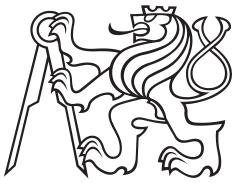
Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra mikroelektroniky

Bezdrátová telemetrie pro elektrickou formuli

Patrik Bachan

Vedoucí: Ing. Vít Záhlava, Csc
Obor: Elektronika a komunikace
Studijní program: Aplikovaná elektronika
Červen 2020

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Bachan** Jméno: **Patrik** Osobní číslo: **420181**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra mikroelektroniky**

Studijní program: **Elektronika a komunikace**

Studijní obor: **Elektronika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Bezdrátová telemetrie pro elektrickou formuli

Název diplomové práce anglicky:

Wireless Telemetry for Electric Formula

Pokyny pro vypracování:

1) Seznamte se s možnostmi bezdrátového přenosu dat mezi pohyblivými body, vyberte vhodnou technologii.
2) Navrhněte a realizujte systém bezdrátového přenosu telemetrických dat v reálném čase pro elektrickou formuli.
3) Změřte vlastnosti spojení a zhodnoťte dosaženévýsledky

Seznam doporučené literatury:

[1] Záhlava, V.: Návrh a konstrukce desek plošných spojů, Principy a pravidla praktického návrhu, BEN - technická literatura, Praha 2011
[2] Formula Student Rules 2020 [online] https://www.formulastudent.de/fsg/rules/
[3] Paul Horowitz and Winfield Hill, The Art of Electronics (Third ed. 2015), Cambridge University Press
[4] D. Tse, P. Viswanath: Fundamentals of Wireless Communications, Cambridge University Press, 2005

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Vít Záhlava, CSc., katedra mikroelektroniky FEL

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

Datum zadání diplomové práce: **20.09.2019** Termín odevzdání diplomové práce: **14.08.2020**

Platnost zadání diplomové práce: **19.02.2021**

Ing. Vít Záhlava, CSc. podpis vedoucí(ho) práce	prof. Ing. Pavel Hazdra, CSc. podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. 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

Poděkování

Tímto bych chtěl poděkovat Ing. Vítovi Záhlavovi, CSc za vedení práce.

Velké poděkování bych chtěl věnovat studentskému týmu elektrické formule eForce za poskytnutí podmínek, zdrojů a zázemí pro vývoj elektroniky a za několik sezón plných zajímavých výzev a adrenalinu. Děkuji všem bývalým i současným členům za podporu. Jmenovitě bych chtěl poděkovat Bc. Janu Sixtovi za poskytnutí vývojových desek a Vojtovi Michalovi za cenné rady v oblastech jazyka C++.

Poděkování též patří Ing. Vítovi Hlinovskému, CSc. a katedře pohonů za zapůjčení spektrálního analyzátoru pří ztížených podmínkách v roce 2020.

Nakonec chci poděkovat rodině, která mi byla oporou po celou dobu studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 10. června 2020

Abstrakt

Tato práce si klade za cíl analyzovat možnosti bezdrátového přenosu dat mezi pohyblivým objektem v podobě vozidla kategorie Formula Student Electric a pevnou stanicí. Na základě této analýzy je vybrána vhodná technologie vyhovujícím požadavkům samotného vozidla a provozním podmínkám týmu.

Práce dále obsahuje výběr stěžejních hardwarových komponent, návrh designu desek plošných spojů a stejně tak zástavbu do vhodné krabičky. V další kapitole práce je rozebrán návrh softwaru pro tento spoj, kde je kladen důraz na univerzálnost a rozšiřitelnost.

V závěru práce jsou změřeny vlastnosti realizace spoje a jsou nastíněny možnosti rozšíření do budoucna.

Klíčová slova: fomula student, formula, eforce, wireless telemetry

Vedoucí: Ing. Vít Záhlava, Csc

Abstract

This work aims to analyze the possibilities of wireless data transmission between a moving object in the form of a vehicle of the Formula Student Electric category and a fixed station. Based on this analysis, a suitable technology is selected to meet the requirements of the vehicle itself and the operating conditions to suit the team.

The work also includes a selection of key hardware components, design of printed circuit boards, as well as installation in a suitable box. The next part of the work discusses the design of software for this connection, where the emphasis is on versatility and extensibility.

At the end of the work, the properties of the connection are measured and the possibilities of expansion into the future are outlined.

Keywords: formula student, formule, eforce, bezdrátová telemetrie

Title translation: Wireless telemetry for electric formula

Obsah

Obrázky

Tabulky

Kapitola 1

Úvod

1.1 Tým

eForce FEE Prague Formula je tým studentů účastnící se celosvětové soutěže Formula Student. Tým byl založen roku 2010 a funguje pod záštitou ČVUT FEL. Je složen převážně ze studentů elektrotechnické a strojní fakulty ČVUT. Vznikají zde specializovaná vozidla s elektrickým pohonem splňující požadavky Formula SAE.

Obrázek 1.1: Tým ze sezóny FSE.08.

Obrázek 1.2: Formule FSE.08 na trati.

1.2 Souťěž

Cílem soutěže Formula student Electric je poskytnou studentům příležitost vyzkoušet si inženýrskou práci na reálném zařízení. Studenti tak získají zkušenosti v mnoha oborech spojených s návrhem a výrobou elektrického vozidla, ale i s vedením týmu, získáváním sponzorů a propagací. Výsledky své každoroční práce v podobě elektrických formulí pak obhajují na mnoha závodech [1] po celém světě, během letních prázdnin. Na těchto závodech týmy získávají body ze statických i dynamických disciplín, které následně určují jejich umístění v celosvětovém žebříčku.

1.3 Aktuální stav

V celé formuli je několik řídících jednotek vyvinutých v týmu, které spolu navzájem komunikují po dvou sběrnicích CAN. Jsou zde dostupné informace o důležitých prvcích auta jako jsou zbývající energie a teploty akumulátoru, stav motorů a dalších důležitých částí. Jsou zde také data z inerciální jednotky, která slouží jako zdroj referenční rychlosti pro kontrolu trakce. Po stejných sběrnicích putují i řídící data od pedálů, přes jednotku dynamiky auta až do měničů. Všechny důležité informace jsou tedy na jednom místě. Pro detailní pochopení chování auta, jeho zlepšování a také analýzu závad jsou nyní všechny zprávy z obou sběrnic CAN logovány komečním CAN loggerem VECTOR GL1010.

Z takto získaných dat je například možné zjistit vhodnost nastavení parametrů kontroly trakce a vektorování momentu. Nyní jsou tato data dostupná až

[1] Oficiálně se nejedná o závody, ale o události inženýrské soutěže

po testovacím jezdění. Rychlost iterace procesu jezdění, analýzy a případných úprav je tedy nízká. Další možností je přístup k datům na sběrnicích v reálném čase ať už pomocí komerčních CAN-Universal Serial Bus (USB) převodníků VECTOR VN1610 1.3 nebo v týmu vyvinutých CAN-USB převodníků Ocarina 1.4. Ačkoliv toto řešení poskytuje detailní informace o sběrnicích jako je přesné časování zpráv a analýzy chyb, lze jich využit pouze pokud není formule v pohybu.

Obrázek 1.3: Převodník vector VN1610. Převzato z [24].

Obrázek 1.4: Převodník Ocarina IV vyvinutý v týmu.

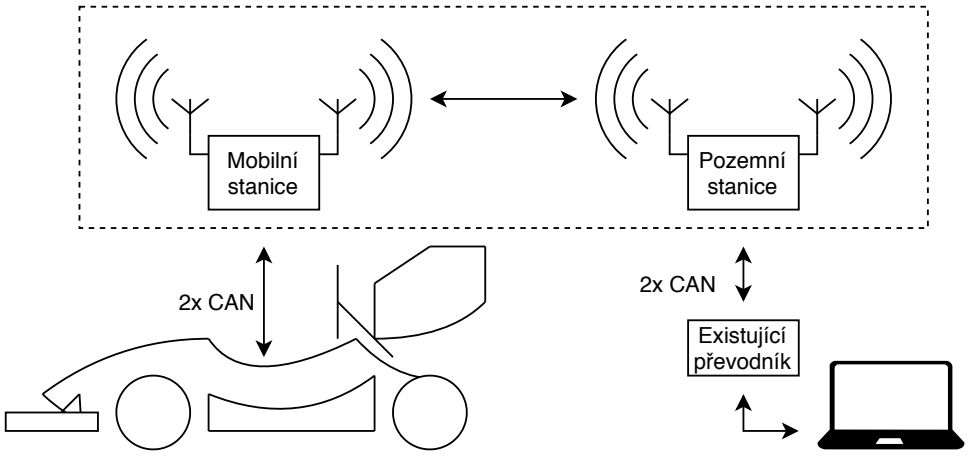
Možností jak překlenout tuto překážku je bezdrátový spoj mezi pozemní stanicí a pohybující se formulí pro přenos dat v reálném čase. [2]

[2]Tato problematika se v týmu řešila již v minulosti [22], ale kvůli svým nedostatkům nebyla použita.

1.4 Požadavky

Byly stanoveny tyto požadavky:

- Obousměrný přenos dat dvou CAN sběrnic. Filtrování přenášených zpráv pro snížení vytížení spoje je možné.
- Dosah spoje alespoň 200m ve volném prostranství.
- napájení z palubního napětí 24V
- Minimální požadavky na změnu nástrojů. Spoj by měl být v mezích možností průhledný pro již existující nástroje. (Bezdrátový spoj v čárkovaném rámečku1.5 se z venku chová jako kabelové spojení.)
- Konfigurace přes CAN nevyžadující nástroje omezené na specifický OS.
- Nezávislé na službě třetích stran jako jsou mobilní operátoři nebo časově omezené licence.
- Univerzální, bez potřeby provádět modifikace pro funkci v jiné generaci formule.
- Diverzita (více)externích antén kvůli jejich rozmístění. Antény nemají přímou viditelnost ve všech orientacích formule.
- Možnost rozšíření o hlasovou komunikaci s pilotem.



Obrázek 1.5: Přenosový řetězec.

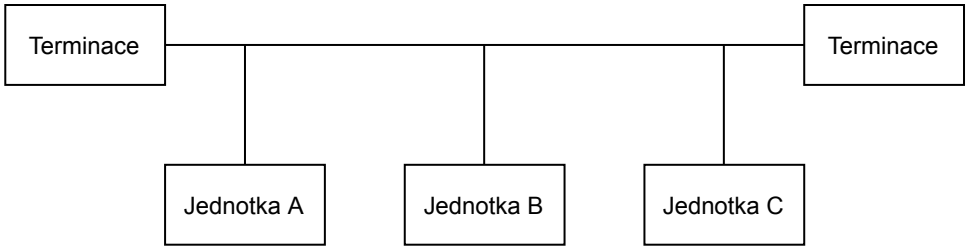
Kapitola 2

Sběnice CAN

Controller Area Network je automotive standardem vyvinutý společností BOSCH pro komunikaci mezi více mikrokontroléry a zařízeními. CAN byl navržen s cílem sjednotit používané sběrnice v automobilech, ale má velké nasazení i v jiných odvětvích průmyslu. První verze CAN standardu byla vydána již v roce 1986. Během své existence se standard dočkal několika rozšíření, které byly možné díky flexibilně navrženému formátu komunikace. Od roku 1991 je aktuální standard verze 2.0 [2].

2.1 Topologie

Po sběrnici CAN lze přenášet data mezi N jednotkami. Jedná se o multi-master topologii sítě, kde jsou si funkcí všechny jednotky rovny. V obrázku 2.1 je znázorněna topologie sběrnice. O přidělování přístupu k médiu se stará samotný protokol. Maximální množství jednotek omezuje pouze schopnost budičů vybudit sběrnici. Kvůli časování a vlastnostem fyzické vrstvy má CAN topologii sběrnice. Tzn., že jednotky jsou umisťovány přímo na vedení s minimálními odbočkami.



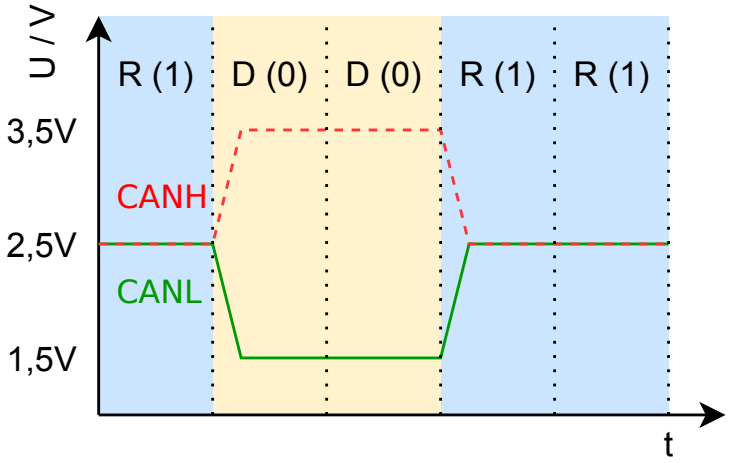
Obrázek 2.1: Topologie sběrnice CAN.

2.2 Fyzická vrstva

Jako nosné médium pro sběrnici CAN se používá kroucená dvojlinka o charakteristické impedanci 120 Ω. Jedná se tedy o diferenciální sběrnici. Vodič kladné polarity se označuje **CANH** a záporné **CANL**. Na každém konci je zakončena terminačním odporem pro potlačení odrazů. Z toho též vyplývá, že jakákoliv delší odbočka se chová jako nezateminované vedení a může způsobovat odrazy a poškozovat tak integritu dat na sběrnici.

Standard definuje dvě úrovně signálu. Způsobem definování stavů připomíná signáty typu otevřený kolektor, kdy je jedna úroveň definována slabě rezistorem a druhá úroveň silně kterýmkoliv zařízením na sběrnici. Nemůže tedy nastat zkrat.

Maximální bitrate sběrnice CAN je omezen na 1 Mbit

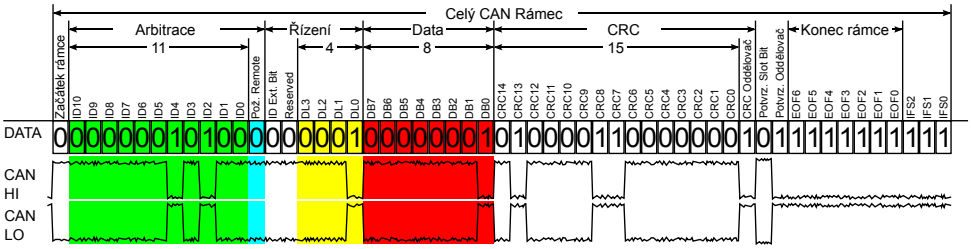


Obrázek 2.2: Stavy sběrnice CAN.

- Dominantní - stav, kdy je sběrnice některou jednotkou aktivně buzena. $U_{CANH} - U_{CANL} > 0\text{V}$. Vyjadřuje logickou úroveň 0.
- Recessivní - klidový stav, který je definován přítomností terminačních odporů, kdy žádná jednotka nebudí sběrnici. $U_{CANH} - U_{CANL} \leq 0\text{V}$ Vyjadřuje logickou úroveň 1.

2.3 Rámec

Komunikace po sběrnici probíhá v celcích nazývaných rámce nebo také označovanách jako CAN zprávy. Struktura je detailne vidět na obrázku 2.3.



Obrázek 2.3: Rámec sběrnice CAN se standardním ID. Převzato z [1].

Během arbitrační fáze bežně sběrnici ovládá více jednotek. Při tom sledují reální stav sběrnice. Pokud jednotka vysílá recesivní bit avšak na sběrnici vidí dominantní, rozpozná kolizi s jinou jednotkou vysílající ve stejnou chvíli a odmlčí se až do konce přenosu zprávy. Arbitraci vyhrává jednotka vysílající rámec s nejnižší hodnotou arbitrační části, nazývané také ID. Hodnota ID je přenášena od nejvíce významného bitu po nejméně významný. Existují dvě varianty ID zprávy. Standardní má délku 11 bitů (2.0A) a rozšířené ID pak 29 bitů (2.0B). Lze je na jedné sběrnici libovolně kombinovat. Následuje několik řídících bitů včetne informace o délce datového bloku. Pak je vyslán blok dat o maximální délce 8 bytů, který je validován kontrolním součtem. Za každým rámcem je vždy krátká odmlka kvůli synchronizaci. Pokud některá jednotka zprávu přijme, potvrzuje ji vysláním dominantního potvrzovacího bitu. Standard popisuje i chybové stavy sběrnice a jejich řešení. Pokud například některí jednotka generuje velké množství chyb, stane se na sběrnici pasivní, dokud není schopna přijímat validní zprávy.

2.4 Použití v eForce

Formule týmu eForce již několik generací používají dvě sběrnice CAN. Jedna bývá vyhrazena pro zprávy nutné k provozu formule. Na druhé sběrnici jsou data spíše ladící charakteru pro zpětné vyhodnocení stavu a chování formule. Tímto rozdělením, snížit vytížení první sběrnice a tak i latenci kritických zpráv.

Aby si všechny jednotky na sběrnici rozumněly, byl v týmu vyvinut nástroj pro definici obsahu a formátu zpráv canDB. Po ustálení formátu zpráv si každý vývojář nechá z canDB vygenerovat kód, který se stará o sestavování a parsování informací z hrubých CAN zpráv. V případě změny formátu se pouze dotčeným jednotám aktualizuje tento generovaný kód.

[1] `https://en.wikipedia.org/wiki/CAN_bus`

2.5 Varianta FD

Propustnosti sběrnice CAN postupně přestávají stačit stále se zvyšujícím datovým tokům. Jako řešení problému byla navržena rozšiřující varianta CAN Flexible Datarate (CAN FD). Navyšuje maximální množtví dat v jedné zprávě až na 64 bytů a dovoluje pro datovou část rámce přepnout bitrate až na 8 Mbit. Je tak zvýšena propustnost, se zachováním přenosového média. CAN FD je zpětně kompatibilní se sběrnicí CAN, ale ne naopak.

Kapitola 3

Bezdrátové spoje

V dnešní době se velké množství datových spojů realizuje jako bezdrátové. Motivací může být nižší pořizovací cena, tehdy se nejčastěji jedná o spoj mezi fixními stanicemi. V jiném případě bývá motivací fakt, že některá nebo i více stanic je mobilních. Příkladem takového spoje je například Wireless LAN (WLAN).

3.1 Komerčně dostupná řešení

Pro bezdrátové propojení dvou míst komunikující po sběrnici CAN již existují hotová komerční řešení. Žádné však nesplňují všechny nebo alespoň téměř všechny body zadání. Nejčastějším problémem je absence diverzity antén, nastavování přes aplikací fungující pouze pod operačním systémem Windows nebo závislosti na síti mobilního operátora. Z dokumentace těchto komerčních řešení nebylo možné zjistit, zda se snaží o rekonstrukci časování zpráv.

3.2 Wi-Fi

Wi-Fi je spoj ze skupiny WLAN, se kterým se nejčastěji setkávají koncoví uživatelé. Aktuálně nejnovější standard 802.11ax disponuje teoretickou maximální propustností téměř 7 Gbit/s [10] [1]. Technologie Wi-Fi je převážně určena pro pokrytí vnitřních prostor budov a připojení velkého množství uživatelů k títi Local Area Network (LAN). Nejčastěji dále do internetu. Wi-Fi operuje v pásmech 2,4 GHz a 5 GHz.

Zařízení jako jsou jednodeskové počítače běžně disponují vestavěnou Wi-Fi [1]. Horší je pak dostupnost CAN rozhraní. Nejjednodušším způsobem je připojení externí CAN periferie přes rozhraní SPI. Na těchto jednodeskových počítačích nejčastěji běží některá z distribucí GNU/Linuxu. Ačkoliv by se

[1] Teoretická rychlost v ideálním laboratorním prostředí.

jednalo o jedno z jednodušších řešení na realizaci, u CAN periferií připojených přes SPI k počítači s běžným os (bez realtime záplat) hrozí riziko ztrát CAN rámců [3] při vyšším vytížení. Další podstatnou nevýhodou těchto počítačů je dlouhá doba startu, která se řádově pohybuje od jednotek až po desítky sekund, což by v případě krátkodobé ztráty napájení kterékoliv stanice znamenalo dlouhý výpadek spoje.

Existují i mikrokontroléry s integrovaným Wi-Fi rozhraním. Rozšířeným zástupcem je například ESP8266 a jeho nástupce ESP32 od firmy Espressif Systems nebo mikrokontroléry z rodiny CC32xx od Texas Instruments.

3.3 Ostatní řešení

V neposlední řadě je možné využít některé z bezlicenčních ISM pásem [23] a k tomu vhodný transceiver. Jakékoliv řízení toku dat pak nejčastěji přechází na zodpovědnost řídícího firmwaru v mikrokontroléru. Transceiver samotný se pouze stará o modulaci/demodulaci, synchronizaci a přenos rámců. Tento přístup umožňuje větší kontrolu nad vysokofrekvenční částí spoje.

3.4 Pracovní pásmo

Bezlicenční bezdrátový přenos může být provozován v několika pásmech. V Evropě jsou to pásma 433 MHz, 868 MHz a 2400 MHz. Každé má své výhody i nevýhody. Výsledný výběr pracovního pásma musí být kompromisem mezi všemi ovlivňujícími faktory.

3.4.1 Ztráty ve volném prostoru

Jedním z hlavních rozdílů mezi pásmy jsou ztráty ve volném prostoru, které jsou úměrné frekvenci [26] str. 1320.

$$\mathrm{FSPL} = \left(\frac{4\pi df}{c}\right)^2 \tag{3.1}$$

Rovnici lze upravit.

$$\mathrm{FSPL(dB)} = 20\log_{10}(d) + 20\log_{10}(f) + 20\log_{10}\left(\frac{4\pi}{c}\right) \tag{3.2}$$

Vypočtené hodnoty útlumu podle rovnice 3.1 na pro několik vzdáleností jsou uvedeny v tabulce 3.2.

Frekvence [MHz]	$FSPL_{10m}$ [dB]	$FSPL_{100m}$ [dB]	$FSPL_{200m}$ [dB]
433	45	65	71
868	51	71	77
2400	60	80	85

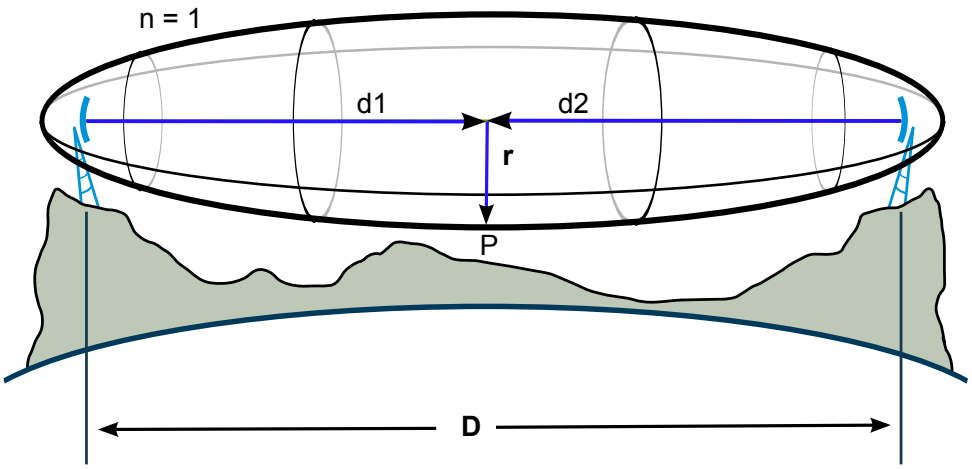
Tabulka 3.1: Hodnoty FSPL pro různé vzdálenosti a ISM pásma.

Samotné ztráty ve volném prostoru se s rostoucí frekvencí zvyšují. Parametr FSPL tedy naznačuje, že nižší frekvence jsou vhodnější pro realizaci spoje na delší vzdálenosti.

3.4.2 Fresnelova zóna

Energie vlny není rozložena pouze na spojnici mezi přijímačem a vysílačem. Většina energie se nachází v první Fresnelově zóně. Tvar této zóny ve směru šíření je kruhového průřezu s poloměrem závislým na frevenci a pozici mezi přijímačem a vysílačem. Tvar zóny je znázorněn na obrázku 3.1

Překážky v první Fresnelově zóně mají značný vliv na přenášenou vlnu. Překážky nevodivého charakteru mají tendenci energii vlny pohlcovat, překážky vodivého charakteru zase vlnu odrážet.



Obrázek 3.1: Tvar Fresnelovy zóny. Převzato z [2].

Rovnice 3.3 popisuje výpočet maximálního poloměru fresnelovy zóny.

$$r_n = \sqrt{\frac{\lambda\, n\, d_1\, d_2}{d_1 + d_2}} \tag{3.3}$$

Pro výpočet maximálního poloměru první Fresnelovy zóny, kdy $d_1 = d_2 = d$ a $n = 1$, lze rovnici zjednodušit 3.4.

[2] https://commons.wikimedia.org/wiki/File:FresnelSVG1.svg

$$r_{max} = \sqrt{\frac{\lambda d}{2}} \tag{3.4}$$

Frekvence [MHz]	$r_{max100m}$ [m]	$r_{max200m}$ [m]
433	4,2	5,9
868	2,9	4.2
2400	1,8	2,5

Tabulka 3.2: Maximální poloměry první Fresnelovy zóny pro různé frekvence a ISM pásma.

Podle tabulky lze usuzovat, ze spoje realizované na vyšší frekvenci mají užší Fresnelovů zónu a tedy nižší nároky na okolí spoje.

3.4.3 Dopplerův jev

Pro fungování bezdrátového spoje musí vysílač i přijímač pracovat se stejnou nosnou frekvencí. Jelikož v této aplikaci bude jedna strana spoje umístěna na elektrické formuli dosahující až rychlosti 120 km/h a elektromagnetická vlna se šíří konečnou rychlostí, je třeba zjistit dopad Dopplerova jevu na změnu nosné frekvence. Rovnice 3.5 popisuje změnu přijímané frekvence pevnou stanicí při pohybu vysílače. Rovnice 3.6 pak popisuje opačnou situaci, kdy je vysílač v klidu a pohybuje se přijímač. Je potřeba zvážit obě možnosti, protože plánovaný tlemetrický spoj je obousměrný.

$$f = \left(\frac{c}{c \pm v_s}\right) f_0 \tag{3.5}$$

$$f = \left(\frac{c \pm v_r}{c}\right) f_0 \tag{3.6}$$

- f_0 - frekvence vlny na straně vysílače
- c - rychlost šíření vlny (rychlost světla)
- $v_{s/r}$ - rychlost pohybu vysílače nebo přijímače
- f - frekvance vlny na straně přijímače

Tabulka 3.3 obsahuje vypočtené frekvenční odchylky $f_d = f_0 - f$ při maximální rychlosti formule 33 m/s pro všechny tři zvažovaná ISM pásma.

Frekvence [MHz]	f_{ds+} [Hz]	f_{ds-} [Hz]	f_{dr+} [Hz]	f_{dr-} [Hz]
433	47	-47	47	-47
868	95	-95	95	-95
2400	264	-264	264	-264

Tabulka 3.3: Vliv pohybu vysílače a přijímače na frekvenci přijímané vlny.

Výsledné frekvenční odchylky způsobené pohybem vysílače $f_{ds\pm}$ a přijímače $f_{dr\pm}$ jsou v jednotkách ppm nosné frekvence. Tolerance frekvencí krystalů používaných jako zdroj hodin pro transceivery je bežně v desítkách ppm. Přijímače musí takový rozdíl frekvencí brát při zpracování signálu v potaz a být k němu tolerantní. Vliv Dopplerova jevu pro tento spoj může být nadále považován za bezvýznamný a ignorován.

3.4.4 Odhad energetické bilance

Výpočetem energetické bilance spoje, lze zjistit přibližné nároky na komponenty celého přenosového řetězce nebo odhadnou maximální vzdálenost spoje. Rovnice 3.7 ukazuje příspěvky jednotlivých částí spoje.

$$L_{RES} = P_{TX} - L_{TX} + G_{TX} - L_{FSPL} + G_{RX} - L_{RX} - P_{RX} \tag{3.7}$$

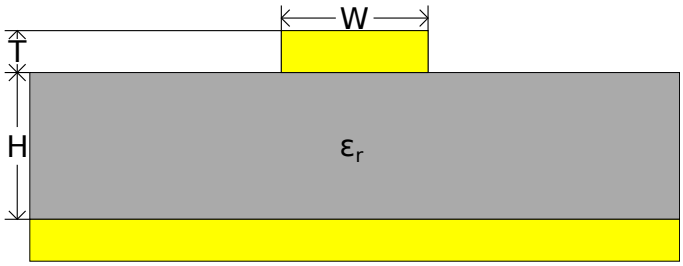
- L_{RES} - výkonová rezeva spoje pro pokrytí ztrát jako jsou překážky v cestě
- P_{TX} - vysílaný výkon
- L_{TX} - ztráty ve vedení mezi vysílačem a anténou
- G_{TX} - zisk antény vysílače
- L_{FSPL} - ztráty šířením ve volném prostoru
- G_{RX} - zisk antény přijímače
- L_{RX} - ztráty ve vedení mezi anténou a přijímačem
- P_{RX} - přijímaný výkon/citlivost přijímače

3.5 VF signály na DPS

Vysokofrekvenční obvody a prvky běžně pracují s vstupní a výstupní impedancí 50 Ω. Při propojování takovýchto prvků mimo DPS se lze setkat s koaxiálními kabely a vlnovody. Pro realizaci obdobných VF propojů na desce existuje několik planárních strulktur [12] kapitola *Transmission Lines and Waveguides*. vybrané struktury jsou zmíněny dále.

3.5.1 Mikropáskové vedení

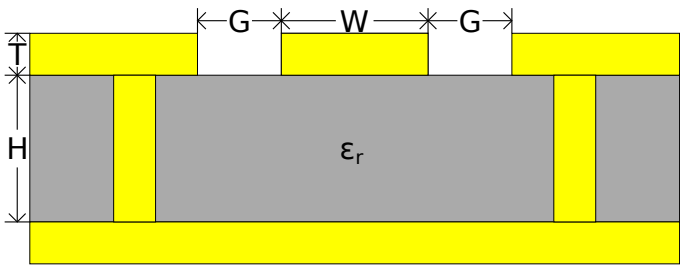
Skládá se ze signálového pásku a zemnící plochy mezi nimiž je dielektrický substrát. Řez strukturou je na obrázku 3.2. Jelikož je většina energie mezi zemnící plochou a mikropáskem, je mikropáskové vedení tolerantnější k tloušťce nakovení DPS. Nevýhodami je potřebný velký odstup od okolních komponent, vyšší přeslechy a ztráty ve vyšších frekvencích.



Obrázek 3.2: Řez mikropáskovým vedením.

3.5.2 Koplanární vedení

Oproti mikropáskovému vední je kolem vysokofrekvenční cesty (na obrázku 3.3 uprostřed) rozlitá zem. Nejčastěji používanou variantou koplanárnáho vedení je ta se zemí i na druhé straně substrátu, podobně jak je tomu u mikropáskového vedení. Vlnovodný kanál je pak z boků uzavřen prokovy. Největší část energie šířené vlny se nachází v mezerách mezi signálovou cestou a obklopující zemí. To má za následek vyšší citlivost na tloušťku nakovené vrstvy T. V porovnání s mikropáskovým vedením vychází šířka signálového vodiče W menší.



Obrázek 3.3: Řez koplanárním vedení se zemí.

3.6 RF vysílač/přijímač

První možností je použití mikrokontroléru s integrovaným Wi-Fi hardwarem. Díky rostoucímu Internet Of Things (IOT) odvětví roste i nabídka takových mikrokontroléru. Jejich výhodou je často vysoký stupeň integrace. Běžně obsahují mikrokontorlér s velkým množstvím periferií, další baseband koprocesor

pro obsluhu bezdrátového hwardwaru a vysokofrekvenční část s výkonovými zesilovači a přepínači. Parametry vybraných mikrokontrolérů jsou v tabulce 3.4.

Obvod	[Mbit/s]	citlivost [dBm]	Výstupní výkon [dBm]	RAM [KiB]	Flash	Periferie
ESP8266	11 54 72.2	-91 -75 -72	20 17 14	80	až 16 MiB ext.	UART, SDIO, SPI, I2C, I2S, IR, GPIO, ADC, PWM, DMA
ESP32	1 6 6.5 13.5 11 54 72 150	-98 -93 -93 -90 -88 -75 -73 -70	19.5 13	520	448 KiB int. až 16 MiB ext.	UART, SDIO, SPI, I2C, I2S, IR, GPIO, ADC, PWM, DAC, CAN, HALL, DMA, ETHERNET, TOUCH
CC32xx	1 2 6 6.5 9 11 18 36 54 65	-95 -94 -90 -89 -89 -88 -86 -81 -74 -79	18.0 18.0 17.3 17.3 18.3 17 16 14.5 13	256	až 1 MiB int. až 16 MiB ext.	DMA, SDIO, I2C, I2S, ADC, PWM, GPIO

Tabulka 3.4: Srovnání mikrokontroléru s Wi-Fi. (ESP8266[19], ESP32 [20], CC3220[21])

Druhou možností je transceiver pracující v některém ISM pásmu. V tabulce 3.5 jsou parametry vybraných transceiverů. Toto řešení nabízí větší variabilitu komponent protože neintegruje uživatelský mikrokontrolér.

Obvod	Nosná frekvence [MHz]	Datový tok [kbit/s]	Výstupní výkon [dBm]	Citlivost [dBm]	Diverzita antén
nRF24L01	2400 ∼2500	1000 2000	0	-85 -82	ne
SPIRIT1	150 ∼174, 300 ∼348, 387 ∼470, 779 ∼956	500	16	-88	ne
Si446x	142 ∼175, 284 ∼350, 420 ∼525, 850 ∼1050	500 1000	20	-97 -88	ano

Tabulka 3.5: Srovnání ISM transceiverů. (NRF24L01[14], SPIRIT1[6]), SI446x[17]

Kapitola 4

Komponenty

4.1 Transceiver

V minulosti jsem již jsem měl možnost pracovat s mikrokontroléry rodiny CC32xx. Jejich dokumentace vždy počítá s použitím Hardware Abstraction Layer (HAL) knihoven výrobce, které obalují nízkoúrovňový přístup k periferiím. Běžně není tento předpoklad omezením, avšak způsob jakým je napsána značně znesnadňuje ladění. Způsob zabezpečení obsahu paměti flash a tedy i programu je vhodný pro komerční produkty a v této aplikaci je spíše na obtíž. Komunita vývojářů kolem CC32xx také není příliš rozsáhlá a v případě problémů je vývojář odkázán na podporu od výrobce, kde ne vždy najde řešení. Kvůli těmto negativům jsem mikrokontroléry rodiny CC32xx nakonec vyloučil.

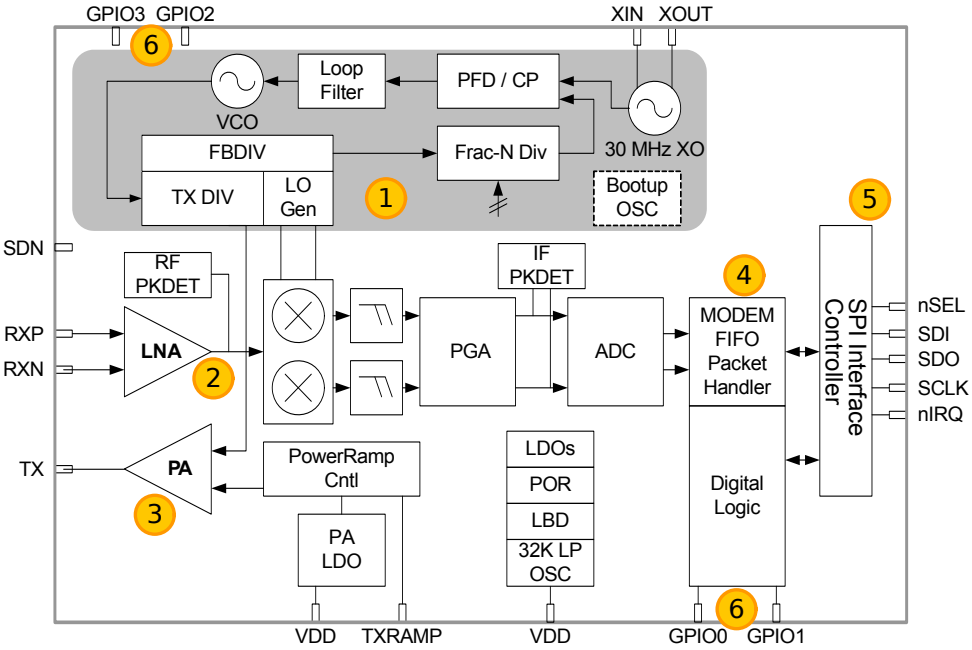
Mikrokontroléry ESP jsou velmi rozšířené v hobby odvětví, ale mají i velké nasazení v produktech IOT. Rozsáhlá komunita poskytuje rychlé řešení na mnohém problémy. Existuje také mnoho otevřených projektů a příkladů kódu. ESP32 disponuje jednou CAN periferií, což je nedostačující a musela by se připojit externí přes SPI. ESP32 lze pořídit na rozličných modulech avšak v době tvorby této práce žádný z modulů nedisponoval přepínačem antén a jejich vyvedením na konektor. Jistě by šlo poskládat design ze základních komponent bez použití hotového modulu, ale kladlo by to velké nároky na návrh plošného [1] spoje materiály i potřebné vybavení.

4.1.1 Si4463

Nakonec jsem zvolil transceiver Si4463. Disponuje nejvyšším vysílacím výkonem a lze k němu jednoduše připojit externí přepínač antén, výkonový zesilovač i nízkošumový zesilovač pro příjem. Transceiver pracuje v několik pásmech od 119 MHz do 1050 MHz. Často používanými ISM pásmy jsou

[1] A tedy jako první zkušenost s VF návrhem nevhodné.

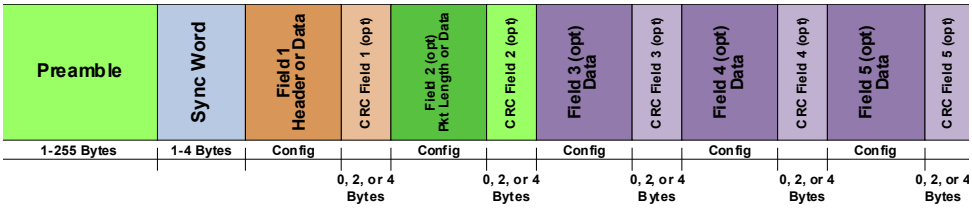
433 MHz, 868 MHz a 2400 MHz. Vyšší frekvence kladou vyšší nároky na návrh desek plošných spojů, na použité materiály a komponenty. Nižší frekvence zase vyžadují větší antény a jsou více zarušeny. Zároveň je pásmo 433 MHz hojně využíváno ve spotřební elektronice. Výrobce poskytuje vývojové desky a také vypočtené filtry pro několik frekvencí a 868 MHz je mezi nimi. Rozhodl se pro využití pásma 868 MHz.



Obrázek 4.1: Vnitřní blokové schéma transceiveru Si446x.

Vnitřní schéma na obrázku 4.1 ukazuje základní bloky transceiveru.

1. Přeladitelná, neceločíselná násobička hodin (Phase Locked Loop (PLL)) poskytuje transceiveru referenční hodnotu nosné frekvence. Jejím zdrojem je externí krystal.
2. Nízkošumový zesilovač s diferenciálním vstupem v přijímací cestě. Jeden z klíčových prvků určující citlivost zesilovače.
3. Výstupní zesilovač s výkonem až 20 dBm.
4. Paketizér se stará o sestavování paketů při vysílání a o jejich dekódování při příjmu. Struktura paketu je na obrázku 4.2.
   - Každý paket uvádí preamble, což je periodická sekvence dat (z pravidla opakující se 0b01) určená k bitové synchronizaci přijímače a doladění případného frekvenčního ofsetu. Po dobu jejího příjmu může transceiver přepínat antény a zjišťovat, která je vhodnější pro příjem.
   - Synchronizační slovo (Sync Word) ukončuje preambli. Jedná se o fixní, dohodnutou hodnotu. Jakmile transceiver najde synchronizační slovo, dojde bytové synchronizaci.



Obrázek 4.2: Struktura paketu transceiveru si446x. Převzato z [17]

- Až 5 datových polí o velikosti až 8191 bytů se nachází hned za synchronizačním slovem. Jejich velikost může být předem dohodnutá a tedy fixní nebo s proměnlivou délkou. Při použití pole proměnlivé délky je jedno pole vyhrazeno pro délku a kterékoliv vybrané následující pak má tuto délku. Ostatní pole musí být fixní nebo vypnutá.
- CRC neboli Cyclic redundancy check je volitelná položka v paketu, která slouží pro jednoduchou kontrolu integrity obsahu datového pole. Paketizér je schopný CRC generovat i automaticky kontrolovat.

5. Komunikace s nadřazeným systémem probíhá přes rozhraní SPI o maximální rychlosti 12,5 MHz.
6. Multifunkční General Purpose Input Output (vstupně výstupní pin) (GPIO) piny lze přepnout mnoha módů, jako jsou například:
   - digitální vstup/výstup
   - analogový vstup pro AD převodník
   - řízení externího přepínače antén
   - řízení externího výkonového zesilovače (PA)
   - indikace připravenosti na další příkaz
   - signál přerušení při různých událostech
   - serializovaná data z paketizéru

4.2 Koncový zesilovač

Vybraný transceiver podporuje široké možnosti řízení externích komponent jako jsou přepínače antén nebo řízení externího výkonového zesilovače (PA). Jeho vlastnosti jako je vysílaný výkon nebo citlivost jde přidáním externích komponent zlepšit. Kvůli minimalizaci vysokofrekvenčních komponent a tak zjednodušení motivu desky jsem hledal vysoce integrované řešení, které by obsahovalo ji zmíněný přepínač antén, výkonový výstupní zesilovač a nízkošumový zesilovač pro příjem.

4.2.1 SE2435L

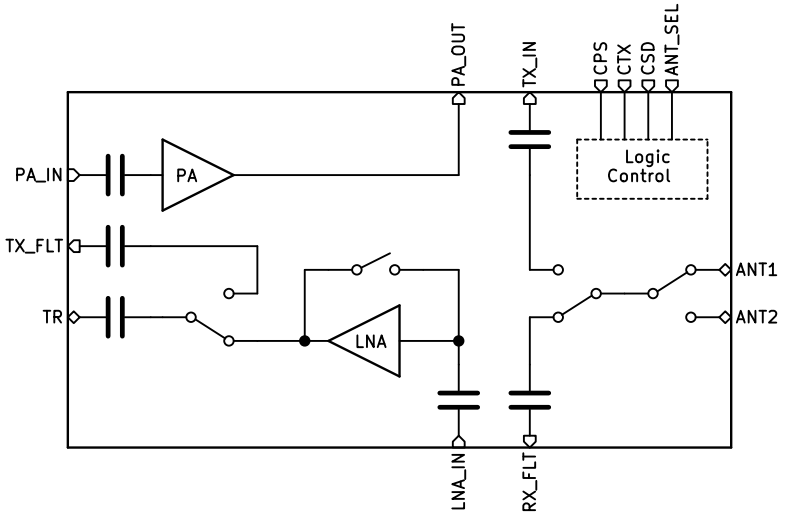
Tento vysokofrekvenční front-end je použit na vývojových deskách přijímače SPIRI1 od ST microelectronics. Při použití lze tedy vycházet z doporučených

motivů DPS, ale také použít již vypočtené filtry apod. Disponuje všemi požadovanými částmi a ke své funkci potřebuje minimum externích komponent.

Výstupní zesílení	+dBm	27
Napájecí napětí	V	2 až 4.8
Vstupní zesílení	dB	16
Frekbenční rozsah	MHz	860 až 930

Tabulka 4.1: Základní parametry frontendu SE2435L.[18]

Na obrázku 4.3 je vidět zjednodušené vnitřní zapojení tohoto obvodu.



Obrázek 4.3: Vnitřní blokové zapojení frontendu SE2435L. Překlresleno z [18].

4.3 Mikrokontrolér

Mikrokontrolér je centrálním prvkem celé jednotky. Řídí všechny periferní obvody jako je transceiver, audio kodek nebo display a komunikuje s vnějším světem v podobě dvou CAN sběrnic. V týmu eFroce již několik generací formule stavíme na mikrokontrolérech rodiny STM32. Jsou to mikrokontroléry založené na jádrech ARM Cortex-M. Portfolio mikrokontrolérů STM32 je rozmanité a pohybuje je od modelů s nízkou spotřebou založených na jádrech Cortex-M0 v pouzdrech UQFN-28 až po výkonné Cortex-M7 v pouzdrech BGA. Při výběru vyvstaly tyto požadavky:

- 2 integrované CAN 2.0B periferie pro komunikaci se zbytkem formule
- Alespoň 2 SPI rozhraní pro připojení periferních obvodů
- Serial Audio Interface (SAI) rozhraní pro možnost budoucího rozšíření spoje o hlasovou komunikaci. Oproti rozhraní Inter-IC Sound (I2S) je SAI flexibilnější.

- podpora aritmetiky čísel s plovoucí desetinou čárkou (Floating Point Unit (FPU))
- minimální ERRATA nebo alespoň s minimálním dopadem na použité periferie
- Rozhraní Quad SPI (QSPI) pro připojení externí flash paměti k ukládání obrázků a zvuků.

4.3.1 STM32F446RC

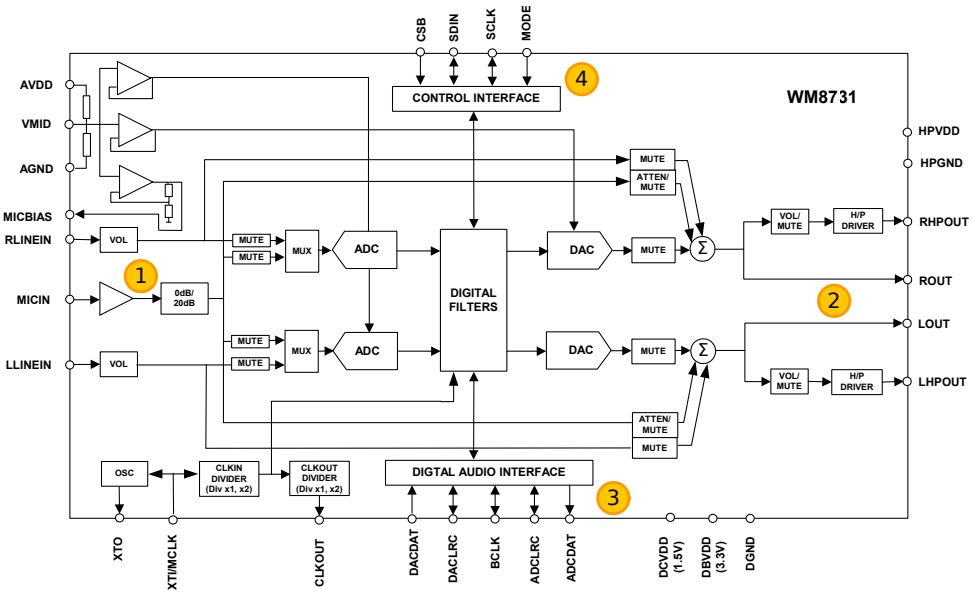
Při zvážení všech požadavků a zkušeností jsem zvolil mikrokontrolér STM32F446. Splňuje všechny kladené požadavky. V tabulce 4.2 je výčet základních parametrů.

Jádro	Cortex-M4
Maximální hodinová frekvence	180 MHz
Flash paměť	256 KiB
RAM paměť	128 KiB
Počet GPIO	114
Čítačů časovačů	14
CAN 2.0B	2
SPI	4
QSPI	1
SAI	2

Tabulka 4.2: Základní parametry mikrokontrolérů STM32F446RC.[8]

4.4 Audio kodek

Jednou z budoucích vlastností spoje by měl být obousměrný přenos zvuku. K tomuto účelu jsem hledal kodek s integrovaným koncovým zesilovačem pro sluchátka.



Obrázek 4.4: Vnitřní blokové zapojení audio kodeku WM8731. Převzato z [9]

Důležité části pro aplikaci:

1. Mono vstup mikrofonu včetně výstupu pro jeho napájení.
2. Stereo výstup s koncovými zesilovači a nastavitelnou hlasitostí.
3. Audio rozhraní kompatibilní s I2S a dalšími formáty dat.
4. Ovládací rozhraní SPI pro konfiguraci parametrů.

Kapitola 5

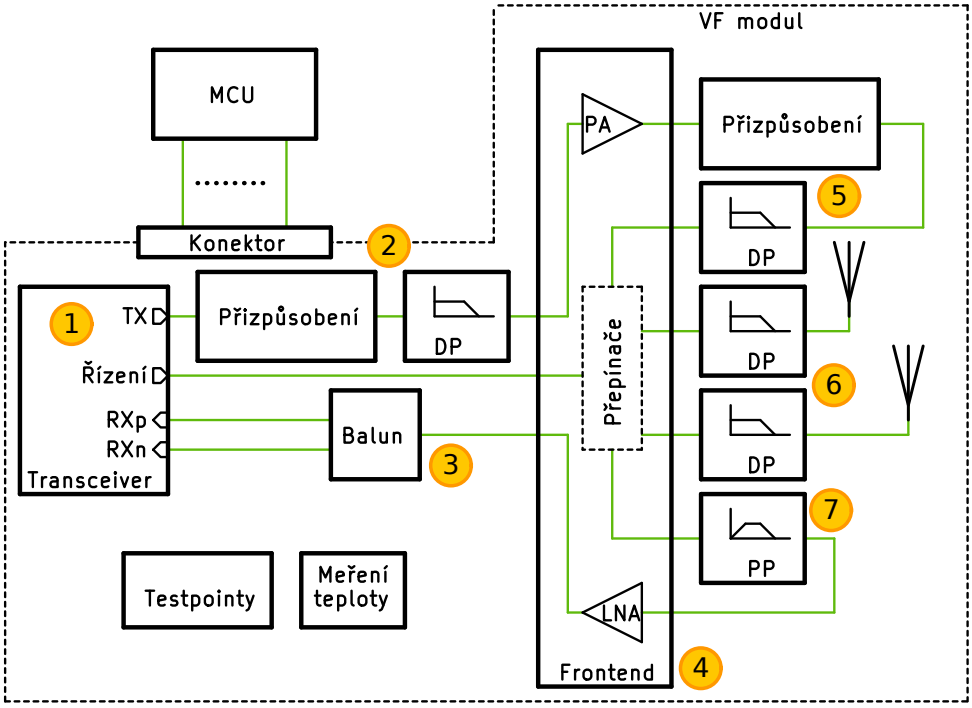
Návrh

V této kapitole je popsán návrh hardwaru elektroniky spoje. Od zapojení všech komponent až po návrh desek plošných spojů. Celý spoj je navrhován s cílem minimalizovat množství rozdílných komponent mezi pozemní jednotkou a tou umístěnou ve formuli. Lišit by se měly maximálně v tom, které komponenty jsou osazeny a které ne, nikoliv však v motivech desek nebo typu komponent. Takto lze zkrátit dobu potřebnou k vývoji.

Po zvážení nedostatků komerčních řešení jsem se rozhodl pro vývoj vlastního zařízení, které bude maximálně vyhovovat požadavkům.

5.1 Vysokofrekvenční část

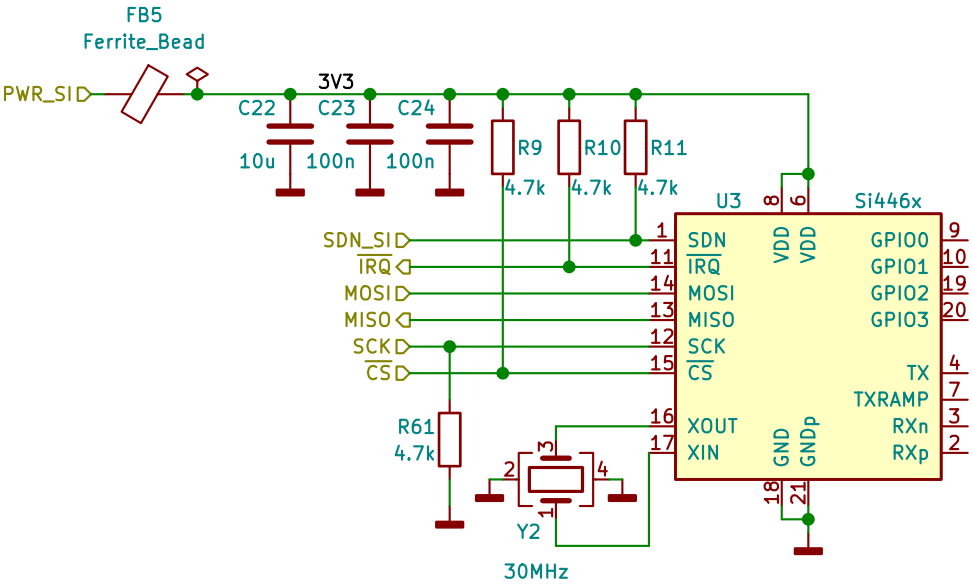
VF modul jsem navrhoval s cílem umožnit jeho jednoduchou výměnu v případě nutných změn ve VF části. Je teoreticky možné tento modul zcela vyměnit za jiný, založených na jiných VF obvodech. Oddělení jsem zvolil i pro případ, že bude nutné provést změny na VF modulu a vyhnout se opětovnému osazování zbytku elektroniky jako je mikrokontrolér nebo zdroje. S nimi je pak propojen jediným konektorem. Zjednodušené blokové schéma je na obrázku 5.1



Obrázek 5.1: Blokové schéma vysokofrekvenční části.

5.1.1 Transceiver

V obrázku 5.1 blok 1 je vybraný transceiver Si4463 4.1.1.



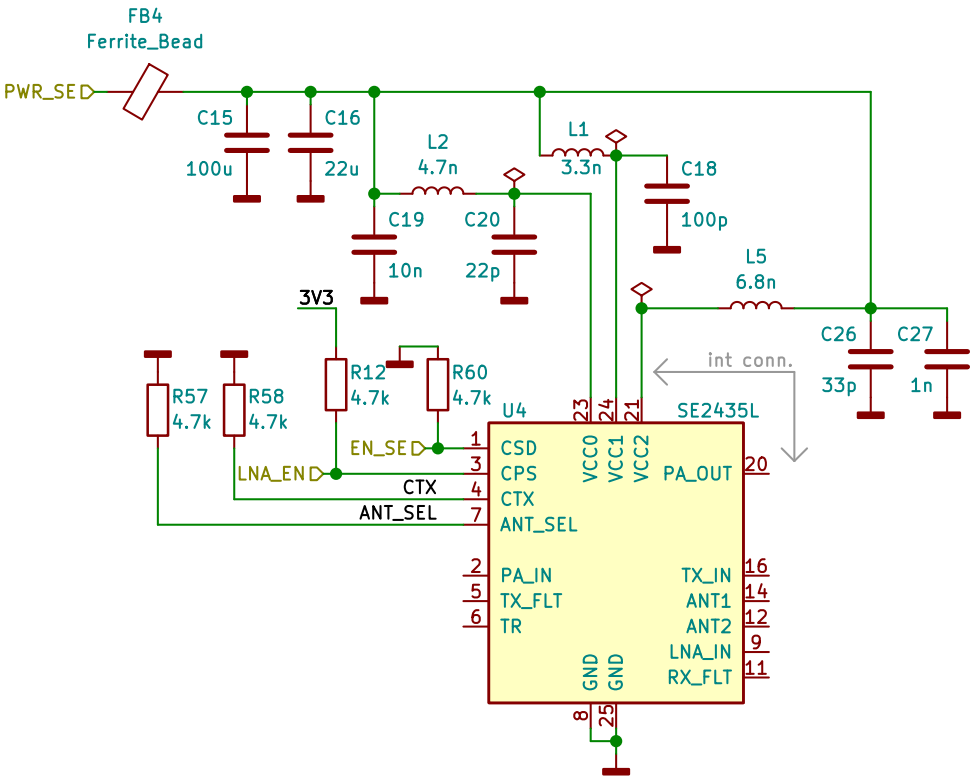
Obrázek 5.2: Schéma zapojení napájení a SPI rozhraní VF transceiveru.

Nepočítaje VF části, potřebuje transceiver k provozu pouze minimum

externích komponent. Schémat napájení a SPI rozhraní je na obrázku 5.2. Kondenzátory C22, C23 a C24 slouží jako lokální zásoba energie pro situace kdy spotřeba transceiveru skokově vzroste. To se děje z pravidla při přechodu do režimu vysílání. Kondenzátory musí být umístěny blízko samotného transceiveru kvůli minimalizaci indukčnosti cest mezi samotným transceiverem a kondenzátory. Feritová perlička FB1 naopak zvyšuje impedanci napájecí cesty ve vyšších frekvencích a brání tak proudovým pulzům způsobující rušení v okolí napájecí větvě dále na desce. Odpory R9, R10, R11 a R61 zajišťují aby byly na SPI rozhraní vždy definované úrovně, pokud mikrokontrolér neřídí SPI sběrnici. To se může stát při jeho resetu nebo než dojde k nakonfigurování jeho GPIO pinů. Poslední komponentou pro funkci transceiveru je externí krystal s frekvencí 30 MHz.

5.1.2 FrontEnd

V obrázku 5.1 blok 4 je frontend SE2435L 4.2.1.

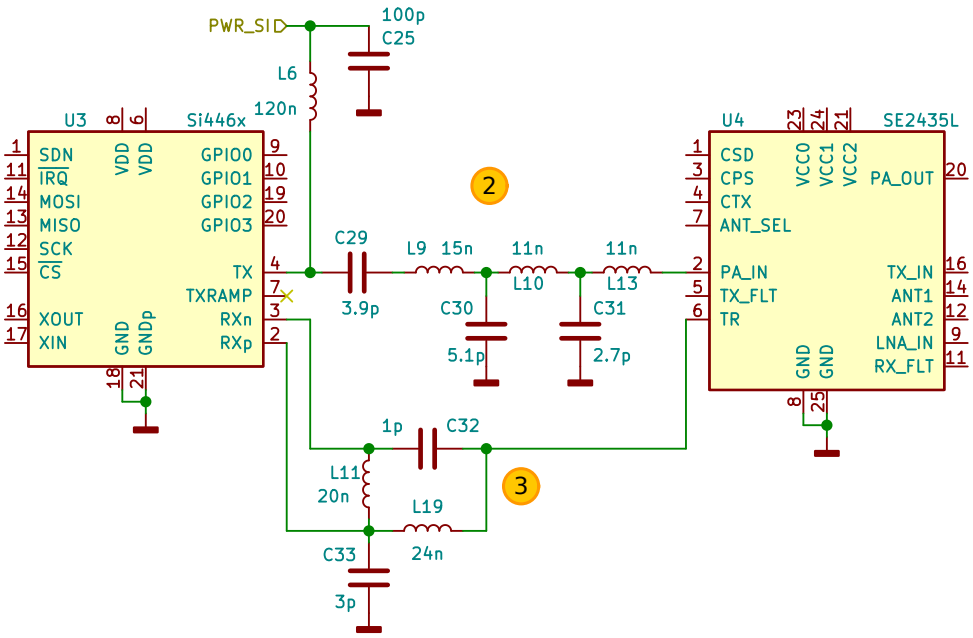


Obrázek 5.3: Schéma zapojení napájení a SPI rozhraní VF transceiveru.

Kromě VF částí popsaných v samostatných kapitolách potřebuje ke své funkci vhodně filtrované napájení. Schéma napájení je na obrázku 5.3. Hlavní zásobou energie jsou kondenzátory C15 a C16. Feritová perlička obdobně jako u transceiveru brání šíření VF rušení dále do desky. Zbylé filtry pro napájecí větve jsou převzaty z desky s přijímačem SPIRIT1 [7].

Logické signály pro zapínání transceiveru **EN_SE** a pro přemosťování **LNA_EN** jsou řízeny přímo z mikrokontroléru. Signály **CTX** přepínající mezi vysílacím a přijímacím režimem a **ANT_SEL** vybírající připojenou anténu jsou řízeny z transceiveru. Všechny signály mají opět definovánu úroveň pomocí rezistorů pro případ, že z nějakého důvodu nejsou řízeny z jejich zdroje.

5.1.3 VF propojení transceiveru a frontendu



Obrázek 5.4: Schéma propojení transceiveru a frontendu.

Hodnoty součástek vycházejí z doporučení výrobce a jsou uvedeny v aplikační poznámce [16].

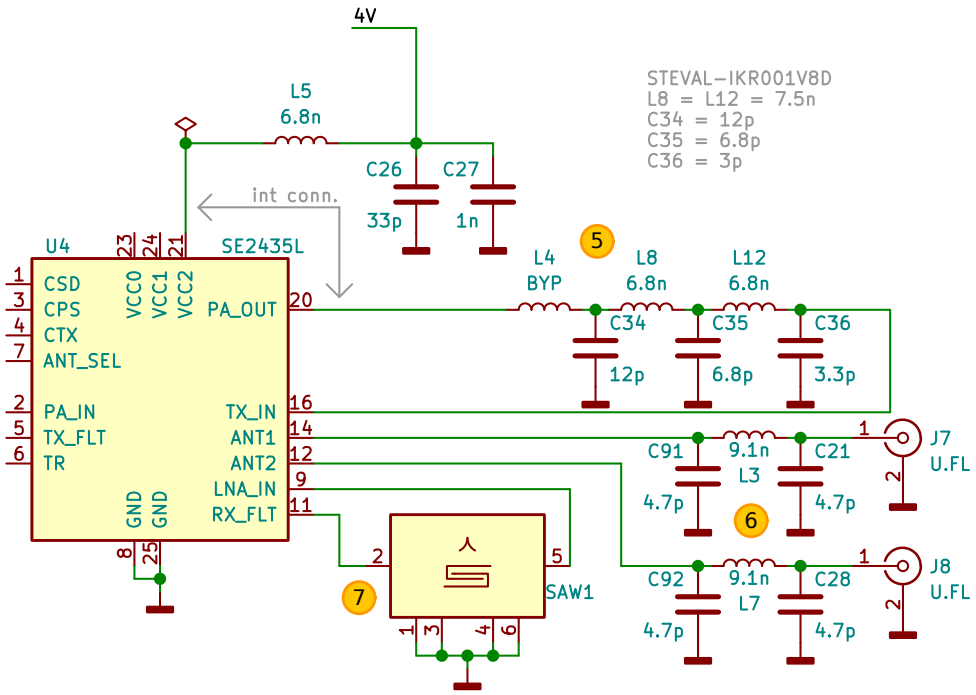
Vysílací cesta

Výstup transceiveru **TX** nelze přímo připojit na 50 Ω vstup frontendu **PA_IN** a je zde minimálně vyžadováno impedanční přizpůsobení. Zároveň je ve vysílací cestě zařazen filtr typu dolní propust pro potlačení vyšších harmonických. Obojí je znázorněno v blokovém schématu 5.1 i v detailním schématu 5.4 jako blok 2.

Přijímací cesta

Opět 50 Ω výstup frontendu **TR** nelze přímo připojit do rozdílového vstupu transceiveru **RXp/n**. Pro převod signálu ze symetrického, šířeného po dvou vodičích s identickou impedancí vůči zemi na signál nesymetrický, šířící se po jednom vodiči vztaženém vůči zemi a naopak se používá zapojení označováno jako balun (balanced-unbalanced). Zapojení je znázorněno v blokovém schématu 5.1 i v detailním schématu 5.4 jako blok 3.

5.1.4 Filtry VF frontendu



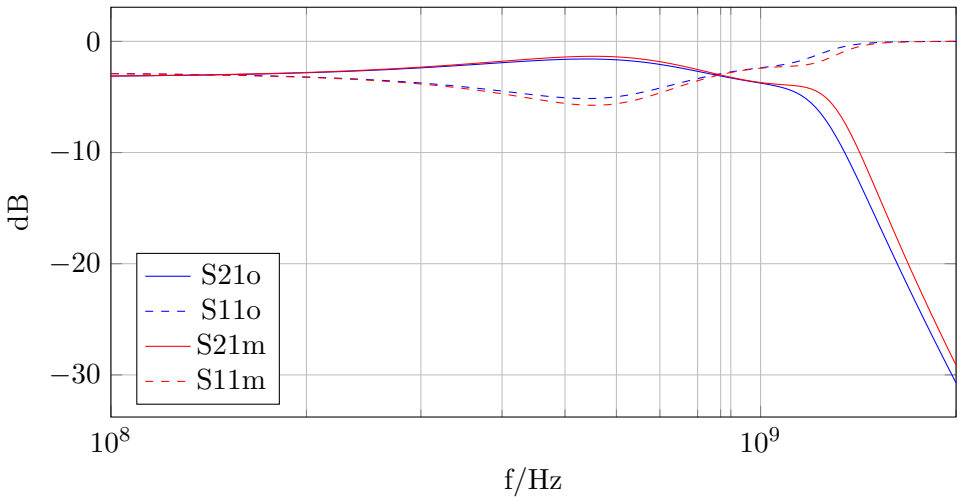
Obrázek 5.5: Schéma filtrů kolem frontendu.

Filtr výkonového zesilovače

Samotný frontend kromě VF přepínačů obsahuje další výkonový zesilovač podobný tomu v transceiveru. Opět je potřeba přizpůsobit výstupní impedanci zesilovače na portu **PA_OUT**, kterou výrobce uvádí jako 8 Ω v pásmu kolem 868 MHz, na hodnotu 50 Ω pro následující filtr a port frontendu **TX_IN**. Impedanční přizpůsobení a filtr jsou znázorněny v blokovém schématu 5.1 i v detailním schématu 5.5 jako blok 5.

Pro tuto VF část jsem se snažil hodnotami součástek přiblížit k těm v ve vývojové desce s transceiverem SPIRIT1 [7]. Kvůli momentální nedostupnosti některých hodnot jsem filtr musel upravit. Filtr s novými hodnotami

jsem porovnal v simulaci. V grafu 5.6 je vidět, že na 868 MHz nedošlo v S parametrech prakticky k žádné změně. Zlomová frekvence se lehce zvýšila.



Obrázek 5.6: Porovnání závislosti parametrů S21 a S11 obou variantu filtru. Přípona značí *o* původní verzi filtru a přípona *m* upravenou verzi.

Anténní filtry

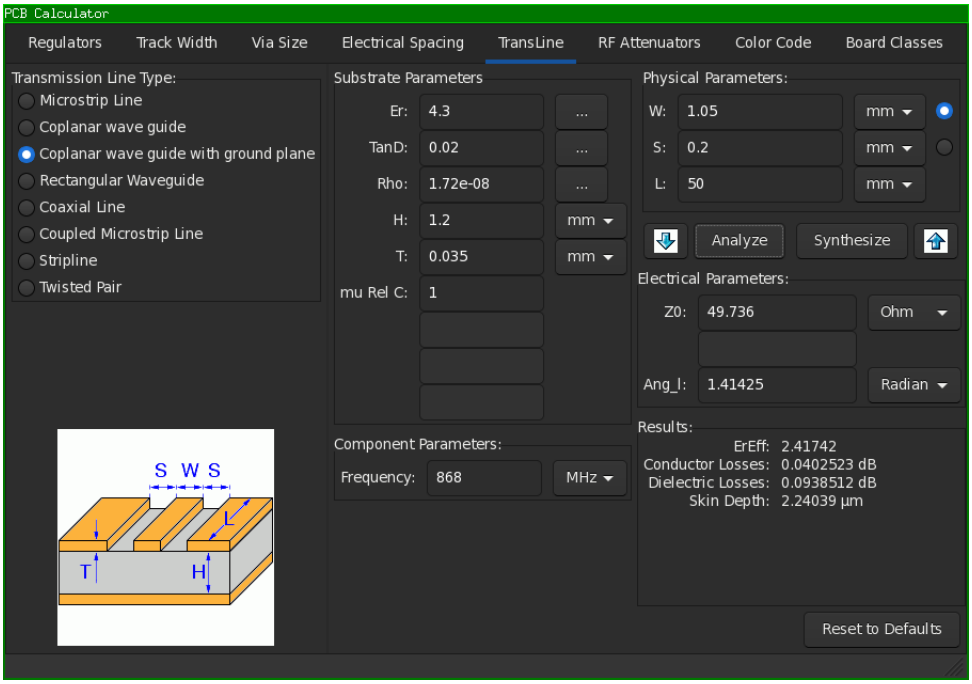
V blokovém schématu 5.1 i v detailním schématu 5.5 jsou označeny jako blok 6. Při vysílání jsou tyto filtry poslední v řetězci a potlačují vyšší harmonické vznikající na spínačích uvnitř frontendu. Při příjmu jsou tyto filtry v řetězci první a potlačují frekvence nad pracovním pásmem VF frontendu. Hodnoty filtru jsou převzaty z [17].

Přijímací SAW filtr

Jedná se o SAW filtr typu pásmová propust, není nezbytný, ale umí velmi selektivně propustit signál jen ve vybraném pásmu a s menším útlumem než LC filtry. Za následek to má snížení nároků na digitální filtry v transceiveru.

5.1.5 Návrh DPS

Všechny VF komponenty jsou na samostatném modulu. Návrhu DPS pro VF obvody vyžaduje dodržení několika základních pravidel. Použité komponenty vyžadují propojení vedením o impedanci 50 Ω. Referenční návrhy, ze kterých vycházím, používají koplanární vedení se zemí (Grounded Coplanar Waveguide (GCPW)) 3.5.2. Jedná se u plošně úsporné VF vedení. Jeden z požadavků GCPW je celistvost zemní plochy. Jelikož na i na VF modulu vedou různé řídící signály, které kvůli úspoře místa musí křížit VF cesty, rozhodl jsem se pro 4 vrstvou desku. K výpočtu geometrie GCPW jsem použil nástroj v

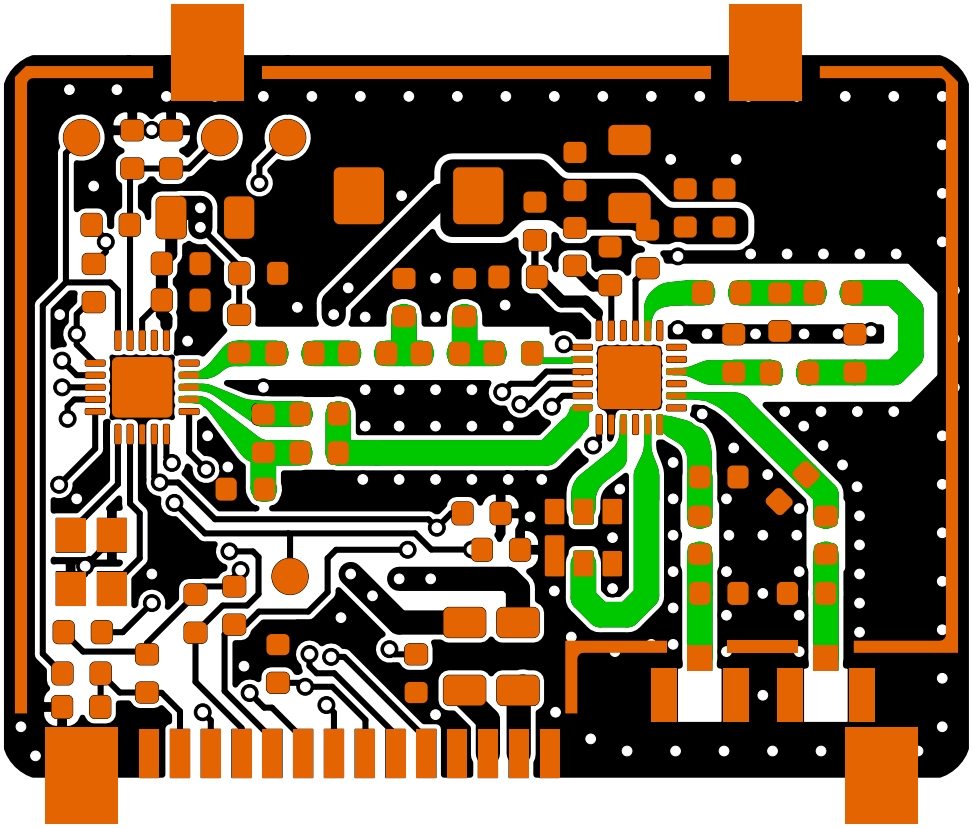


Obrázek 5.7: Nástroj z balíku EDA KiCAD pro výpočet parametrů VF vedení.

EDA softwaru KiCAD 5.7. Jedním ze vstupních parametrů výpočtu, které musí být předem dány je tloušťka substrátu a jeho materiál. Firma u které tým eForce vyrábí desky používá substrát ITEQ158 [5]. Vzdálenost krajních vrstev od sebe je pouze 80 µm, což vychází na velmi tenké GCPW spoje. S VF vrstvou tedy nemůže být v páru žádná další. Tímto je návrh omezen na využití pouze 3 vrstev. Tloušťka substrátu u čtyřvrstvé desky je 1,2 mm. s těmito rozměry a parametry substrátu vychází 50 Ω GCPW cesta s šířkou 1,05 mm a odstupem od okolní zemnící plochy 0,2 mm

Výrobce transceiveru doporučuje [15] kolem pasivních komponent, tj impedančních přizpůsobení a filtrů, zvětšit odstup země. Sníží se tak vazební kapacita mezi terminály jednotlivých pasivních součástek ve VF cestách, hlavně pak u cívek, kde je tato vlastnost silně nežádoucí.

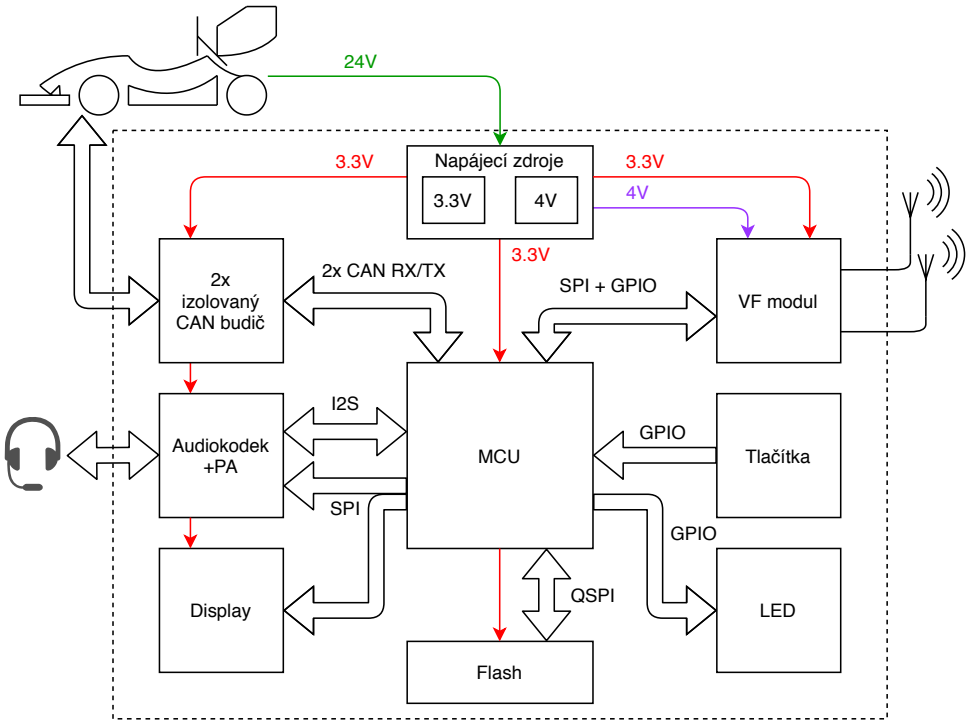
Na obrázku 5.8 je vidět horní vrstva motivu deky. Všechny VF cesty jsou právě v této vrstvě. Zelené cesty jsou navrhovány s jako GCPW s impedancí 50 Ω, ostatní cesty jsou černé. Oranžová znázorňuje masku. I když se jedná o druhou verzi, má stále nedostatky a v budoucnu bude vyrobena opravná verze. Další vrstva je zcela vynechána, vnitřní vrstva z druhé strany desky je celistvá zem a na spodní vrstvě jsou další spoje pro zajištění řízení a napájení.



Obrázek 5.8: Horní vrstva VF modulu.

5.2 Základní deska

Tato deska nese všechny ostatní komponenty a VF modul samotný. Blokové schéma desky je na obrázku 5.9.



Obrázek 5.9: Blokové schéma celé jednotky.

5.2.1 Napájecí zdroje

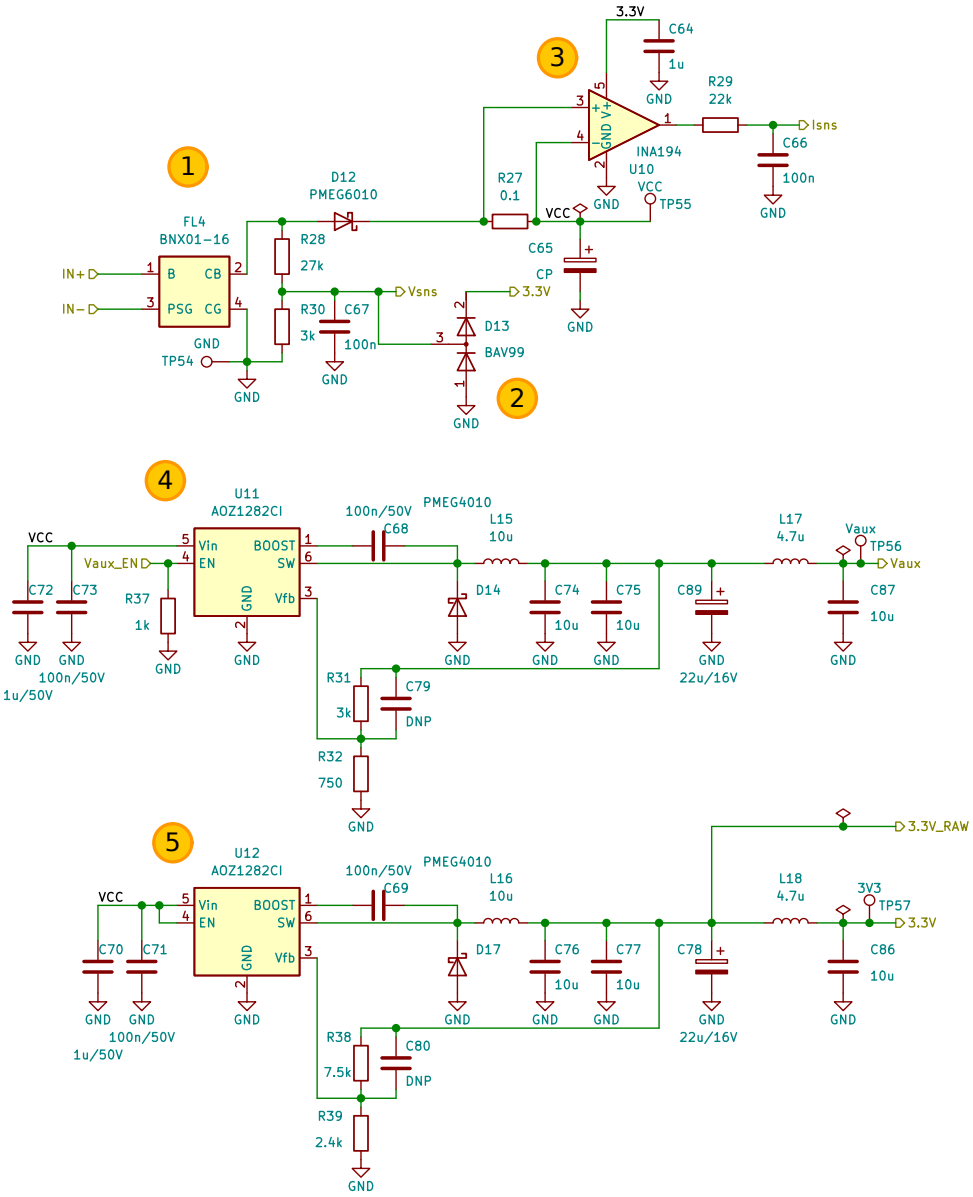
Celá jednotka telemetrie má být napájena ze stejnosměrného systému formule o napětí 24V. Pozemní stanice pak ze samostatného zdroje 12 V až 24 V. Potřebné napájecí větve na desce jsou 3,3 V pro mikrokontrolér a většinu periferií a 4 V pro VF frontend. Schéma napájecí části je na obrázku 5.10.

1. Ihned u konektoru je umístěn EMI filtr FL4. Zabraňuje šíření rušení po napájení formule. Brání průniku rušení z venku dovnitř a naopak.

2. Ještě před diodou D12 je dělič pro měření vstupního napětí. V případě ztráty napájení tak může systém ještě chvíli fungovat z energie v kondenzátoru C65 a například uložit nastavení než dojde k úplné ztrátě napětí.

3. Proudový odběr celé jednotky je snímán na rezistoru R27 a zesílen v obvodu U10.

4. Jako snižující spínané DC/DC zdroje používáme v týmu obvody AOZ1282CI, které se prokázaly jako spolehlivé a cenově dostupné. Konkrétně blok 5 blok zajišťuje napájení pro VF transceiver 4 V a je zapínán signálem **Vaux_EN** z mikrokontroléru.



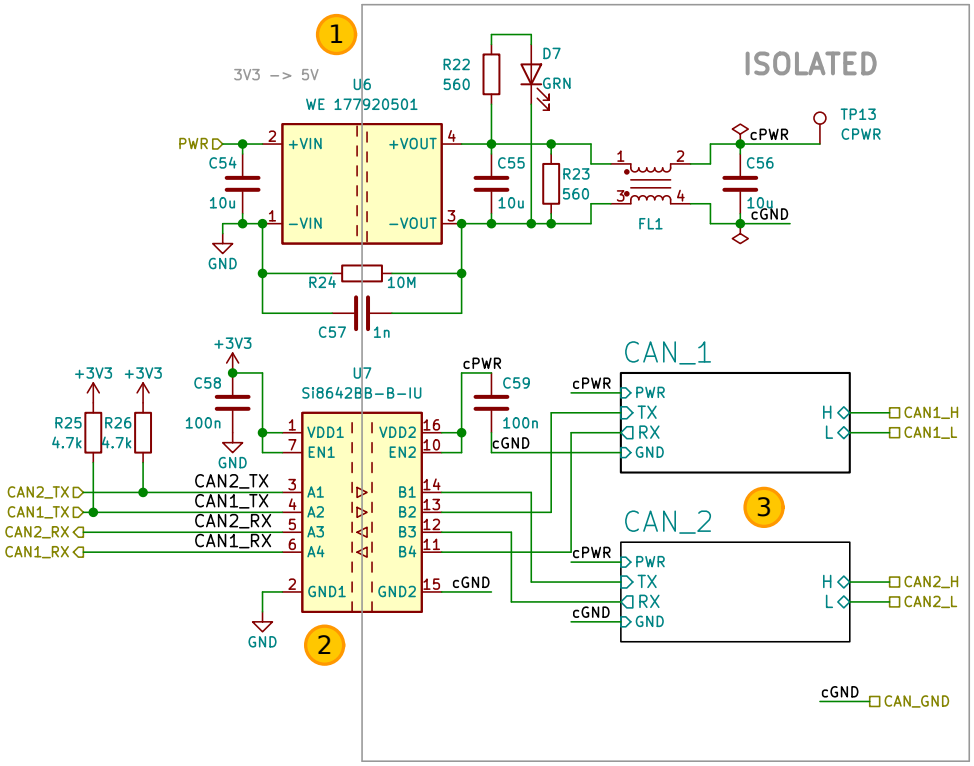
Obrázek 5.10: Schéma napájení jednotky.

5. Tento blok je téměř identický s předchozím blokem 4. Je však v provozu vždy při přivedení napájení a děličem R38 a R39 je nastaven na výstupní napětí 3,3 V.

5.2.2 Izolované rozhraní CAN

Jak již bylo zmíněno v dřívější kapitole 1.3, formule týmu eForce používá dvě sběrnice CAN 2.0. V minulých sezónách byly problémy s integritou signálu a to přesto, že se jedná o diferenciální sběrnici s kroucenou dvojlinkou. Některé jednotky ve formuli jsou schopné generovat značné elektromagnetické rušení

(Electromagnetic Interference (EMI)). Jedná se o frekvenční měniče pro trakční motory, jednotky spínající čerpadla a ventilátory chladících systémů a další. Jako řešení se ukázalo galvanické oddělení CAN sběrnice. V důsledku mají dva kroucené páry k sobě navíc vlastní zemnící vodič. Toto zároveň ukazuje, že rušení CAN sběrnic bylo souhlasného charakteru a šířilo se pravděpodobně po napájecí síti formule. Výsledkem je unifikované schéma rozhraní CAN, které používají všechny týmem vyvinuté jednotky. Na obrázku 5.11 je schéma zapojení izolace. Existují i CAN budiče, které mají izolaci integrovanou, avšak značné skladové zásoby běžných budičů a dostupnost izolovaných DC/DC měničů od sponzora zvýhodňují naše řešení. Tento přístup také snižuje množství speciálních komponent používaných v týmu, protože stejné izolované zdroje a digitální izolátory jsou pak použity i v jiných aplikacích.



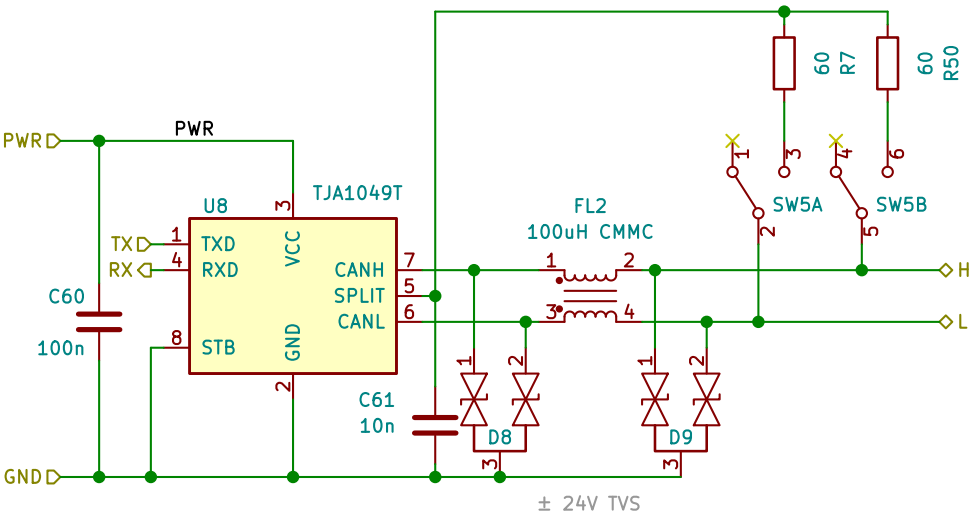
Obrázek 5.11: Schéma izolace rozhraní CAN.

1. O izolované napájení se stará integrovaný DC/DC měnič U6, který je napájený z 3,3 V větve a na izolované straně poskytuje napětí 5 V. Izolační pevnost zdroje zde není příliš důležitá, protože galvanické oddělení slouží pouze k odstranění zemních smyček. Rozdíl potenciálů mezi zemí napájení a zemí CAN sběrnic je zde minimální, ideálně nulový.

2. Digitální signály CAN RX a TX jsou izolovány obvodem U7. Zároveň je tento digitální izolátor použit k překladu napěťových úrovní. Mikrokontrolér pracuje s napěťovými úrovněmi 3,3 V a budiče na izolované straně s úrovněmi 5 V. Opět jeho izolační pevnost není příliš podstatná a proto

také používáme verzi v úzkém pouzdře SOIC-16.

3. schéma CAN budičů je podrobněji na obrázku 5.12.

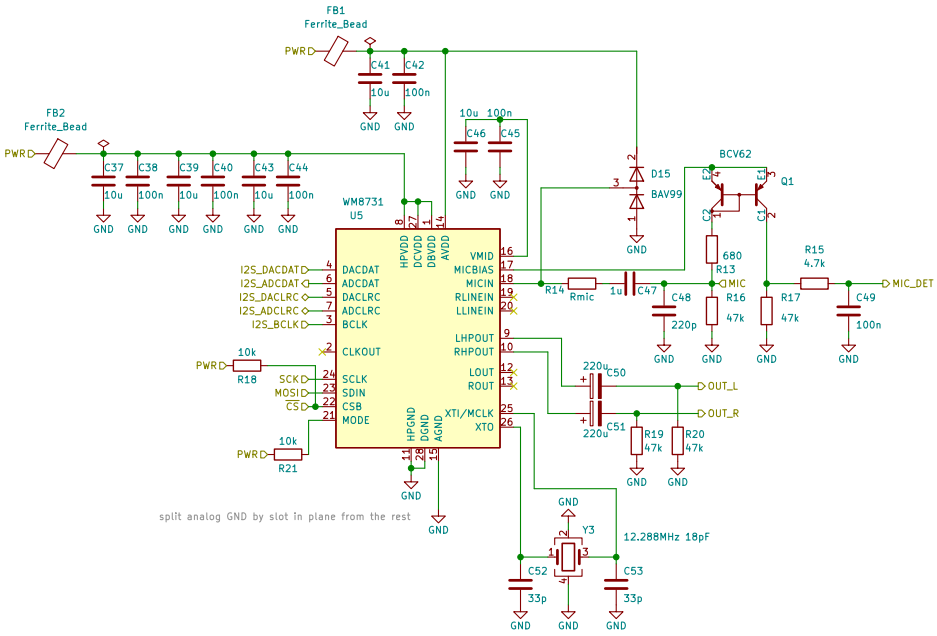


Obrázek 5.12: Schéma budiče rozhraní CAN.

Budiče používané v eForce jsou TJA1049T nebo TJA1057. Oba typy jsou vhodné pro automotive prostředí, snesou napětí na terminálech **CANL** a **CANH** až ±42 V. I tak jsou pro posílení ochrany proti elektrostatickým výbojům (Electrostatic Discharge (ESD)) přidány bipolární transily D8 a D9. Indukčnost FL2 potlačuje souhlasné rušení diferenciálního signálu [4]. Spínačem SW5 pak lze připojit ke sběrnici terminační odpory. Tato funkce je výhodná při testování jednotek mimo auto.

5.2.3 Audio kodek

Dalším možným využitím telemetrického spoje do budoucna je i obousměrný přenos hlasové komunikace. Proto návrh obsahuje i audio kodek s koncovým zesilovačem. Schéma zapojení kodeku je na obrázku 5.13 a z většiny vychází z doporučení od výrobce [9]. Úpravou je přidané proudové zrcadlo Q1, které napájecí proud tekoucí do mikrofonu přes port **MIC** zrcadlí do rezistoru R17, na kterém se promítne jako měřitelný úbytek napětí. Toto zapojení má za cíl eliminovat potřebu pomocného kontaktu v konektoru headsetu k detekci jeho připojení.

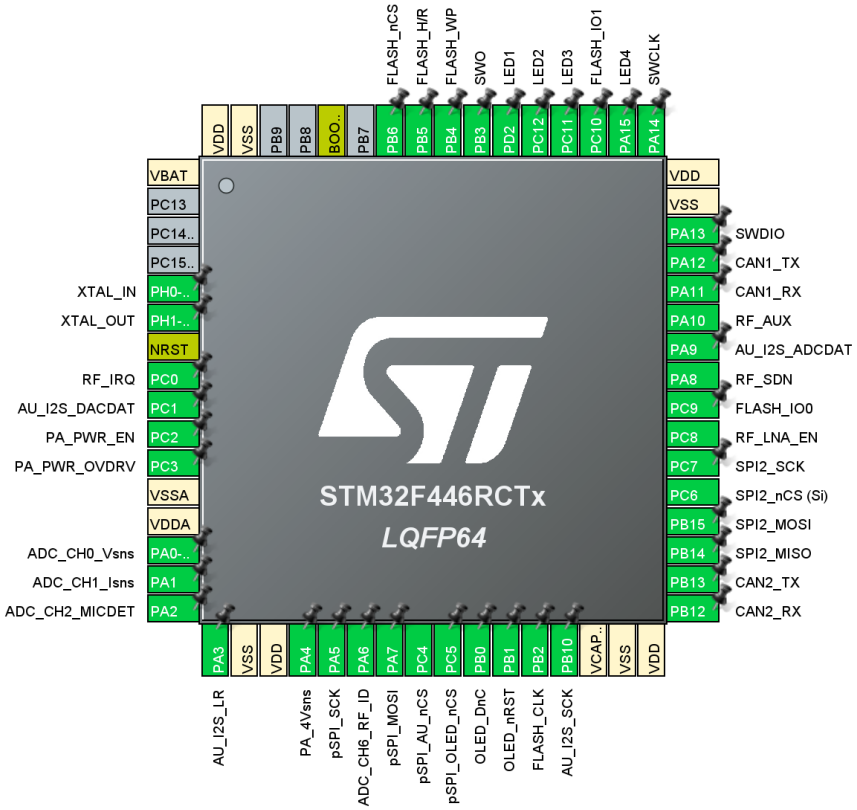


Obrázek 5.13: Schéma zapojení audio kodeku.

5.2.4 Mikrokontrolér

Vybraný mikrokontrolér STM32F446RC[8] od firmy ST-Microelectronics disponuje velkým množstvím periferií. Při plánování zapojení pinů jsem použil nástroj STM32CubeMX. Tento nástroj bere v potaz možnosti konfigurace všech pinů a pomáhá hledat jejich alternativní umístění. Při hrubé představě o rozložení komponent na desce si lze takto zjednodušit práci při následném routování cest. Hotový pinout je vidět na obrázku 5.14. Primární funkcí STM32CubeMX je generování kódu pro nastavení periferií, ale tuto funkci jsem nevyužil, více v kapitole 6.2. Mapování jsem pak 1:1 zreplikoval ve schématu.

Na obrázku 5.15 je vidět zapojení mikrokontroléru. V datasheetu [8] doporučuje výrobce ke každému napájecímu pinu umístit blokovací 100 nF kondenzátor. Další lokalizovanou zásobu energie pro pokrytí špiček v odběru poskytuje kondenzátor C13. Napájení mikrokontroléru je odděleno feritovou perličkou FB6 pro omezení šíření rušení po sdílené napájecí větvi. Obdobně je samostatně filtrováno napájení pro AD převodník na portu **VDDA**. Konektor J1 slouží k programování a ladění programu mikrokontroléru. Jeho pinout je v eForce též unifikován aby se předešlo výrobě a pak možné záměně různých programovacích kabelů. Signály v konektoru J1 jsou chráněny proti ESD vysokorychlostní ESD ochranou D1. I když mikrokontrolér obsahuje vnitřní zdroj hodin v podobě RC oscilátoru, jeho stabilita a přesnost je limitovaná. To může působit problémy na asynchronních sběrnicích jako je CAN nebo Universal Asynchonous Receive Transmit (UART). Hodiny lze přepnout na externí zdroj, kterým je zde krystal Y1. Libovolný zdroj hodin umí vnitřní



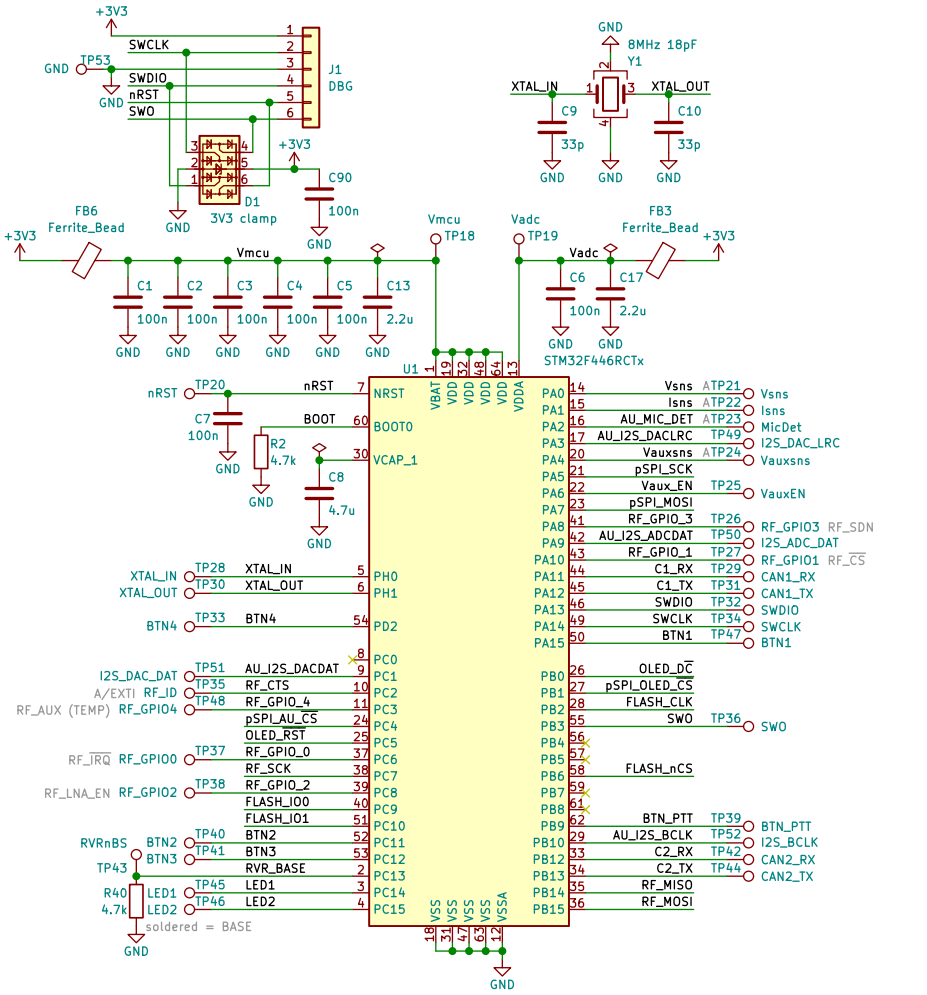
Obrázek 5.14: Mapování funkcí půnu v STM32CubeMX.

blok PLL vynásobit pro vyšší pracovní frekvenci, která u tohoto mikrokontroléru činí 180 MHz [1]. Jedním z rozdílů mezi pozemní jednotkou a mobilní jednotkou ve formuli, je osazení nebo neosazení odporu R40. Tímto způsobem firmware pozná o jakou jakou stranu spoje se jedná a podle toho může upravit své schování.

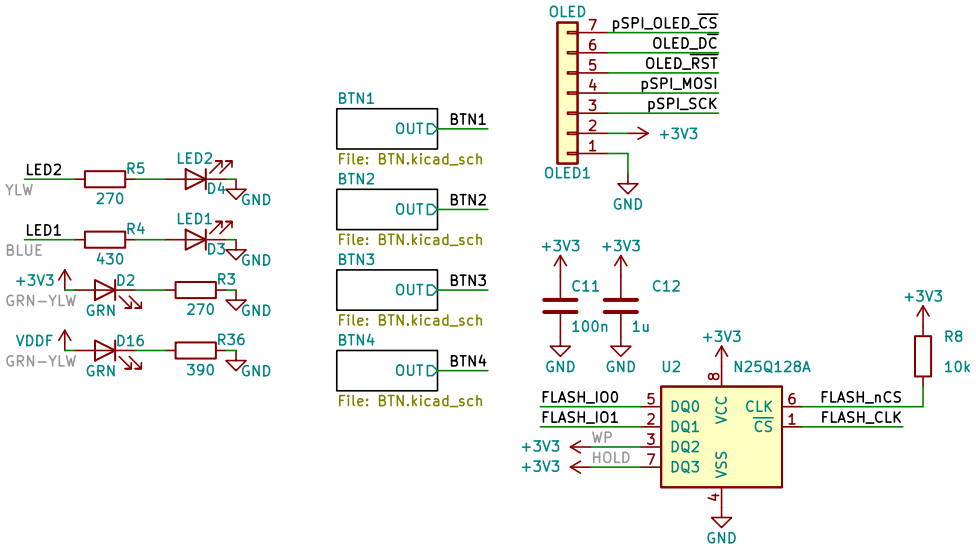
5.2.5 Ostatní periferie

Kromě dříve zmíněných periferií je na ní ještě několik podpůrných obvodů, které nemusí být osazeny. Jedná se o tlačítka, která mohou vystupovat z krabičky. Indikační Light Emitting Diode (LED), 2 na napájecích větvích a 2 řízené mikrokontrolérem a flash paměť na sběrnici QSPI, viz obrázek 5.16.

[1] Nedopatřením jsem mikrokontrolér používal na frekvenci 240 MHz bez znatelného dopadu na spolehlivost. Avšak bylo tomu za pokojových podmínek a výrobce za funkčnost mimo doporučení rozsah frekvencí neručí.



Obrázek 5.15: Schéma zapojení mikrokontroléru.

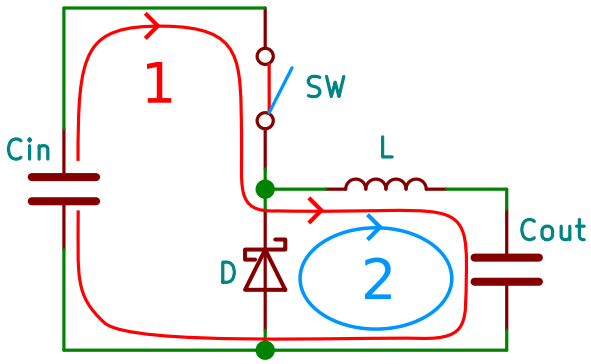


Obrázek 5.16: Schéma ostatních periferních obvodů.

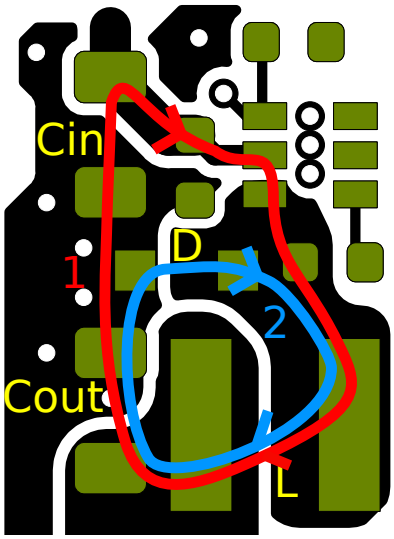
5.2.6 Návrh DPS

Většina komponent na základní desce není příliš citlivá na motiv DPS. Výjimku tvoří jen spínané zdroje. Pro minimalizaci vyzařování spínaných zdrojů je při návrhů důležitá minimalizace plochy proudových smyček. Použitý snižující spínaný zdroj pracuje ve dvou fázích, viz obrázek 5.17. Poměr doby trvání těchto fází určuje výstupní napětí (za předpokladu fixních parametrů ostatních součástek).

1. V první fázi, je spínač sepnut a proud z vstupního kondenzátoru Cin jím prochází dále skrz cívku L až do výstupního kondenzátoru Cout a zpět. Proud cívkou L roste a kondenzátor Cout se nabíjí.
2. V druhé fázi, je spínač rozpojen a proud z cívky se uzavírá přes diodu D1. Cívka při tom ztrácí energii, kterou předává výstupnímu kondenzátoru Cout.



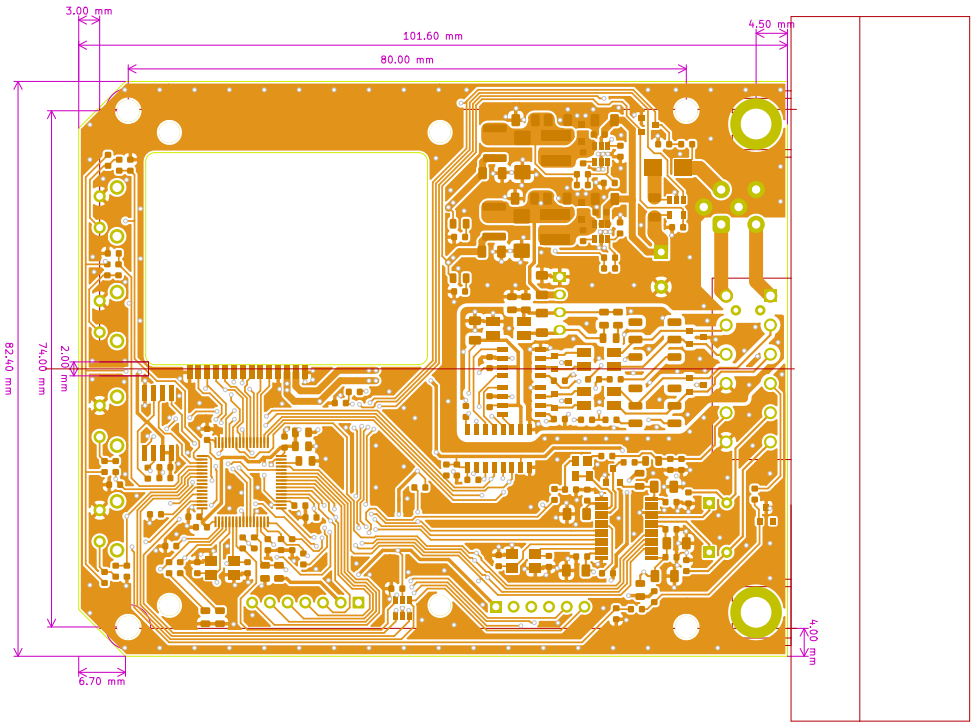
Obrázek 5.17: Dvě fáze činnosti DC/DC zdroje.



Obrázek 5.18: Proudové smyčky na DPS.

V našem případě je spínač realizován tranzistorem a integrován v obvodu AOZ1282CI-1 [13], který zajišťuje jeho řízení. Na obrázku 5.18 je vidět detail proudových smyček na desce. Zobrazená oblast má v reálných rozměrech přibližně 10 mm na šířku.

Desku jsem navrhoval jako dvouvrstvou. Nebyly zde žádné požadavky na specifickou impedanci cest a lze tak snížit náklady a zrychlit výrobu. Pro zachování minimální indukčnosti zemnící cesty jsem rozlil zemnící polygon po celé spodní vrstvě. Polygon je narušen jen minimálně a to na místech, kde bylo potřeba v signálové vrstvě křížit signály. Náhled horní vrstvy je na obrázku 5.19

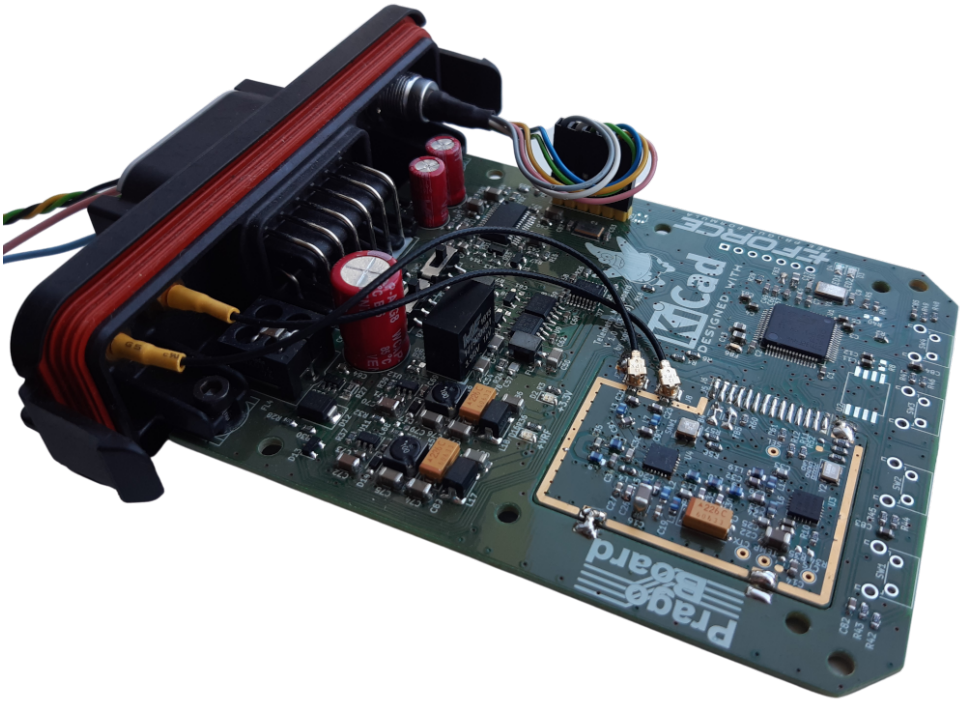


Obrázek 5.19: Horní vrstva základní desky.

5.3 Zástavba

Z důvodu praktičnosti a ochrany desek jsem vybral automotive krabičku tolerantní k hrubšímu zacházení. Všechna elektronika je při neosazení tlačítek vodotěsně chráněna před okolními vlivy. Originální čelo s konektorem nepočítá s VF konektory. Čelo jsem upravil a umístil na něj SMA konektory, které jsou napojeny přímo na VF desku. Na druhé straně čela je navíc programovací konektor. Pokud tedy není vyžadován mechanický zásah, může i pro přeprogramování zůstat krabička uzavřena. Detail montáže je na obrázku 5.20. Na stejném obrázku je i vidět umístění VF modulu. Ten je v základní desce zapuštěný to výřezu, který odpovídá tvaru modulu a je připájen na několika

místech po obvodu a cínovými propojkami propojen ze zbytkem elektroniky. Kompletní jednotka zastavěná v krabičce je na obrázku 5.21.



Obrázek 5.20: Hotové desky přimontované k čelu.

Obrázek 5.21: Pohled na výslednou jednotku v krabičce.

Kapitola 6

Firmware

Tato kapitola se firmwarwm jednotky telemetrie, vysvětluje použité technologie, popisuje strukturu a funkce.

6.1 C++

Po několika letech zkušeností s psaním firmwaru pro embedded zařízení v jazyce C jsem se rozhodl přejít na jazyk, který by nabídl vyšší bezpečnost kódu a odstranil většinu nedostatků kódu psaného v jazyce C. Jazyk Rust [1] se nabízí jako logická možnost. Jedná se moderní jazyk zaměřený na rychlost a implicitní bezpečnost ve vícevláknových aplikacích. Je podporován na mnoha mikrokontrolérech včetně STM32. Rust má ale zcela jinou syntaxi než jazyk C a i odlišný styl přístupu. Z časových důvodů jsem se nakonec uchýlil k alternativě a pro tuto aplikaci zvolil C++. Oproti C má mnohé výhody a syntakticky je s ním v mnoha ohledech kompatibilní. Standard jazyka C++20 přinesl mnoho funkcí užitečných při práci nejen s embeded. Zmínit lze například std::span.

6.1.1 Použité techniky

Jazyk C++ se vyznačuje mnoha technikami a nástroji pro lepší a bezpečnější správu zdrojů jako jsou chytré ukazatele, šablony, Resource Acquisition Is Initialization (RAII) a další.

Šablony

Šablony, anglicky templates je způsob jazyka C++, jak psát obecný kód nezávislý na typu dat se kterými pracuje. Během kompilace vytvoří kompilátor

[1] `https://www.rust-lang.org/`

potřebné specializace pro datové typy, které šablonu používají. Například lze takto napsat jedinou implementaci digitálního filtru a až dle použití se provede specializace pro konkrétní datové typy. Ukázka šablonové funkce pro generování bitové masky je v ukázce kódu 1

```
template<typename T>
constexpr T fillFromLSB(std::uint8_t bits) {
    // kontrola předpokladů datového typu
    static_assert(std::is_integral_v<T>,
        "T must be integral type");
    static_assert(not std::is_signed_v<T>,
        "T must be unsigned type");
    if (bits >= (sizeof(T) * 8)) { return -1; }
    return ((static_cast<T>(1)) << bits) - 1;
}

// ověření funkčnosti několika testy
static_assert(fillFromLSB<std::uint32_t>(0) == 0,
    "fillFromLSB broken");
static_assert(fillFromLSB<std::uint32_t>(9) == 0x000001FFUL,
    "fillFromLSB broken");
static_assert(fillFromLSB<std::uint32_t>(32) == 0xFFFFFFFFUL,
    "fillFromLSB broken");
static_assert(fillFromLSB<std::uint32_t>(33) == 0xFFFFFFFFUL,
    "fillFromLSB broken");

// použití v kódu
auto mask = fillFromLSB<std::uint32_t>(23);
// ...
```

Listing 1: Ukázka šablony v C++.

constexpr

Velké množství funkcí a dat splňující některé požadavky, včetně konstruktorů lze označit klíčovým slovem `constexpr`. Takto označené funkce a data může kompilátor vykonat už při kompilaci a v binárním firmwaru se již ani nemusí objevit. V ukázce 1 je takto označena funkce *fillFromLSB*. Důsledkem je možnost kontrolování chování takových funkcí již při kompilaci jak je vidět na několika řádcích `static_assert` v ukázce.

RAII

RAII je technika pro zprávu různých zdrojů v závislosti na životnosti objektu. Takto technika je hojně využita pří řízení přístupu ke sdíleným zdrojům jako

```
// bez použití RAII
Context context {};
Ar::Mutex access{};

void reset() {
    mutex.get(); // manuální zamknutí mutexu
    if (context.rxQueue.empty()) {
        mutex.put();
        // nutné manuální uvolnění mutexu
        // u všech návratů z funkce
        return;
    }
    context.rxQueue.clear();
    context.lastCreatedPacketId = 0;
    context.lastPoppedPacketId = 0;
    mutex.put(); //manuální uvolnění mutexu
}
```

Listing 2: Ukázka sdíleného přístupu bez RAII.

jsou různé zásobníky, fronty ale i periferie jako SPI nebo Direct Memory Access (DMA). Srovnání je ukázáno v úryvcích kódu 2, 3 4.

- Ukázka 2 bez RAII je nejzákladnější způsob realizace přístupu. Programátor musí ručně zamknout i odemknout mutex. Nesmí zapomenout ani na jedno. U delších funkcí s více místy návratu lze tak jednoduše zapomenout na odemknutí mutexu.
- V ukázce 3 se již RAII využívá a to konkrétně u objektu *mtx*. Jeho konstruktor zamyká mutex *access* a jeho destruktor při návratu z libovolného místa funkce zase mutex uvolňuje. Nemůže se tedy stát, že programátor zapomene mutex uvolnit.
- V posledním případě 4 je navíc s metodou RAII využita šablona třídy, která zprostředkovává přístup k objektu a automaticky zamyká i odemyká mutex.

```
// při použití RAII
Context context {};
Ar::Mutex access{};

void reset() {
    Ar::ScopedMutexAcquisition mtx(access);
    // vznik objektu mtx zamkne mutex access
    if (context.rxQueue.empty()) {
        //object mtx zaniká
        // a v jeho destruktoru se odemkne mutex
        return;
    }
    context.rxQueue.clear();
    context.lastCreatedPacketId = 0;
    context.lastPoppedPacketId = 0;
    //object mtx zaniká (končí jeho scope)
    // a v jeho destruktoru se odemkne mutex
}
```

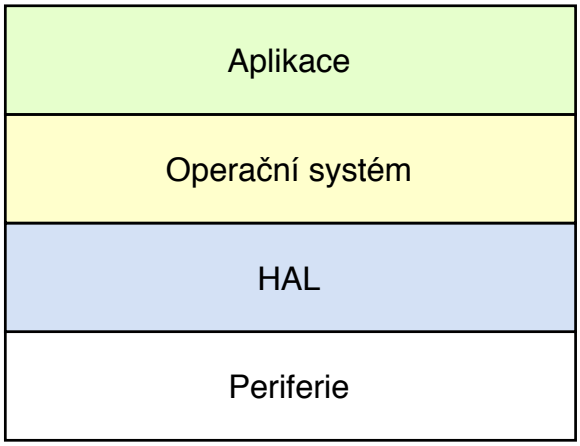
Listing 3: Ukázka sdíleného přístupu s RAII.

```
// použití pomocné šablony specializované pro typ Context
Ar::ProtectedAccess<Context> context {};

void reset() {
    // objekt ctx je instancí třídy zprostředkovávající
    // přístup ke sdilenému zdroji
    // vznik objektu ctx zamkne mutex
    // uvnitř objektu context
    auto ctx = context.access();

    // přítup k datům třídy Context
    // probíhá přes proxy objekt ctx
    if (ctx->rxQueue.empty()) {
        // object ctx zaniká
        // a v jeho destruktoru se odemkne mutex
        return;
    }
    ctx->rxQueue.clear();
    ctx->lastCreatedPacketId = 0;
    ctx->lastPoppedPacketId = 0;
    // object ctx zaniká (končí jeho scope)
    // a v jeho destruktoru se odemkne mutex
}
```

Listing 4: Ukázka sdíleného přístupu s RAII a proxy objektem.

6.2 HAL

Hardware Abstraction Layer je sada knihoven poskytující vrstvu abstrakce na přímým přístupem k periferiím mikrokontroléru. Často má také jednotné rozhraní napříč různými mikrokontroléry stejného výrobce. Většina výrobců ke svým mikrokontrolérům poskytuje právě svoje HAL knihovny. Typické uspořádání abstrakčních vrstev systému je na obrázku 6.1. Operační systém nemusí být vždy přítomný.



Obrázek 6.1: Abstrakční vrstvy firmwaru.

Pro mikrokontroléry STM32 existuje starší HAL knihovna Standard Peripheral Library (STL) někdy také nazývána STDperiph. Pro nové rodiny mikrokontrolérů není již vyvíjena a je tak na ústupu. Je charakteristická spíše menší úrovní abstrakce a vývojář je tak při psaní kódu blíže k periferiím. Nástupcem STL je nová knihovna s názvem HAL[2]. Poskytuje uživateli vyšší abstrakci a pro nízkoúrovňový přístup k periferiím používá další knihovnu LL.

Existují i opensource knihovny se se stejným cílem. Jejich zástupcem je například libopencm3[3]. Projekt nabízí abstrakci nad periferiemi a to i napříč mikrokontroléry od různých výrobců založených na jádrech Cortex-M.

Při psaní firmwaru pro telemetrii jsem se snažil o co nejvíce kódu napsaného v C++ kvůli plnému využití jeho možností. Zároveň psaní nízkoúrovňových knihoven, například pro přístup k periferiím patří k mým zájmům. Ve výsledku jsem nepoužil ani jednu z implementací HAL knihoven a pro využívané periferie jsem napsal vlastní.

[2] Ano, název této implementace HAL knihovny je HAL.
[3] `https://libopencm3.org/`

6.3 Komponenty

6.3.1 Externí knihovny

ETL

Neboli Embedded Template Library[4] je C++ šablonová knihovna zaměřená na omezené zdroje embedded zařízení. Nabízí různé šablony s kontejnery, algoritmy a dalšími nástroji kompatibilními s šablonami ze standardní šablonové knihovny STL jazyka C++. Hlavním rozdílem je pak fixní velikost kontejnerů a tedy absence dynamické alokace. Rozšířena je o kontejnery vhodné pro komunikaci mezi běžným vláknem a rutinou obsluhy přerušení, které lze využít třeba jako frontu zpráv CAN rozhraní.

printf

Jedná se o alternativní implementaci [5] funkce printf ze standardní knihovny s cílem minimalizace velikosti a poskytnutí nastavitelnosti podporovaných funkcí. Programátor pak její výstup může přesměrovat na libovolné rozhraní jako je UART, Serial Wire Output (SWO) nebo i CAN.

CMSIS

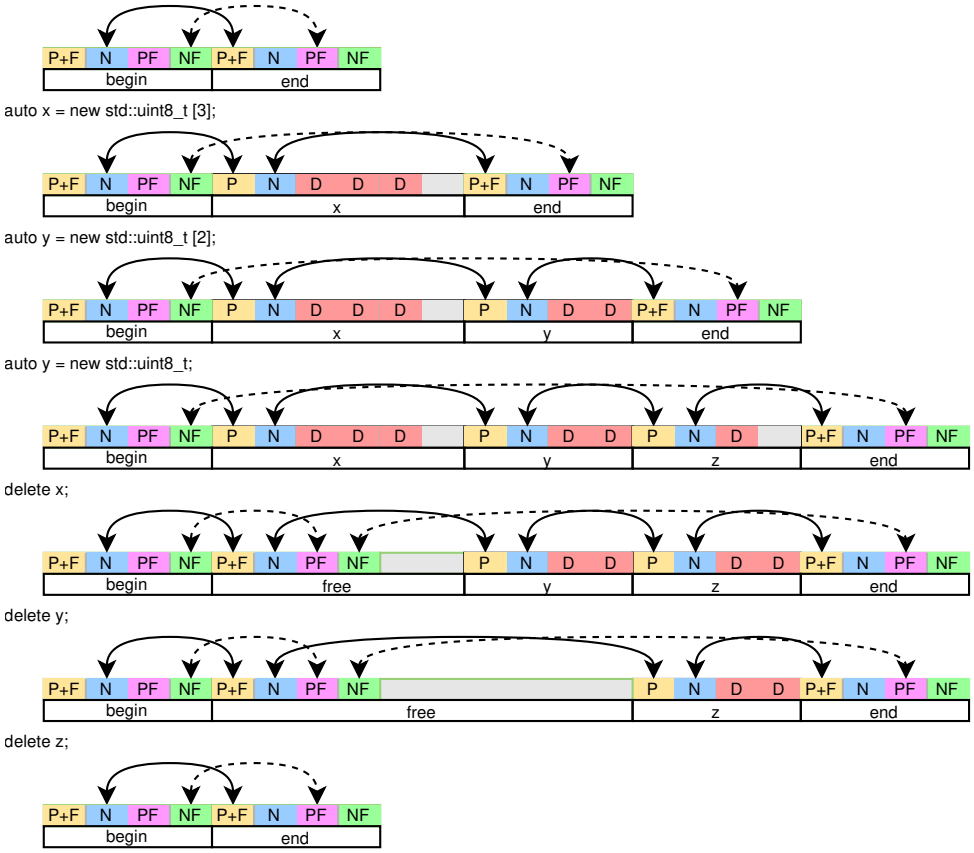
Mnozí výrobci mikrokontrolérů nevyvíjí vlastní jádra a licencují do svých mikrokontrolérů jádra Arm. Samotný Arm dodává knihovny pro zjednodušení přístupu k funkcím jako je integrovaní kontrolér přerušení Nested Vector Interrupt Controller (NVIC). Součástí knihovny CMSIS jsou také optimalizované implementace algoritmů jako maticové operace, digitální filtry nebo volání instrukcí Single Instruction Multiple Data (SIMD) pro paralelní výpočty.

6.3.2 xalloc

I když se dynamické alokaci paměti v embedded systémech většina vývojářů vyhýbá, je velmi užitečným způsobem jak pracovat s daty bez předem známé velikosti. Dynamickou alokaci vyžadují i kontejnery ze standardní knihovny C++. Kompilátor poskytuje dynamický alokátor, který běžně všechny alokované bloky zarovnává na hranici 16 B. Vlastní implementace s názvem xalloc, má minimální zarovnání 4 B a při požadavku dokáže alokovaný blok zarovnat na libovolnou mocninu 2 B větších než 4. Alokované bloky připomínají

[4] https://www.etlcpp.com/
[5] https://github.com/mpaland/printf

"double linked list"neboli po sobě jdoucí bloky mají vždy odkaz na předchozí a následující blok. Kvůli ušetření paměti nejsou v xalloc odkazy realizované jako ukazatele, které mají na 32 bitové platformě velikost 4 B, ale pouze jako pozice indexovaná od počátku paměti přidělené xallocu 16 bitovým číslem s krokem nejmenšího možného bloku, tj. 8 B. Taktovéto adresování omezuje maximální velikost dynamicky alokované oblasti, ale pro embedded zařízení je často dostačující. Při požadavku na větší dynamicky alokovanou oblast lze zvětšit velikost bloků. Pro rychlý pohyb po volných blocích si tyto bloky na místě dat drží odkaz na další a předchozí volné bloky. Obrázek 6.2 ukazuje chování xallocu při alokaci a dealokaci paměti. Obrázek je pouze schematický a nerespektuje minimální zarovnání.



Obrázek 6.2: Příklad průběhu dynamické alokace paměti.

Čárkované šipky znázorňují vzájemné odkazy volných bloků. Šipky plnou čarou pak vzájemné spoje libovolných bloků.

- P(+F) - odkaz na předchozí blok, +F indikuje, že daný blok je označen jako volný.
- N - odkaz na další blok
- PF - odkaz na předchozí volný blok, pouze u volných bloků

- NF - odkaz na následující volný blok, pouze u volných bloků
- D - data

1. Paměť alokátoru obsahuje pouze počáteční (begin) a koncový (end) prvek. Ty jsou vždy přítomny.
2. Při požadavku na alokaci 3 bytů, začne xalloc prohledávat volné bloky od počátečního. Pohybuje se pouze po odkazech na volné bloky PF a NF. V tomto případě nenajde vhodný volný blok a dorazí na konec. Vytvoří posunutý nový koncový blok a do mezery v paměti umístí alokovaný blok. Následné upraví odkazy aby ukazovaly na platné pozice. Kvůli požadavku o velikosti, která není násobkem minimálního zarovnání je na konci alokované paměti volné místo. Úpravy odkazů provádí alokátor po každém zásahu do paměti a dále nebudou zmiňovány.
3. Při další alokaci je algoritmus stejný, akorát velikost alokovaného bloku je násobkem minimálního zarovnání a nevzniká tedy žádný nevyužitý prostor.
4. Tato alokace je svým chováním stejná jako v bodě 2.
5. Při dealokaci paměti je blok označen jako prázdný a v případě dalších volných bloků v okolí je s nimi sloučen. Zde žádný takový není.
6. Další dealokace paměti, ale tentokrát se volné bloky sloučily do jednoho.
7. Po dealokaci posledního bloku dat zjistí alokátor, že mezi počátečním a koncovým blokem je jediný blok a ten je prázdný. V důsledku toho přesune koncový blok hned za počáteční

Implementace alokátoru xalloc má několik pravidel při hledání vhodného bloku pro nově alokovaná data s cílem minimalizovat fragmentaci paměti.

6.3.3 RTOS

Během psaní firmwaru jsem narazil na situaci kdy program musel čekat na událost. Aby nedocházelo k blokování celého programu při čekání na tuto událost, musí být program napsán tak, že se obsluhy všech částí programu volají ve smyčce periodicky. Informace o tom, kde minule obsluha skončila, neboli kontext, se nejčastěji udržuje ve formě stavového automatu. Tento přístup se stává neúměrně komplikovaný u částí programů, kde jsou stavů desítky nebo by musely obsahovat více stavových automatů. Alternativním řešením neblokujícího běhu je nasazení Real Time Operating System 6.3.3 (RTOS). Operační systém může přepínat mezi vlákny programu podle jejich priority nebo podle jejich požadavků na buzení a uspávání, tzv. multitasking. Lze tak psát program, který při čekání na událost uspí vlákno a poskytne procesorový čas ostatním vláknům do doby, než je uspané vlákno odblokováno očekávanou událostí.

RTOS podporující architekturu Arm je velké množství. Z nejvíce zastoupených lze zmínit FreeRTOS[6], ChibiOS[7], nuttx[8], Zephyr[9], eCos[10], CMSIS-RTOS[11]. Ani jeden neposkytuje C++ API a ani není v C++ napsán. Většina již s sebou nese svoji implementaci nebo ST HAL knihovny.

Nakonec jsem zvolil RTOS jménem Argon[12], se kterým jsem již delší dobu experimentoval. Jedná se minimalistickou implementaci v C++, která zajišťuje multitasking a poskytuje několik užitečných funkcí jako jsou mutexy, semafory, fronty, kanály a další. Nenese s sebou žádný HAL ani jiné komponenty specifické pro konkrétní Micro-Controller Unit - (mikrokontrolér) (MCU).

6.3.4 Ovladač Si4463

Transceiver Si4463 je komplexní obvod s velkým množstvím nastavení a příkazů. Ovládá se pomocí 29 příkazů a má přes 170 nastavovaných parametrů. Výrobce poskytuje grafický nástroj Wireless Development Suite (WDS), který běží pouze pod operačním systémem Windows, ve kterém lze navolit parametry spoje a ten následně vygeneruje blok dat, který se po startu nahraje do transceiveru.

Součástí je i ovladač transceiveru, který kromě inicializace pomocí vygenerovaného bloku dat řídí i další funkce transceiveru. Nevýhodou tohoto přístupu je, že při každé změně nastavení je nutné nové vygenerování kódu nástrojem WDS. Samotný ovladač od výrobce je napsaný v jazyce C a je svojí strukturou a přístupem spíše optimalizovaný na 8 bitové mikrokontroléry.

Po zvážení výhod i nevýhod softwaru od výrobce jsem se rozhodl napsat vlastní ovladač v jazyce C++. Cílem bylo poskytnout praktické rozhraní pro posílání libovolných příkazů nastavování všech parametrů.

Dokumentace Application Programming Interface (API) se všemi příkazy a nastaveními je poskytována ve formě html stránky. Ruční přepis všech těchto informací by zabral neúnosně dlouhou dobu a nepochybně by zanesl těžko odhalitelné chyby do kódu. Proto jsem vytvořil skript, který projde celou html stránku a vygeneruje potřebné struktury dat reprezentující všechny příkazy a nastavení. Ikdyž jsem většinu generování zautomatizoval, ruční post-processing byl pořád nezbytný. Důvodem je fakt, že dokumentace je pravděpodobně generována z dat, která stále psal člověk.

Implementace ovladače je zaměřena na vysokou optimalizaci již během kompilace. Nepřidává tedy příliš režie oproti řešení od výrobce a umožňuje jednoduchou změnu nastavení transceiveru za běhu. Viz příklad nastavení

[6] https://www.freertos.org/
[7] https://www.chibios.org/
[8] https://nuttx.apache.org/
[9] https://www.zephyrproject.org/
[10] https://www.ecoscentric.com/ecos/
[11] https://arm-software.github.io/CMSIS_5/RTOS2/html/index.html
[12] https://github.com/flit/argon-rtos

parametrů části paketu 5. Klíčové slovo `constexpr` vynucuje vykonání konstruktoru již při kompilaci a za běhu má program k dispozici už sestavený blok dat reprezentující požadované nastavení. Ve velké míře jsou k dosažení tohoto výsledku nasazeny šablony.

Pokud nejsou všechny parametry známy při kompilaci, provede se konstruktor a tedy i převod do hrubých dat až za běhu programu.

```
{
    using namespace Si446x::PROPERTIES::PKT_CONFIG1;
    constexpr auto prop = Payload(
        BIT_ORDER::MSBIT_FIRST,
        CRC_ENDIAN::MSBYTE_FIRST,
        CRC_INVERT::NO_INVERT,
        MANCH_POL::PATTERN_10,
        FSK4::ENABLE,
        PH_RX_DISABLE::RX_ENABLE,
        PH_FIELD_SPLIT::FIELD_SHARED);
    if (auto result = si.setProperty(prop); !result) { xassert(); }
}
```

Listing 5: Ukázka nastavení parametrů transceiveru.

Přestože lze všechno nastavení provést pomocí nového ovladače, malá část inicializace pomocí bloku dat zůstala. Jedná se hlavně o nahrání binární záplaty pro mikrokontrolér uvnitř transceiveru a konfiguraci parametrů modemu, ke kterému výrobce neposkytuje dostatečnou dokumentaci a nevysvětluje jejich optimální nastavení.

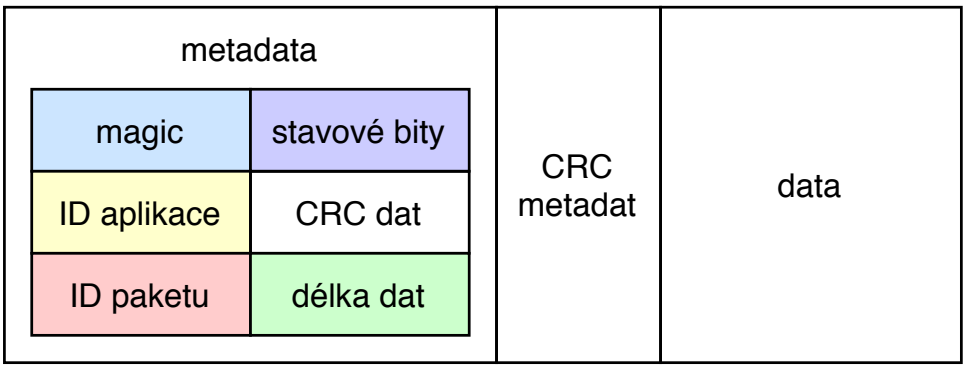
Komunikace s transceiverem probíhá formou požadavek - čekání - odpověď, kdy informaci o připravené odpovědi lze získat buďto cyklickým čtením stavu (polling), což je neefektivní a blokuje program, nebo čekáním na změnu stavu signálu Clear To Send (CTS), při kterém se vlákno ovladače uspí a probudí jej až přerušení od změny signálu CTS.

6.4 Paketizace

Celý komunikační protokol ja navržen tak, aby přidávání dalších zdrojů dat znamenalo minimální nebo žádné změny v kódu. Protokolová vrstva se snaží imitovat vlastnosti Transmission Control Protocol (TCP) a User Datagram Protocol (USP) spojení v Internet Protocol (IP) sítích. Například přenos CAN zpráv vyžaduje zachování pořadí a doručení všech zpráv, oproti tomu přenos zvuku kódovaného například kodekem Opus [25] je tolerantní k výpadku paketů.

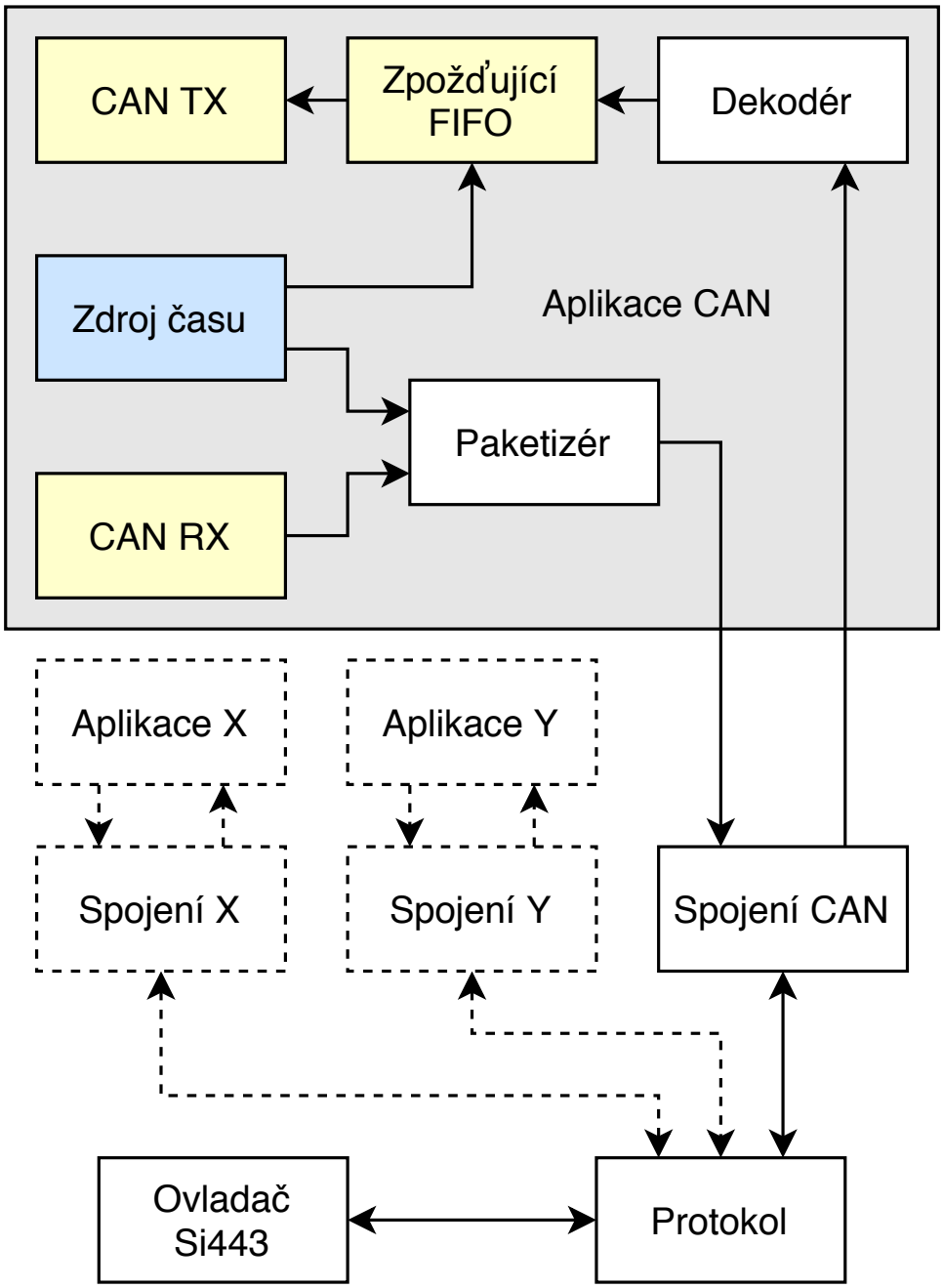
Transceivery řady Si446x dokáží najednou přenést rámec proměnlivé délky

o maximální velikosti 8191 B. Při komunikaci nejčastěji nastane situace, kdy je ve frontě dat k přenesení několik menších bloků. Kvůli maximalizaci využití datového toku je výhodné více těchto paketů spojit do rámce až do jeho maximální délky. Struktura jednoho paketu je vidět na obrázku 6.3.



Obrázek 6.3: Struktura paketu.

- Magic - dohodnutá konstanta pro základní ověření validity paketu.
- ID Aplikace - rozlišuje pakety od různých aplikace jako je přenos CAN zpráv nebo přenos zvuku.
- ID paketu - číslo určující pořadí paketu. Slouží pro zachování pořadí paketů při typu přenosu, když je požadované doručení všech paketů a ve správném pořadí.
- Stavové bity - obsahují informace o typu paketu a stavu spojení.
- CRC dat - kontrolní součet datové oblasti
- délka dat - určuje délku datového bloku, slouží také k dopočítání začátku dalšího paketu při deserializaci
- CRC metadat - kontrolní součet bloku metadat
- data - blok dat proměnlivé délky



Obrázek 6.4: Blokové schéma spojové části firmwaru.

Na obrázku 6.4 je schématicky znázorněno členění firmwaru zajišťující spojení. Jednotlivé bloky jsou detailněji popsány níže.

6.4.1 Protokol

Tento blok je centrálním bodem celého paketizačního řetězce. Udržuje si seznam aktivních spojení a monitoruje stav spojení s protistranou. Pokud dojde k výpadku spojení delší než zvolená doba, dojde k resetu všech spojení. Odesílané pakety všech spojení se v zde finalizují, tj výpočet kontrolních součtů CRC. Příchozí pakety se zde zase ověřují a třídí.

Při příjmu dostává blok Protokol segmenty dat od ovladače transceiveru a postupně dochází k rekonstrukci celých paketů. Pokud má přijatý paket poškozena metadata, což je detekováno při neshodě jejich kontrolního součtu CRC, je zbytek přijímaných dat ignorován až do konce aktuálního přenosu, protože nelze s jistotou určit začátek dalšího paketu. Jestliže jsou metadata v pořádku a je poškozen jen datový blok, dojde k zahození aktuálního paketu, ale příjem dalších paketů z aktuálního rámce transceiveru pokračuje dál. Validní pakety jsou předány jednotlivým spojením.

Při vysílání se nejdříve ovladač transceiveru dotáže bloku Protokol na délku dostupných dat připravených k odeslání. Poté dávkově předává data ovladači transceiveru, který periodicky plní vnitřní First In First Out (FIFO) zásobníky v transceiveru.

Další funkcí bloku Protokol je potvrzování přijetí paketů, u kterých je to patřičným bitem vyžádáno. K tomuto řízení toku existuje speciální typ paketu, který potvrzuje přijetí paketů z předchozího přenosu v opačném směru a to pomocí párů ID aplikace a ID paketu. Zároveň tento paket nese informaci o počtu paketů v celém přenosovém rámci a počtu těch, které budou vyžadovat potvrzení. Potvrzené pakety jsou vymazány z odesílací fronty.

6.4.2 Spojení

Každá aplikace má svoji instanci bloku Spojení. Při odesílání paketu tento blok označí paket identifikátorem příslušné aplikace, vypočítá ID paketu a předá jej bloku Protokol. Při obdržení paketu jej zařadí do své fronty přijatých paketů a vygeneruje událost o přijetí, která může probudit vlákno čekající na data. Pokud je u spojení vyžadováno zachování pořadí paketů, je událost vygenerována, až přijde paket, který má správné ID v sekvenci.

6.4.3 Aplikace

Pro demonstraci je v obrázku 6.4 nastíněna aplikace přenosu CAN zpráv. Bloky příjmu a časovaného odesílání CAN zpráv jsou součástí ovladače CAN periferie a jsou zobrazeny pro úplnost. Paketizér aplikace plní pakety přijatými zprávami a předává je k odeslání. Dekodér naopak přijímá pakety a sestavuje z nich jednotlivé CAN zprávy. U přijatých zpráv upravuje časy odeslání, taky aby byla ve zpožďujícím FIFO bufferu dostatečná zásoba paketů k odeslání

během krátkých výpadků. Vysílané CAN zprávy mají stejné časování jako u odesílatele, jen jsou časově zpožděné.

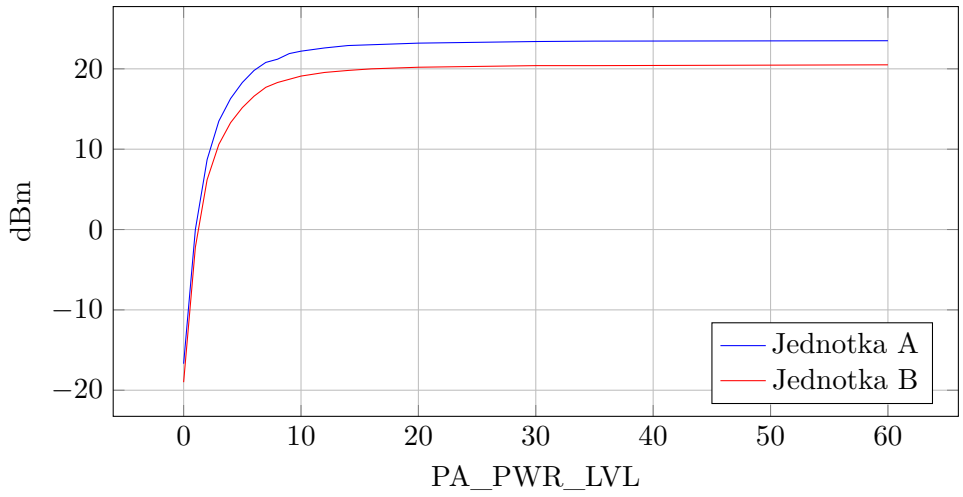
Kapitola 7

Měření spoje

Tato kapitola pojednává o změřených vlastnostech spoje a porovnává je s očekávanými parametry.

7.1 Výstupní výkon

Jedním z prvních měření, která jsem provedl po oživení hardwaru bylo zjištění převodu mezi parametrem transceiveru *PA_PWR_LVL* nastavující výstupní výkon a reálným výstupním výkonem z celé jednotky telemetrie. Měření byla provedena spektrálním analyzátorem ROHDE & SCHWARZ FSP 1093.4495.03. Výsledek měření je v grafu 7.1.



Obrázek 7.1: Závislost měřeného výstupního výkonu na parametru transceiveru *PA_PWR_LVL*.

I když jsou VF moduly identické a byly osazovány současně, vznikl mezi nimi značný rozdíl. Lze pozorovat, že mezi vysílanými výkony je rozdíl přibližně 3 dB, tj. výstupní výkon *Jednotky A* je dvojnásobný oproti *Jednotce*

B. Tolerance parametrů pasivních součástek mohou způsobit velké rozdíly v chování filtrů a impedančních přizpůsobení. I přesto se rozdíl 3 dB jeví jako značný a bude vyžadovat další analýzu.

Maximální dosažený výkon u *Jednotky A* dosahuje 23,5 dBm. Teoretické maximum koncového zesilovače ve frontendu [18] je 27 dBm, což je o 3,5 dB více než maximální dosažený výkon. V měřícím řetězci figurovaly tyto komponenty: VF modu, konektor U.fl, 10 cm koaxiálního kabelu součástí pigtailu, SMA konektor, 1 m kabelu RG316 a redukce z SMA na konektor N. Útlum kabelu RG316 o délce 1 m je podle katalogu [11] přibližně 1,25 dB. Zbývající útlum 2,25 dB je způsoben buď zbývajícími komponentami v měřícím řetězci nebo nedostatkem na VF modulu. Nalezení problému bude vyžadovat další analýzu.

Přesto, že maximální dosažené výstupní výkony nedosahují očekávaných hodnot, budou pro splnění požadavků spoje postačovat.

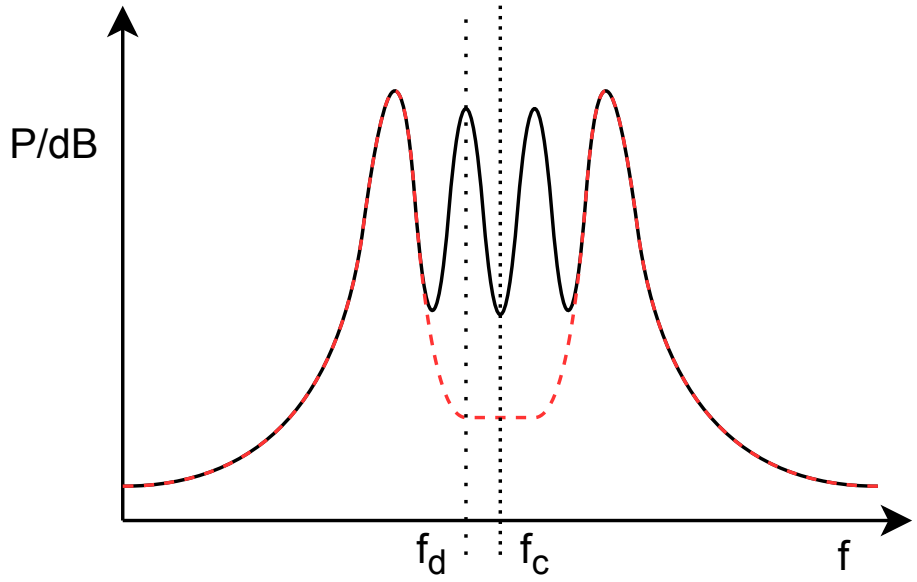
7.2 Datová propustnost

Datová propustnost byla měřena více způsoby. Obě testované modulace pracují na stejné symbolové rychlosti 500 ks/s (maximum transceiveru). Modulace 4GFSK je 4 stavová frekvenční modulace s modulačním kokem 83,3 kHz $f_s = f_c - f_d$, tj. frekvenčním rozdílem nejbližších symbolů od nosné frekvence. Díky čtyřem rozdílným stavům kóduje 2 bity do 1 symbolu a má tedy dvojnásobnou datovou propustnost, tj 1000 kbit/s. Na obrázku 7.2 plnou černou čarou. Modulace 2GFSK využívala stejné nastavení jako 4GFSK, může mít však pouze 2 stavy a tedy přenáší 1 bit na symbol. Na obrázku 7.2 přerušovanou červenou čarou.

- Test A - rámce transceiveru vždy naplněny na maximální hodnotu transceiveru a přeneseny. Přijatá data na druhé straně byla ignorována. Takto se oba konce spoje střídaly.
- Test B - podobný testu A avšak propustnost je vypočtena pouze s bloků o velikosti 500 B, které byly přeneseny bez chyby.
- Test C - byla přenášena data s využitím řízení toku popsaném v kapitole Protokol 6.4.1. Výsledný datový tok byl aproximován jako dvojnásobek datového toku přijatých neporušených paketů s předpokladem, že chyby postihují stejně oba směry přenosu.

Z dat testů v tabulce 7.1 je vidět, že s modulací 2GFSK lze dosáhnout vyššího poměru reálného datového ku maximálnímu. Při nasazení kontroly toku byla propustnost limitována malou pamětí mikrokontroléru, ve které musí pakety čekat až do jejich potvrzení a někdy dochází k jejímu zaplnění.

Při modulaci 4GFSK je poměr reálné propustnosti ku nominální nižší. Důvodem může být vyšší režie kolem paketizace a vyšších nároků na paměť. I při krátké komunikační vzdálenosti se u modulace 4GFSK projevovala poměrně vysoká chybovost paketů. Pří posílání paketů o velikosti 1 kB byla chybovost Packet Error Rate (PER) přibližně 5 %. Jednotlivé symboly mají mezi sebou menší frekvenční rozestup než při modulaci 2GFSK a může snáze docházet k jejich záměně.



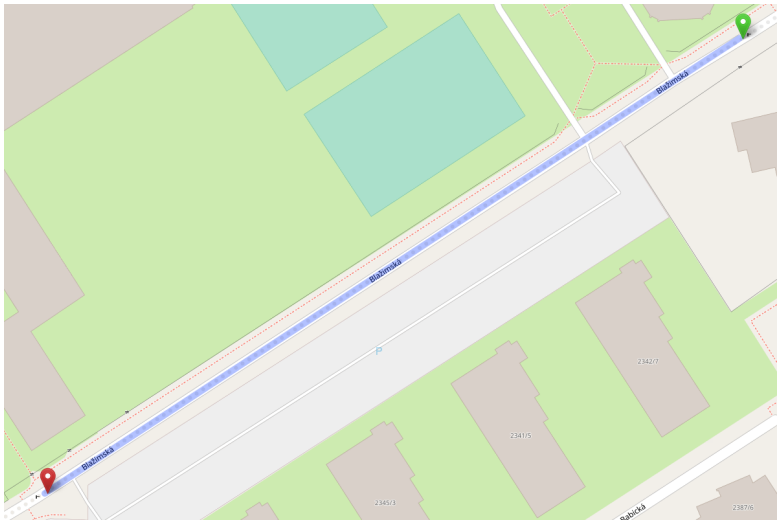
Obrázek 7.2: Teoretické výkonové spektrum modulací 4GFSK a 2GFSK.

Modulace	A [kbit/s]	Bg[kbit/s]	C [kbit/s]
2GFSK	489	490	416
4GFSK	896	881	742

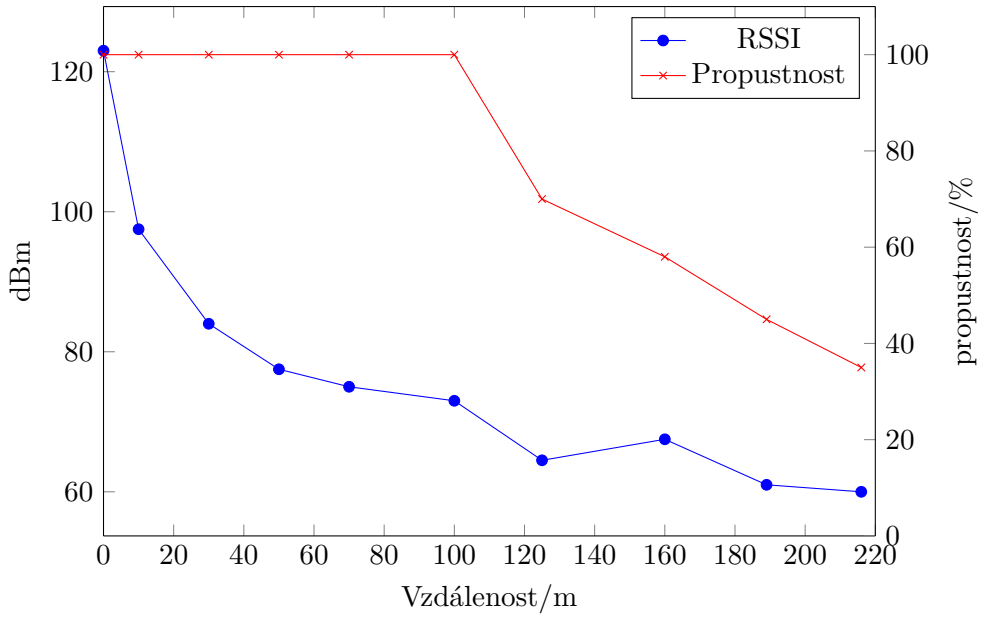
Tabulka 7.1: Datové propustnosti spoje.

7.3 Dosah

Jako poslední bylo provedeno měření dosahu bezdrátového spoje. Kvůli problémům s modulací 4GFSK na krátké vzdálenosti byly parametry měřeny pouze s modulací 2GFSK. Posílány byly rámce obsahující jediný paket o velikosti dat 34 B + 4 B CRC. Obě jednotky se střídaly maximální rychlostí a saturovaly tak spoj. Výstupní výkon byl 20 dBm a byly použity antény o zisku 3 dBi. Samotné měření probíhalo na uzavřeném úseku silnice na obrázku 7.3.



Obrázek 7.3: Mapa úseku měření.



Obrázek 7.4: Závislost RSSI a propustnosti spoje na vzdálenosti

Poznámka: absolutní honodta RSSI není kalibrována.

V grafu 7.4 lze pozorovat skokový pokles propustnosti jakmile hodnota RSSI klesla pod přibloně 70 dBm za vzdáleností 100 m. Přestože množství přenosů začalo klesat, ty které se uskutečnily, přenesly paket bez poškození. Tento fakt indikuje, že nenastávají chyby při demodulaci, nýbrž se synchronizací kdy přijímač začínal mít problém detekovat vysílanou preambli. Vhodnou úpravou parametrů transceiveru by mělo jít tento jev minimalizovat.

Kapitola 8

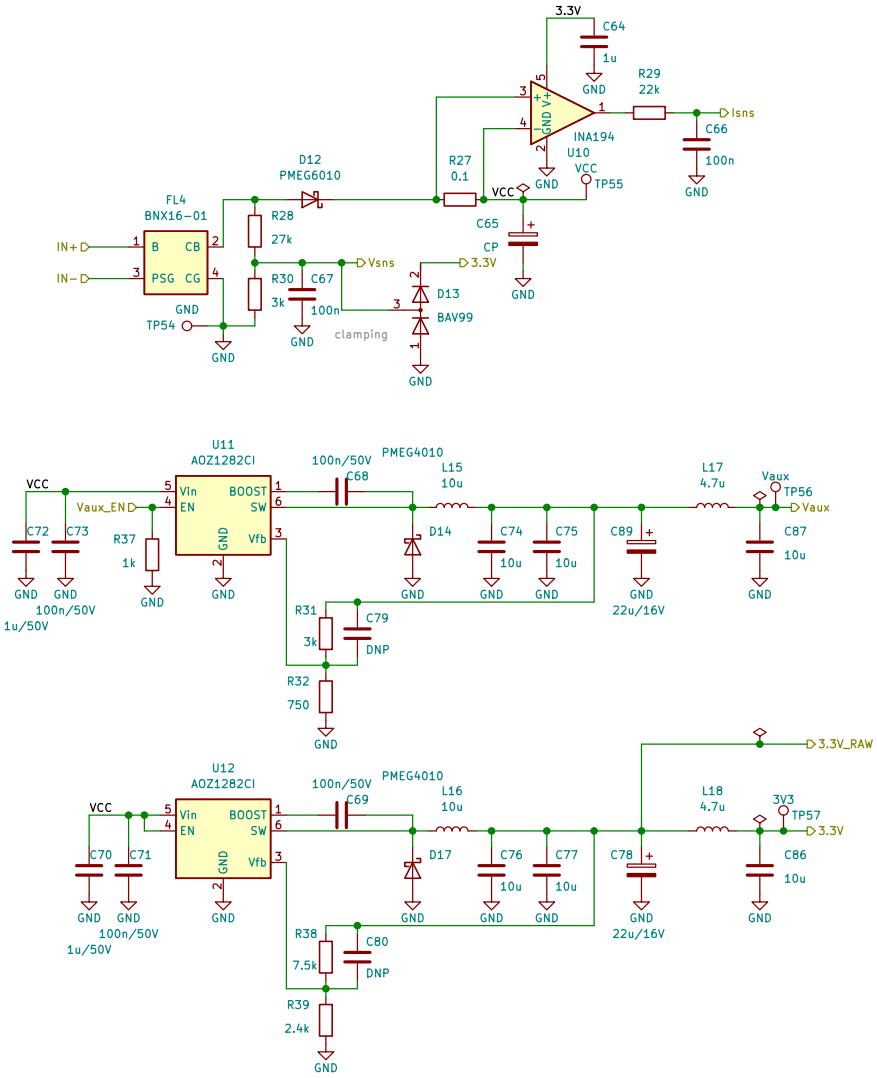
Závěr

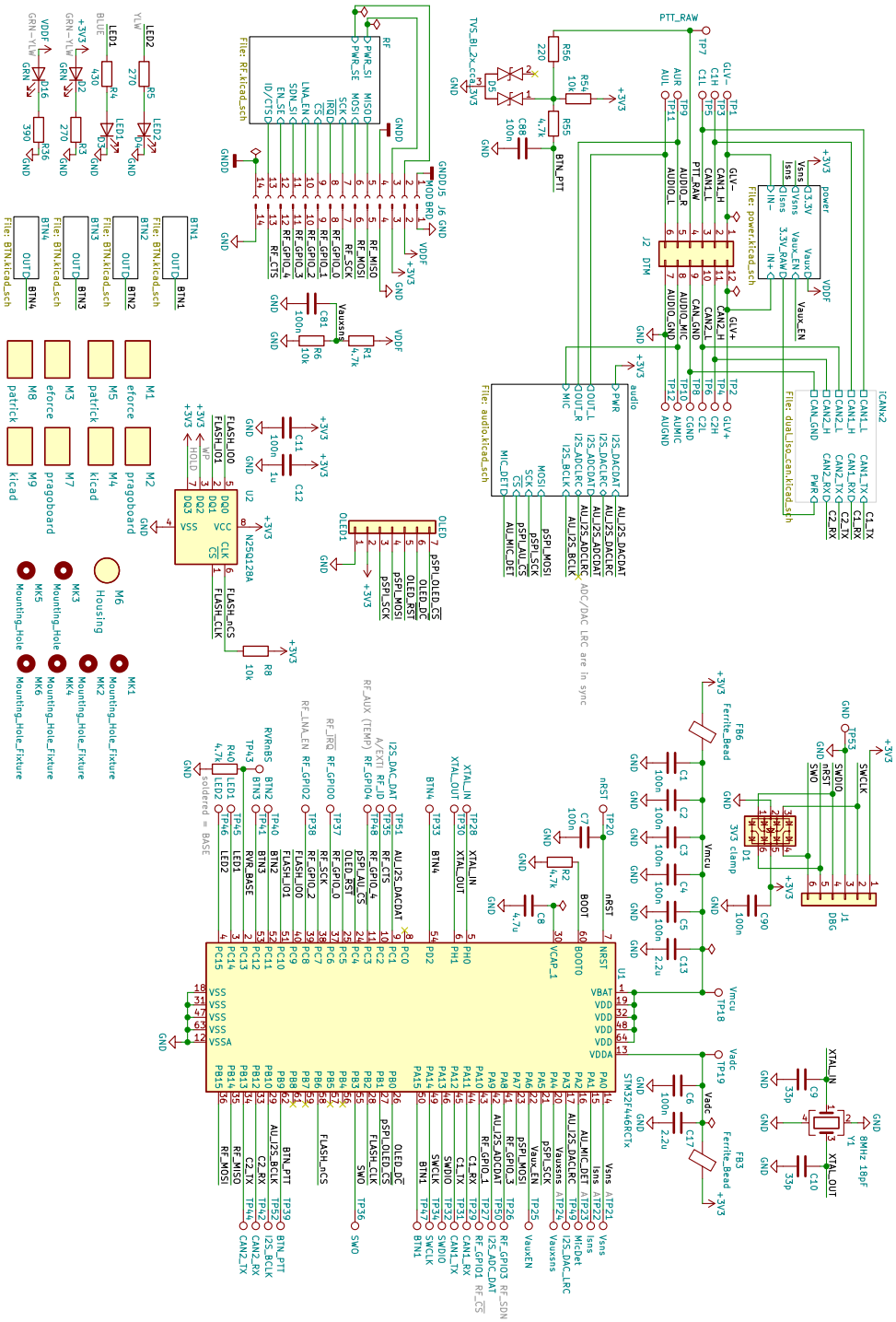
Cílem práce bylo analyzovat možnosti realizace bezdrátového spoje pro vozidlo typu Formula Student Electric. Na základě vstupních požadavků, parametrů dostupných komponent a předešlých zkušeností byly vybrány komponenty systému tvořené samostatným, transceiverem Si4463 [17] s externím přepínačem [18] a řízeným mikrokontrolérem STM32F446 [8]. Práce ukázala realizaci hardwaru a poukázala na problematické části návrhu. Desky jsou navrženy s cílem modulárnosti a testu s alternativními komponentami do budoucna. Velké úsilí bylo věnováno vývoji firmwaru zajišťujícího spolehlivý přenos dat. Při vývoji byly použity moderní technologie v podobě jazyka C++ ve standardu 20 a nasazení RTOS pro efektivnější vývoj a využití procesorového času. Během vývoje a měření byly nalezeny některé nedostatky zmíněné v kapitole *Měření spoje*7. Jejich vyřešení bude vyžadovat další práci a vývoj jednotek spoje. Po odstranění těchto nedostatků bude v blízké době spoj otestován na elektrické formuli týmu eForce. Následně bude implementována a vyhodnocena použitelnost spoje pro přenos zvuku. Pro autora byla toto první zkušenost s vývojem vysokofrekvenční elektroniky a návrhem bezdrátového spoje což se také projevilo na nedostatcích VF části elektroniky.

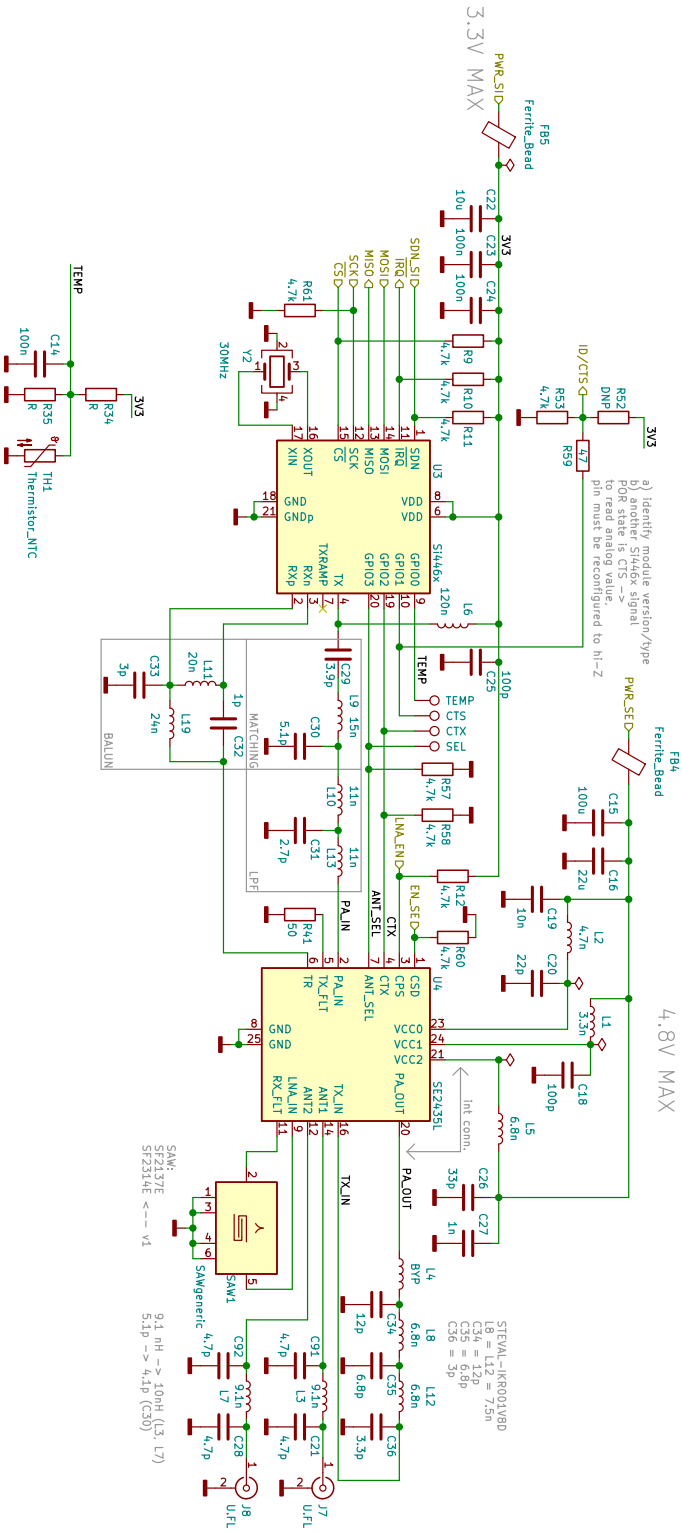
Tato práce ukázala, že možných řešení spoje existuje několik. Do budoucna by bylo vhodné analyzovat možnosti technologie Wi-Fi a porovnat se současným řešením.

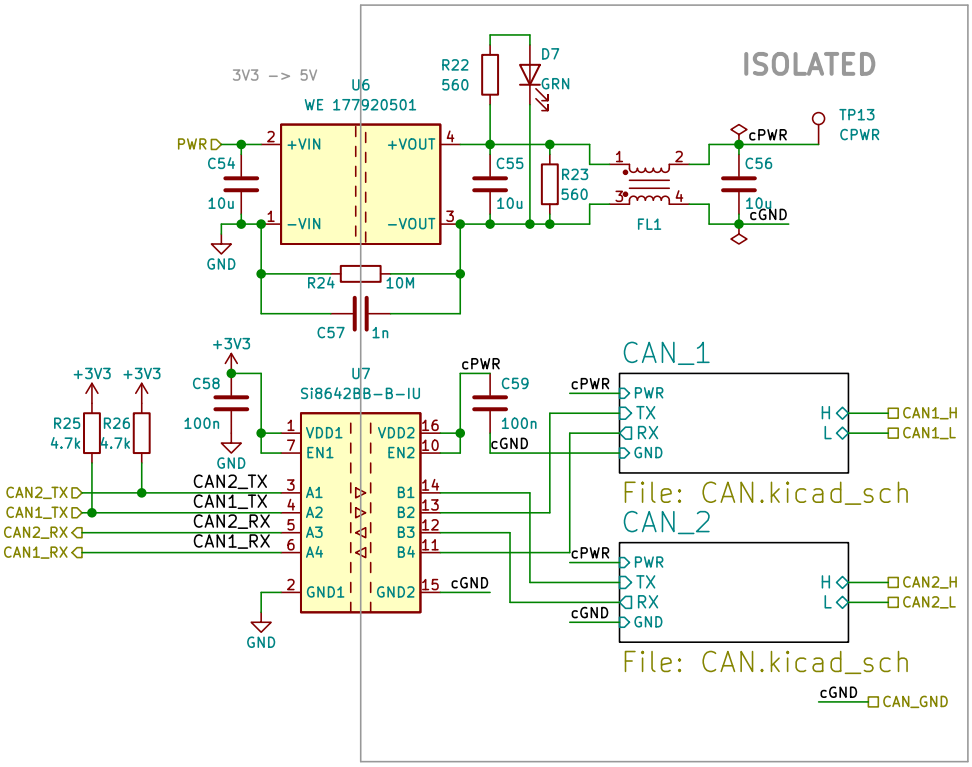
Příloha A

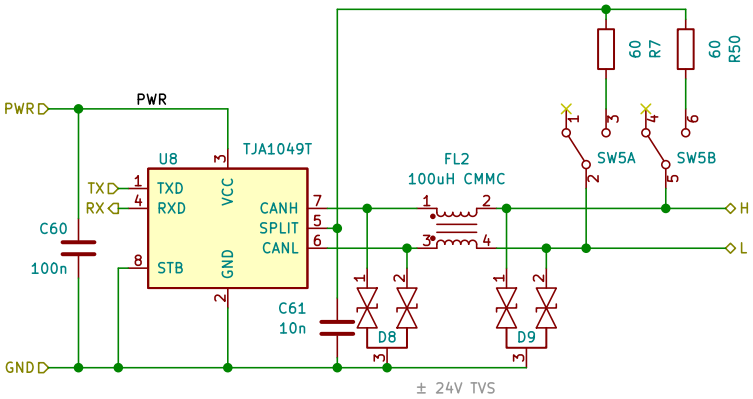
Schémata

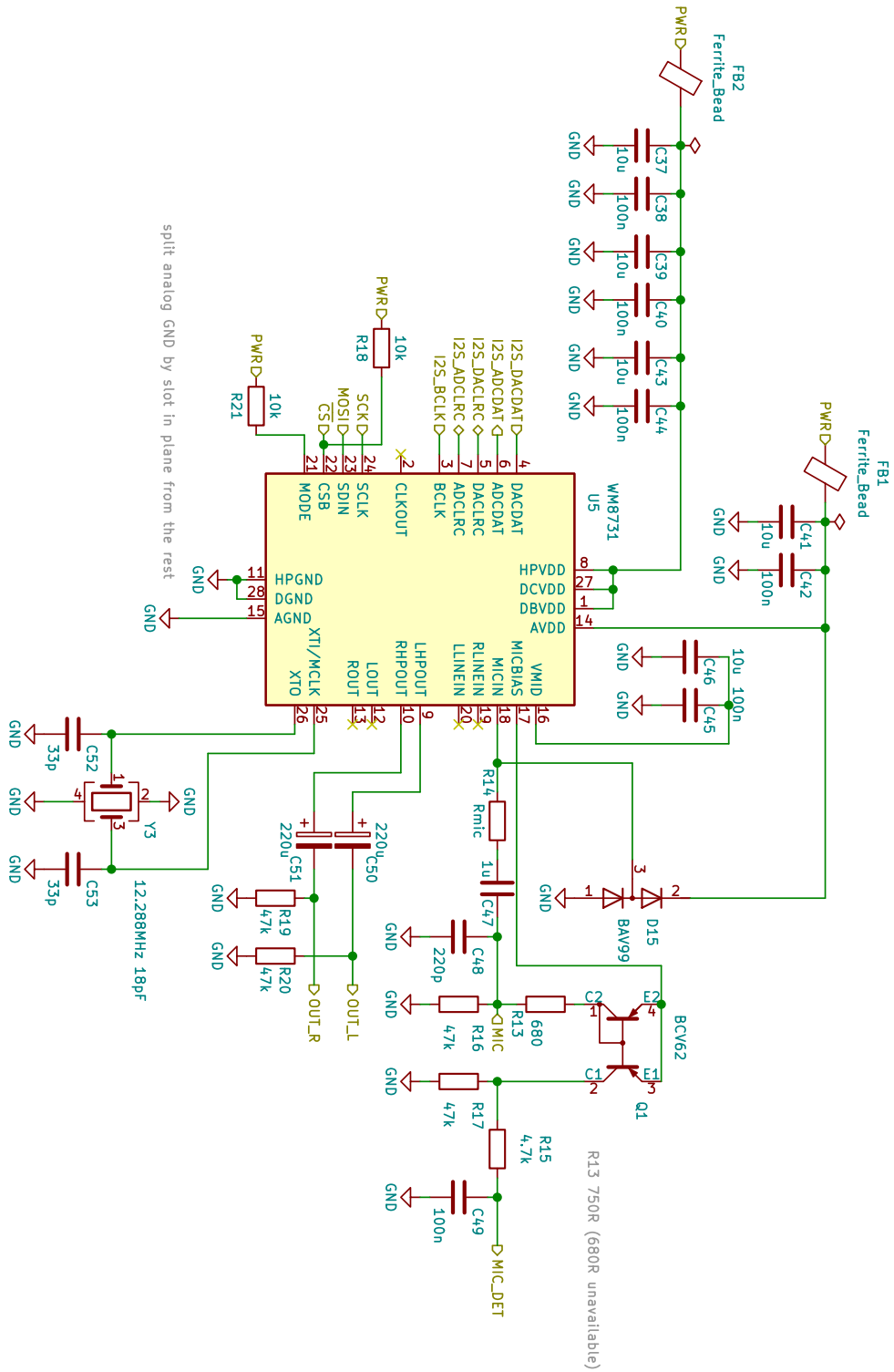





3.3V MAX
4.8V MAX
PWR_SIG
PWR_SEL
FB5
FB4
Ferrite_Bead
3V3
TEMP
ID/CTS
SDN_SIG
IRQ
MOSI
MISO
SCK
CS
30MHz
Y2
TH1
Thermistor_NTC
U3
Si446x
U4
SE2435L
XOUT
XIN
GND
GNDp
VDD
SDN
GPIO0
GPIO1
GPIO2
GPIO3
TX
TXRAMP
RXn
RXp
CSD
CPS
CTX
ANT_SEL
PA_IN
TX_FLT
TR
VCC0
VCC1
VCC2
PA_OUT
TX_IN
ANT1
ANT2
LNA_IN
RX_FLT
EN_SE
LNA_EN
BALUN
MATCHING
LPF
int conn.
SAW1
SAWgeneric
SAW:
SF2137E
SF2314E <--- v1
9.1 nH -> 10nH (L3, L7)
5.1p -> 4.1p (C30)
STEVAL-IKR001V8D
L8 = L12 = 7.5n
C34 = 12p
C35 = 6.8p
C36 = 3p
J8
J7
U.FL
a) identify module version/type
b) another Si446x signal
POR state is CTS -->
to read analog value,
pin must be reconfigured to hi-Z


ISOLATED
3V3 -> 5V
U6
WE 177920501
+VIN
+VOUT
-VIN
-VOUT
PWR
GND
C54
10u
C55
10u
R22
560
D7
GRN
R23
560
FL1
C56
10u
cPWR
cGND
TP13
CPWR
R24
10M
C57
1n
+3V3
C58
100n
U7
Si8642BB-B-IU
VDD1
EN1
GND1
VDD2
EN2
GND2
C59
100n
cPWR
cGND
R25
4.7k
R26
4.7k
CAN2_TX
CAN1_TX
CAN2_RX
CAN1_RX
A1
A2
A3
A4
B1
B2
B3
B4
CAN_1
PWR
TX
RX
GND
H
L
CAN1_H
CAN1_L
File: CAN.kicad_sch
CAN_2
CAN2_H
CAN2_L
CAN_GND


PWR
U8
TJA1049T
TXD
RXD
VCC
CANH
SPLIT
CANL
STB
GND
TX
RX
C60
100n
C61
10n
FL2
100uH CMMC
SW5A
SW5B
R7
60
R50
60
D8
D9
± 24V TVS


WM8731
U5
split analog GND by slot in plane from the rest
R13 750R (680R unavailable)
12.288MHz 18pF
Y3
BAV99
D15
BCV62
Q1
FB1
Ferrite_Bead
FB2
Ferrite_Bead
PWR
GND
MIC_DET
OUT_L
OUT_R
MIC

Příloha B

Seznam zkratek

ADC Analog to Digital Converter (A/D převodník). 15

API Application Programming Interface. 51

CAN Controller Area Network. ix, 2–10, 15, 17, 20, 21, 32–35, 48, 52, 53, 55, 56, 69

CAN FD CAN Flexible Datarate. 8

CTS Clear To Send. 52

DAC Digital to Analog Converter (D/A převodník). 15

DMA Direct Memory Access. 15, 45

DPS Deska Plošných Spojů. ix, 13, 14, 20, 28, 38

EMI Electromagnetic Interference. 33

ESD Electrostatic Discharge. 34, 35

FIFO First In First Out. 55

FPU Floating Point Unit. 21

GCPW Grounded Coplanar Waveguide. 28, 29

GPIO General Purpose Input Output (vstupně výstupní pin). 15, 19, 21

HAL Hardware Abstraction Layer. 17, 47, 51

I2C Inter-Integrated Circuit. 15

I2S Inter-IC Sound. 15, 20, 22

IOT Internet Of Things. 14

IP Internet Protocol. 52

IR Infra Red. 15

LAN Local Area Network. 9, 70

LED Light Emitting Diode. 36

MCU Micro-Controller Unit - (mikrokontrolér). 51

NVIC Nested Vector Interrupt Controller. 48

PER Packet Error Rate. 59

PLL Phase Locked Loop. 18, 36

PWM Pulse Width Modulation (Pulzne šířková modulace). 15

QSPI Quad SPI. 21, 36

RAII Resource Acquisition Is Initialization. 43–45

RTOS Real Time Operating System 6.3.3. 50, 51, 61

SAI Serial Audio Interface. 20, 21

SDIO Serial Data Input Output. 15

SIMD Single Instruction Multiple Data. 48

SPI Serial Peripheral Interface. ix, 9, 10, 15, 17, 19–22, 24, 25, 45, 70

SWO Serial Wire Output. 48

TCP Transmission Control Protocol. 52

UART Universal Asynchonous Receive Transmit. 15, 35, 48

USB Universal Serial Bus. 3

USP User Datagram Protocol. 52

VF VysokoFrekvenční. ix, 13, 23–25, 27–31, 39, 57, 58, 61

WDS Wireless Development Suite. 51

WLAN Wireless LAN. 9

Příloha C

Bibliografie

[1] All3DP. *2020 Best Single Board Computers/ Raspberry Pi Alternatives*. https://all3dp.com/1/single-board-computer-raspberry-pi-alternative/. Accessed on 2020-07-20. 2020.

[2] BOSCH. *CAN Specification*. http://esd.cs.ucr.edu/webres/can20.pdf. Accessed on 2020-08-10. 1991.

[3] eLinux.org. *RPi CANBus*. https://elinux.org/RPi_CANBus. Accessed on 2020-07-20.

[4] Texas Instruments. *Common Mode Chokes in CAN Networks*. https://www.ti.com/lit/an/slla271/slla271.pdf. Accessed on 2020-08-10. 2008.

[5] ITEQ. *ITEQ158 substrate Datasheet*. http://www.iteq.com.tw/wp-content/uploads/2018/01/IT-158-Data-sheet.pdf. Accessed on 2020-08-04. 2017.

[6] ST Microelectronics. *SPIRIT1 Datasheet*. https://www.st.com/resource/en/datasheet/spirit1.pdf. Accessed on 2020-07-20. Zář. 2016.

[7] ST Microelectronics. *SPIRIT1 Schéma desky STEVAL-IKR001V8D*. https://www.st.com/resource/en/schematic_pack/steval-ikr001v8d_schematic.pdf. Accessed on 2020-08-03.

[8] ST Microelectronics. *STM32F446 Datasheet*. https://www.st.com/resource/en/datasheet/stm32f446rc.pdf. Accessed on 2020-07-20. Čvc 2020.

[9] Wolfson microelectronics. *WM8731 Datasheet*. https://statics.cirrus.com/pubs/proDatasheet/WM8731_v4.9.pdf. Accessed on 2020-07-20. Říj. 2012.

[10] NI. *Introduction to 802.11ax High-Efficiency Wireless*. https://www.ni.com/cs-cz/innovations/white-papers/16/introduction-to-802-11ax-high-efficiency-wireless.html. Accessed on 2020-07-20. Červ. 2020.

[11] Pasternack. *RG316 datasheet*. https://www.pasternack.com/images/productpdf/rg316-u.pdf. Accessed on 2020-08-08. 2019.

[12] David M. Pozar. *Microwave Engineering, Fourth edition*. Wiley, 2011.

[13] Alpha & Omega semiconductor. *AOZ1282CI-1 Datasheet*. http://www.aosmd.com/res/data_sheets/AOZ1282CI-1.pdf. Accessed on 2020-08-05. 2015.

[14] NORDIC Semiconductor. *NRF24L01+ Datasheet*. https://infocenter.nordicsemi.com/pdf/nRF24L01P_PS_v1.0.pdf. Accessed on 2020-07-20. 2008.

[15] Silabs. *AN648 Si4460/61/63/64/67/68 RF ICs Layout Design Guide*. https://www.silabs.com/documents/public/application-notes/AN629.pdf. Accessed on 2020-08-04. 2014.

[16] Silabs. *AN648 Si4X6X AND EZR32 HIGH-POWER PA MATCHING, rev 0.6*. https://www.silabs.com/documents/public/application-notes/AN648.pdf. Accessed on 2020-08-04. 2016.

[17] Silabs. *Si446x Datasheet*. https://www.silabs.com/documents/public/data-sheets/Si4464-63-61-60.pdf. Accessed on 2020-07-20. 2016.

[18] SKYWORKS. *SE2435L Datasheet*. https://www.skyworksinc.com/-/media/SkyWorks/Documents/Products/601-700/SE2435L_202412K.pdf. Accessed on 2020-07-20. Říj. 2019.

[19] Espressif Systems. *ESP8266EX Datasheet*. https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf. Accessed on 2020-07-20. Čvc 2020.

[20] Espressif Systems. *ESP8266EX Datasheet*. https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf. Accessed on 2020-07-20. Břez. 2020.

[21] TexasInstruments. *CC3220 Datasheet*. https://www.ti.com/lit/ds/symlink/cc3220s.pdf. Accessed on 2020-07-20. Lis. 2018.

[22] Ing. Stanislav Tomášek. *Modulární zařízení pro telemetrii a sběr dat z libovolných systémů*. https://dspace.cvut.cz/handle/10467/68630. Accessed on 2020-07-20. Červ. 2016.

[23] Český telekominukační úřad. *všeobecné oprávnění č. VO-R/10/01.2019-1kvyužívání rádiových kmitočtů akprovozování zařízení krátkého dosahu.* https://www.ctu.cz/sites/default/files/obsah/ctu/vseobecne-opravneni-c.vo-r/10/01.2019-1/obrazky/vo-r10-012019-1.pdf. Accessed on 2020-07-20. Led. 2019.

[24] VECTOR. *P/roduktová stránka VN16xx*. https://www.vector.com/int/en/products/products-a-z/hardware/network-interfaces/vn16xx/. Accessed on 2020-07-20.

[25] Koen Vos. *Voice Coding with Opus*. https://www.opus-codec.org/static/presentations/opus_voice_aes135.pdf. Accessed on 2020-08-07. 2013.

[26] Jerry C. Whitaker. *The Electronics Handbook*. CRC Press, 1996.