vypočítán vysílačem CAN zprávy. Přijímač po přijetí spočítá z definovaných polí nové CRC a porovná ho s CRC přijatým ve zprávě. Pokud se tyto dvě hodnoty liší, je vyhlášena chyba. [17], [22]

Bit-stuffing Pole START OF FRAME, ARBITRATION FIELD, CONTROL FIELD, DATA FIELD A CRC jsou kódována za použití metody bit-stuffing. To znamená, že kdykoliv vysílač detekuje v těchto polích řetězec pěti bitů stejné úrovně, doplní za něj jeden bit opačné úrovně. Přijímač provádí na přijatém rámci ”destuffing”(odstranění bit-stuffed bitů). Poté proběhne kontrola rámce, pokud je v rámci řetězec šesti bitů stejné logické úrovně, je vyhlášena chyba. Bit stuffing se využívá pouze ve zprávách typu DATA FRAME a ERROR FRAME. Kromě detekce chyb zajišťuje i dodržení synchronizace. [17], [22]

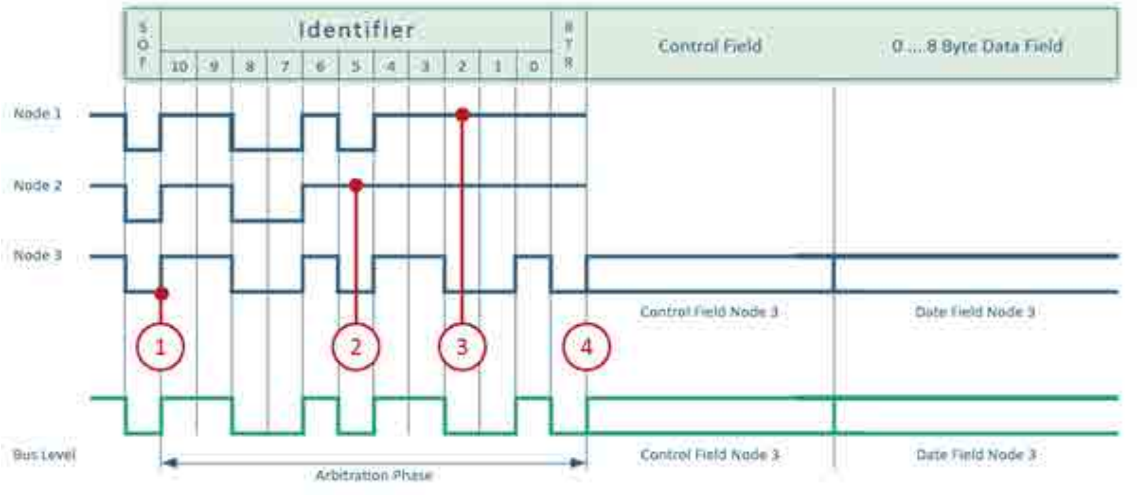
Kontrola zprávy Při této kontrole se ověřují hodnoty polí CRC delimiter, ACK delimiter, END OF FRAME a tři bity mezery mezi zprávami. V případě detekce nepovolené hodnoty je vyslána chybová zpráva. [17], [22]

Potvrzení přijetí zprávy Pokud je zpráva přijata v pořádku, přijímací uzel zapíše dominantní bit do ACK pole zprávy. To poté detekuje vysílač. Pokud je hodnota ACK bitu vysílače recesivní a hodnota ACK bitu přijímače dominantní, je přenos vyhodnocen jako úspěšný. Toto potvrzování je prováděno všemy uzly ve sběrnici. [17], [22]

2.4.6 Princip komunikace

Jak bylo uvedeno v předcházejících částech, CAN je protokol typu Multi-Master. Tedy každý uzel na sběrnici může vystupovat jako master, což má za výhodu zvýšení spolehlivosti (např. při poruše jednoho uzlu zbytek sítě dál funguje). Uzel sběrnice může zahájit vysílání zprávy jen v případě, že je sběrnice v klidovém stavu (reprezentováno recesivní úrovní sběrnice).

Arbitrace Kdykoliv je sběrnice v klidovém stavu, může kterýkoliv z uzlů zahájit vysílání zprávy (hovoříme o tzv. sběrnici s náhodným přístupem). Pokud ve stejném čase začnou zprávu vysílat dva a více uzlů vznikne na sběrnici konflikt, který je vyřešen bitovou arbitrací identifikátoru zpráv. Během arbitrace každý z vysílačů porovnává úroveň vysílaného bitu s úrovní, která je monitorována na sběrnici. Pokud jsou tyto úrovně shodné, vysílač může pokračovat ve vysílání. Pokud je poslán bit s recesivní úrovní a monitorován bit s dominantní úrovní, musí vysílač okamžitě přerušit vysílání zprávy. Hovoříme o tzv. ztrátě arbitrace pro daný vysílač (uzel). Průběh arbitrace je znázorněn na obrázku 14. [2]



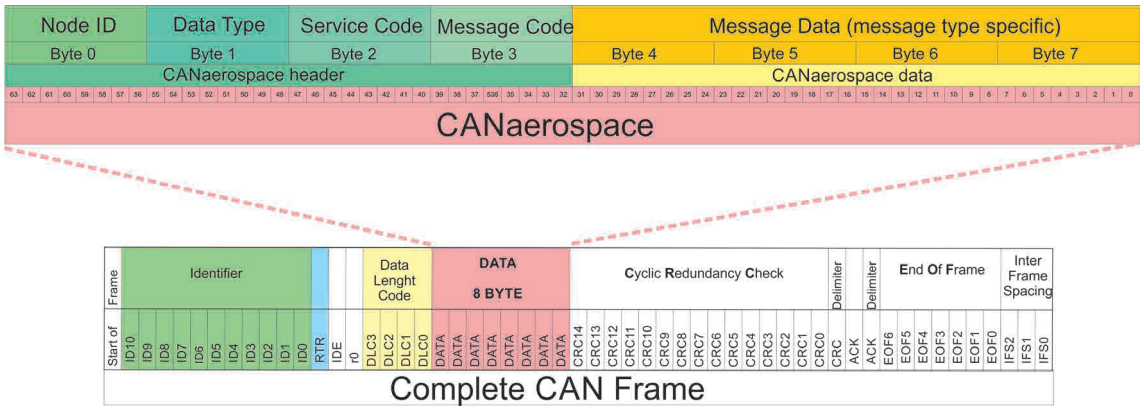
Obrázek 14: Průběh arbitrace ve sběrnici CAN [18] - Obrázek zachycuje průběh arbitrace, kde se vysílače Node1, Node2, Node3 pokouší vyslat zprávu - arbitraci sběrnice vyhraje vysílač Node3.

2.5 CANaerospace

Jedná se o komunikační protokol vyšší vrstvy, který je založen na sběrnici CAN. Byl vyvinut v roce 1998 společností Stock Flight. Tento protokol byl vytvářen s ohledem na to, aby jej bylo možné použít spolu se sběrnicí CAN pro všechny její přenosové rychlosti. Definuje datové typy, komunikační mechanizmus a fyzickou vrstvu dle modelu OSI/ISO. [23]

2.5.1 Datový rámec

Datový rámec CANaerospace je složen z 8 bajů datového pole rámce CAN. První 4 bajty datového rámce jsou označované jako "Message Header / CANaerospace header (záhlaví zprávy)" a obsahují informaci o přenášené zprávě. Poslední 4 bajty obsahují přenášená data ("Message Data / CANaerospace data"). Obrázek 15 zobrazuje jednotlivé bajty, ze kterých se skládá CANaerospace rámec.



Obrázek 15: Formát CANaerospace zprávy [23], [24]

Node-ID Část Node-ID identifikuje odesílatele (EED a NOD zprávy viz tabulka 2), nebo adresáta (NSH/NSL zprávy viz tabulka 2). Pokud je hodnota Node-ID rovná 0, jedná se o zprávu určenou všem uzlům (definice uzlu viz kapitola 2.4).

Datatype (Datový typ) Tento bajt specifikuje kódování dat v posledních 4 bajtech CANaerospace rámce (specifikace datových typů je uvedena v [23]).

Service code (Servisní kód) Pro zprávy typu NOD může být tento bajt využit podle potřeby pro upřesnění přenášených dat. V případě nevyužití jsou všechny bity tohoto bajtu nastavené na 0. Pro zprávy typu NSL/NSH se jedná o servisní kód pro aktuální operaci.

Message code (Kód zprávy) Pro zprávy typu NOD tento bajt obsahuje počítadlo zpráv (s každou další zprávou dojde k jeho inkrementaci o 1). V případě dosažení hodnoty 255 počítadlo začíná opět od 0. Tato funkcionalita umožňuje uzlům v síti určit stáří obdržené zprávy. Pro zprávy typu NSL/NSH je kód zprávy použit pro specifikaci dané služby. [23]

2.5.2 Typy přenášených zpráv

CANaerospace definuje 6 základních typů zpráv, které jsou mohou být přenášeny po této sběrnici. Každý typ zprávy má definovaný rozsah CAN-ID, který určuje jeho prioritu. Jednotlivé typy přenášených zpráv jsou uvedeny v tabulce 2.

CAN Identifier for CANaerospace					
Identifikátor CAN rámce [dekadicky hexadecimálně]	**Zkratka**	**Název**	**Popis**	**Typ komunikace** [Anyone-to-Many Peer-to-Peer]	**Priorita**
0 - 127 000 - $07F_{HEX}$	EED	Emergency Event Data *Datový kanál pro nouzové události*	Transmitted asynchronously whenever a situation requiring immediate action occurs *Tento komunikační kanál se používá pro zprávy, které vyžadují okamžitou akci (tj. Degradaci systému nebo rekonfiguraci) a musí být přenášeny s velmi vysokou prioritou. Data nouzových událostí používají výhradně komunikaci ATM.*	ATM	Highest
128 - 199 080 - $0C7_{HEX}$	NSH	High Priority Node Service Data *Datový kanál služby uzlu s vysokou prioritou*	Transmitted asynchronously or cyclic with defined transmission intervals for operational commands (36 channels) *Tento komunikační kanál se používá pro interakce klient / server pomocí komunikace PTP. Odpovídající služby mohou být typu orientovaného na připojení i typu bez připojení. NSH lze také použít k podpoře testovacích a údržbových funkcí.*	PTP	
200 - 299 0C8 - $12B_{HEX}$	UDH	High Priority User-Defined Data *Uživatelsky definovaný datový kanál s vysokou prioritou*	Message/data format and transmission intervals entirely user-defined *Tento kanál je určen pro komunikaci, kterou nelze kvůli svým specifickým vlastnostem přiřadit jiným kanálům, aniž by byla porušena specifikace CANaerospace. Pokud se použije definovaný rozsah identifikátorů, může návrhář systému pro tyto kanály určit obsah zprávy a typ komunikace (ATM, PTP). K zajištění interoperability se důrazně doporučuje minimalizovat použití těchto kanálů.*	ATM PTP	
300 - 1799 12C - 707_{HEX}	NOD	Normal Operation Data *Normální provozní datový kanál*	Transmitted asynchronously or cyclic with defined transmission intervals for operational and status data *Tento komunikační kanál se používá k přenosu dat, která jsou generována během normálního provozu systému a popsána v seznamu přiřazení identifikátorů CANaerospace. Tyto zprávy mohou být přenášeny periodicky nebo neperiodicky i synchronně nebo asynchronně. Tento kanál musí používat všechny zprávy, které nelze přiřadit jiným komunikačním kanálům.*	ATM	
1800 - 1899 708 - $76B_{HEX}$	UDL	Low Priority User-Defined Data *Uživatelsky definovaný datový kanál s nízkou prioritou*	Message/data format and transmission intervals entirely user-defined *Tento kanál je určen pro komunikaci, kterou nelze kvůli svým specifickým vlastnostem přiřadit jiným kanálům, aniž by byla porušena specifikace CANaerospace. Pokud se použije definovaný rozsah identifikátorů, může návrhář systému pro tyto kanály určit obsah zprávy a typ komunikace (ATM, PTP). K zajištění interoperability se důrazně doporučuje minimalizovat použití těchto kanálů.*	ATM PTP	
1900 - 1999 76C - $7CF_{HEX}$	DSD	Debug Service Data *Servisní kanál*	Transmitted asynchronously or cyclic for debug communication & software download actions. *Tento kanál je určen pro zprávy, které jsou dočasně použity pouze pro účely vývoje a testování a nejsou přenášeny během normálního provozu. Pokud se použije definovaný rozsah identifikátorů, může návrhář systému pro tyto kanály určit obsah zprávy a typ komunikace (ATM, PTP).*	ATM PTP	
2000 - 2031 7D0 - $7EF_{HEX}$	NSL	Low Priority Node Service Data *Datový kanál služby uzlu s nízkou prioritou*	Transmitted asynchronously or cyclic for test & maintenance actions (16 channels). *Tento komunikační kanál se používá pro interakce klient / server pomocí komunikace PTP. Odpovídající služby mohou být typu orientovaného na připojení i typu bez připojení. NSL lze také použít k podpoře testovacích a údržbových funkcí.*	ATM PTP	Lowest

Tabulka 2: CANaerospace - Typy přenášených zpráv [23]

2.6 Přenosový protokol v prostředí LAN/ETHERNET - UDP

User Datagram Protocol (UDP) je protokol transportní vrstvy, přes který mohou jednotlivé počítače komunikovat po síti. [25] Ve své práci jsem jej použil ke komunikaci řídícího PC s avionickými jednotkami v samostatném prostředí LAN/ETHERNET.

2.7 Použitá kódování

Následující kapitola popisuje principy kódování, které jsou být použité pro reprezentaci čísel v binární podobě.

2.7.1 BNR

Binary numbers (BNR) se používá pro kódování datové části ARINC429 rámce. V tomto případě nejvýznamnější (MSB) bit určuje znaménko. Za tímto bitem následuje 18 bitů pro binární vyjádření čísla. [1]

2.7.2 BCD

Binary coded decimal (BCD) se používá pro kódování datové části ARINC429 rámce. Jedná se způsob kódování, při kterém se decimální číslo rozdělí na jednotlivé číslice. Poté se každá číslice binárně zakóduje do 4 bitů.

V následujícím příkladu je uveden převod čísla 915 dle BCD specifikace. Nejprve číslo 915 rozložím na jednotlivé číslice 9,1 a 5, které převedu do binárního tvaru o délce 4 bitů. [26]

$(9)_{10} = (1001)_2$
$(1)_{10} = (0001)_2$
$(5)_{10} = (0101)_2$

Následně se vytvořené binární řetězce složí dohromady. Výsledek převodu čísla 915 dle specifikace BCD je 100100010101.

2.7.3 Oktalové

Oktalové kódování je použito pro reprezentaci labelu (část ARINC429 rámce). Label je složen ze 3 oktalových čísel, které jsou kódované do 8 bitů. V ARINC429 rámci odpovídá nejvýznamější (MSB) bit labelu nejméně významnému (LSB) bitu ARINC429 rámce.

V následujícím příkladu je uveden převod labelu 204 do binární podoby. Nejprve label rozdělím na samostatné číslice, které převedu do binárního řetězce o délce 3 bity.

$(2)_8 = (010)_2$
$(0)_8 = (000)_2$
$(4)_8 = (100)_2$

Následně se vytvořené binární řetězce složí dohromady. Jelikož je label kódován do 8 bitů, je potřeba odebrat první bit zleva (značen červeně viz vztah 1).

$$010\ 000\ 100 \tag{1}$$

Poslední bit binární reprezentace labelu ve vztahu 1 odpovídá nejméně významnému bitu labelu. Pro použití v ARINC429 rámci je nutno převrátit pořadí bitů viz vztah 2. [1]

$$001\ 000\ 01 \tag{2}$$

2.7.4 Single-precision Float (IEEE 754)

Single-precision Float je v mé práci využito pro reprezentaci měřených hodnot zasílaných CAN rámcem. Numerická hodnota je uložena do 32 bitů (4 bajty). Těchto 32 bitů je rozděleno na 3 části v následujícím pořadí: znaménkový bit (1 bit), exponent (8 bitů), mantisa (23 bitů). [27]

1. **znaménkový bit (Z)**: Znaménkový bit se ukládá do 1 bitu. Nabývá hodnoty 1 pro zápornou numerickou hodnotu a hodnoty 0 pro kladnou numerickou hodnotu.
2. **exponent (E)**: Exponent je reprezentován 8 bity. Do nich se ukládá hodnota E posunutá o 127 (E+127). Zavedení tohoto posunutí umožňuje do exponentu ukládat i záporné hodnoty, což zlepší přesnost binární reprezentace i velmi malých čísel.
3. **mantisa (M)**: Mantisa je reprezentována 23 bity. Jedná se o normalizované číslo, tedy hodnota nejdůležitějšího bitu je vždy rovna 1. Tato 1 se nachází na levé straně desetinné čárky. Na pravé straně desetinné čárky se nachází binární řetězec, který je uložen v mantise. [27]

Převod čísla do binární reprezentace dle IEEE 754 V následujícím příkladu je uvedeno převedení čísla 45,45 do binární reprezentace dle standartu IEEE 754.

$$(45,45)_{10} = (101101,01110011001100110011)_2 \tag{3}$$

$$= 1,01101011100110011001100\,11 * 2^5 \tag{4}$$

Z rovnice 4 se určí následující části binární reprezentace:

- Z = $(0)_2$
- E = $(10000100)_2 \rightarrow (5 + 127)_{10}$
- M = $(01101011100110011001100)_2$ $\rightarrow$ 23 bitů binárního řetězce napravo od desetinné čárky v rovnici 4

Výsledná binární reprezentace čísla 45,45 dle standartu IEEE 754 je:

$$01000010001101011100110011001100$$

Převod čísla z binární reprezentace dle IEEE 754 do dekadické soustavy V následujícím příkladu je uvedeno převedení binárního řetězce 01000010001101011100110011001100 do dekadické soustavy dle standartu IEEE 754. Pro tento převod lze použít vzorec ze vztahu 5.

$$(-1)^Z \cdot (1.M)_2 \cdot 2^{E-127} \tag{5}$$

Binární řetězec lze rozdělit do následujících částí:

- Z = $(0)_2$
- E = $(10000100)_2$ = $(132)_{10}$
- M = $(01101011100110011001100)_2$

Po dosazení do vztahu 5 dostáváme následující:

$$(-1)^{Z} * (1.M)_2 \cdot 2^{E-127} \tag{6}$$

$$(-1)^{0} * (1, 01101011100110011001100)_2 \cdot 2^{132-127} \tag{7}$$

$$(101101, 011100110011001100)_2 = 45,45 \tag{8}$$

Výsledná dekadická hodnota binárního řetězce 01000010001101011100110011001100 je 45,45.

2.8 Dekódování ARINC429 a CAN rámců

2.8.1 Příklad dekódování ARINC429 rámce

V následujícím příkladu je uveden postup dekódování ARINC429 rámce (viz vztah 9).

$$10011100000100110010000000100000 \tag{9}$$

ARINC429 rámec ze vztahu 9 má LSB bit na začátku a MSB bit na konci binárního řetězce. Podle specifikace [1] musíme tedy binární řetězec otočit následujícím způsobem (první bit v řetězci je MSB, poslední je LSB):

$$00000100000001001100100000111001 \tag{10}$$

Tento zápis lze dále rozdělit na jednotlivé části datového rámce:

- bity pro label - 00111001
- bity pro SDI - 00
- bity pro data - 00100000001001100 10
- bity pro SSM - 00
- bit parity - 0

Dále lze zkontrolovat lichou paritu datového rámce. Ten obsahuje 9 logických 1 a 23 logických nul. Tedy počet logických 1 v datovém rámci je liché číslo a je splněna podmínka liché parity.

Dekódování labelu: Bity labelu jsou kódovány oktalově. Nejdříve bity otočíme a doplníme zleva na 9 bitů. Poté řetězec následujícím způsobem rozdělíme na trojice:

$$010011100 \tag{11}$$

$$010 \mid 011 \mid 100 \tag{12}$$

Dále každou takto vytvořenou trojici převedeme z binárního tvaru do dekadického:

$(010)_2 = (2)_{10}$
$(011)_2 = (3)_{10}$
$(100)_2 = (4)_{10}$

Dekódovaný label má hodnotu 234, což podle [1] odpovídá barometrickým datům. Datové bity mají BCD kódování a jsou rozděleny na 5 digitů. Rozlišení dat je 0,1 mb.

Dekódování datových bitů: Dle specifikace [1] víme, že data obsahují 5 digitů s BCD kódováním. Nejdříve binární řetězec doplníme zleva o 0 na 20 bitů. Dále zleva po 4 bitech na 5 digitů a převedeme na dekadické číslo:

$$00010000000100110010 \tag{13}$$

$$0001 \mid 0000 \mid 0001 \mid 0011 \mid 0010 \tag{14}$$

$$10132 \tag{15}$$

Jelikož je rozlišení 0,1 mb dostáváme přenášenou hodnotu 1013,2 mb. [22]

2.8.2 Příklad dekódování CAN rámce

Následující kapitola popisuje postup dekódování CAN rámce, jenž má následující binární tvar:

$$00010100000\,1\,00000\,1\,10001000010001101011100110011001100101101001000110101111111111 \tag{16}$$

Rozdělení CAN rámce na jednotlivá pole Před rozdělením CAN rámce na jednotlivá pole (viz kapitola 2.4) je nutné odstranit všechny přidané (Bit-stuffing) bity ve zprávě. Tyto bity jsou znázorněny žlutým podbarvením ve vztahu 16. Na následujícím vztahu 17 je uveden CAN rámec bez přidaných bit-suffing bitů.

$$00010100000000001000100001000110101110011001100110010110100100011010111111111111 \tag{17}$$

Dále CAN rámec rozdělíme do jednotlivých polí. Rozdělení CAN rámce do polí je znázorněno v tabulce 3. Pole jsou řazena dle jejich pořadí v CAN rámci.

Pole	Hodnota
START OF FRAME	0
ARBITRATION FIELD	Identifikátor - 00101000000 rtr - 0
CONTROL FIELD	rezervované bity - 00 DATA LENGHT CODE - 0100
DATA FIELD	0100001000110101110011001100
CRC FIELD	crc - 101101001000110 crc delimiter - 1
ACK FIELD	ACK - 0 ACK DELIMITER - 1
END OF FRAME	1111111
mezera mezi rámci	111

Tabulka 3: Rozdělení CAN rámce do polí

Kontrola rámce Z hodnot v tabulce 3 jsem ověřil, že pole CRC delimiter, ACK delimiter a END OF FRAME obsahují pouze platné hodnoty bitů. Jelikož se jedná o datový CAN rámec, lze ověřit, zda rtr bit zprávy obsahuje dominantní úroveň. Tabulka 4 uvádí očekávané hodnoty a reálné hodnoty bitů CAN rámce. Jak je vidět, u všech ověřovaných polí se shoduje očekávaná hodnota s reálnou.

Pole	Reálná hodnota	Očekávaná hodnota
CRC delimiter	1	1
ACK delimiter	1	1
rtr bit	0	0
END OF FRAME	1111111	1111111
mezera mezi zprávami	111	111

Tabulka 4: Poronání reálných a očekávaných hodnot CAN rámce

Výpočet CRC Pro kontrolu CAN rámce jsem dále ověřil hodnotu přijatého CRC kódu s CRC kódem, který se vypočítá z přijatého CAN rámce. CRC je zbytek po dělení dvou polynomů značených M(x) a G(x). Polynom M(x) je vytvořen z polí START OF FRAME, ARBITRATION FIELD, CONTROL FIELD a DATA FIELD. K těmto polím se poté přidá patnáct nul. Polynom G(x) se označuje jako generující polynom a je uveden ve vztahu 18.

$$G(x) = x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1 \tag{18}$$

Po provedení polynomiálního dělení dostáváme vypočtenou hodnotu CRC = 101101001000110. Tato hodnota je shodná s hodnotou v poli CRC FIELD. Tímto byla ověřena správnost CAN rámce.

Překlad identifikátoru Hodnota identifikátoru je kódovaná dle BNR (viz kapitola 2.7.1). Převedená hodnota identifikátoru je vyjádřena vztahem 19.

$$(00101000000)_2 = (320)_{10} \tag{19}$$

Identifikátor CAN rámce má hodnotu 320. Dle [23] je v CAN rámci přenášen datový typ FLOAT, který obsahuje hodnotu ”Baro corrected altitude”. Jednotky dat, zakódovaných v CAN rámci, jsou metry.

Dekódování dat V předchozí části bylo uvedeno, že je v CAN rámci přenášen datový typ FLOAT. Před převodem lze ověřit hodnotu pole DATA LENGHT CODE (dále značeno DLC). DLC má binární reprezentaci $(0100)_2 = (4)_{10}$. Tato hodnota odpovídá čtyřem bajtům [Byte], které jsou potřeba na zakódování datového typu float. Dekódovaná hodnota dat je 45,45 (princip dekódování FLOAT je uveden v kapitole 2.7.4).

Výsledná dekódovaná hodnota dat přenášená v rámci je 45,45 metrů.

Kapitola 3
Návrh a realizace avionické sítě, převodník sběrnic ARINC429 - CAN

3.1 Koncepce avionické sítě

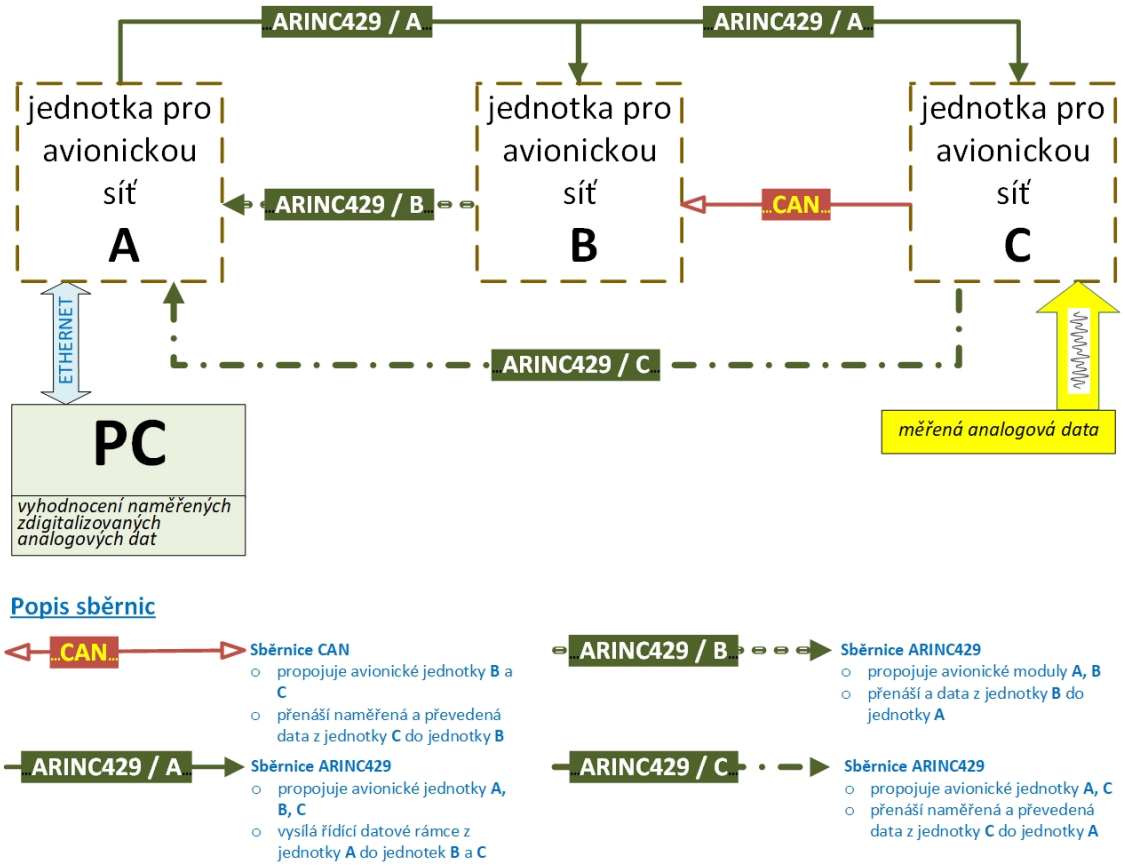
Pro realizaci zadání DP bylo nutno navrhnout avionickou síť minimálně se třemi avionickými jednotkami (hlavní jednotka vyčítající informace po sběrnicích ARINC429, jednotka pro překlad rámců sběrnice CAN na rámce sběrnice ARINC429 a jednotka realizující měření analogových veličin s následným převodem digitalizovaných dat do datových rámců sběrnic CAN a ARINC429. Tato navržená koncepce avionické sítě je popsána v kapitole 3.1.1.

Na základě zkušeností nabytých při realizaci avionické sítě se třemi jednotkami a konzultacích s vedoucím mé diplomové práce bylo původní zadání rozšířeno o návrh avionické sítě pouze se dvěma jednotkami, kde bude možno demonstrovat měření konkrétních avionických veličin (Baro corrected altitude 1, Computed Airspeed, Exhaust Gas Temperature, Pitch Angle) s jejich následným současným přenosem sběrnicemi ARINC429 a CAN v odpovídajících standardech. Tato koncepce je podrobněji rozepsána v kapitole 3.1.2.

3.1.1 Avionická síť se třemi jednotkami

Na obrázku 16 je uvedeno blokové schéma propojení třech avionických jednotek realizující základní zadání mé diplomové práce. Fyzická realizace je uvedena na obrázku 36 v příloze B.1.

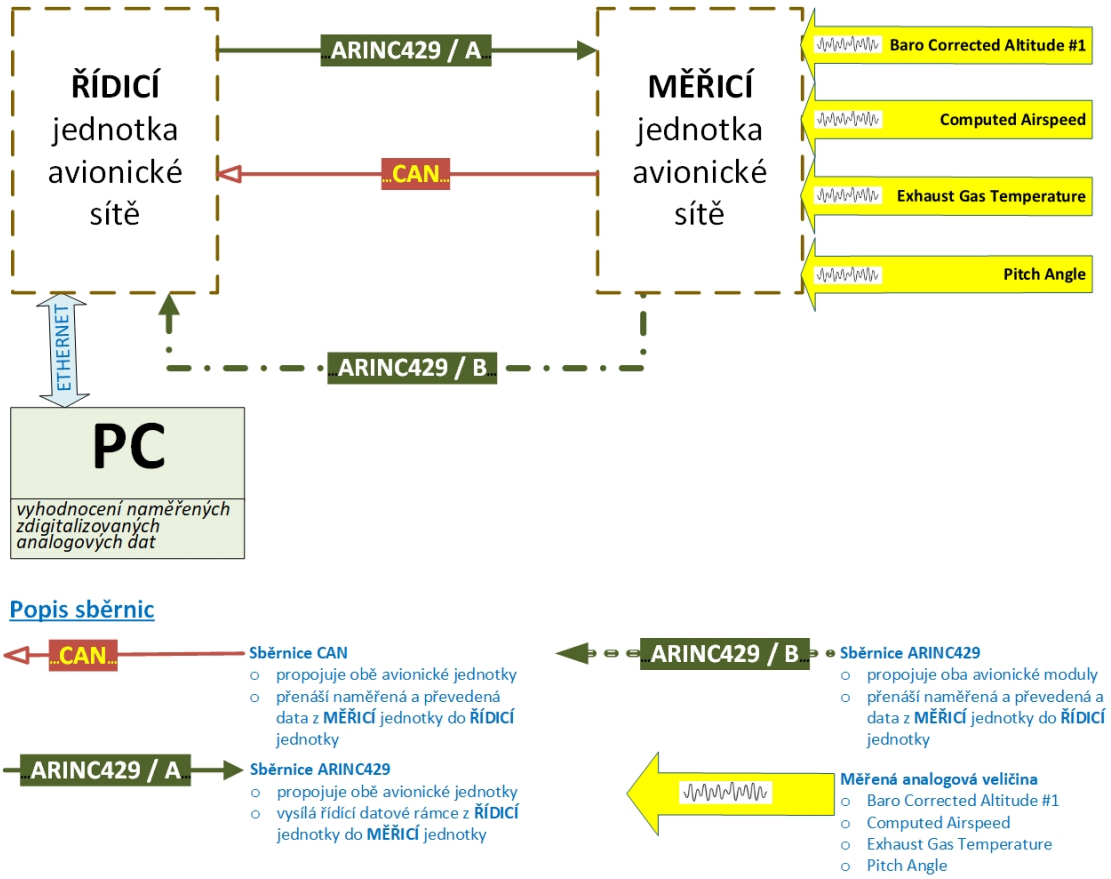
V navrženém propojení jsou všechny jednotky avionické sítě propojeny sběrnicí ARINC429/A. Na této sběrnici řídicí jednotka A vysílá oběma podřízeným jednotkám (B, C) řídicí datové rámce. Sběrnice ARINC429/C propojuje řídicí jednotku C, která vysílá po této sběrnici převedená vyčtená data do podřízené jednotky A. Vyčtená data jsou z jednotky C paralelně vysílána sběrnicí CAN do jednotky B. V jednotce B jsou datové rámce sběrnice CAN převedeny do protokolu sběrnice ARINC429 a vyslány sběrnicí ARINC429/B do podřízené jednotky A. Z jednotky A se načtená data ze sběrnic ARINC42/B a ARINC429/C přenesou do řídícího PC, kde se data dále zpracují a vyhodnotí. Zmíněné scénáře využití jednotlivých sběrnic jsou uvedené na obrázcích v příloze A.



Obrázek 16: Blokové schéma propojení třech avionických jednotek jednotlivými sběrnicemi

3.1.2 Avionická síť se dvěma jednotkami - výuková aplikace

Rozšířené zadání mé diplomové práce spočívá v konfiguraci propojení jednotek avionické sítě tak, aby bylo možno tuto konfiguraci využít ve výuce při demonstraci funkcionalit sběrnic ARINC429 a CAN. Zvolil jsem propojení dvou jednotek uvedené na obrázku 17 (fyzická realizace je uvedena na obrázku 37 v příloze B.1). Tato koncepce umožňuje měřit na každém kanálu A/D převodníku MĚŘICÍ jednotky jednu z avionických veličin uvedených v tabulce 5 a následně tuto změřenou veličinu převést dle požadovaných parametrů do rámců sběrnic ARINC429 a CAN a přenést do ŘÍDICÍ jednotky.



Obrázek 17: Blokové schéma propojení dvou avionických jednotek jednotlivými sběrnicemi

Přijatá data jsou z ŘÍDICÍ jednotky zasílána do grafické aplikace připojeného PC. V grafické aplikaci jsou následně přijatá data zobrazena v binární podobě příslušných datových rámců a student/ka má možnost přijatá data dále zpracovat (například je ručně přeložit a následně ověřit správnost řešení).

A/D kanál	Rozsah	Převáděná veličina	ARINC429			CAN	
			Jednotky	Label	Eq. ID	Jednotka	ID
1	4p8V	Baro corrected altitude #1	ft - Feets	204	006	m - Meters	320
2	4p8V	Computed Airspeed	kn - Knots	206	140	m/s	317
3	150mV	Exhaust Gas Temperature	^oC - Celsius	345	01A	K - Kelvin	520
4	4p8V	Pitch Angle	^{o} - Degrees	324	004	^{o} - Degrees	311

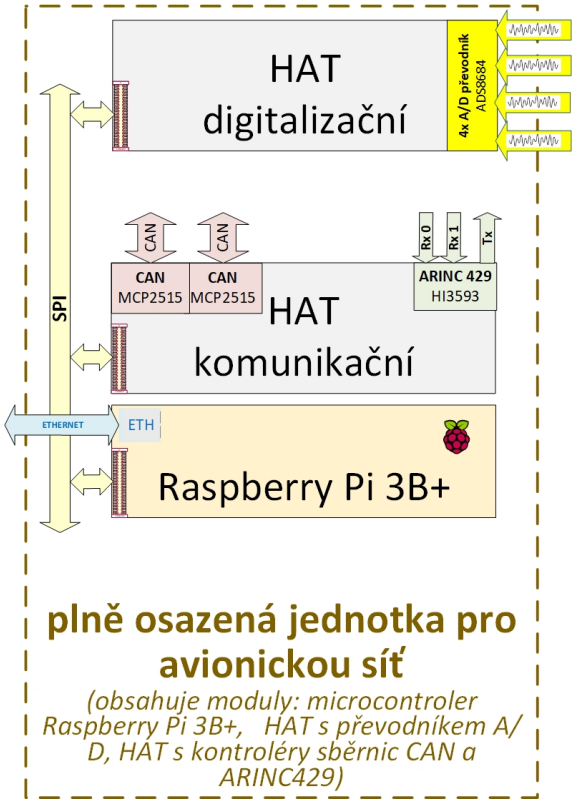
Tabulka 5: Měřené veličiny A/D kanálů a konfigurace datových rámců

3.2 Návrh HW konfigurace jednotek pro avionickou síť

Při návrhu konfigurace avionických jednotek (A, B, C / ŘÍDICÍ, MĚŘICÍ) jsem vycházel z výstupů projektu studenta Patricka Rodriguesa Rocha z brazilské univerzity FEDERAL UNIVERSITY OF MINAS GERAIS s názvem "REALIZATION OF THE AVIONICS NETWORK". Projekt byl zpracován v rámci programu "ACADEMIC EXCHANGE PROGRAM BRAZIL – CZECH REPUBLIC" a jeho výstup byl prezentován formou posteru v květnu 2020 v Praze na ČVUT Fakultě elektrotechnické. Patrick Rodrigues Rocha realizoval projekt na Raspberry Pi 3 B+, ke kterému připojil již dříve navržený modul HAT obsahující kontroléry MCP2515 pro sběrnici CAN a převodník HI3593 pro sběrnici ARINC429 (Hardwarovou realizaci tohoto modulu navrhl student Patrik Bachan).

Aby bylo možno jednotku pro avionickou síť využít k měření analogových signálů rozšířil jsem jednotku navrženou studentem Patrickem Rodriguesem Rochem o další modul HAT s funkcionalitou pro převod analogových vstupních dat do digitální formy. Pro převod analogového signálu do digitální formy obsahuje tento rozšiřující modul obvod ADS8684 se čtyřmi A/D převodníky.

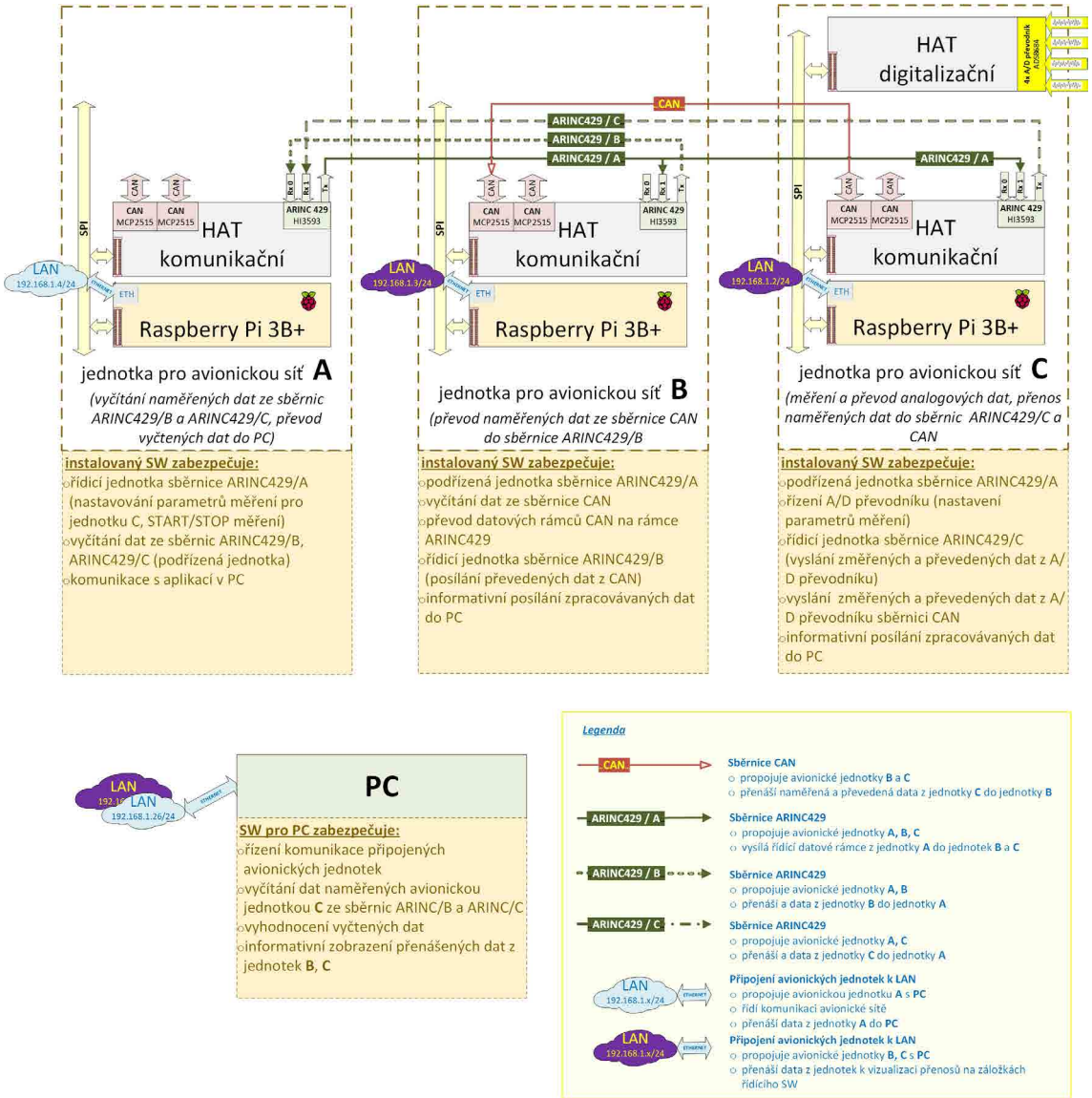
Propojení jednotlivých modulů avionické jednotky je uvedeno na obrázku 18.



Obrázek 18: Blokové schéma propojení všech modulů plně osazené jednotky pro avionickou síť

3.2.1 Propojení třech jednotek pro avionickou síť

V zapojení třech jednotek avionické sítě dle koncepce uvedené na obrázku 16 jsem použil pro jednotky avionické sítě označené A, B navrženou jednotku avionické sítě osazenou pouze komunikačním HATem. Jednotka C avionické sítě je pak osazena i digitalizačním HATem. Výsledné propojení všech modulů jednotlivými sběrnicemi s vyznačeným směrem přenosu datových rámců na jednotlivých sběrnicích je uvedeno na obrázku 19. Jednotka A je s PC propojena protokolem UDP v prostředí LAN/ETHERNET. Toto propojení umožní na PC zobrazovat přijatá data a řídit avionickou síť.



Obrázek 19: Blokové schéma propojení třech avionických jednotek jednotlivými sběrnicemi

3.2 Propojení třech jednotek pro avionickou síť

Aby bylo možno demonstrovat funkcionalitu všech jednotek v řídícím programu na PC, rozhodl jsem se jednotky B a C připojit ke stejné LAN protokolem UDP. Toto propojení není využíváno k datovým přenosům mezi avionickými jednotkami a je na obrázku zobrazeno fialovou barvou.

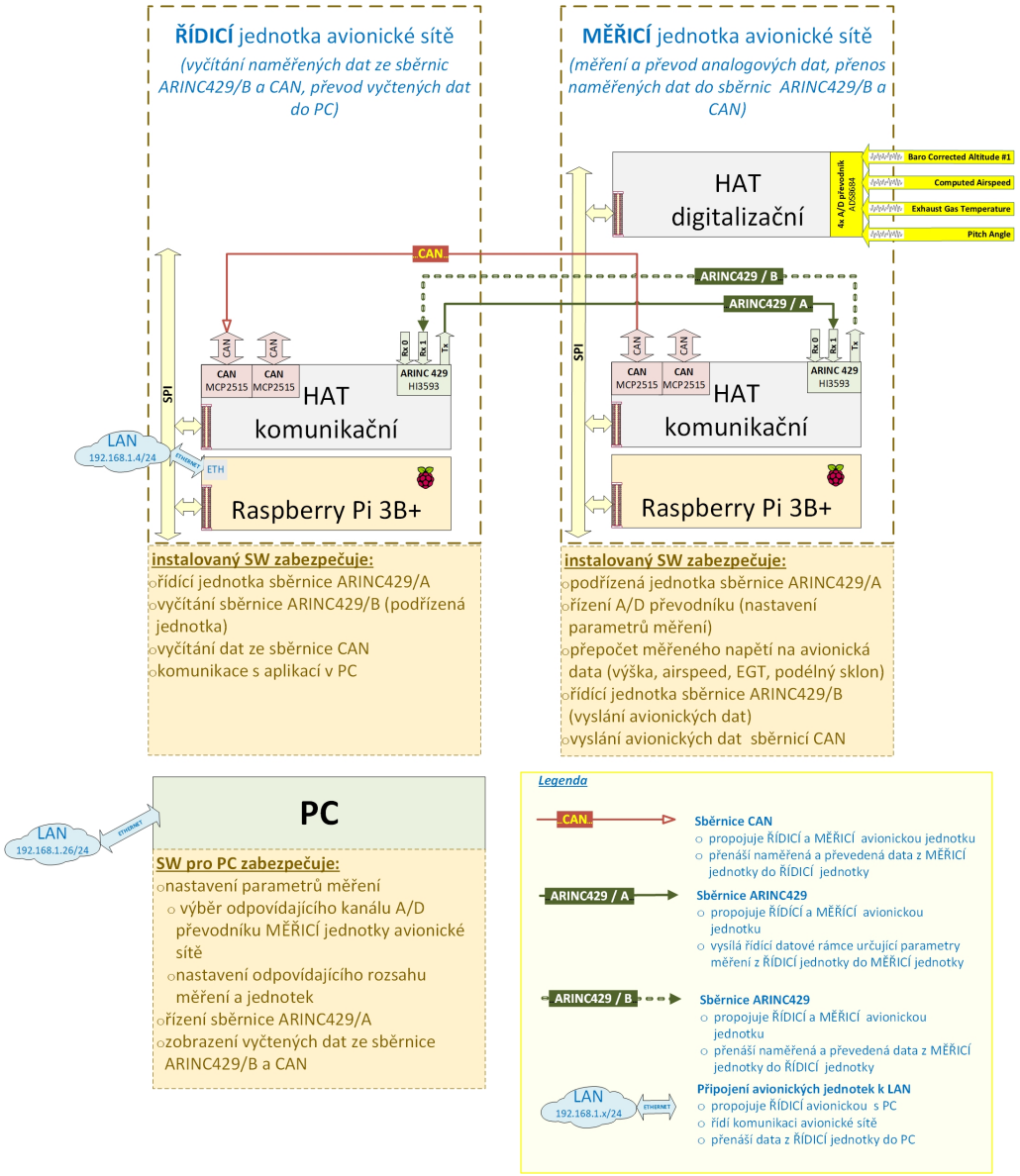
ETHERNET/LAN je navržena jako izolovaná síť bez dalšího propojení k internetu se statickým přidělením IP adres. Parametry sítě jsou uvedeny v tabulce 6.

Zařízení	**Hodnota**	**Čísla portů**
NETWORK: MASK: GATEWAY:	192.168.1.0 255.255.255.0 není potřeba, jedná se o lokální síť	
IP adresa - PC IP adresa - Jednotka A IP adresa - Jednotka B IP adresa - Jednotka C	196.168.1.26 196.168.1.4 196.168.1.3 196.168.1.2	 65000, 64000 63000, 62000 61000, 60000

Tabulka 6: Konfigurace LAN sítě se třemi jednotkami

3.2.2 Propojení dvou jednotek pro avionickou síť - výuková aplikace

Propojení obou modulů jednotlivými sběrnicemi s vyznačeným směrem přenosu datových rámců je zobrazeno na obrázku 20. V tomto zapojení stačí ŘÍDICÍ jednotku osadit pouze komunikačním HATem a MĚŘICÍ jednotku osadit i digitalizačním HATem. ŘÍDICÍ jednotka je s PC propojena protokolem UDP v prostředí LAN/ETHERNET. Toto propojení umožňuje na PC zobrazovat přijatá data a řídit avionickou síť.



Obrázek 20: Blokové schéma zapojení dvou avionických jednotek pro současný přenos naměřených dat sběrnicemi ARINC429 a CAN

ETHERNET/LAN je navržena jako izolovaná síť bez dalšího propojení k internetu se statickým přidělením IP adres. Parametry sítě jsou uvedeny v tabulce 7.

Zařízení	Hodnota	Čísla portů
NETWORK: MASK: GATEWAY:	192.168.1.0 255.255.255.0 není potřeba, jedná se o lokální síť	
IP adresa - PC IP adresa - ŘÍDICÍ Jednotka	196.168.1.26 196.168.1.4	 65000, 64000

Tabulka 7: Konfigurace LAN sítě se dvěma jednotkami

.

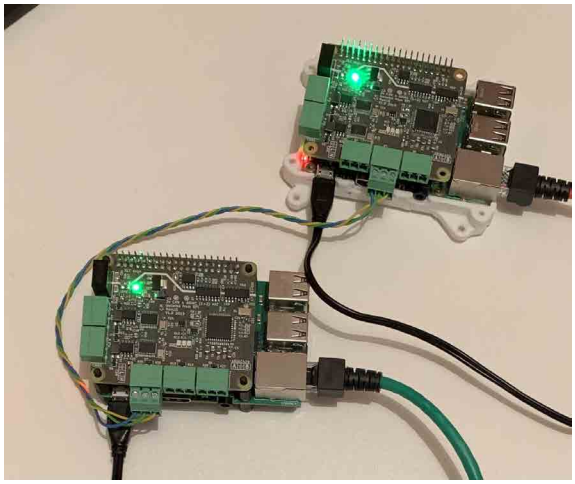
3.3 Oživení a zprovoznění jednotek avionické sítě

Pro vlastní HW realizaci diplomové práce jsem měl k dispozici 3x minipočítač Raspberry Pi 3B+, 2x komunikační HAT s výstupními rozhraními CAN a ARINC429, 1x čtyř-kanálový digitalizační HAT a driver pro komunikační HAT instalovaný na Raspberry Pi 3B+ volně dostupný ze stránek [7].

Abych mohl zrealizovat řešení navržené na obr. 16, musel jsem vyrobit jeden nový modul - komunikační HAT s výstupními rozhraními CAN a ARINC429.

3.3.1 HW oživení a zprovoznění

Během prvního zapojení dodaných modulů minipočítače Raspberry Pi 3B+ jsem zjistil, že jeden z modulů není funkční. Závadu jsem diagnostikoval a odstranil - byl nefunkční měnič DC/DC (součástka značená U6 na desce plošného spoje komunikačního modulu) realizovaný obvodem Recom RFM-0505s. Obvod jsem nahradil funkčně ekvivalentním obvodem SIM1-0505.



Obrázek 21: Propojení Raspberry Pi 3B+ s instalovaným komunikačním HAT sběrnicí ARINC429

3.3.2 Výroba třetího modulu HAT s rozhraními CAN a ARINC429

Součástí diplomové práce bylo vytvoření jednoho komunikačního HATu. Tento HAT jsem vytvořil na základě již vytvořené desky plošných spojů, která byla poskytnuta vedoucím mé diplomové práce. K pájení jsem použil bezolovnatou pájku s obsahem tavidla. Pájel jsem mikropájkou o teplotě 350 °C.

S deskou plošných spojů mi bylo poskytnuto i schéma zapojení. Pro určení konkrétní hodnoty součástek z tohoto schématu jsem vygeneroval BOM soubor (Bill of materials - seznam všech použitých komponent, včetně konkrétních hodnot). Desku plošných spojů jsem osazoval v následujícím pořadí: integrované obvody, pasivní součástky, konektory.

Při osazování integrovaných obvodů je nutné dbát na jejich správnou orientaci. Z tohoto důvodu se používá značení prvního vývodu součástky na jejím pouzdře a na desce plošných spojů. Pro pasivní součástky (jako jsou rezistory a keramické kondenzátory) není nutné dodržovat polaritu. Toto je nutné dodržovat pouze u diod. Po osazení všech součástek povrchové montáže (SMD) jsem osadil konektory, které mají drátové vývody (trough holes). V průběhu osazování jsem všechny pájené spoje kontroloval pod lupou. Požadovaný stav spojů bez defektů musí být hladký, lesklý a v žádném případě nesmí dojít k neúmyslnému spojení dvou plošek.

3.3.3 Základní SW oživení a zprovoznění dodaných HATů

Při základním zprovozňování a SW oživování sestavených jednotek A, B, C (viz výše zmíněné konfigurace) se u všech takto sestavených jednotek vyskytl problém s komunikačním HATem. Tento HAT umožňoval pouze komunikaci po sběrnici CAN (přijímání i odesílání zpráv). Na sběrnici ARINC429 nefungovalo posílání ani příjem zpráv.

Nejprve jsem se rozhodl ověřit, zda je HW tohoto HATu v operačním systému Raspberry Pi správně popsán a zkonfigurován. V operačním systému je popis připojených HW zařízení uložen v tzv. DeviceTree souborech. Tyto soubory popisují potřebné datové struktury pro komunikaci HW s operačním systémem. Pro ověření správně začleněných HW modulů do operačního systému lze použít příkaz ”lsblk”, který na standartní výstup linuxového terminálu vypíše všechny HW moduly dostupné v operačním systému.

Protože modul komunikačního HATu byl v systému správně začleněn (byl v seznamu vypsaného HW příkazem ”lsblk”), rozhodl jsem se ověřit, zda jsou správně nainstalovány příslušné drivery nutné pro správnou funkcionalitu daného HW. Pro ověření jsem použil příkaz ”dmesg”, který na standardní výstup vypíše všechny zprávy obsahující výsledky operací linuxového jádra (jednou z takovýchto operací je i inicializace a provoz ovladačů zařízení, viz kapitola 2.2). Při anylýze těchto zpráv jsem zjistil, že ovladač pro komunikační HAT, není správně nainstalován. Pro korektní instalaci driverů bylo potřeba změnit instalovanou verzi jádra operačního systému na verzi 5.10.92-v7+ a opětovně nainstalovat potřebné ovladače do Raspberry Pi.

3.4 Návrh SW vybavení

Pro tvorbu SW vybavení pro jednotky avionické sítě i pro Řídicí SW na PC jsem se rozhodl použít objektově orientovaný programovací jazyk Python s odpovídajícími knihovnami.

Pro SW instalovaný v avionických jednotkách jsem zvolil řešení, kdy je v každé jednotce instalován identický SW - viz kap. 3.4.1.

Pro Řídicí SW pro PC jsem naopak, z důvodu přehlednosti, zvolil pro každou variantu avionické sítě (viz kap. 3.1) samostatnou SW aplikaci. Pro vytvoření grafického uživatelského rozhraní jsem použil knihovnu "Tkinter", která umožňuje vytvářet přehledné uživatelské grafické rozhraní SW aplikace. Zdrojové kódy aplikace jsem následně přeložil nástrojem "PyInstaller" do spustitelného .EXE souboru v prostředí WINDOWS.

3.4.1 Požadavky na SW funkcionalitu jednotek pro avionickou síť

Jak vyplývá z obou variant zapojení avionických jednotek (viz obr. 19 a 20) má každá z jednotek pro avionickou síť (A, B, C / ŘÍDICÍ, MĚŘICÍ) jinou základní funkcionalitu - jiné požadavky na instalovaný SW v každé jednotce (viz tabulky 8 a 9).

Jednotka	Funkcionalita
A	- řídicí jednotka avionické sítě (sběrnice ARINC429/A) (nastavování start/stop měření a parametrů digitalizačního HATu) - vyčítání dat ze sběrnic ARINC429/B a ARINC429/C - komunikace s řídicí grafickou aplikací v PC
B	- podřízená jednotka avionické sítě (řízena sběrnicí ARINC429/A) - přijímač dat ze sběrnice CAN - převod rámce CAN na rámec ARINC429 - vysílač dat po sběrnici ARINC429/B
C	- podřízená jednotka avionické sítě (sběrnice ARINC429/A) - vysílač dat po sběrnici CAN a ARINC429/C - řízení digitalizačního HATu (nastavení měření průběhu napětí na specifikovaných kanálech A/D převodníku)

Tabulka 8: Základní funkcionalita jednotlivých jednotek avionické sítě se 3 jednotkami

Z výše uvedeného by se tedy dalo usoudit, že je optimální vyvinout pro každou jednotku specifický SW, který ji bude obsluhovat. Toto řešení je ale velice nepraktické - neumožní snadnou záměnu jednotlivých jednotek (HW výměnu jednotky připojené ke sběrnici), neboť by bylo vždy nutno do každé jednotky tento specifický SW instalovat.

Abych tuto nevýhodu vyřešil, tak jsem zvolil řešení, že v každé jednotce bude nainstalován stejný SW a dle nasazení jednotky (A, B, C / ŘÍDICÍ, MĚŘICÍ) se budou aktivovat pouze potřebné specifické funkce. Toto navržené řešení má následující výhody:

• **Jednoduchá modifikovatelnost:** Při rozdělení jednoho programu řešícího komplexní zadání na dílčí spolupracující pod-programy/procesy obsluhující jednotlivé konkrétní funkcionality se zlepšuje celková modifikovatelnost SW programu. V případě potřeby je možné modifikovat pouze

Jednotka	Funkcionalita
ŘÍDICÍ	- řídicí jednotka avionické sítě (sběrnice ARINC429/A) - vyčítání dat ze sběrnic ARINC429/B a CAN - komunikace s řídicí grafickou aplikací v PC
MĚŘICÍ	- podřízená jednotka avionické sítě (sběrnice ARINC429/A) - řízení digitalizačního HATu (nastavení měření průběhu napětí na specifikovaném kanále A/D převodníku) - přepočet měřeného napětí na avionická data (Baro corrected altitude #1, Computed Airspeed, Exhaust Gas Temperature, Pitch Angle) - posílání leteckých dat po sběrnici ARINC429/B a CAN

Tabulka 9: Základní funkcionalita jednotlivých jednotek avionické sítě se dvěmi jednotkami

daný pod-program/proces, který definuje konkrétní funkcionalitu bez potřeby složitých zásahů do komplexního zdrojového kódu. Kromě úprav stávajících funkcionalit poskytuje toto řešení možnost jednoduššího definování a vytváření nových funkcí daného zařízení.

• **Modularita:** Navržené pod-programy/procesy jsou vytvořené tak, aby mohly fungovat pro každou jednotku v avionické síti. V programovém kódu jednotlivých jednotek v avionické sběrnici, jsou následně použity pouze ty pod-programy/procesy, které jsou potřebné pro její funkcionalitu. Při přidání dalšího HW vybavení do dané jednotky, či změně její funkce v avionické sběrnici stačí pouze použít už definované pod-programy/procesy bez nutnosti složitých zásahů do zdrojového kódu SW.

• **Zpracování časově kritických úloh:** Rozdělením komplexní úlohy na dílčí pod-programy/procesy lze tyto vykonávat paralelně a tím eliminovat případnou ztrátu dat způsobenou postupným vykonáváním jednotlivých úloh. Potřebné činnosti obsluhované pod-programem/procesem jsou vykonávány v čase jejich vzniku a nedochází ke zdržování jinými nesouvisejícími činnostmi. Takovouto typickou úlohou je zejména zpracování asynchronních událostí. U jednotky pro avionickou sběrnici navrženou v předchozí kapitole se jedná prakticky o všechny obsluhované činnosti (posílání dat na sběrnici ARINC429, vyčítání dat z A/D převodníku, ...).

• **Čitelnost zdrojového kódu:** Rozdělením komplexní úlohy na dílčí pracovní činnosti se zlepšuje celková čitelnost kódu. Každá pracovní činnost je ve zdrojovém kódu definovaná pouze jednou v samostatném procesu.

3.5 SW vybavení pro jednotku avionické sítě

Jak již bylo zmíněno v předchozí kapitole 3.4.1, je obsluha jednotlivých činností v jednotce pro avionickou síť realizována paralelními procesy. V prostředí jazyka Python jsem jednotlivé procesy podle jejich funkcionality rozdělil do tříd (tzv. Classes), viz Tabulka 10. Pro realizaci avionické sítě jsem tedy vytvořil 4 třídy, kde metody těchto tříd představují jednotlivé procesy. Aby bylo zajištěno paralelní vykonávání procesů, jsou jednotlivé metody spouštěny jako vlákna knihovny "threading". Popisy jednotlivých tříd jsou uvedeny v následujících podkapitolách.

Třída	**Metody**	**Popis funkcionality**
ARINC_handler	RX0	Příjem ARINC429 dat z přijímače RX0 obvodu HI-3593
	RX1	Příjem ARINC429 dat z přijímače RX1 obvodu HI-3593
	TX	Vysílání ARINC429 dat vysílačem obvodu HI-3593
CAN_handler	RX	Příjem CAN dat
	TX	Odesílání CAN dat
UDP_handler	RX	Příjem zpráv z grafické aplikace PC
	TX	Odesílání zpráv do grafiké aplikace PC
ADC_handler	set_channels	Nastavení rozsahu kanálů A/D převodníku
	measurement	Zpracování a měření kanálů A/D převodníku

Tabulka 10: Seznam vytvořených tříd a jejich metod

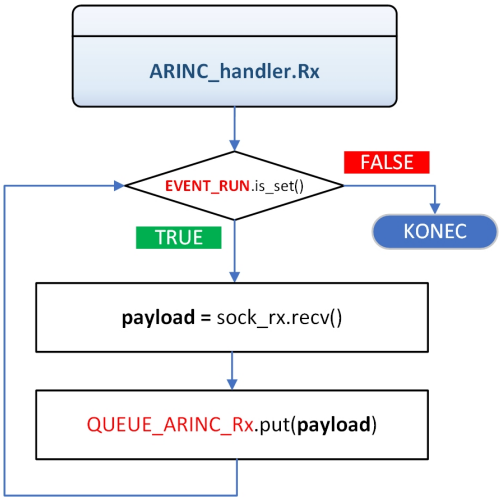
3.5.1 ARINC_handler - *třída pro příjem a posílání ARINC429 zpráv*

ARINC_handler - třída pro příjem a posílání ARINC429 zpráv		
Atributy třídy	QUEUE_ARINC_RX	- *FIFO fronta, do které jsou ukládány přijaté zprávy*
	QUEUE_ARINC_TX	- *FIFO fronta, do které jsou ukládány zprávy k odeslání*
	EVENT_RUN	- *Sdílená proměnná pro start/konec vykonávání procesů*
Metody třídy	RX0	- *Příjem ARINC429 dat*
	RX1	- *Příjem ARINC429 dat*
	TX	- *Odesílání ARINC429 dat*
Vstupní proměnné	sock_rx0	- *socket pro komunikaci s ARINC přijímačem*
	sock_rx1	- *socket pro komunikaci s ARINC přijímačem*
	sock_tx	- *socket pro komunikaci s ARINC vysílačem*
Použité knihovny	-	

Tabulka 11: Parametry třídy ARINC_handler

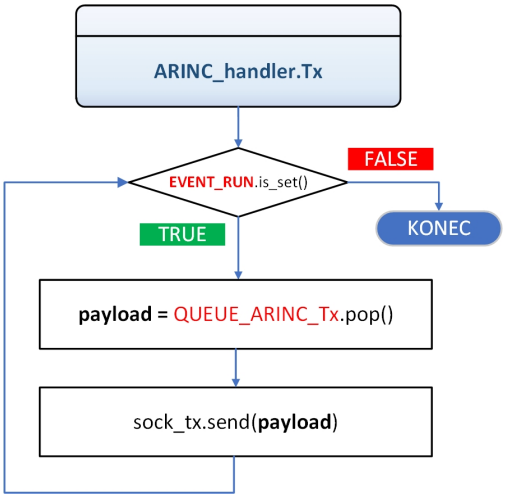
Procesy RX0, RX1: Pro proces čtení datových rámců ze sběrnice ARINC jsem použil knihovnu pro socketovou komunikaci. Metoda "sock_rx.recv()" monitoruje aktivitu na daném přijímači obvodu HI-3593 a čeká na příchozí ARINC429 data. V okamžiku přijetí platného datového rámce ARINC429 obvodem HI-3593 je poskytuje "sock_rx.recv()" data, která mají podobu datového typu "Byte-like array". Prvních osm bajtů obsahuje čas, kdy byl přijat datový rámec ARINC429 (tzv. timestamp). V následujícím zpracování tuto informaci nepoužívám.

Následující 4 přijaté bajty obsahují přijatý ARINC429 rámec. Ten je pro další zpracování převeden do hexadecimální podoby a uložen do FIFO fronty QUEUE_ARINC_RX. Poté následuje návrat na začátek procesu čtení. Data z fronty QUEUE_ARINC_RX mohou být podle použité konfigurace následně vyčítána procesem pro nastavení rozsahů kanálů A/D převodníku (jednotka C), nebo procesem pro převod ARINC429 rámce na rámec CAN (jednotka B).



Obrázek 22: Vývojový diagram procesu RX pro obsluhu sběrnice ARINC429

Proces TX: Použití FIFO fronty slouží k synchronizaci a k bezpečnému předávání dat mezi procesy avionické sítě. Příkladem takového předání dat je proces pro odesílání ARINC429 rámců. Tento proces čeká, dokud do FIFO fronty QUEUE_ARINC_TX není uložen ARINC429 rámec. Uložený rámec je následně vyčten a posílán vysílačem obvodu HI-3593 do jedné ze sběrnic (ARINC429/A, ARINC429/B, ARINC429/C).



Obrázek 23: Vývojový diagram procesu TX pro obsluhu sběrnice ARINC429

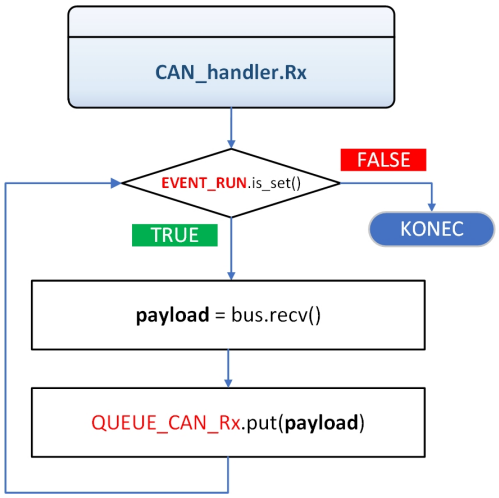
3.5.2 CAN_handler - *třída pro příjem a posílání CAN zpráv*

CAN_handler - třída pro příjem a posílání CAN zpráv		
Globální proměnné	QUEUE_CAN_RX	- *FIFO fronta, do které jsou ukládány přijaté zprávy*
	QUEUE_CAN_TX	- *FIFO fronta, do které jsou ukládány zprávy k odeslání*
	QUEUE_ARINC_TX	- *FIFO fronta, do které jsou ukládány zprávy k odeslání*
	EVENT_RUN	- *Sdílená proměnná pro start/konec vykonávání procesů*
Metody třídy	RX	- *Příjem CAN dat*
	TX	- *Odesílání CAN dat*
	CAN_to_ARINC	- *Převod CAN rámce do ARINC429*
Vstupní proměnné	can	- *Objekt knihovny CAN, který definuje zvolenou sběrnici*
Použité knihovny	ARINC.py - *vytvořená v rámci mé práce, stará se o kódování/dekódování ARINC429 rámců*	
	python-can [28]	

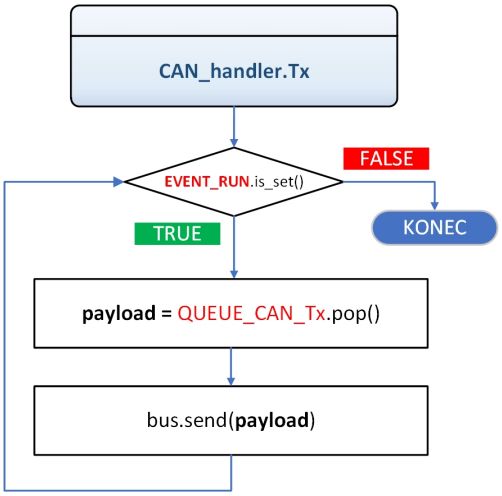
Tabulka 12: Parametry třídy CAN_handler

Procesy RX, TX: Jak lze vidět z obrázků 29 a 30, procesy pro přijímání a odesílání CAN rámců mají obdobnou funkcionalitu jako v případě sběrnice ARINC429. Liší se pouze forma komunikace s obvodem MCP2515. Pro tuto komunikaci jsem použil standardní knihovnu ”python-can”, která poskytuje abstrakci pro sběrnici CAN.

Knihovna ”python-can” definuje instanci ”can.Bus()”, což je instance použitá jako abstrakce CAN sběrnice, která opět přes socket komunikuje s obvodem MCP2515. Pro odeslání a příjem CAN rámců do/ze sběrnice jsou použity metody s obdobným názvem jako v případě sběrnice ARINC429 (pro odeslání metoda ”send”, pro příjem metoda ”recv”). Přijaté rámce a rámce k odeslání jsou opět ukládány do FIFO front QUEUE_CAN_TX resp. QUEUE_CAN_RX podle požadované funkcionality.

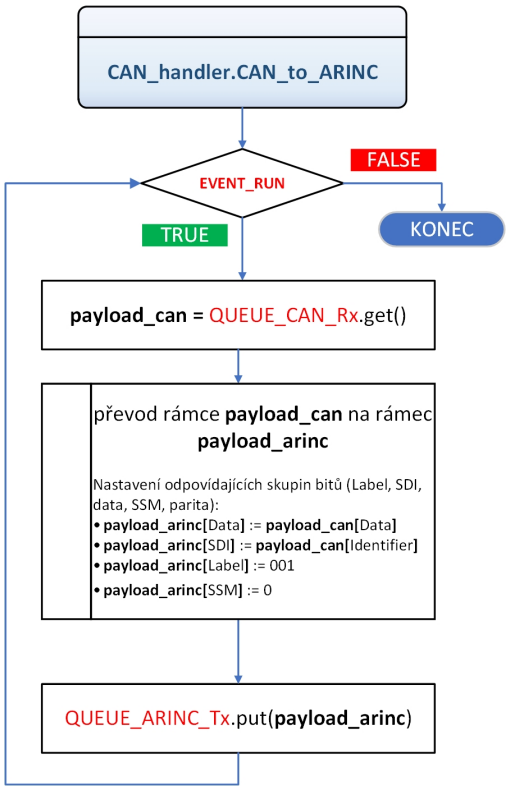


Obrázek 24: Vývojový diagram procesu RX pro obsluhu sběrnice CAN



Obrázek 25: Vývojový diagram procesu TX pro obsluhu sběrnice CAN

CAN_to_ARINC Proces pro převod CAN rámců na ARINC429 rámce čeká na data ve FIFO frontě QUEUE_CAN_RX. Přijaté CAN rámce z FIFO fronty následně dekóduje a použije pro tvorbu ARINC429 rámců. Pro pole SDI rámce ARINC429 je použito pole arbitration_id CAN rámce. Pro datové pole ARINC429 rámce jsou použita data z datového pole CAN rámce. Pro pole Label ARINC429 rámce jsem použil hodnotu '001'. Datová zpráva je ukládána do 18 signifikantních bitů.



Obrázek 26: Vývojový diagram procesu CAN_to_ARINC pro převod datových rámců

3.5.3 ADC_handler - *třída pro vyčítání zpráv z digitalizačního HATu*

ADC_handler - třída pro vyčítání zpráv z digitalizačního HATu		
Globální proměnné	QUEUE_CAN_RX	- *FIFO fronta, do které jsou ukládány přijaté zprávy*
	QUEUE_CAN_TX	- *FIFO fronta, do které jsou ukládány zprávy k odeslání*
	QUEUE_ARINC_RX	- *FIFO fronta, do které jsou ukládány přijaté zprávy*
	QUEUE_ARINC_TX	- *FIFO fronta, do které jsou ukládány zprávy k odeslání*
	EVENT_RUN	- *sdílená proměnná pro start/konec vykonávání procesů*
	ADC_RANGES	- *list s jednotlivými A/D kanály a jejich rozsahy pro měření*
Metody třídy	set_channels	- *nastavuje rozsahy A/D kanálů pro měření*
	measurement	- *měření hodnot na kanálech digitalizačního HATu*
Vstupní proměnné	send_ARINC	- *True/False, poslat měření přes ARINC429?*
	send_CAN	- *True/False, poslat měření přes CAN*
Použité knihovny	ARINC.py python-can [28] ADC_TCPtoUDP.py	- *upravená knihovna poskytnutá s digitalizačním HATem*

Tabulka 13: Parametry třídy ADC_handler

set_channels: Digitalizační HAT umožňuje měření napětí v závislosti na nastavení od rozsahu 150 mV až do rozsahu 38.4 V.

Pro komunikaci s digitalizačním HATem jsem vytvořil vlastní definici rámců ARINC429. Rámec pro nastavení měřeného rozsahu napětí A/D převodníku má hodnotu label 001. V části SDI je zakódovaný kanál A/D převodníku, pro který je platná zaslaná hodnota rozsahu. Hodnota SDI může nabývat hodnot od 0 do 3, kde 0 odpovídá prvnímu kanálu A/D převodníku a hodnota 3 odpovídá čtvrtému kanálu A/D převodníku. Požadovaný rozsah napětí je zakódován do 18-ti bitů datové části rámce ARINC429.

Definice jednotlivých polí rámce ARINC429 prosběrnici **ARINC429/A**						
Label	**SDI**		**Data**		**SSM**	**Popis, význam**
	hodnota	význam	hodnota	význam		
	měřící kanál A/D převodníku		***rozsah měření na kanálu A/D převodníku***			
001	**0** **1** **2** **3**	kanál 1 kanál 2 kanál 3 kanál 4	**0** **1** **2** **3** **4** **5** **6** **7** **8** **9**	OFF *150 mV* *300 mV* *600 mV* *1.2 V* *2.4 V* *4.8 V* *9.6V* *19.2 V* *38.4 V*	**0**	Nastavení parametrů pro jednotlivé kanály A/D převodníku digitalizačního HATu

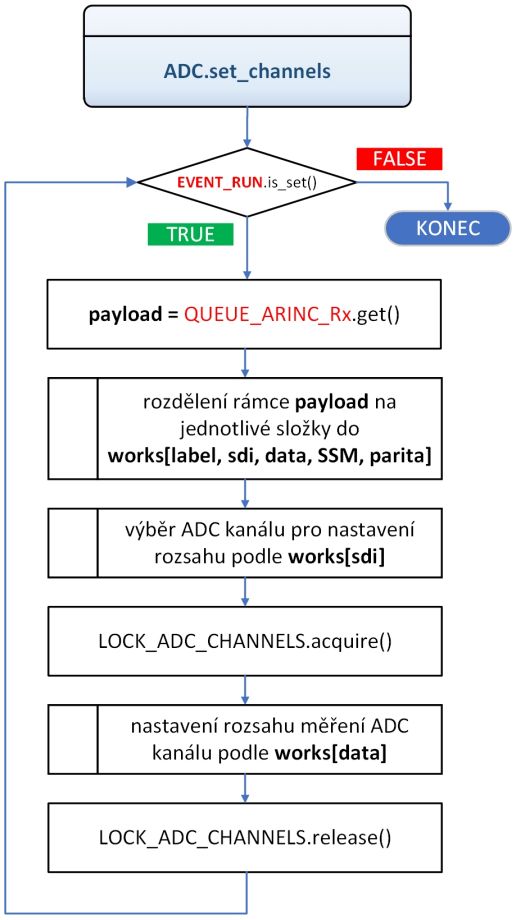
Tabulka 14: Specifikace rámce pro nastavení rozsahů A/D převodníku

Proces pro nastavování požadovaného rozsahu pro A/D měření vyčítá data z FIFO fronty QUEUE_ARINC_RX. Vyčtená data mají podobu ARINC429 rámce. Po vyčtení rámce z FIFO fronty je dekódován label přijatého rámce. Pokud má label hodnotu 001, dojde k dekódování zbytku ARINC429 rámce (viz tabulka 14).

Hodnoty rozsahu kanálů A/D převodníku jsou ukládány do datové struktury ”list” s názvem ADC_ranges. Tento list obsahuje čtyři hodnoty kanálů, které jsou řazeny jako [kanál 1, kanál 2, kanál 3, kanál 4]. Hodnoty, které odpovídají jednotlivým rozsahům, jsou uvedené v tabulce 14.

Jelikož je list ADC_ranges používán i v procesu ”measurement” je nutné zajistit, aby k tomuto listu v jeden okamžik přistupoval pouze jeden proces. V opačném případě by mohlo docházet k přepisování dat uložených v listu během doby, kdy jsou data využívána jiným procesem. Aby tato situace nenastala, používám funkcionalitu knihovny ”threading” s názvem ”Lock”.

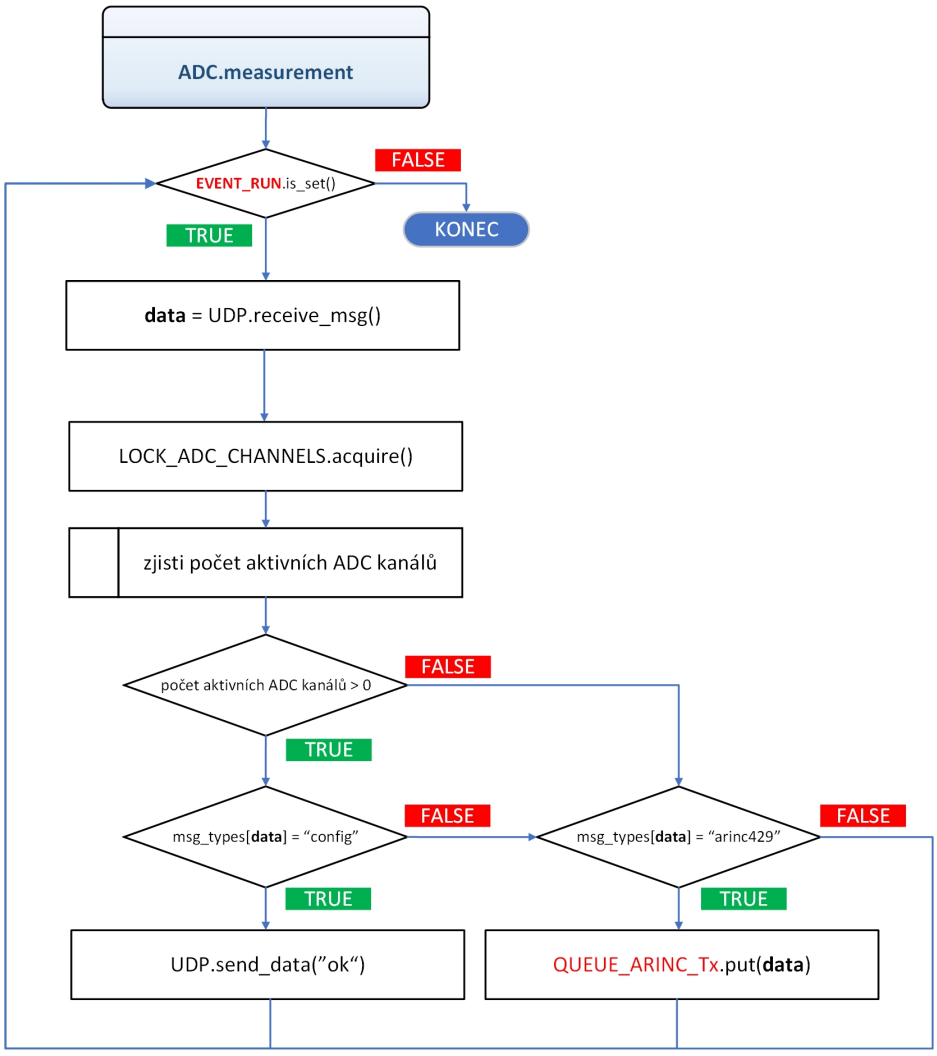
Jedná se o objekt, který je sdílený mezi procesy a může nabývat hodnot zamknuto/odemknuto. Před nastavením hodnot ADC_ranges se v procesu volá metoda ”Lock.acquire()”. Tato metoda mění hodnotu ”Lock” z hodnoty odemknuto na hodnotu zamknuto. V případě, že je ”Lock” využíván jiným procesem, metoda ”acquire” pozastaví vykonávání procesu, dokud nedojde k jeho uvolnění (hodnota ”Lock” bude odemknuto / metoda ”Lock.release()”). Po obdržení objektu ”Lock” dojde k zápisu dat do listu ARINC_ranges a uvolnění objektu ”Lock”.



Obrázek 27: Vývojový diagram procesu set_channels pro A/D převodník

measurement: Proces pro měření dat kontroluje nastavené hodnoty rozsahů v listu ADC_ranges. Pokud je v tomto listu rozsah alespoň jednoho A/D kanálu nastaven na nenulovou hodnotu dojde k odeslání socketu s informacemi o požadovaném měření do digitalizačního HATu a k vyčtení ADC hodnot nastavených kanálů.

Vyčtené hodnoty jsou následně podle nastavení programu zakódovány do CAN a ARINC429 rámců a uloženy do příslušných FIFO front.



Obrázek 28: Vývojový diagram procesu measurement pro převodník A/D

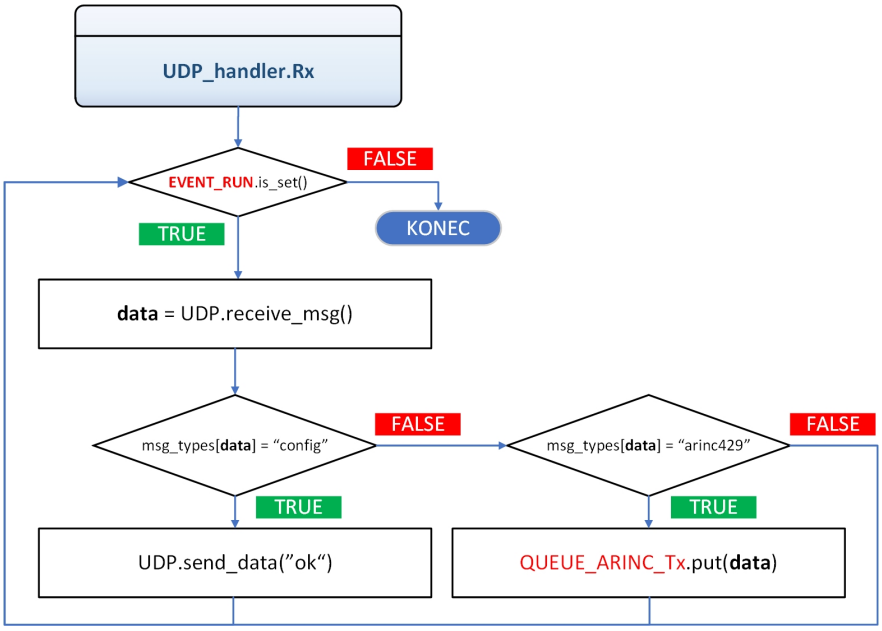
3.5.4 UDP_handler - *třída pro komunikaci s řídicím SW v PC*

ADC_handler - třída pro komunikaci s řídicím SW v PC		
Globální proměnné	QUEUE_UDP_RX QUEUE_UDP_TX EVENT_RUN	*- FIFO fronta, do které jsou ukládány přijaté zprávy* *- FIFO fronta, do které jsou ukládány zprávy k odeslání* *- sdílená proměnná pro start/konec vykonávání procesů*
Metody třídy	RX TX	*- příjem zpráv z grafické aplikace PC* *- odesílání zpráv do grafické aplikace PC*
Vstupní proměnné	-	
Použité knihovny	UDP.py	

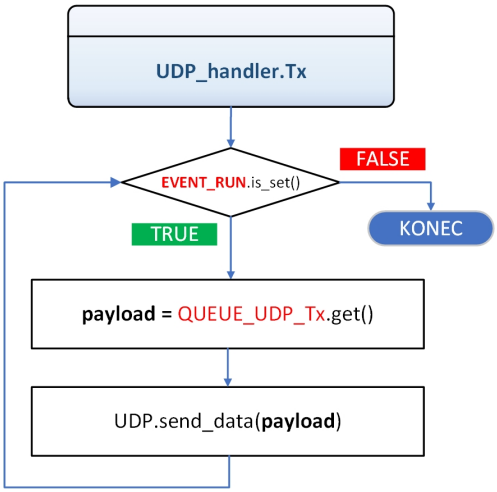
Tabulka 15: Parametry třídy UDP_handler

Procesy RX, TX: Princip procesů pro příjem a odesílání zpráv z/do grafické aplikace je téměř totožný s procesy pro příjem a odesílání ARINC429 rámce. Jediná změna je, že přijaté UDP zprávy mohou být dvojího typu tzv. konfigurační zpráva a zpráva ARINC429.

Konfigurační zpráva slouží k ověření úspěšného spojení grafické aplikace s programem avionické sítě v jednotce A, resp. ŘÍDICÍ. Po jejím obdržení RX proces odešle zpět do grafické aplikace potvrzující ”OK” zprávu. V případě přijetí ARINC429 dat dojde k jejich uložení do FIFO fronty QUEUE_ARINC_TX, odkud jsou data zpracována výše popsanými procesy.



Obrázek 29: Vývojový diagram procesu RX pro obsluhu UDP



Obrázek 30: Vývojový diagram procesu TX pro obsluhu UDP

3.6 SW pro PC (GUI_v.1) - avionická síť se třemi jednotkami

Při návrhu této verze GUI jsem se rozhodl pro vytvoření tří záložek (**Jednotka A**, **Jednotka B**, **Jednotka C**), kde každá záložka odpovídá dané jednotce v avionické síti. Na záložce **Jednotka A** je umožněno ovládání avionické sběrnice (zahájení/ukončení měření) a jsou zde zobrazována naměřená a přenesená data ze sběrnic ARINC429/B a ARINC429/C. Na záložkách **Jednotka B** a **Jednotka C** je možno informativně zobrazit zpracovávaná data danou jednotkou získaná z propojení daných jednotek do sítě LAN protokolem UDP (viz kap. 3.2.1). Pro vytvoření záložek jsem použil objekt "notebook" z knihovny "Tkinter".

Záložka	Funkcionalita
Jednotka A	Ovládání komunikace na avionické síti. Zobrazování přijatých dat z avionické sítě
Jednotka B	Zobrazování dat, které jsou použité v jednotce B
Jednotka C	Zobrazování měřených hodnot z digitalizačního HATu,

Tabulka 16: Popis záložek grafické aplikace pro avionickou síť se třemi jednotkami

3.6.1 Ovládání aplikace

Jak již bylo uvedeno v kapitole 3.4 je avionická síť řízena zprávami, které jsou posílané jednotkou A po sběrnici ARINC429/A. Ty umožňují adresování a řízení připojených avionických jednotek. Pro jednotku B je v současné verzi implementováno potvrzení přijetí zprávy zaslané z jednotky A (toto potvrzení je vypsáno do terminálu jednotky B). Dále je provedena příprava, která umožňuje, po připojení digitalizačního HATu k jednotce B, ovládat měření na tomto HATu. Pro jednotku C je implementována možnost zahájení/ukončení měření na digitalizačním HATu. (definice zpráv, které jsou použité pro popsanou funkcionalitu jsou uvedené v tabulce 17).

Definice jednotlivých polí rámce ARINC429 pro sběrnici **ARINC429/A**

Label	SDI		Data		SSM	Popis, význam
	hodnota	význam	hodnota	význam		
	měřicí kanál A/D převodníku		***rozsah měření na kanálu A/D převodníku***			
001	**0** **1** **2** **3**	kanál 1 kanál 2 kanál 3 kanál 4	**0** **1** **2** **3** **4** **5** **6** **7** **8** **9**	OFF *150 mV* *300 mV* *600 mV* *1.2 V* *2.4 V* *4.8 V* *9.6V* *19.2 V* *38.4 V*	**0**	Nastavení parametrů pro jednotlivé kanály A/D převodníku digitalizačního HATu
	měřicí kanál A/D převodníku					
002	**0**	konstanta	0	konstanta	**0**	Požadavek na potvrzující zprávu od jednotky A v jednotce B

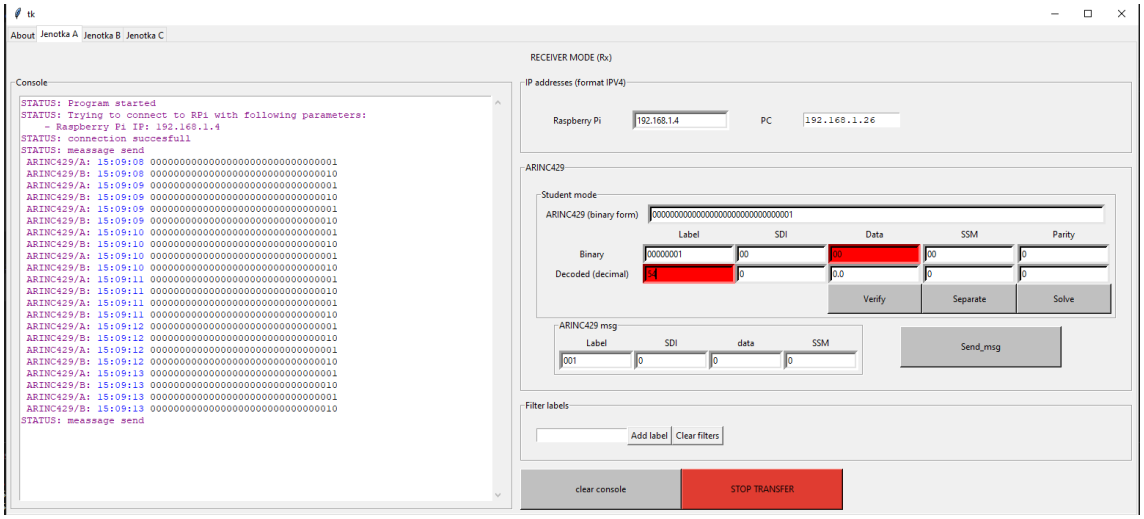
Tabulka 17: Návrh datových rámců pro sběrnici ARINC429/A - síť se třemi avionickými jednotkami

ARINC429 zprávy, kterými jednotka A řídí provoz avionické sítě, jsou vytvářeny v PC aplikaci na záložce ”Jednotka A” v části ”Arinc429 msg” (viz obrázek 31). Po stisknutí tlačítka ”Send_msg” (ve stejné části) dojde k zaslání UDP zprávy z PC do jednotky A. Zaslaná UDP zpráva obsahuje binární řetězec vytvořené ARINC429 zprávy.

Před tím, než je možné zahájit měření na digitalizačním HATu jednotky C, je nutné spustit proces grafické aplikace pro vyčítání a zobrazování UDP zpráv a z jednotky A. Tento proces se spouští stisknutím tlačítka ”START_TRANSFER”na záložce ”Jednotka_A”. Po jeho stisknutí dojde ke změně textu tlačítka na ”STOP_TRANSFER” (opakované stisknutí tohoto tlačítka ukončí měření na digitalizačním HATu jednotky C a ukončí proces pro čtení UDP zpráv z jednotky A).

Po zahájení měření na digitalizačním HATu jednotky C dojde ke spuštění komunikace v avionické síti (viz kapitola 3.4). Pro možnost debuggování a kontroly této komunikace bylo přidáno posílání těchto zpráv z jednotlivých jednotek na příslušné záložky PC aplikace. Definice a popis těchto zpráv je uveden v následující kapitole 3.6.2.

Kromě řízení komunikace po avionické síti byla do PC aplikace implementována část pro dekódování přijatých ARINC429 zpráv z jednotky A. Tato funkcionalita se nachází na záložce ”Jednotka A” v části ”Student mode” (viz obrázek 31). Jedná se o widget, který umožňuje rozdělení ARINC429 rámce na jednotlivé části, jeho dekódování, nebo ověření správnosti zadaného řešení.



Obrázek 31: PC aplikace - grafická podoba záložky ”Jednotka A”

3.6.2 Typy hlášek v PC aplikaci

PC aplikace (dále značeno jako GUI_v.1) umožňuje vypisování hlášek, které informují o průběhu komunikace v avionické síti. Tyto hlášky lze rozdělit na statusové, chybové a datové (tabulka 18 uvádí jednotlivé typy hlášek a jejich popis). Statusové a chybové hlášky jsou vytvářené v rámci běhu PC aplikace a informují uživatele o stavu v avionické síti.

Datové hlášky jsou vytvářeny avionickými jednotkami a posílány jako UDP zprávy do PC aplikace. V případě jednotek A, B se v hláškách přenáší binární reprezentace CAN/ARINC429 rámce a systémový čas jednotky, při kterém byl tento rámec přijat. V případě jednotky C se místo binární reprezentace datových rámců přenáší seznam naměřených hodnot z digitalizačního HATu.

Aby bylo možné tyto údaje poslat jako UDP zprávu, je použita knihovna ”pickle” [31] pro serializaci datového typu ”list”. Z těchto dat se následně vytvoří datový typ ”bytes”, který je použit jako argument funkce ”send” (funkce použitá pro posílání UDP zpráv) knihovny socket. [29]

Záložka	Typ hlášky	Složení hlášek	Popis hlášky
společné	status	definované oznamovací hlášky GUI (fialová barva)	provozní informace grafické aplikace
	chybová	definované chybové hlášky GUI (červená barva)	chybové hlášky grafické aplikace
Jednotka A	datová	- sběrnice avionické sítě (fialová barva) - čas přijetí ARINC429 rámce jednotkou A (modrá barva) - ARINC429 rámec (černá barva)	přijaté ARINC429 rámce z jednotky A
Jednotka B	datová	- název zprávy (černá barva) - čas přijetí CAN rámce jednotkou B (modrá barva) - CAN rámec (černá barva)	přijaté CAN rámce v jednotce B
Jednotka C	datová	- název hlášky (černá barva) - čas přijetí měřených dat jednotkou C (modrá barva) - list měřených hodnot na A/D převodníku (černá barva)	přijaté hodnoty měření napětí na A/D převodníku digitalizačního HATu

Tabulka 18: Typy hlášek v PC aplikaci pro ovládání avionické sítě třemi jednotkami

Záložka Jednotka A - datová hláška Grafická aplikace z jednotky A přijímá UDP zprávy, které obsahují rámce ze sběrnice ARINC429/C (definice rámce viz tabulka 19) a timestamp (systémový čas přijetí rámce v jednotce A). Přijaté UDP zprávy obsahují řetězec serializovaných dat s kódováním UTF-8. Před zobrazením dat v grafické aplikaci je potřeba tato data dekódovat a vytvořit ze serializovaných dat list, který obsahuje timestamp a přijatý ARINC429 rámec. Dekódování přijatých dat a vytvoření listu je implementováno ve vytvořené knihovně UDP.py, vytvoření ARINC429 rámce z přijaté UDP zprávy je implementováno v knihovně ARINC.py.

Definice jednotlivých polí rámce ARINC429 pro sběrnici **ARINC429/C**						
Label	**SDI**		**Data**		**SSM**	**Popis, význam**
	hodnota	význam	hodnota	význam		
	měřící kanál A/D převodníku		***měřená data na kanálu A/D převodníku***			
002	**0** **1** **2** **3**	kanál 1 kanál 2 kanál 3 kanál 4	19 bit / BNR	Změřená data daného A/D kanálu v kódování BNR s rozlišením 0,001	**0**	Změřená data posílaná na sběrnici ARINC429/C avionickou jednotkou C

Tabulka 19: Návrh datových rámců pro sběrnici ARINC429/C - síť se třemi avionickými jednotkami

Záložka Jednotka B - datová hláška Grafická aplikace z jednotky B přijímá UDP zprávy, které obsahují rámec CAN (vytvořený z rámce ze sběrnice ARINC429/B - definice viz tabulka 20) a timestamp (systémový čas přijetí převáděného ARINC429 rámce v jednotce B). Definice CAN rámce je uvedena v tabulce 21. Přijaté UDP zprávy obsahují řetězec serializovaných dat s kódováním UTF-8. Před zobrazením dat v grafické aplikaci je potřeba tato data dekódovat a vytvořit ze serializovaných dat list, který obsahuje timestamp a CAN rámec. Dekódování přijatých dat a vytvoření listu je implementováno ve vytvořené knihovně UDP.py, vytvoření CAN rámce z přijaté UDP zprávy je implementováno v knihovně canframe.py, jejímž autorem je Ken Tidell. [30]

Definice jednotlivých polí rámce ARINC429 pro sběrnici **ARINC429/B**						
Label	**SDI**		**Data**		**SSM**	**Popis, význam**
	hodnota	význam	hodnota	význam		
	měřící kanál A/D převodníku		***měřená data na kanálu A/D převodníku***			
001	**0** **1** **2** **3**	kanál 1 kanál 2 kanál 3 kanál 4	19 bit / BNR	Změřená data daného A/D kanálu v kódování BNR s rozlišením 0,001	**0**	Změřená data posílaná na sběrnici ARINC429/B avionickou jednotkou B

Tabulka 20: Návrh datových rámců pro sběrnici ARINC429/B - síť se třemi avionickými jednotkami

Definice jednotlivých polí rámce **CAN**				
Identifier		**Data**		**Popis, význam**
hodnota	význam	hodnota	význam	
měřící kanál A/D převodníku		***měřená data na kanálu A/D převodníku***		
0 **1** **2** **3**	kanál 1 kanál 2 kanál 3 kanál 4	19 bit / BNR	Změřená data daného A/D kanálu v kódování BNR s přesností 0,001	Změřená data posílaná na sběrnici CAN avionickou jednotkou C

Tabulka 21: Návrh datových rámců pro sběrnici CAN - síť se třemi avionickými jednotkami

Záložka Jednotka C - datová hláška Grafická aplikace z jednotky C přijímá UDP zprávy, které obsahují list měřených hodnot napětí z digitalizačního HATu. Přijaté UDP zprávy obsahují řetězec serializovaných dat s kódováním UTF-8. Před zobrazením dat v grafické aplikaci je potřeba tato data dekódovat a vytvořit ze serializovaných dat list, který obsahuje čas přijetí měřených hodnot jednotkou C a měřená data. Dekódování přijatých dat a vytvoření listu je implementováno ve vytvořené knihovně UDP.py.

3.7 SW pro PC (GUI_v.2) - avionická síť se dvěma jednotkami

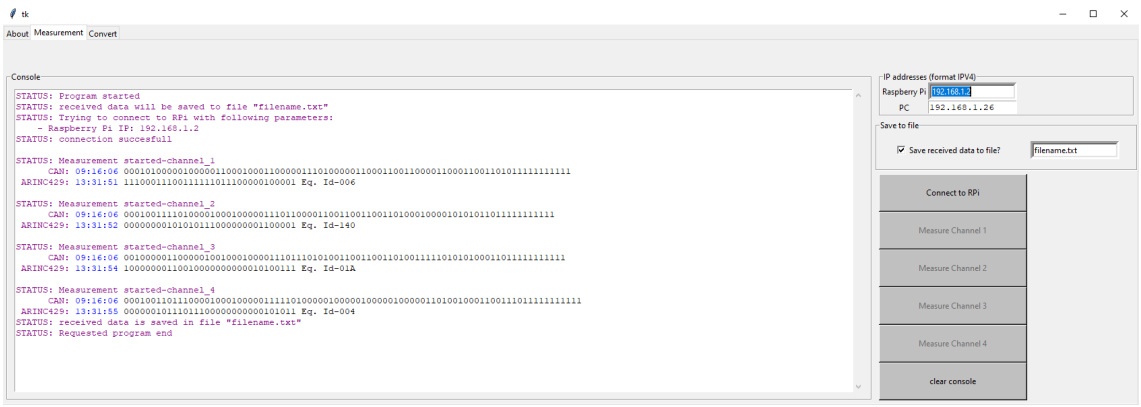
Při návrhu této verze GUI jsem se rozhodl pro vytvoření 2 záložek (**Measurement**, **Convert**). Na záložce **Measurement** je umožněno ovládání avionické sběrnice (zahájení/ukončení měření) a jsou zde zobrazována naměřená a přenesená data ze sběrnic ARINC429/B a CAN. Na záložkách **Convert** je implementována funkcionalita pro dekódování ARINC429 a CAN rámců. Pro vytvoření záložek jsem použil objekt "notebook" z knihovny "Tkinter."

Záložka	Funkcionalita
Measurement	Ovládání komunikace na avionické síti. Zobrazování přijatých dat z měřicí jednotky
Convert	Testování správnosti převodu ARINC429 a CAN rámce

Tabulka 22: Záložky grafické aplikace pro avionickou síť se dvěmi jednotkami

3.7.1 Ovládání aplikace

PC aplikace pro řízení avionické sítě se dvěmi jednotkami (dále značeno jako GUI_v.2) je založena na obdobém principu, jako GUI_v.1 (popsaná v kapitole 3.6). PC aplikace posílá do ŘÍDICÍ jednotky UDP zprávy obsahující ARINC429 rámce pro řízení avionické sítě. V této verzi jsou ovšem tyto ARINC429 rámce vytvářeny PC aplikací na základě stisknutého tlačítka (tlačítka "Measure channel _" na záložce "Measurement" - viz obrázek 32). ARINC429 rámce jsou vytvořené podle definice v tabulce 23, tabulka 24 zobrazuje typy zpráv, které jsou zobrazovány v GUI_v.2.



Obrázek 32: PC aplikace - grafická podoba záložky Measurement

Definice jednotlivých polí rámce ARINC429 prosběrnici **ARINC429/A**						
Label	**SDI**		**Data**		**SSM**	**Popis, význam**
	hodnota	význam	hodnota	význam		
	měřící kanál A/D převodníku		***rozsah měření na kanálu A/D převodníku***			
001	**0** kanál 1 **1** kanál 2 **2** kanál 3 **3** kanál 4		**0** OFF **1** 150 *mV* **2** *300 mV* **3** *600 mV* **4** *1.2 V* **5** *2.4 V* **6** *4.8 V* **7** *9.6V* **8** *19.2 V* **9** *38.4 V*		**0**	Nastavení parametrů pro jednotlivé kanály A/D převodníku digitalizačního HATu

Tabulka 23: Návrh datových rámců pro sběrnici ARINC429/A - síť se dvěmi avionickými jednotkami

Komunikace po avionické síti začíná stisknutím tlačítka ”Connect to RPi”. Po jeho stisknutí dojde ke spuštění procesu GUI_v.2 pro vyčítání a zobrazování UDP zpráv z MĚŘICÍ jednotky. Stisknutí tohoto tlačítka dále aktivuje tlačítka ”Measure channel _”, jejichž stisknutím je možné řídit start/stop měření na digitalizačním HATu MĚŘICÍ jednotky. Přijaté ARINC429 a CAN zprávy (viz tabulka 24) jsou následně zobrazeny v konzoli a záložce **Measurement**.

Kromě řízení komunikace po avionické síti byla do PC aplikace implementována část pro dekódování ARINC429 a CAN zpráv. Tato funkcionalita se nachází na záložce **Convert**. Jedná se o widget, který umožňuje rozdělení ARINC429 a CAN rámce na jednotlivé části, jeho dekódování nebo ověření správnosti zadaného řešení.

3.7.2 Typy hlášek v PC aplikaci

Typ hlášky	Složení hlášky	Popis hláška
status	definované oznamovací hlášky PC aplikace (fialová barva)	provozní informace PC aplikace
chybová	definované chybové hlášky PC aplikace (červená barva)	chybové hlášky PC aplikace
ARINC429	- řetězec ”ARINC429” (fialová barva) - čas přijetí ARINC429 rámce ŘÍDICÍ jednotkou (modrá barva) - ARINC429 rámec (černá barva) - equipment ID přijatého ARINC429 rámce (černá barva)	přijaté ARINC429 rámce v ŘÍDICÍ jednotce
CAN	- řetězec ”CAN” (fialová barva) - čas přijetí CAN rámce ŘÍDICÍ jednotkou (modrá barva) - CAN rámec (černá barva)	přijaté CAN rámce v ŘÍDICÍ jednoce

Tabulka 24: Typy zpráv v PC aplikaci pro ovládání avionické sítě se dvěmi jednotkami

GUI_v.2 umožňuje vypisování hlášek, které informují o průběhu komunikace v avionické síti. Tyto hlášky lze rozdělit na statusové, chybové, ARINC429 a CAN (tabulka 24 uvádí jednotlivé typy hlášek a jejich popis). Statusové a chybové hlášky jsou vytvářeny v rámci běhu PC aplikace

a informují uživatele o stavu v avionické síti.

CAN a ARINC429 hlášky jsou vytvářeny avionickými jednotkami a posílány jako UDP zprávy do PC aplikace. V těchto hláškách se přenáší binární reprezentace CAN/ARINC429 rámce a časový údaj přijetí tohoto rámce ŘÍDICÍ jednotkou. Aby bylo možné tyto údaje poslat jako UDP zprávu, je použita knihovna pickle [31] pro serializaci datového typu ”list”. Z těchto dat se následně vytvoří datový typ ”bytes”, který je použit jako argument funkce ”send” (funkce použitá pro posílání UDP zpráv) knihovny ”socket”. [29]

ARINC429 hláška Grafická aplikace z MĚŘICÍ jednotky přijímá UDP zprávy, které obsahují ARINC429 rámce zaslané z MĚŘICÍ jednotky (definice rámců viz tabulka 25). Přijaté UDP zprávy obsahují řetězec serializovaných dat s kódováním UTF-8. Před zobrazením dat v grafické aplikaci je potřeba tato data dekódovat a vytvořit ze serializovaných dat list, který obsahuje čas přijetí ARINC429 rámce v MĚŘICÍ jednotce a přijatý ARINC429 rámec. Dekódování přijatých dat a vytvoření listu je implementováno ve vytvořené knihovně UDP.py, vytvoření ARINC429 rámce z přijaté UDP zprávy je implementováno v knihovně ARINC.py.

Definice jednotlivých polí rámce ARINC429 pro sběrnici **ARINC429/B**						
Label	**SDI**		**Data**		**SSM**	**Popis, význam**
	hodnota	význam	hodnota	význam		
	měřící kanál A/D převodníku		***naměřená data na kanálu A/D převodníku***			
204	**0**	kanál 1	17 bit / BNR	Avionická data daného A/D kanálu s rozlišením *1,0*	**0**	Měření **Baro corrected altitude #1** na kanálu 1 A/D převodníku
206	**1**	kanál 2	14 bit / BNR	Avionická data daného A/D kanálu s rozlišením *0,0625*	**0**	Měření **Computed Airspeed** na kanálu 2 A/D převodníku
345	**2**	kanál 3	12 bit / BNR	Avionická data daného A/D kanálu s rozlišením *0,5*	**0**	Měření **Exhaust Gas Temperature** na kanálu 3 A/D převodníku
324	**3**	kanál 4	14 bit / BNR	Avionická data daného A/D kanálu s rozlišením *0,01*	**0**	Měření **Pitch Angle** na kanálu 4 A/D převodníku

Tabulka 25: Návrh datových rámců pro sběrnici CAN - síť se dvěmi avionickými jednotkami

CAN hláška Grafická aplikace z MĚŘICÍ jednotky přijímá UDP zprávy, které obsahují CAN rámce rámce zaslané z MĚŘICÍ jednotky (definice rámců viz tabulka 26). Přijaté UDP zprávy obsahují řetězec serializovaných dat s kódováním UTF-8. Před zobrazením dat v grafické aplikaci je potřeba tato data dekódovat a vytvořit ze serializovaných dat list, který obsahuje čas přijetí CAN rámce v MĚŘICÍ jednotce a přijatý CAN rámec. Dekódování přijatých dat a vytvoření listu je implementováno ve vytvořené knihovně UDP.py, vytvoření CAN rámce z přijaté UDP zprávy je implementováno v knihovně canframe.py, jejímž autorem je Ken Tidell. [30]

Definice jednotlivých polí rámce **CAN**				
Identifier		**Data**		**Popis, význam**
hodnota	význam	hodnota	význam	
		měřená data na kanálu A/D převodníku		
320		32 bit / IEEE754	Avionická data daného A/D kanálu s přesností 0,1	Měření **Baro corrected altitude #1** na kanálu 1 A/D převodníku
317		32 bit / IEEE754	Avionická data daného A/D kanálu s přesností 0,1	Měření **Computed Airspeed** na kanálu 2 A/D převodníku
520		32 bit / IEEE754	Avionická data daného A/D kanálu s přesností 0,1	Měření **Exhaust Gas Temperature** na kanálu 3 A/D převodníku
311		32 bit / IEEE754	Avionická data daného A/D kanálu s přesností 0,1	Měření **Pitch Angle** na kanálu 4 A/D převodníku

Tabulka 26: Návrh datových rámců pro sběrnici CAN - síť se dvěmi avionickými jednotkami

Kapitola 4
Závěr

Tato diplomová práce se zabývá návrhem a realizací avionické sítě se sběrnicemi ARINC429 a CAN. Avionická síť je realizována za použití minipočítače Raspberry Pi 3 B+, ke kterému jsou připojené zařízení (zvaná HAT - Hardware Atachement on Top) rozšiřující jeho funkcionalitu o sběrnice CAN, ARINC429 a A/D převodník.

V teoretické části práce je uveden popis použitých sběrnic. Jsou zde rozebrány způsoby kódování binární reprezentace rámců jednotlivých sběrnic. Na základě těchto kódování je v teoretické části popsán princip dekódování CAN a ARINC429 rámců. Dále je zde vysvětlen princip použití rozšiřujících modulů HAT a jejich konfigurace v Raspberry Pi.

V praktické části práce byla navržena a realizována avionická síť sestávající se ze třech avionických jednotek (minipočítač Raspberry Pi 3 B+ s připojenými HATy) a PC aplikace. Tato síť obsahuje jednu řídící jednotku, ke které jsou sběrnicí ARINC429 připojené dvě podřízené jednotky. Součástí této konfigurace avionické sítě je také návrh ARINC429 rámců, kterými lze řídit komunikaci v avionické síti. Navržená avionická síť umožňuje měření analogových hodnot napětí a jejich převod do digitální podoby. Dále umožňuje přenos digitalizovaných dat z podřízených jednotek do řídicí jednotky. Pro tento přenos je možné použít sběrnice CAN a ARINC429. V avionické síti je také navržen a implementován způsob převodu CAN rámce na rámec sběrnice ARINC429. Vyvinutá PC aplikace je spustitelná pod operačním systémem Windows a umožňuje nastavování řídicích rámců v řídicí jednotce. Dále umožňuje monitorování stavu podřízených jednotek avionické sítě.

V rámci konzultací s vedoucím mé diplomové práce bylo zadání práce rozšířeno o návrh a realizaci další avionické sítě pro účely použití ve výuce. Tato síť sestává z PC aplikace a jedné řídicí avionické jednotky, ke které je připojena jedna podřízená (MĚŘICÍ) jednotka umožňující měření analogových signálů (pro měření byly definovány čtyři očekávané avionické veličiny) a jejich převod do digitální podoby. Definované avionické veličiny jsou ”Baro corrected altitude #1”, ”Computed Airspeed”, ”Exhaust Gas Temperature”, ”Pitch Angle”.

Kromě řízení komunikace v realizovaných avionických sítích je v PC aplikaci implementována funckionalita pro testování správného dekódování CAN a ARINC429 rámců. Ta umožňuje zadat libovolný CAN/ARINC429 rámec a předpokládané dekódované řešení. Poté aplikace vyhodnotí správnost zadaného řešení.

Při návrhu a realizaci SW části avionické sítě bylo dbáno na možnost jednoduché modifikace, modularitu a zpracování časově kritických úloh. Úlohy v avionické síti byly z pohledu jejich funkcionality rozřazeny do skupin, kde pro každou takovouto skupinu byly vytvořeny procesy zajišťující správné vykonávání činností v korektní časové posloupnosti.

Tento přístup realizace avionických sítí umožňuje jednoduché přidávání nových jednotek do avionické sítě, a proto umožňuje snadnou změnu funkcionality podle budoucích požadavků.

Součástí diplomové práce byla i fyzická realizace jednoho HATu, který rozšiřuje funckionalitu Raspberry Pi o sběrnice ARINC429 a CAN. Tento HAT byl vytvořen na základě dříve vyvinuté a navržené desky plošných spojů, která byla navržena v rámci činnosti výzkumného týmu NavLIS.

Literatura

[1] Mark 33 Digital Information Transfer System (DITS) Part 1, Functional Description, Electrical Interface, Label Assignments and Word Formats, ARINC SPECIFICATION 429 PART 1-17, 2014, Aeronautical Radio, Inc.

[2] Bosch, C. A. N. (1991). Specification version 2.0. Published by Robert Bosch GmbH (Září 1991)

[3] Raspberry pi documentation (2022) Raspberry Pi hardware. Dostupné z: https://www.raspberrypi.com/documentation/computers/raspberry-pi.html (cit. Leden 2, 2023).

[4] Corbet, J., Rubini, A. and Kroah-Hartman, G. (2010) Linux device drivers. Beijing etc., China: O'Reilly.

[5] Python Software Foundation (2022) Socket - Low-level networking interface. Dostupné z: https://docs.python.org/3/library/socket.html (cit. Prosinec 28, 2022).

[6] Smith, M. et al. (2021) Introducing raspberry pi hats, Raspberry Pi. Dostupné z: https://www.raspberrypi.com/news/introducing-raspberry-pi-hats/ (cit. Prosinec 28, 2022).

[7] Eidsness, C. (2022) Ccxtechnologies/driver-avionics: ARINC-429 driver for linux, GitHub. Dostupné z: https://github.com/ccxtechnologies/driver-avionics (cit. Prosinec 28, 2022).

[8] HOLT Integrated circuits Inc. (2023) "3.3V ARINC 429 Dual Receiver, Single Transmitter with SPI Interface,"HI-3593 datasheet, (Leden 6, 2023)

[9] Microchip Technology Inc. (2018) Stand-Alone CAN Controller with SPI Interface, MCP2515 datasheet, (Srpen 15, 2018)

[10] Texas Instruments Inc. (2015) ADS868x 16-Bit, 500-kSPS, 4- and 8-Channel, Single-Supply, SAR ADCs with Bipolar Input Ranges, ADS8684, ADS8688 datasheet, (Duben 21, 2015)

[11] Texas Instruments Inc. (2009) PGA280 Zerø-Drift, High-Voltage, Programmable Gain Instrumentation Amplifier, PGA280, PGA280 datasheet, (Červen 2009)

[12] Gossner, K. (2013) Add a digitally controlled PGA with noise filter to an ADC, Texas Instruments Incorporated - Data converters [Preprint].

[13] ARINC 429 Tutorial - ARINC 429 Specification Overview - AIM Online. Home Page of AIM Online - Professional Avionics Databus Solutions. Dostupné z: https://www.aim-online.com/products-overview/tutorials/arinc-429-tutorial/ (cit. Prosinec 12, 2022)

[14] Cary R. Spitzer: Digital Avionics Handbook, Second Edition, Avionics: Development and Implementation, CRC Press, 2007, ISBN: 978-0-8493-8441-7.

[15] ARINC 429 (2022) Wikipedia. Wikimedia Foundation. Dostupné z: https://en.wikipedia.org/wiki/ARINC_429 (cit. Leden 5, 2023).

[16] GILBOA-MARKEVICH, Nimrod a Avishai WOOL. Hardware Fingerprinting for the ARINC 429 Avionic Bus. In: CHEN, Liqun, Ninghui LI, Kaitai LIANG a Steve SCHNEIDER, ed. Computer Security – ESORICS 2020. Cham: Springer International Publishing, 2020, 2020-09-13, s. 42-62. ISBN 978-3-030-59012-3.

[17] Cook, J. A., Freudenberg, J. S. (2007). Controller area network (CAN). EECS, 461, 1-5.

[18] CAN fieldbus — CAN Technology — Ixxat. Ixxat — Data communication for Embedded, Safety and Energy [online]. Copyright © 2022 HMS Networks. Dostupné z: https://www.ixxat.com/technologies/fieldbuses/can (cit. Leden 3, 2023)

[19] Corrigan, S. (2008) Controller Area Network Physical Layer Requirements, Texas Instruments Inc. [Preprint].

[20] SC31, I. S. O. ISO 11898-1: 2015, Road vehicles-Controller area network (CAN)-Part 1: Data link layer and physical signalling."

[21] CSS Electronics (2021) Can bus explained - a simple intro [2022], CSS Electronics. Available at: https://www.csselectronics.com/pages/can-bus-simple-intro-tutorial (Accessed: January 5, 2023).

[22] Šipoš, M. (2020/2021) Úvod, přenos dat, ARINC 429, ARINC 629, MIL-STD-1553, CSDB, ASDB, CAN, CANaerospace [PowerPoint] České Vysoké Učení Technické v Praze, Dostupné z: https://moodle.fel.cvut.cz/pluginfile.php/293925/mod_resource/content/1/INA_Lecture_01.pdf,

[23] Stock, M. (2006) CANaerospace - Interface specification for airborne CAN applications V 1.7.

[24] Szurman, Karel Kastil, Jan Straka, M. Kotásek, Zdeněk. (2013). Fault tolerant CAN bus control system implemented into FPGA. 289-292. 10.1109/DDECS.2013.6549837.

[25] Postel, J. (1980) User datagram protocol. Available at: https://doi.org/10.17487/rfc0768.

[26] Shahid, M. (2022) Binary coded decimal - BCD, Electronics. Dostupné z: https://www.electronics-lab.com/article/binary-coded-decimal-bcd/ (cit. Leden 3, 2023).

[27] Rajaraman, V. (2016) IEEE standard for Floating point numbers, Resonance, 21(1), pp. 11–30. Dostupné z: https://doi.org/10.1007/s12045-016-0292-x.

[28] Divo, F. (2021) python-can 4.1.0 documentation. Dostupné z: https://python-can.readthedocs.io/en/stable/development.html (cit. Prosinec 28, 2022).

[29] Python Software Foundation (datum neuvedeno) Socket - low-level networking interface, Python documentation. Dostupné z: https://docs.python.org/3/library/socket.html (cit. leden 5, 2023).

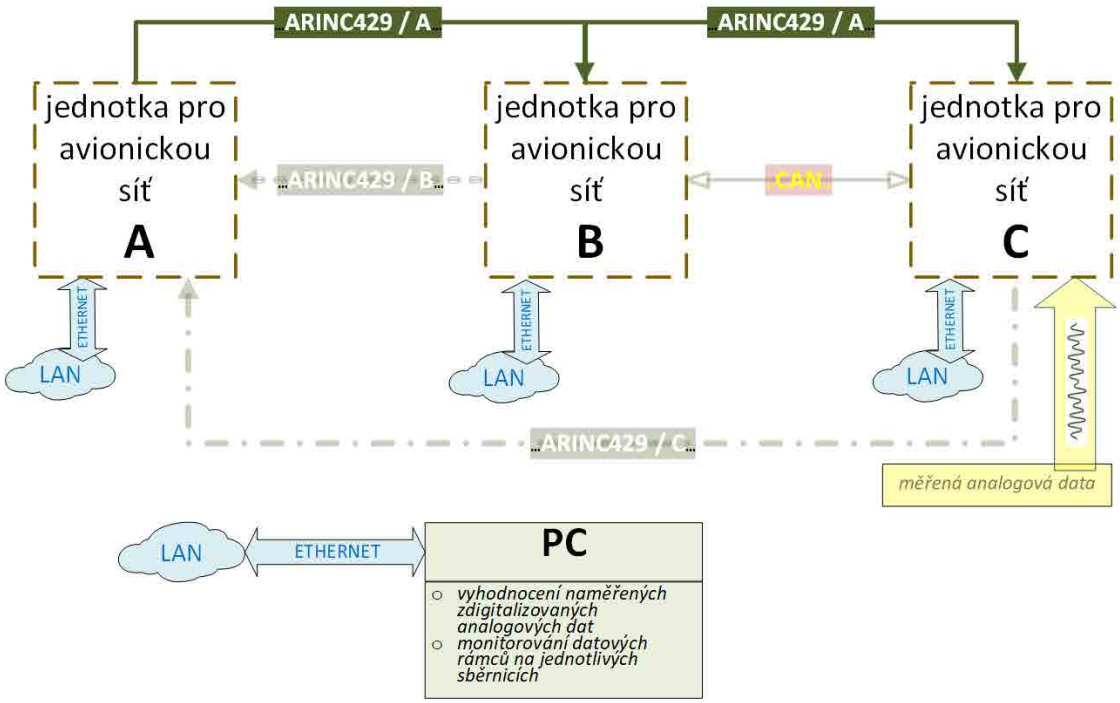
[30] Tindell, K. and Gardiner, B. (2022) Kentindell/canhack: The yes we can project of Canis Labs, GitHub. Dostupné z: https://github.com/kentindell/canhack (cit. Prosinec 28, 2022).

[31] Python Software Foundation (datum neuvedeno) Pickle - python object serialization, Python documentation. Dostupné z: https://docs.python.org/3/library/pickle.html (cit. Leden 5, 2023).

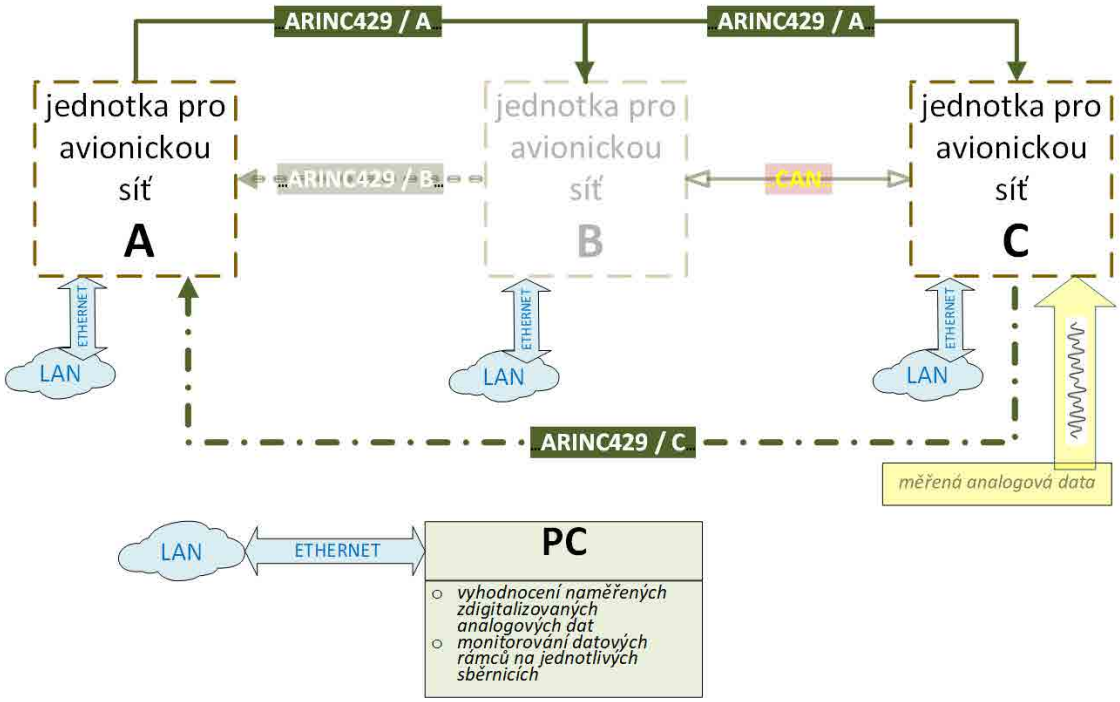
Příloha A

PŘÍLOHA: Scénáře přenosů dat

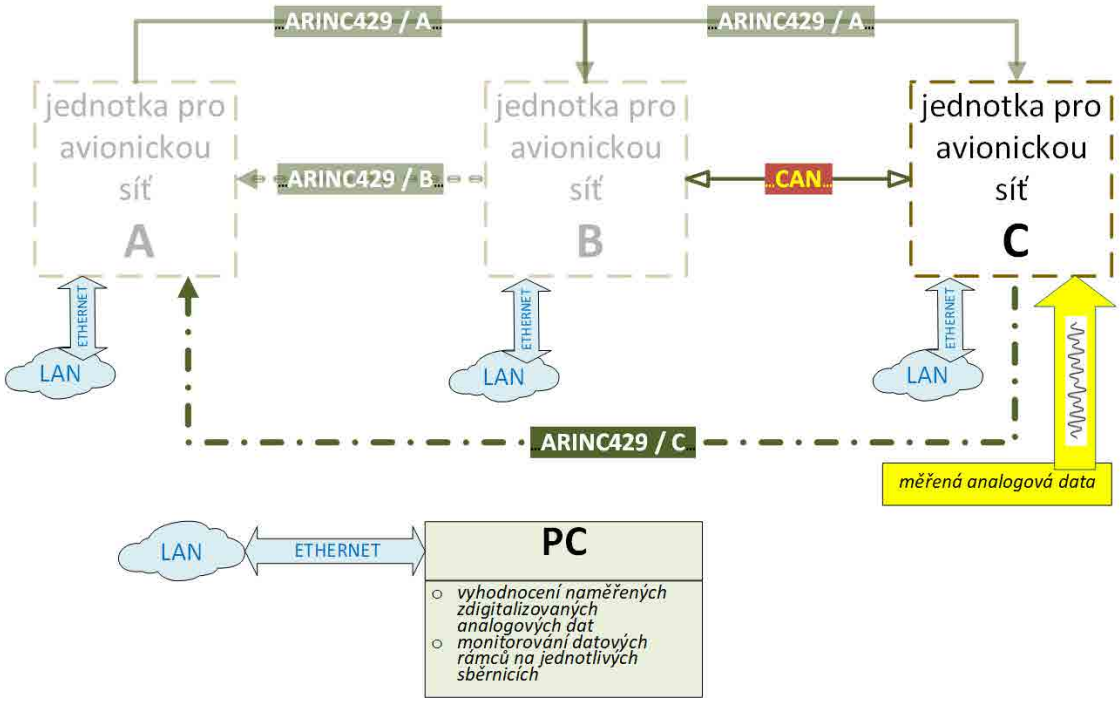
A.1 Navržené zapojení v režimu tří komunikujících avionických jednotek po sběrnici ARINC429/A



A.1 Navržené zapojení v režimu tří komunikujících avionických jednotek po sběrnici ARINC429/A

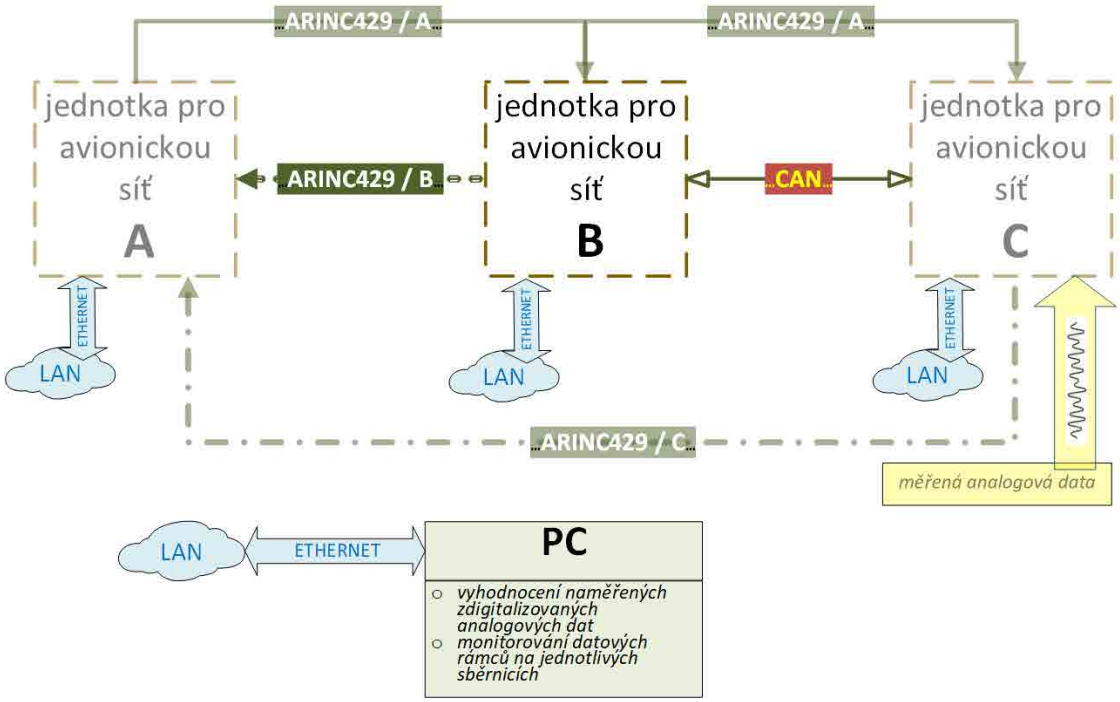


Obrázek 33: Navržené zapojení v režimu obousměrné komunikace avionických jednotek (A, C) po sběrnicích ARINC429 (ARINC429/A, ARINC429/C)



Obrázek 34: Blokové schéma využívající avionickou jednotku C pro měření a digitalizaci dat s jejich následným odesláním po sběrnicích ARINC429/C a CAN

A.1 Navržené zapojení v režimu tří komunikujících avionických jednotek po sběrnici ARINC429/A

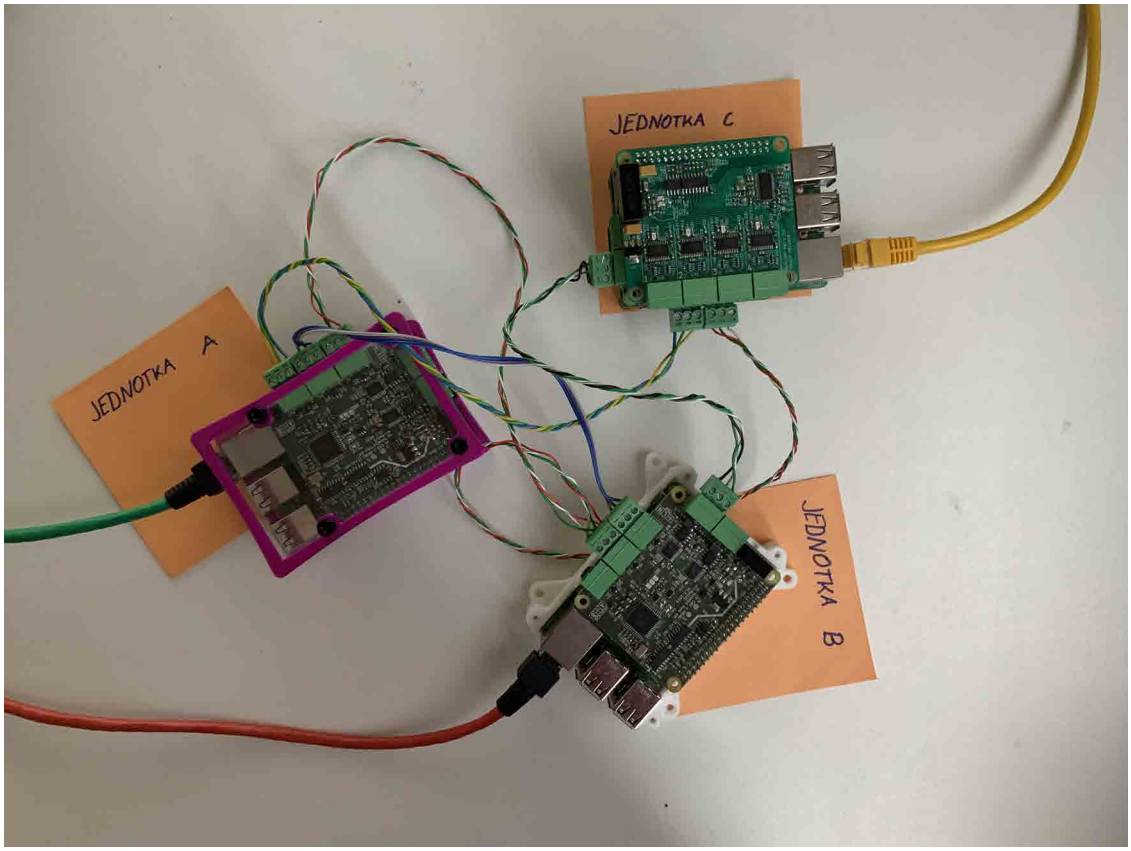


Obrázek 35: Blokové schéma využívající avionickou jednotku B pro převod dat ze sběrnice CAN a jejich následného odeslání sběrnicí ARINC429/B

Příloha B

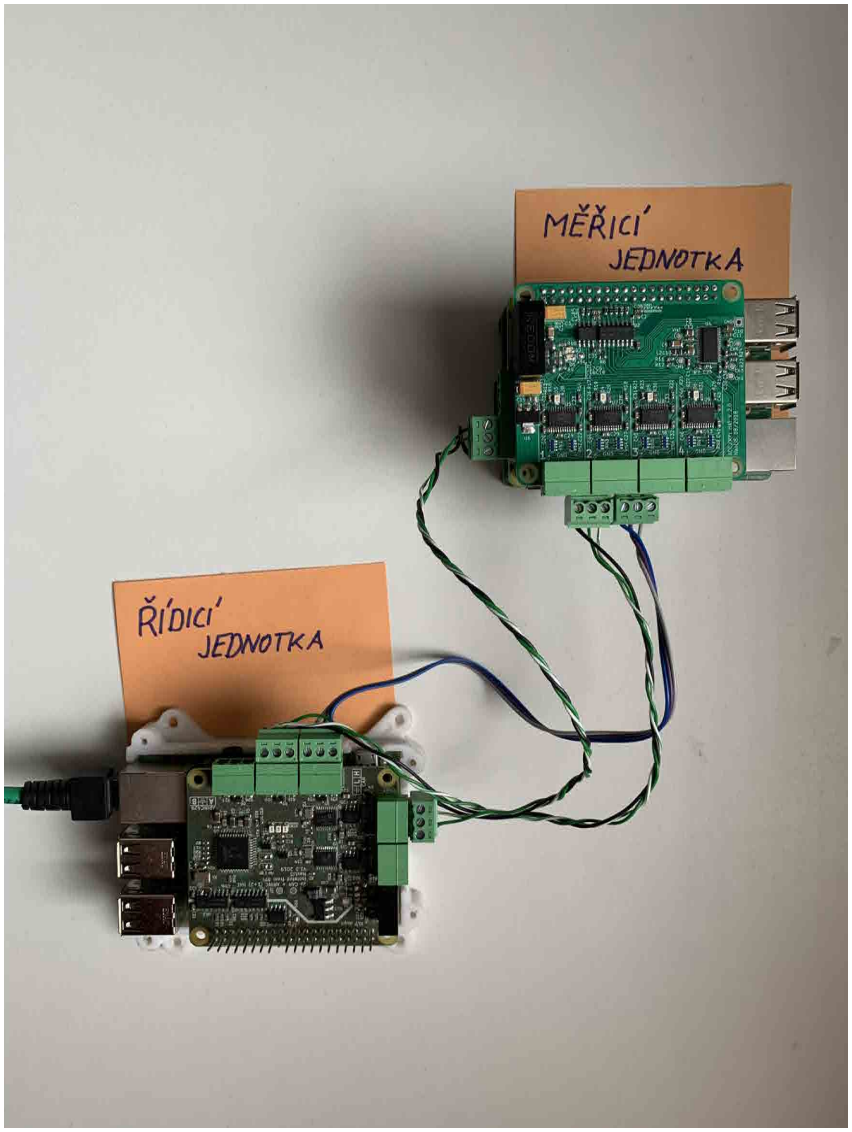
PŘÍLOHA: Realizované zapojení vytvořených avionických sítí

B.1 Avionická síť se třemi jednotkami



Obrázek 36: Uspořádání avionické sítě se třemi avionickými jednotkami

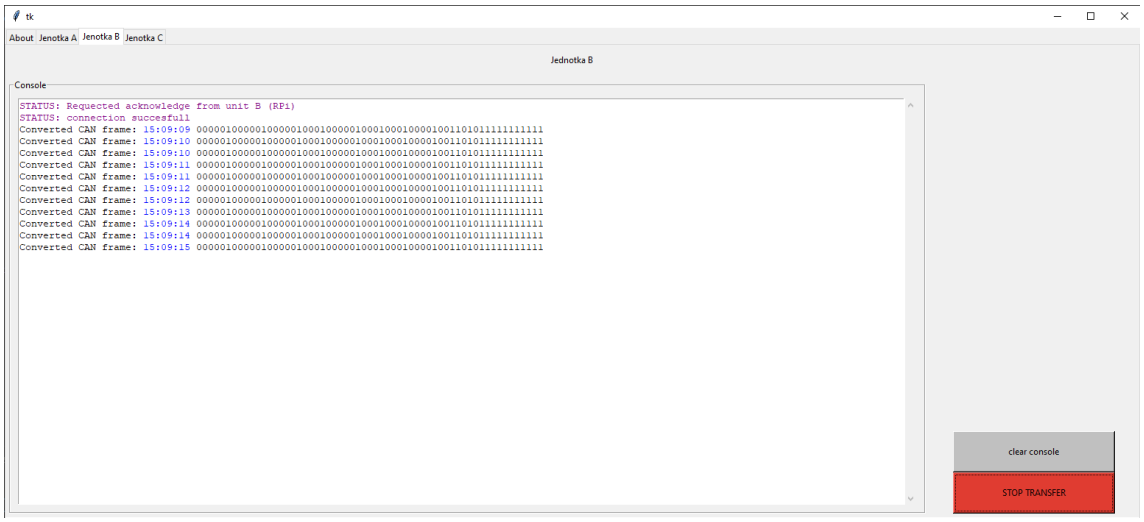
B.2 Avionická síť se dvěmi jednotkami



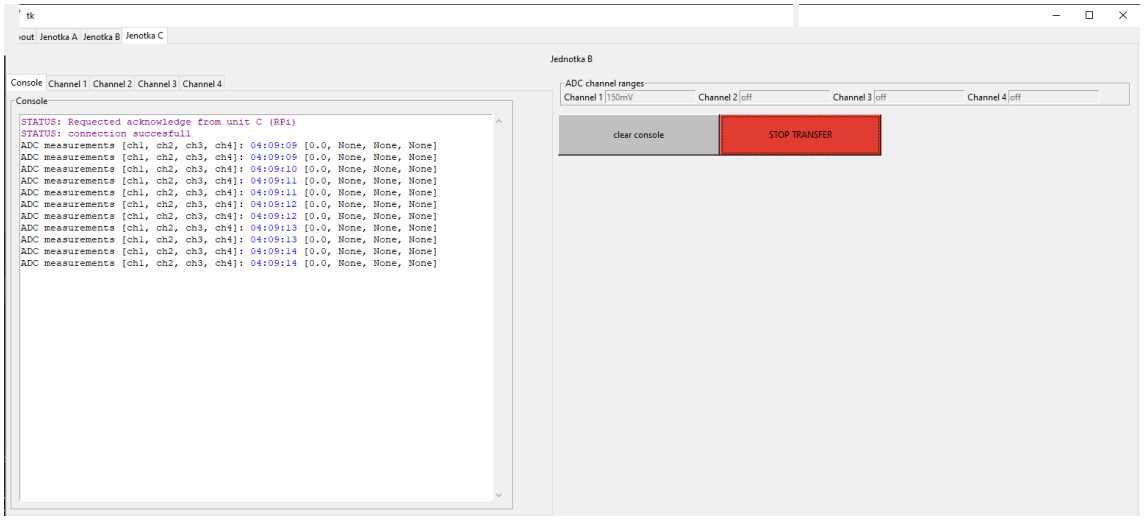
Obrázek 37: Uspořádání avionické sítě se dvěmi avionickými jednotkami

Příloha C
PŘÍLOHA: Grafické výstupy PC aplikace pro avionickou síť

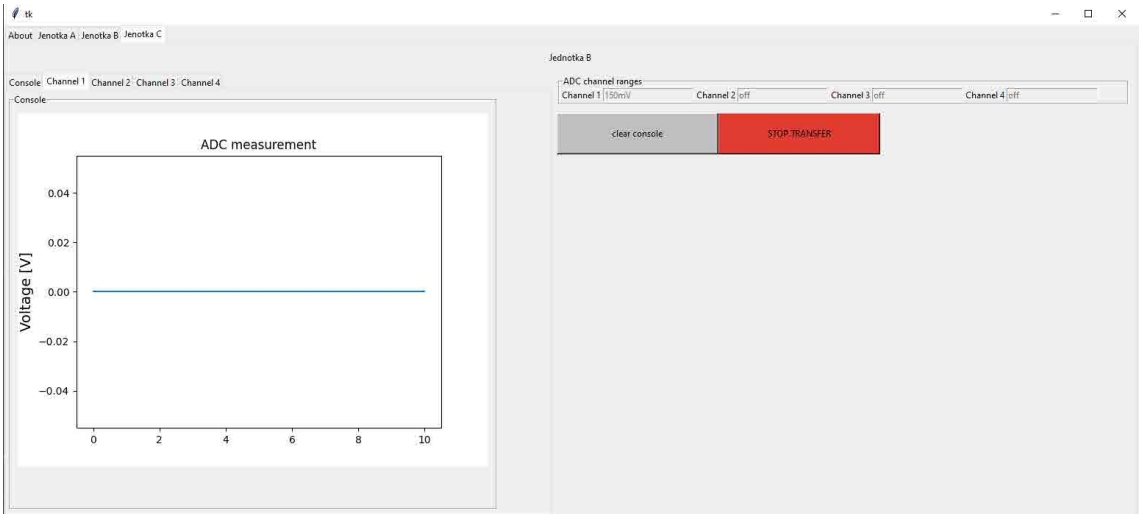
C.1 Avionická síť se třemi jednotkami



Obrázek 38: Záložka B - bitové zobrazení zpracovávaných CAN rámců

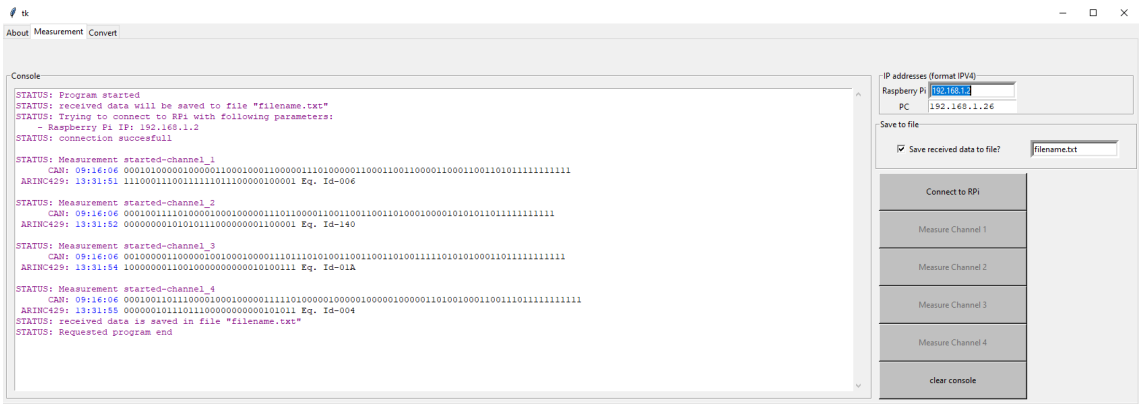


Obrázek 39: Záložka C - naměřené a převedené hodnoty na jednotlivých A/D kanálech - číselné zobrazení



Obrázek 40: Záložka C - naměřené a převedené hodnoty na jednotlivých A/D kanálech - grafické zobrazení

C.2 Avionická síť se dvěmi jednotkami



Obrázek 41: Záložka Measurement

Obrázek 42: Záložka Convert