Fakulta elektrotechnická
Katedra aplikované elektroniky a telekomunikací

Diplomová práce

Vzdáleně řízená průmyslová komunikační jednotka na platformě STM32

Autor práce: Bc. Lukáš Manda
Vedoucí práce: Ing. Kamil Kosturik, Ph.D.

Plzeň 2020

ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta elektrotechnická

Akademický rok: 2019/2020

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Lukáš MANDA**
Osobní číslo:	**E18N0028P**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Elektronika a aplikovaná informatika**
Téma práce:	**Vzdáleně řízená průmyslová komunikační jednotka na platformě STM32**
Zadávající katedra:	**Katedra aplikované elektroniky a telekomunikací**

Zásady pro vypracování

Navrhněte a realizujete komunikační jednotku (analyzátor průmyslových sběrnic) pro vzdálenou správu a diagnostiku zařízení připojených k sběrnicím RS-232 a CAN. Jednotka bude připojena prostřednictvím rozhraní Ethernet. Pro implementaci použijte mikrokontrolér řady STM32F4xx na jádře ARM Cortex-M4. Pro připojený nadřazený počítač navrhněte aplikaci umožňující monitorování a řízení aktuálního stavu na sběrnicích (příjem a odesílání dat).

1. Seznamte se s architekturou a vývojovými prostředky mikrokontroléru STM32F4xx.
2. Prostudujte specifikace komunikačních sběrnic RS-232, CAN a komunikačních protokolů TCP/IP.
3. Navrhněte hardwarové řešení daného zařízení (analyzátoru).
4. Definujte konfigurační, stavová a datová rozhraní komunikační jednotky. Implementujte programové vybavení mikrokontroléru v jazyce C, případně C++.
5. Navrhněte aplikaci pro nadřazený počítač, která umožní sledovat aktuální dění na sběrnicových rozhraních jednotky.

Zvažte možnosti rozšíření rozhraní jednotky i o jiné sběrnicové standardy.

Rozsah diplomové práce: **40 – 60 stran**
Rozsah grafických prací: **podle doporučení vedoucího**
Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. Dostálek, L., Kabelová, A.: Velký průvodce protokoly TCP/IP a systémem DNS. Computer Press, 2000. ISBN 80-7226-323-4
2. IEEE 802.3 Standard – Part 3 – Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications
3. Kállay, F.; Peniak, P.: Počítačové sítě a jejich aplikace. Grada, 1999. ISBN 80-7169-407-X
4. Pfeiffer, O., et. al.: Embedded Networking with CAN and CANopen. RTC Books, 2003. ISBN 0-929392-78-7
5. CANopen Communication Profile For Industrial System Based on CAL, CiA Draft Standard 301 – Revision 3.0. CAN in Automation, 1996

Vedoucí diplomové práce: **Ing. Kamil Kosturik, Ph.D.**
Katedra aplikované elektroniky a telekomunikací

Datum zadání diplomové práce: **4. října 2019**
Termín odevzdání diplomové práce: **28. května 2020**

Západočeská univerzita v Plzni Fakulta elektrotechnická L.S.

Prof. Ing. Zdeněk Peroutka, Ph.D.
děkan

Doc. Dr. Ing. Vjačeslav Georgiev
vedoucí katedry

V Plzni dne 4. října 2019

Abstrakt

Tato diplomová práce je zaměřena na návrh a realizaci komunikační jednotky pro vzdálenou správu zařízení. Komunikační jednotka je postavena na vývojové desce NUCLEO-F429ZI od výrobce STMicroelectronics. Vytvořená komunikační jednotka umožňuje monitorovat komunikaci na RS-232, RS-485, CAN a přeposílat tuto komunikaci prostřednictvím Ethernetového rozhraní. Začátek práce popisuje základní specifikace komunikačních sběrnic, pokračuje návrhem komunikační jednotky, implementací softwaru pro komunikační jednotku s využitím lwIP TCP/IP a měřením elektromagnetické kompatibility.

Klíčová slova

STM32, lwIP, ARM,komunikační jednotka

Abstract

Manda, Lukáš. *Remote Controlled Industrial Communication Unit on the STM32 Platform* [*Vzdáleně řízená průmyslová komunikační jednotka na platformě STM32*]. Pilsen, 2020. Master thesis (in Czech). University of West Bohemia. Faculty of Electrical Engineering. Department of Applied Electronics and Telecommunications. Supervisor: Kamil Kosturik

---

This diploma thesis is focused on design and realization of communication unit for remote device management. The communication unit is based on the development board NUCLEO-F429ZI from the manufacturer STMicroelectronics. The created communication unit allows to monitor communications of RS-232, RS-485, CAN and forward these communications via Ethernet interface. The beginning of the thesis describes basic specifications of communication buses, continues with design of communication unit, implementation of software for communication unit using lwIP TCP/IP and measurement of electromagnetic compatibility.

Keywords

STM32, lwIP, ARM, Communication Unit

Prohlášení

Předkládám tímto k posouzení a obhajobě diplomovou práci, zpracovanou na závěr studia na Fakultě elektrotechnické Západočeské univerzity v Plzni.

Prohlašuji, že jsem svou závěrečnou práci vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 270 trestního zákona č. 40/2009 Sb.

Také prohlašuji, že veškerý software, použitý při řešení této diplomové práce, je legální.

V Plzni dne 28. března 2020

Bc. Lukáš Manda

.....

Podpis

Poděkování

V první řadě bych rád poděkoval vedoucímu práce panu Ing. Kamilu Kosturikovi, Ph.D., který mi byl při řešení této práce velmi nápomocen a zároveň bych chtěl poděkovat panu Ing. Petru Kristovi, Ph.D. za cenné rady při návrhu a realizaci zařízení. Dále bych chtěl poděkovat panu Doc. Ing. Jiřímu Skálovi, Ph.D. a panu Ing. Zdeňkovi Kubíkovi, Ph.D. za pomoc pří měření elektromagnetické kompatibility vytvořené komunikační jednotky. V neposlední řadě bych rád poděkoval své rodině a přátelům, kteří mě podporovali po celou dobu mého studia.

Obsah

Seznam obrázků

Seznam tabulek

Seznam symbolů a zkratek

CAN	Controller Area Network
CSMA/CD-AMP ...	Carrier Sense Multiple Access / Collision Detection with Arbitration on Message Priority
CSMA/CR	Carrier Sense Multiple Access with Collision Resolution
DTE	Data terminal equipment.
DCE	Data circuit-terminating equipment.
EIA	Electronic Industries Alliance.
ISO	International Organization for Standardization.
lwIP	Lightweight TCP/IP stack
RS	Recommended Standard
UART	Universal asynchronous receiver-transmitter
USART	Universal synchronous and asynchronous receiver-transmitter

1

Úvod

Cílem této diplomové práce je návrh a realizace komunikační jednotky pro vzdálenou správu zařízení připojených ke standardům RS-232 a CAN. Komunikační jednotka má za cíl posílat zprávy na rozhraní RS-232 a CAN prostřednictvím Ethernetového rozhraní.

V teoretické části jsou popsány základní specifikace komunikačních sběrnic, konkrétně standardy RS-232, RS-485 a CAN, jejich napěťové úrovně, průběhy signálů a formáty zpráv.

V kapitole Návrh komunikační jednotky je popsána navržená komunikační jednotka. Navržená komunikační jednotka je rozdělena do několika obvodových bloků. U těchto bloků jsou popsány zvolené obvodové součástky a jejich zapojení. Všechny obvodové bloky jsou navzájem galvanicky odděleny.

V kapitole Implementace softwaru pro komunikační jednotku je popsán lwIP TCP/IP stack jeho vlastnosti, funkce a možnosti použití. V další části této kapitoly je popsána struktura softwaru a jeho implementace v jazyce C++. Dalším krokem je popis navázání spojení s komunikační jednotkou a ukázkovými zprávami pro konfiguraci komunikační jednotky. Poslední částí této kapitoly je klientská aplikace, která umožňuje konfigurovat, ovládat komunikační jednotku a odesílat, přijímat zprávy. V poslední kapitole je popsáno měření elektromagnetické kompatibility komunikační jednotky a následné vyhodnocení výsledků měření.

2

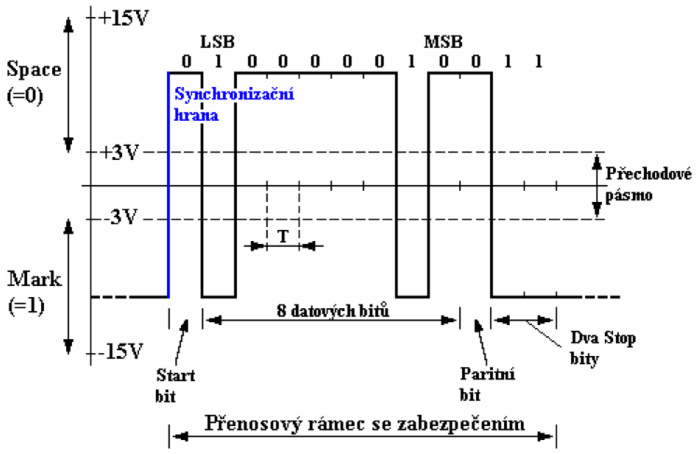
Základní specifikace komunikačních sběrnic

V této kapitole jsou popsány základní specifikace průmyslových standardů RS-232, RS-485 a CAN. Tyto vybrané průmyslové standardy budou použity při návrhu komunikační jednotky, která je cílem této diplomové práce.

2.1 RS-232

Standard RS-232 je jeden z nejpoužívanějších standardů sériové komunikace. Standard byl definován společenstvím EIA v roce 1962 za účelem vytvoření sjednoceného rozhraní mezi dvěma uzly konkrétně mezi DTE a DCE, pro přenos do vzdálenosti desítek metrů v závislosti na zvolené přenosové rychlosti a vybraném kabelu. V průběhu let prošel standard šesti aktualizacemi (RS-232-A, RS-232-B, RS-232-C, RS-232-D, RS-232-E, RS-232-F). Standard RS-232 díky svým vlastnostem byl nejpoužívanějším sériovým rozhraním osobních počítačů do nástupu standardu USB. Aktuálně se využívá RS-232 pro komunikaci dvou embedded zařízení, popřípadě embedded zařízení s PC pomocí převodníku USB<=>RS-232. Standard RS-232 definuje mechanické, elektrické a funkční parametry. Jedná se o duplexní asynchronní komunikaci typu bod-bod, to znamená, že jeden vysílač komunikuje s jedením přijímačem. Přenos dat se provádí na nesymetrickém vedení s použitím negativní logiky. [1, 2] Časový průběh jednoho rámce se zabezpečením a napěťové úrovně RS-232 jsou přehledně znázorněny na obr. 2.1.

Vysílač zahájí vysílaní tzv. start bitem. Start bit slouží pro detekci začátku přenosu dat a k synchronizaci přijímací stanice. Jeho hodnota je logická 0, to odpovídá napěťové úrovni vysílače +5V až +15V. Po odvysílání start bitu následuje přenos 8 bitů tj. 1 byte v pořadí od nejméně významného bitu LSB až po nejvýznamnější bit MSB. Přenášená data mohou být zabezpečena paritním bitem. Paritní bit je generován řadičem jako lichá nebo sudá parita. Konec přenosu je detekován jako jeden nebo více stop bitů. Jeho hodnota je logická 1, to odpovídá napěťové úrovni -5V až -15V.



Obr. 2.1: Průběh signálu RS-232 |Převzato z [2]|

Přijímač detekuje následující napěťové úrovně:

- $-3V$ až $-25V$ je logická úroveň 1
- $+3V$ až $+25V$ je logická úroveň 0
- $-3V$ až $+3V$ úroveň není definována

Maximální přenosová rychlost sběrnice základního standardu RS-232 je specifikována na 19200 bit/s pří délce komunikačního vedení 15 m. Reálná vzdálenost je závislá na dodržení maximální zatěžovací kapacitě vedení, která nesmí být větší než 2500 pF. V aktualizovaných verzích RS-232-A, RS-232-B a RS-232-C byla zvýšena komunikační rychlost až na 115,2 kbit/s, za předpokladu kratšího přenosového vedení. Maximální napěťové úrovně byly sníženy z 25V na 12V a dále na 5V. [2]

Standard RS-232 definuje přenos datových signálů pro vysílání a příjem a dále stavové a řídící signály. V některých aplikacích jsou využívány pouze datové signály. Přiřazení signálů k 9-pinovému CANON konektoru se nachází v tabulce 2.1.

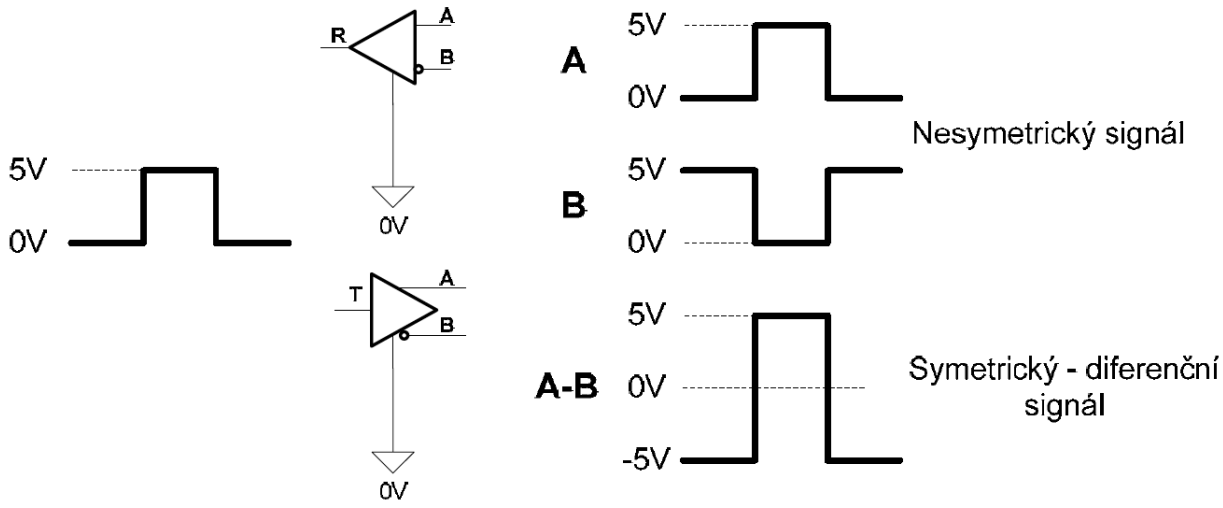
Číslo pinu	Signál	Zdroj	Typ signálu	Popis
1	CD	DCE	řídící	Carrier detect
2	RX	DCE	datový	Receive data
3	TX	DTE	datový	Transmit data
4	DTR	DTE	řídící	Data terminal ready
5	SG	-	-	Signal ground
6	DSR	DCE	řídící	Data set ready
7	RTS	DTE	řídící	Request to send
8	CTS	DCE	řídící	Clear to send
9	RI	DCE	řídící	Ring Indicator

Tab. 2.1: Přiřazení signálů RS-232 k 9-pinovému CANON konektoru |Převzato z [1]|

2.2 RS-485

Komunikační standard RS-485 byl definován institucí EIA v roce 1983. Vzhledem ke svým vlastnostem, se stal standard jednou z nejvíce používaných fyzických vrstev, široce využívaných v průmyslových komunikačních protokolech vyšších vrstev. Standard RS-485 lze využít v aplikacích, kde je zapotřebí komunikovat na vzdálenost až 1200 m nebo rychlostí až 10 Mbit/s oproti standardu RS-232. Komunikační sběrnici RS-485 umožňuje obousměrnou komunikaci až s 32 uzly připojených ke sdílenému médiu s využitím kroucené dvojlinky. Sběrnicový standard používá komunikační vedení o vlnové impedanci 120 Ω.

Standard RS-485 je využíván v průmyslovém prostředí díky své odolnosti vůči elektromagnetickému rušení. Standard definuje pouze elektrické charakteristiky transceiveru, připojeného ke komunikační sběrnici, nespecifikuje konektory, kabely ani protokol přenášených dat. [1, 3] Specifikace RS-485 je založena na přenosu diferenčního symetrického signálu. Princip této přenosové metody je znázorněn na obr. 2.2.



Obr. 2.2: Průběh signálu RS-485 |Převzato z [3]|

Vysílaná data jsou přivedena na vstup T, vysílač vytvoří dvojici komplementárních signálů A a B. Signál A je vysílán ve fázi a signál B v proti fázi. Jako přenosové medium je převážně využívána stíněná kroucená dvojlinka. Pří použití tohoto typu přenosového média se zajistí eliminace rušivých signálů na přenosové cestě. Na přijímací straně jsou signály A a B od sebe odečteny a výsledný signál je k dispozici na výstupu R. Přijímač vyhodnocuje rozdíl mezi dvojicí signálových vodičů, nikoli jednotlivé nesymetrické signály A a B.

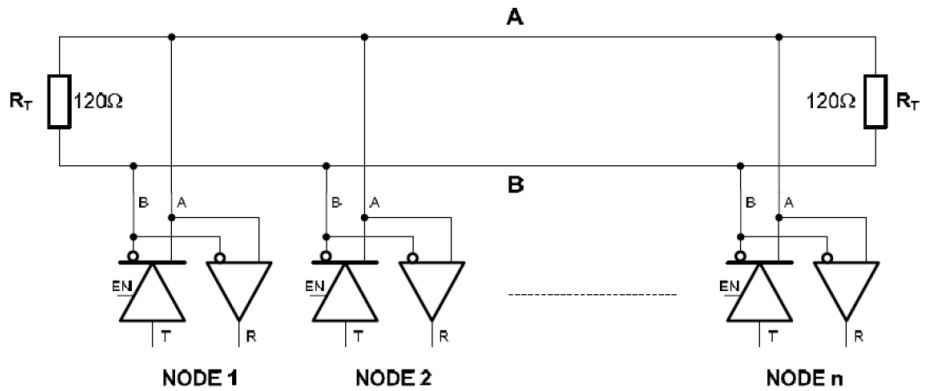
Jak bylo uvedeno v úvodu kapitoly, standard RS-485 nespecifikuje konektory ani kabely. Avšak existuje dohodnuté zapojení konektorů. Přiřazení signálů k 9-pinovému CANON konektoru se nachází v tabulce 2.2.

Číslo pinu	Poloduplexní režim	Duplexní režim
1	Stínění, kostra	Stínění, kostra
2	Rezervováno pro pomocné napájení	Y
3	A	A
4	Řídící signál A (pro opakovač)	
5	0 zdroje	0 zdroje
6	Napájení pro zakončovací děliče vedení +5V	
7	Rezervováno pro pomocné napájení	Z
8	B	B
9	Řídící signál B (pro opakovač)	

Tab. 2.2: Přiřazení signálů RS-485 k 9-pinovému CANON konektoru |Převzato z [4]|

2.2.1 Dvouvodičové zapojení RS-485

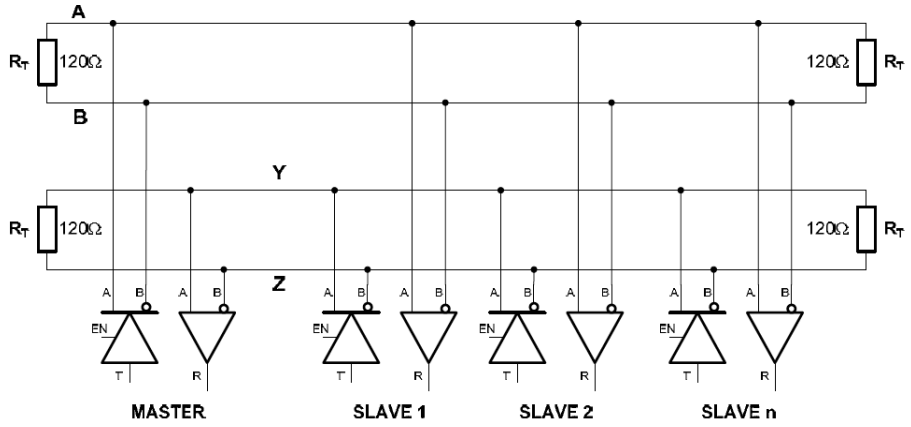
Základní variantou RS-485 je tzv. dvouvodičové zapojení. Toto zapojení umožnuje obousměrnou komunikaci v poloduplexním režimu. Všechny uzly jsou připojené k sdílené sběrnici pomocí jednoho páru vodičů. Sběrnice musí být zakončená na obou stranách impedancí 120 Ω. U dvouvodičového zapojení RS-485 je zapotřebí zajistit, aby v určitém časovém intervalu vysílal pouze jeden uzel. O tuto funkci se stará vyšší protokolová vrstva. Během vysílání je zapotřebí přepnout ostatní vysílací uzly do třetího stavu. Pro tento účel jsou vybaveny transceivery řídícím pinem EN, který přepne vysílač do třetího stavu. Některé transceivery mají implementované řídící piny jak pro vysílací tak i pro přijímací část transceiveru. V takovém případě se řídící piny označují RE a DE.



Obr. 2.3: Dvouvodičové zapojení RS-485 |Převzato z [3]|

2.2.2 Čtyřvodičové zapojení RS-485

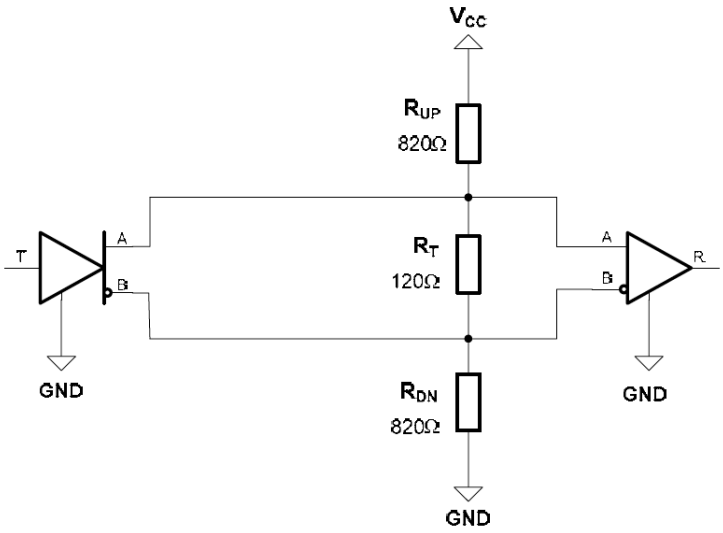
Čtyřvodičové zapojení RS-485 umožnuje připojení jedné master stanice a n slave stanic. U tohoto zapojení je využíván jeden pár vodičů pro přenos dat od master stanice ke všem slave stanicím. Druhý pár vodičů slouží pro přenos od všech slave stanic k přijímači master stanice. U čtyřvodičového zapojení RS-485 není potřeba master stanici přepínat do třetího stavu, avšak u slave stanic je při vysílání potřeba řídit přístup ke sběrnici pomocí třístavového budiče. [3]



Obr. 2.4: Čtyřvodičové zapojení RS-485 |Převzato z [3]|

2.2.3 Terminace vedení s definicí stavu

Základním prvkem komunikace RS-485 je obousměrný budič, pracující v diferenčním módu s rozkmitem diferenčního signálu v rozmezí $1,5V$ až $5V$. Budič je napájen napětím $+5V$. Diferenční přijímač má na svém výstupu definovanou hodnotu, pokud rozdíl vstupních napětí je větší než $\pm 200\,\mathrm{mV}$. Nižší diferenční napětí spadá do tzv. pásma neurčitosti. Pro zamezení uvedeného stavu má každý přijímač vstupní impedanci $12\,\mathrm{k\Omega}$ a interní pull-up, pull-down odpory pro definování klidového stavu log. 1. Připojením zakončovacích odporů $120\,\Omega$ na vedení, dojde k narušení vnitřního děliče obvodu a k nedefinovaným stavům na sběrnici v době, kdy jsou všechny vysílače odpojeny od sběrnice tj. vysílače se nacházejí v třetím stavu. Z tohoto důvodu jsou na sběrnici připojeny odpory R_T, R_{UP} a R_{DN} viz obr. 2.5. Tyto odpory tvoří napěťový dělič v takovém poměru, aby na R_T bylo napětí větší než 200 mV. [3]



Obr. 2.5: Terminace vedení RS-485 s definicí klidového stavu |Převzato z [3]|

2.3 CAN

Komunikační standard CAN byl vyvinut společnosti Bosch v roce 1986 původně za účely automobilového průmyslu. Z původního automobilového průmyslu se CAN také rozšířil do průmyslového řízení, robotiky, medicínských zařízení a do některých leteckých systémů.

CAN umožňuje využívat přenosové rychlosti od 20kb/s do 1Mb/s. Přenosová rychlost se odvíjí od délky sběrnice a rychlosti transceiveru. CAN je zajímavým řešením embedded řídících systémů z důvodu nízké ceny, jednoduchosti protokolu, arbitráže, implementované funkce pro detekci chyb a možností automatického opakování přenosu. [3, 6]

Komunikační standard CAN byl definován pro zajištění deterministické komunikace v komplexních distribučních systémech s následujícími funkcemi a možnostmi:

- Přiřazení priority ke zprávám a garantování maximálního zpoždění
- Multicastová komunikace s bitově orientovanou synchronizací
- Konzistence dat napříč celým systémem
- Multimaster přístup ke sběrnici
- Detekce chyb a schopnost automatického opakovaného přenosu zamítnutých zpráv, princip arbitráže
- Detekce trvalého chybového uzlu a jeho automatické odpojení

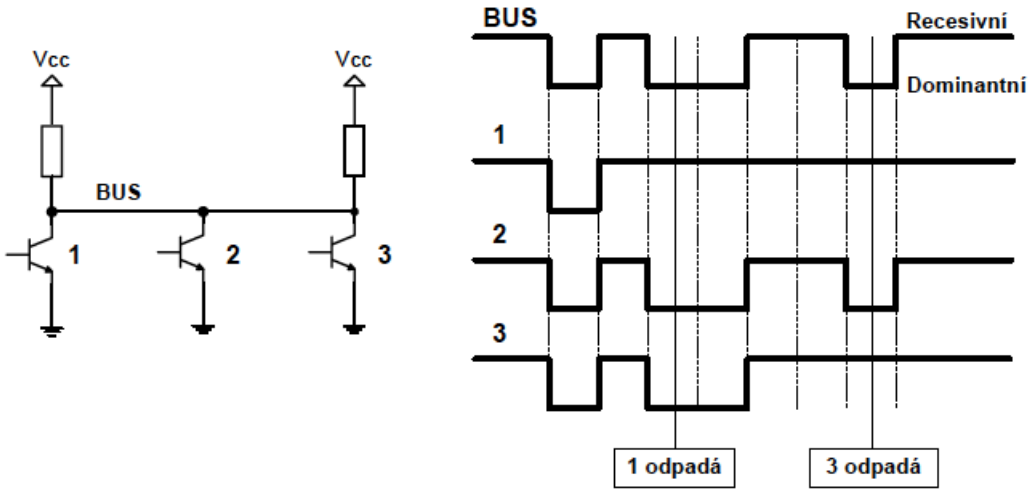
Původní specifikace CAN, vyvinutá firmou Bosch, obsahuje pouze fyzickou a linkovou vrstvu. Později ISO vydalo svou vlastní specifikaci protokolu CAN s podrobnější implementací fyzické vrstvy. Fyzická vrstva má na starost definování kódování bitů na elektrické signály s definovanými fyzikálními charakteristikami. V Bosch CAN standardu je však popis omezen na definici bit timing, bit encoding a synchronizaci. Specifikace fyzického přenosového média obsahuje požadavky na napěťové úrovně signálů, konektorů a další vlastnosti, které jsou nezbytné pro definování transceiveru. Ve standardu není specifikován konkrétní typ vodiče, ale převážně se používá kroucená dvojlinka. Další reference a implementace zaplnily tuto mezeru a poskytly řešení pro praktickou implementaci tohoto protokolu. Linková vrstva se skládá z podvrstev Logical Link Control (LLC) a Medium Access Control (MAC). Podvrstva LLC poskytuje všechny služby pro streamování bitů od zdroje k cíli, zejména následující definice:

- Služby pro přenos dat a vzdálený požadavek na data
- Podmínky, za nichž má být zprava přijatá, včetně filtrování zpráv
- Mechanismy pro řízení obnovy a řízení toku

Podvrstva MAC je zodpovědná za message framing, arbitráž, správu potvrzení a detekci chyb. Pro vyšší spolehlivost je MAC pod dohledem řídící entity, která sleduje chybové stavy a v případě trvale chyby omezí provoz daného uzlu. [6]

2.3.1 Princip arbitráže

Jako přístupová metoda je použita metoda CSMA/CD-AMP, v některých pramenech rovněž uváděná jako CSMA/CR, vycházející z toho, že všechny stanice připojené na sběrnici monitorují její provoz, a jakmile stanice, která potřebuje vysílat, detekuje klidový stav na sběrnici, spustí vysílání. [3] Pokud při vysílání zprávy dojde k detekci kolize identifikátorů, bude odeslána zpráva s nejnižším identifikátorem tj. s nejvyšší prioritou. Odmítnutá stanice může opakovat vysílaní zprávy v dalším kroku, kdy bude sběrnice v klidovém stavu. Princip přidělení přístupu na sběrnici CAN je realizován pomocí bitové arbitráže, která je znázorněna na obr. 2.6.



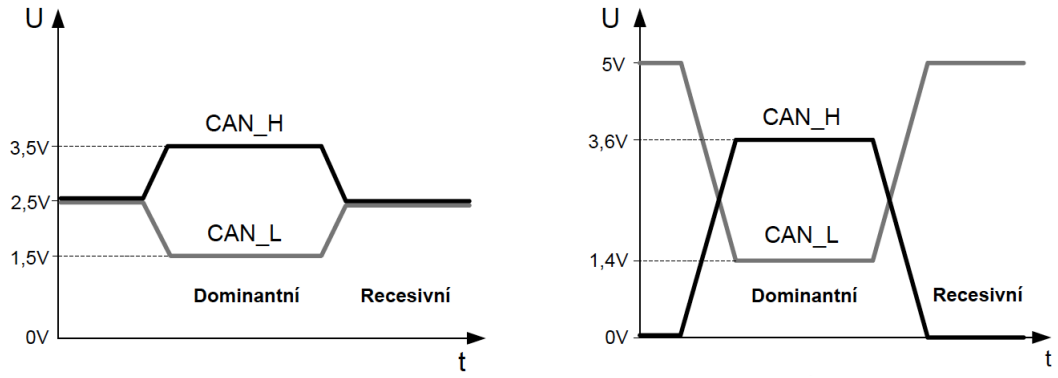
Obr. 2.6: Princip arbitráže na sběrnici CAN |Převzato z [3]|

Popis zahájení vysílání stanic č. 1,2,3 (z obr. 2.6): Všechny stanice začnou vysílat ve stejný okamžik, je odvysílán 1. bit v dominantním stavu. Stanice detekují na sběrnici stejnou hodnotu a pokračují stejným způsobem i u 2. bitu. Problém nastane u 3. bitu, kdy stanice č. 2,3 vyslaly dominantní stav a stanice č. 1 recesivní stav. V tomto okamžiku detekuje stanice č. 1 kolizi a dojde k odpojení od sběrnice. V dalším kroku odešlou stanice č. 2,3 bity 4,5,6. Při odesílání 7. bitu dojde ke kolizi a dojde k odpojení stanice č. 3. Sběrnici pro odesílaní zprávy získala stanice č. 2.

Z uvedeného popisu je patrné, že jsou data vysílaná v přímém formátu na komunikační sběrnici, výjimku tvoří stav, kdy vysílaná zpráva obsahuje více než 5 po sobě jdoucích stejných bitových hodnot stejné úrovně. V takovém případě je na straně vysílače do bitového toku vložená inverzní bitová úroveň a na přijímací straně je tato hodnota odebrána. Tento stav se nazývá bit stuffing.

2.3.2 Diferenční signály a napěťové úrovně sběrnice CAN

Sběrnice CAN využívá dvojici signálů CAN_H a CAN_L. Tato dvojice tvoří komplementární dvojici signálů, které jsou nesymetrické vůči referenční zemi, ale symetrické navzájem a definující vzájemné diferenční napětí. [3] Sběrnice využívá dva standardy High-Speed a Low-Speed. Napěťové úrovně standardů High-Speed CAN a Low-Speed CAN jsou znázorněny na obr. 2.7. Oba standardy jsou navzájem nekompatibilní.



(a) High-Speed (b) Low-Speed

Obr. 2.7: Napěťové úrovně na sběrnici CAN |Převzato z [3]|

High-Speed CAN

- Recesivní stav je definován pro napětí $V_{CAN\_H} = 2,5V$ a $V_{CAN\_L} = 2,5V$.
- Dominantní stav je definován pro napětí $V_{CAN\_H} = 3,5V$ a $V_{CAN\_L} = 1,5V$.

Low-Speed CAN

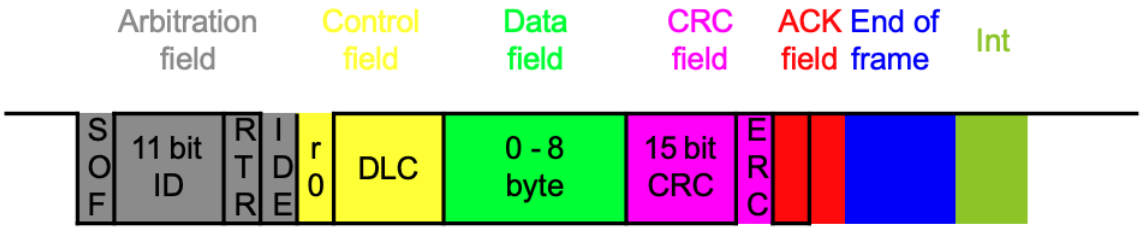
- Recesivní stav je definován pro napětí $V_{CAN\_H} = 0V$ a $V_{CAN\_L} = 5V$.
- Dominantní stav je definován pro napětí $V_{CAN\_H} = 3,6V$ a $V_{CAN\_L} = 1,4V$.

2.3.3 Formát zpráv sběrnice CAN

Komunikační protokol CAN podporuje dva základní formáty zpráv CAN 2.0A Standard a CAN 2.0B Extended. Rozdíl mezi formáty zpráv je v délce identifikátoru, CAN 2.0A využívá délku identifikátoru 11 bitů a CAN 2.0B 29 bitů. Délka identifikátoru jednoznačně určuje maximální počet objektů, které mohou být přenášeny na komunikační sběrnici. U základního formátů CAN 2.0A je maximální počet 2^{11} a pro rozšířený formát CAN 2.0B je maximální počet 2^{29}. Oba formáty mohou být využívány současně na jedné sběrnici, avšak CAN 2.0A má vyšší prioritu. [3, 6]

CAN 2.0A Standard

Formát rámce CAN 2.0A Standard je znázorněn na obr. 2.8. Rámec je složen z následujících částí [3]:



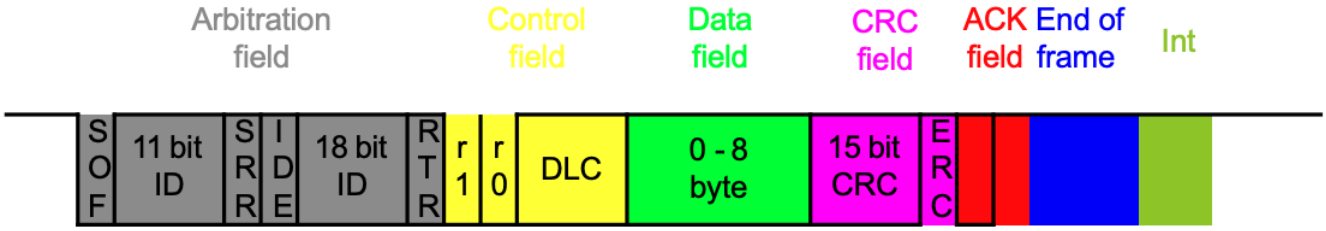
Obr. 2.8: CAN 2.0A Standard |Převzato z [5]|

- **Start of Frame** - 1 bit v dominantním stavu, který značí začátek rámce a slouží k synchronizaci komunikačních uzlů připojených ke komunikační sběrnici.
- **ID** - 11 bitů, které určují prioritu zprávy na sběrnici. Vyšší prioritu má zpráva s nižší binární hodnotou. Identifikátor je vysílán od nejvíce významného bitu až po nejméně významový bit.
- **Remote Transmission Request** - 1 bit, který rozlišuje zda se jedná o klasický datový rámec (recesivní stav) nebo zda se jedná o žádost o data (dominantní stav).
- **Identifier Extension** - 1 bit, který identifikuje, zda je využíván CAN 2.0A nebo CAN 2.0B
- **r0** - 1 bit, rezerva
- **Data Length Code** - 4 bity, určující počet přenášených datových bajtů (0 až 8)
- **Data** - 0 až 8 bajtů přenášených dat
- **Cyclic Redundancy Check** - 15 bitů, určujících cyklický zabezpečovací kód bloku přenášených dat na základě generujícího polynomu: $G(x) = x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1$
- **End of CRC** - 1 bit, ukončující CRC pole
- **Ack slot bit** - 1 bit, potvrzující dominantním stavem, že nenastaly žádné chyby během přenosu.
- **Acknowledge Delimiter** - 1 bit oddělující potvrzovací bit

- **End of Frame** - 7 bitů v recesivním stavu označující konec rámce vysílané zprávy. Délka 7 bitů je zvolena s ohledem na možnost generovaná tzv. chybového rámce, který muže být v tomto intervalu vyslán jakoukoliv přijímací stanicí, která detekuje chybu při přijímání zprávy, převážně z důvodu nesouhlasu přijatého CRC pole.
- **Interframe Space** - Mezera mezi zprávami v minimální délce 3 bitů v recesivním stavu.

CAN 2.0B Extended

Formát rámce CAN 2.0B Extended je znázorněn na obr. 2.9. Rámec obsahuje totožná pole jako CAN 2.0A se shodným významem s tím, že navíc definuje 3 přídavná pole [3]:



Obr. 2.9: CAN 2.0B Extended |Převzato z [5]|

- **Substitute Remote Request** - 1 bit, který nahrazuje původní RTR bit.
- **Identifier Extension** - 18 bitů, druhá část identifikátoru zprávy
- **r1** - rezerva pro budoucí využití

3

Návrh komunikační jednotky

Ze zadání diplomové práce je patrné, že komunikační jednotka má obsahovat standardy RS-232 a CAN. Komunikace z těchto standardů bude možné odesílat a přijímat přes rozhraní Ethernet. Po domluvě s vedoucím diplomové práce byl návrh komunikační jednotky rozšířen o standard RS-485, stavové a řídící signály. Pro implementaci komunikační jednotky byla zvolena vývojová deska NUCLEO-F429ZI s mikrokontrolérem STM32F429. Tento mikrokontrolér disponuje následujícími vlastnostmi:

- Arm® 32-bit Cortex®-M4 CPU s FPU
- Maximální frekvence 180 MHz
- FLASH 2 MB
- RAM 256 KB
- UART
- CAN
- 10/100 Ethernet MAC s přístupem pomocí DMA

Pří návrhu bylo zvoleno řešení umožňující vyměnit vývojovou desku NUCLEO-F429ZI za jiný model např. NUCLEO-F767ZI nebo ESP32, který disponuje Ethernet i WIFI rozhraním. Navržená komunikační jednotka má galvanicky oddělené standardy RS-232, RS-485, CAN, stavové a řídící signály. Navržená komunikační jednotka je rozdělena do několika obvodových bloků. Jedná se o RS-232, RS-485, CAN, stavové a řídící signály a napájecí obvody.

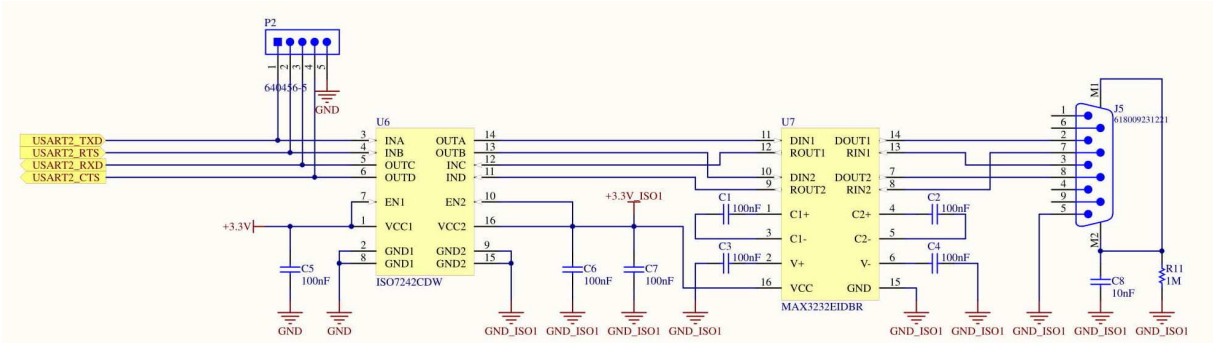
3.1 Zapojení RS-232

Komunikační standard RS-232 je u této jednotky navržen ve dvou zapojení. První část zapojení je klasické zapojení RS-232. Tedy komunikace mezi mikrokontrolérem a připojenou externí jednotkou. U tohoto zapojení jsou použity obvody ISO-7242 a MAX3232.

Obvod ISO-7242 je čtyřkanálový digitální izolátor se dvěma vstupními a dvěma výstupními kanály na obou stranách obvodu s ESD ochranou 4 kV. [7] Zvolený obvod slouží ke galvanickému oddělení signálů z a do mikrokontroléru a RS-232 z důvodu možné nestability zemních potenciálů.

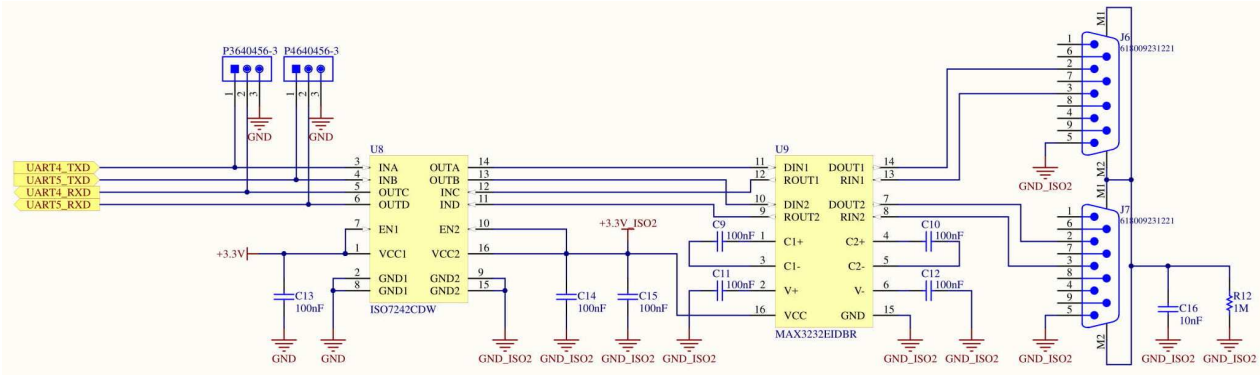
Obvod ISO-7242 umožňuje vysokorychlostní komunikaci až 25 Mbit/s s možností odpojení výstupních kanálů pomocí vstupů EN1 a EN2. V tomto návrhu není potřeba této funkce využít, a proto jsou tyto vstupy připojeny k napájecímu napětí. K obvodu jsou zapojeny kondenzátory C5 a C6 jako blokovací kondenzátory o hodnotě 100 nF.

Obvod MAX3232 je vícekanálový řadič RS-232 s ESD ochranou 15 kV s možnou komunikaci až do 250 kbit/s. [8] K obvodu jsou připojeny čtyři kondenzátory C1, C2, C3 a C4 pro funkci nábojové pumpy o hodnotě 100 nF a blokovací kondenzátor C7 o hodnotě 100nF. Kondenzátor C8 a rezistor R11 slouží k potlačení rušivých napětí.



Obr. 3.1: Zapojení RS-232

Druhá část zapojení RS-232 využívá identické obvody ISO-7242 a MAX3232 jako první část avšak toto zapojení umožňuje komunikaci mezi dvěma externími jednotkami. Komunikace prochází skrze dvojici sériových linek do mikrokontroléru. Takovéto zapojení umožňuje odposlouchání provozní komunikace a přeposlání této komunikace přes rozhraní Ethernet.



Obr. 3.2: Zapojení RS-232

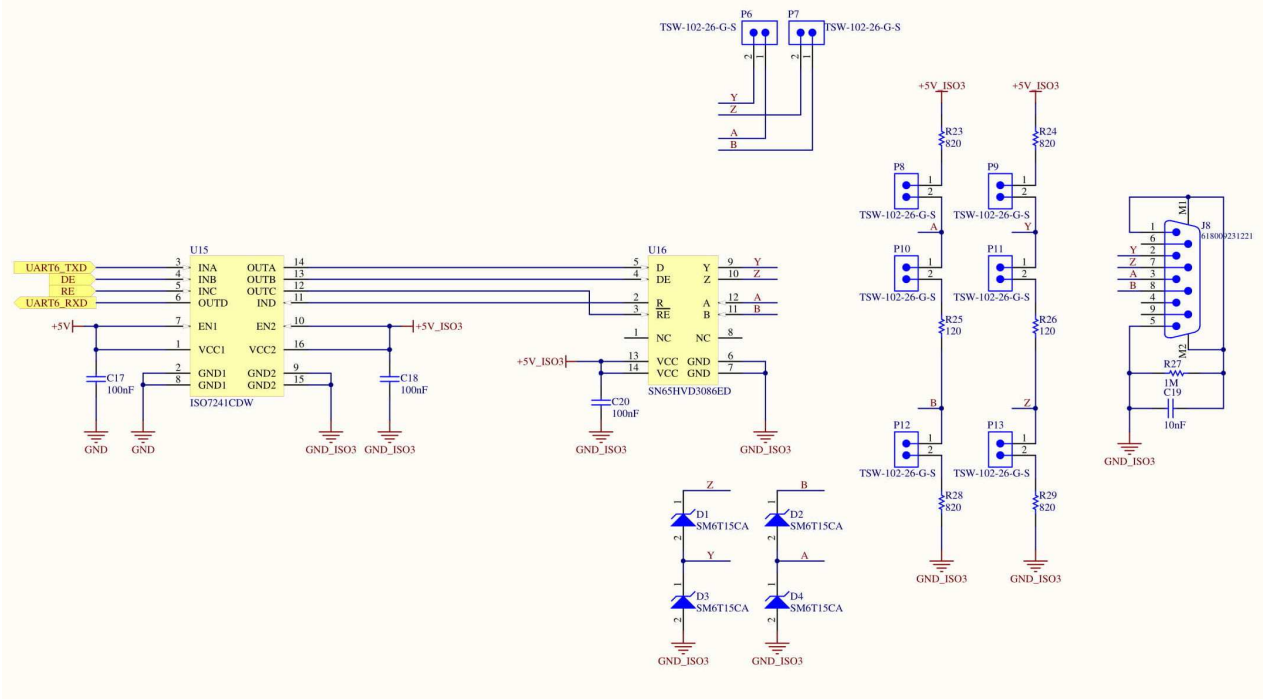
3.2 Zapojení RS-485

Při návrhu komunikačního standardu RS-485 byly použity obvody ISO-7241 a SN65HVD3086E. Obvod ISO-7241 je čtyřkanálový digitální izolátor s ESD ochranou 4 kV. Obvod disponuje třemi vstupními a jedním výstupním kanálem je jedné straně a třemi výstupními a jedním vstupním kanálem na druhé straně.

Obvod ISO-7241 umožňuje vysokorychlostní komunikaci až 25 Mbit/s s možností odpojení výstupních kanálů pomocí vstupů EN1 a EN2. V tomto návrhu není potřeba této funkce využít, a proto jsou tyto vstupy připojeny k napájecímu napětí. K obvodu jsou zapojeny blokovací kondenzátory C17 a C18 o hodnotě 100 nF. [7]

Obvod SN65HVD3086E je řadič RS-485 pro čtyřvodičové zapojení s ESD ochranou 16 kV pro sběrnicové piny. [9] Obvod SN65HVD3086E obsahuje piny D a R pro odeslání a přijímání dat, piny DE a RE slouží pro řízení komunikace. Piny A,B a Y,Z slouží k připojení ke sběrnici.

V případě potřeby dvouvodičového zapojení je potřeba zapojit propojky P6 a P7. Pro terminaci vedení o hodnotě 120 Ω slouží propojky P10 a P11. K definování napěťové úrovně v případě, kdy je řadič odpojen od sběrnice, slouží propojky P8,P9,P12 a P13. Pro přepěťovou ochranu slouží oboustranné transily D1,D2,D3 a D4.



Obr. 3.3: Zapojení RS-485

3.3 Zapojení CAN

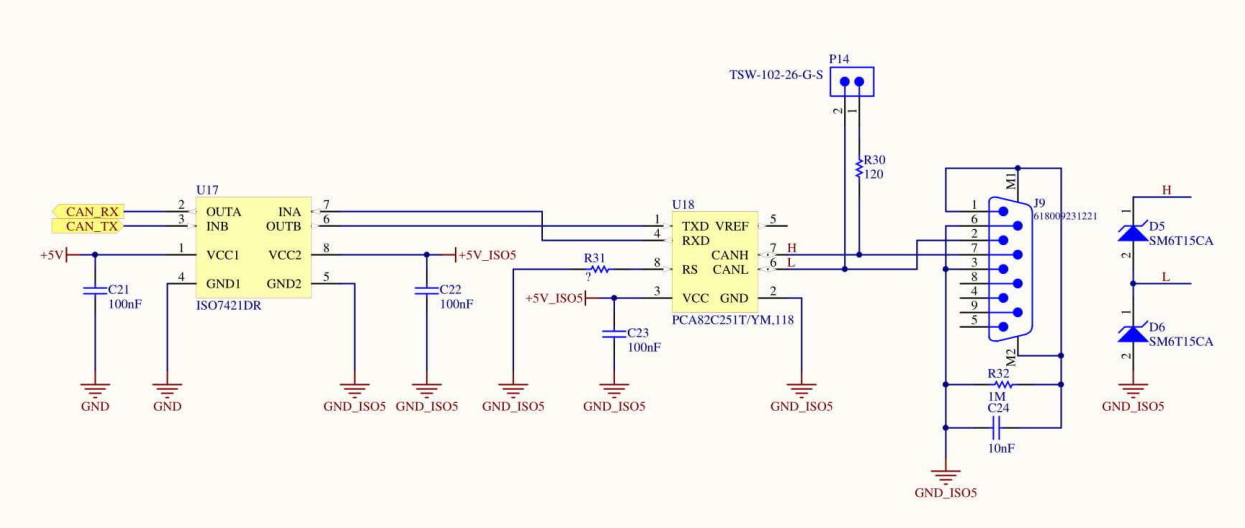
Pří návrhu komunikační sběrnice CAN byly použity obvody ISO-7421 a PCA82C251.Obvod ISO-7421 je dvoukanálový digitální izolátor s jedním vstupním a jedním výstupním ka-

nálem na obou stranách obvodu. Obvod ISO-7421 umožňuje komunikaci až 1 Mbit/s. K obvodu jsou zapojeny blokovací kondenzátory C21 a C22 o hodnotě 100 nF. [10]

Obvod PCA82C251 je rozhraní mezi řadičem protokolu CAN a fyzickou sběrnicí s komunikační rychlostí až 1 MBd a ESD ochranou 2.5 kV. Pomocí pinu RS je možné měnit tří režimy obvodu:

- Standby: $V_{Rs} > 0.75V_{CC}$
- Slope control: $0.4V_{CC} < VRs < 0.6V_{CC}$
- High-speed: $V_{Rs} < 0.3V_{CC}$

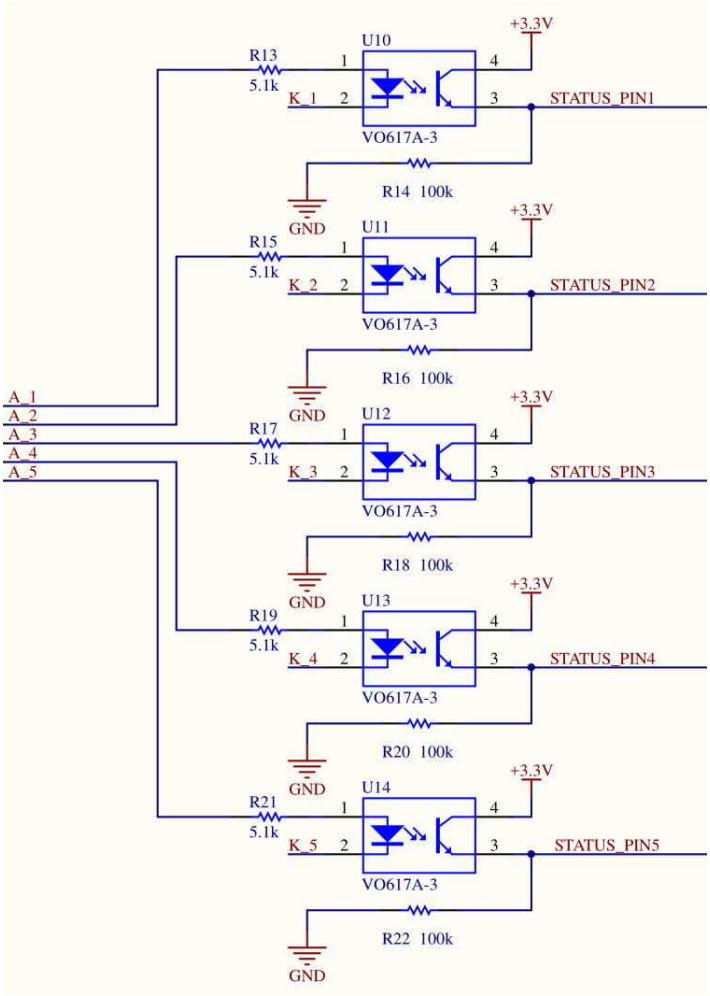
V tomto návrhu je obvod využíván ve stavu High-speed. [11] Obvod PCA82C251 obsahuje piny pro TXD a RXD pro odeslání a příjem CAN zpráv ze strany mikrokontroléru a piny CAN H a CAN L slouží k připojení ke sběrnici. Pro přepěťovou ochranu slouží oboustranné transily D5 a D6.



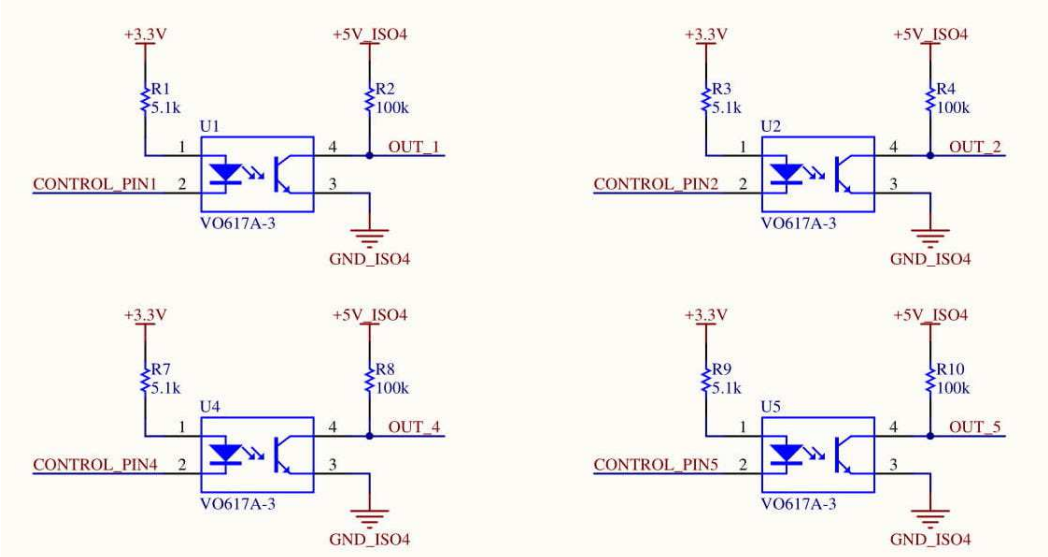
Obr. 3.4: Zapojení CAN bus

3.4 Zapojení stavových a řídících signálů

Pro galvanické oddělení stavových a řídících signálů byly použity optrony. Stavové signály umožňují sledovat externí logické signály s maximálním vstupním napětím 24V. Fototranzistory jsou zapojeny jako emitorové sledovače. Výstupy z emitorů jsou zapojeny na vstupní piny mikrokontroléru. Řídící signály umožňují ovládat řídící signály externí jednoty. Výstupní napětí z optronů je stanoveno na 5V.



Obr. 3.5: Zapojení stavových signálů



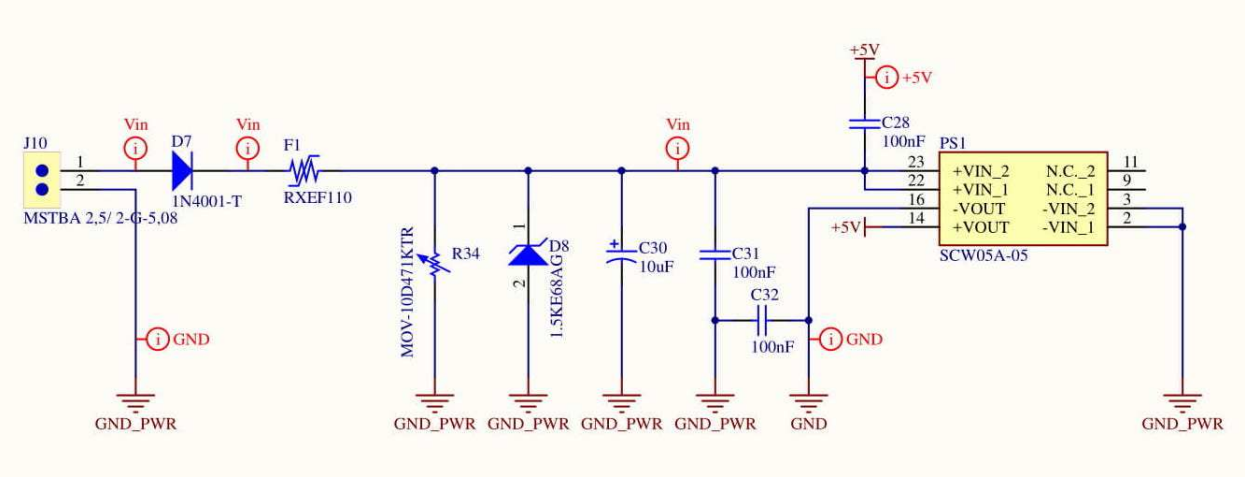
Obr. 3.6: Zapojení řídících signálů

3.5 Zapojení napájecích obvodů

O napájení celé jednotky se stará hlavní napájecí zdroj MEAN WELL SCW05A-05, který má tyto specifikace:

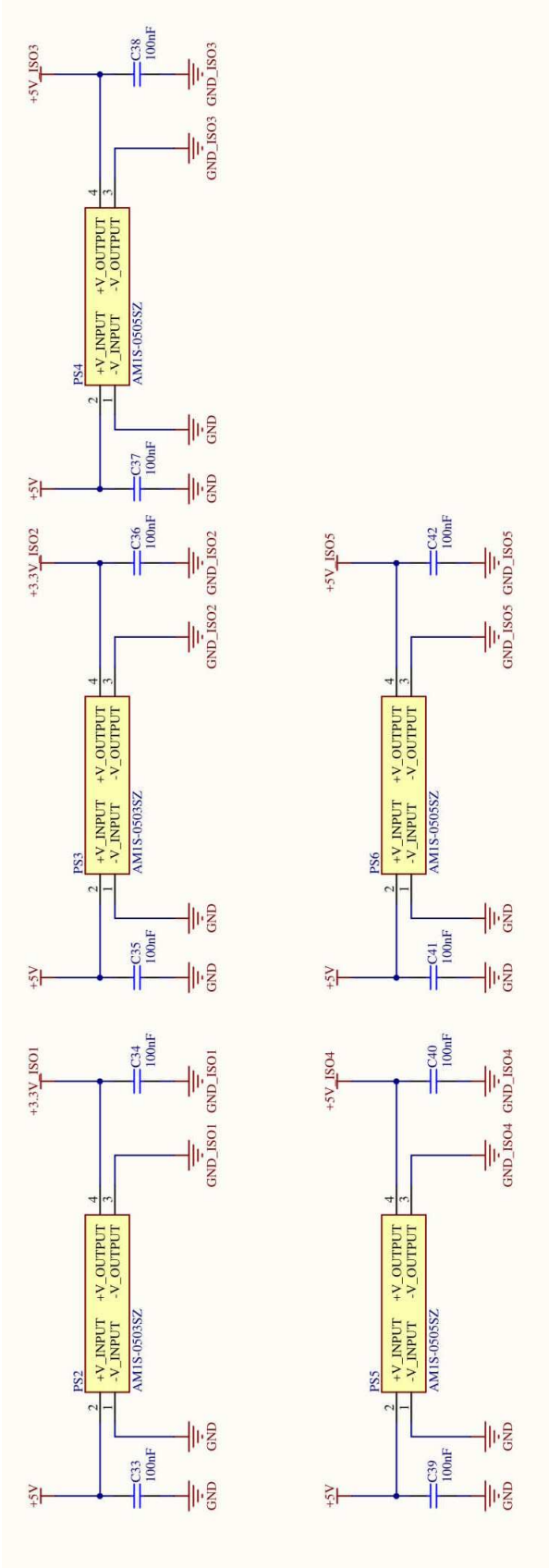
- Vstupní napětí 9-18 V
- Výstupní napětí 5V
- Výstupní proud 1000mA
- Galvanická izolace 1000V DC

Ke zdroji jsou zapojeny blokovací kondenzátory C25, C30 o hodnotě 10 uF a C27, C31 o hodnotě 100 nF. Při návrhu byly do schéma zapojení přidány kondenzátory C28 a C32 pro možnost odvedení rušivých proudů, ale pří osazení nebyly tyto kondenzátory osazeny z důvodu možného přerušení galvanického oddělení. Pro přepěťovou ochranu byl použit varistor R34 a transil D8.



Obr. 3.7: Zapojení hlavního napájecího zdroje

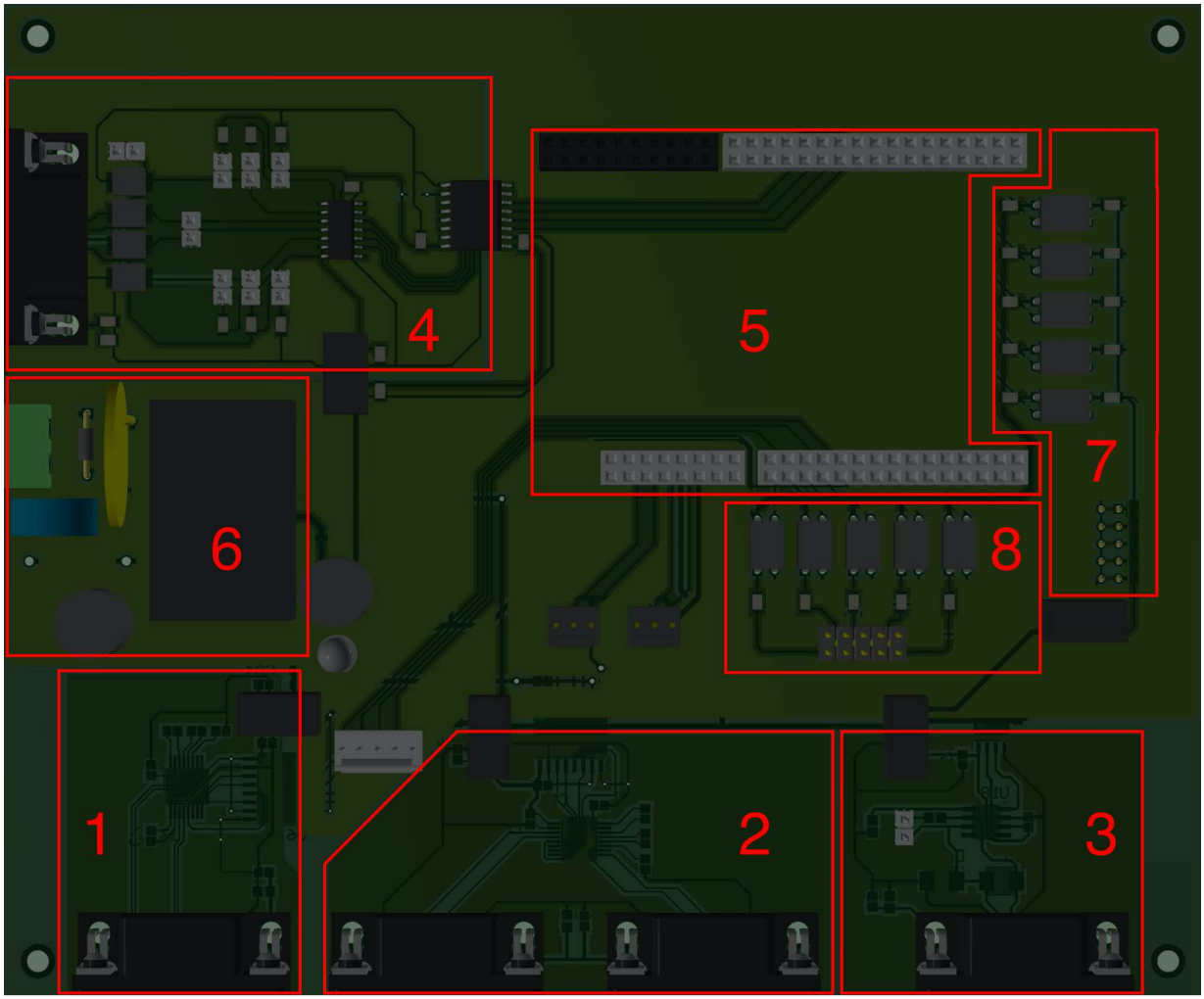
K hlavnímu zdroji jsou připojeny DC/DC zdroje, které slouží k napájení obvodů pro galvanické oddělení sběrnicových, řídících a stavových signálů. Pro napájení těchto obvodu byly zvoleny dva DC/DC zdroje AIMTEC AM1S-0503SZ 5V/3.3V DC s proudovou zátěží do 303 mA a tři DC/DC zdroje AIMTEC AM1S-0505SZ 5V/5V DC s proudovou zátěží do 200mA. Tyto zdroje disponují galvanickou izolací 1 kV. Ke zdrojům jsou připojeny blokovací kondenzátory o hodnotě 100 nF.



Obr. 3.8: Zapojení DC/DC zdrojů

3.6 Návrh desky plošného spoje

Návrh desky plošného spoje je znázorněn na obr. 3.9. V návrhu jsou vyznačeny jednotlivé obvodové bloky. V prvním bloku se nachází komunikační standard RS-232 pro komunikaci mezi mikrokontrolérem a externí jednotkou. Ve druhém bloku se nachází dva kanály RS-232, které umožňují odposlech komunikace mezi dvěma externími jednotkami. Ve třetím bloku se nachází CAN bus s možností zapojení terminace. Ve čtvrtém bloku se nachází standard RS-485 s možným zapojením terminace a stanovením napěťových úrovní v případě, kdy je řadič odpojen od sběrnice. V pátém bloku se nachází konektory pro připojení vývojové desky NUCLEO-F429ZI s ethernetovým rozhraní. V šestém bloku se nachází hlavní napájecí zdroj s přepěťovou ochranou. V sedmém bloku se nachází řídící a v osmém bloku stavové signály.



Obr. 3.9: Návrh desky plošného spoje

4

Implementace softwaru pro komunikační jednotku

V této kapitole budou popsány funkce lwIP TCP/IP stacku, dále struktura softwaru pro komunikační jednotku s popisem jak navázat spojení s komunikační jednotkou a ukázka klientské aplikace pro snadnou obsluhu komunikační jednotky.

4.1 lwIP TCP/IP stack

LwIP je odlehčená implementace TCP/IP protokolu, kterou vytvořil Adam Dunkels v laboratořích Computer and Networks Architectures ve Swedish Institute of Computer Science. LwIP je volně k dispozici pod licencí BSD. Zdrojové kódy LwIP jsou napsané v programovacím jazyce C. Cílem lwIP TCP/IP implementace je snížit využití RAM při zachování plného rozsahu TCP/IP. Díky tomu je lwIP vhodné pro využití u embedded systémů s požadavky na desítky kilobajtů volné paměti RAM a čtyřiceti kilobajty pamětí ROM. Od svého uvedení vyvolává lwIP velký zájem a je využíván v mnoha komerčních produktech. Zdrojové kódy lwIP je možné využít na více platformách s nebo bez použití operačního systémů. [12] LwIP obsahuje následující protokoly a funkce:

- **IP** (Internet Protocol) s možností předávání paketů přes více síťových rozhraní
- **ICMP** (Internet Control Message Protocol) pro údržbu a ladění sítě
- **IGMP** (Internet Group Management Protocol) pro správu multicastového provozu
- **UDP** (User Datagram Protocol) je protokol pro přenos dat mezi počítači v sítí bez záruky doručení datagramu
- **TCP** (Transmission Control Protocol) nejpoužívanější transportní protokol v sadě protokolů TCP/IP s garancí spolehlivého doručení zpráv
- **DNS** (Domain Name System) překlad domény na IP adresu

- **SNMP** (Simple Network Management Protocol) je protokol pro průběžný sběr dat pro potřeby síťové správy a jejich následného vyhodnocení
- **DHCP** (Dynamic Host Configuration Protocol) je protokol pro dynamické přiřazení IP adresy a konfiguračních parametrů síťovému prvku
- **ARP** (Address Resolution Protocol) je komunikační protokol pro provázání IP adresy a adresy síťového zařízení

LwIP poskytuje tři rozhraní pro komunikaci s TCP/IP kódem. První rozhraní je Raw API někdy nazývané jako Native API je událostmi-řízené API pro použití lwIP bez operačního systému. Raw API má implementované tzv. callback funkce, které jsou vyvolány při navázání spojení, přijímání, odesílání zpráv a ukončení spojení. Raw API má nejnižší hardwarové nároky. Druhé rozhraní je Netconn API, které je o úroveň vyšší než událostmi-řízené Raw API avšak toto API vyžaduje použití operačního systému s využitím vláken. Třetí rozhraní je Socket API o úroveň vyšší než Netconn API, vyžaduje použití operačního systému s využitím vláken. Socket API je známý zejména z Unixových systémů a má nejvyšší hardwarové nároky. [13]

4.2 Struktura softwaru

Struktura softwaru pro komunikační jednotku vychází ze základní šablony zdrojového kódu Ethernet a knihovny lwIP v konfiguraci Raw API, která byla poskytnuta Ing. Petrem Kristem, Ph.D. Navržená implementace softwaru pro komunikační jednotku se skládá z poskytnuté šablony zprovozněného Ethernetu a nově vytvořenou sadou tříd v jazyce C++:

- TCP_SERVER
- CAN_BUS
- USART
- GPIO
- IO_BLOCK

4.2.1 Třída TCP_SERVER

Třída TCP_SERVER vychází z ukázkového programu TCP_ECHOSERVER z Git repositáře. [14] Ukázkový program byl přepsán do C++ a rozšířen o funkcionality potřebné pro vytvořenou komunikační jednotku. V následující části je popsán zdrojový kód třídy TCP_SERVER.

```
class TCP_SERVER {
public :
        TCP_SERVER();
        bool init(const uint16_t port);

        void sendData(const char *c);
        bool availableData();
        char* getData();
private :

        struct TCP_SERVER_STRUCT {
                uint8_t state;
                struct tcp_pcb *pcb;
                struct pbuf *p;
                char data[128];
                bool flagData;
        };

        enum TCP_SERVER_STATE {
                TCP_NONE = 0,
                TCP_ACCEPTED,
                TCP_RECEIVED,
                TCP_CLOSING
        };

        struct tcp_pcb *tcp_server_pcb;
        struct TCP_SERVER_STRUCT tss;

        static void acknowledge(struct tcp_pcb *tpcb, struct TCP_SERVER_STRUCT *tss);
        static err_t server_accept(void *arg, struct tcp_pcb *newpcb,err_t error);
        static err_t server_recv(void *arg, struct tcp_pcb *tpcb, struct pbuf *p, err_t error);
        static void  server_error(void *arg, err_t err);
        static err_t server_poll(void *arg, tcp_pcb *tpcb);
        static err_t server_sent(void *arg, tcp_pcb *tpcb, uint16_t len);
        static void  server_send(struct tcp_pcb *tpcb, struct TCP_SERVER_STRUCT *tss);
        static void  server_connection_close(struct tcp_pcb *tpcb, struct TCP_SERVER_STRUCT *tss);

};
```

Třída TCP_SERVER se skládá z veřejného konstruktoru a metod init, sendData, availableData, getData. Metoda init je volána s parametrem port, který říká, na kterém portu chceme danou instanci TCP spojení. Metoda init provede inicializaci ukazatele tcp_server_pcb a následnou kontrolu zda ukazatel není NULL z důvodu nedostatku paměti. Následuje volání funkce tcp_bind, která prováže ukazatel tcp_server_pcb s IP adresou a portem. Pokud návratová hodnota předchozí funkce je ERR_OK je možné zavolat funkci tcp_listen, která začne naslouchat na zadané IP adrese a portu. Oproti ukázkovému kódu v Git repositáři [14] je zapnutá možnost udržení spojení v případě, kdy po dobu 30 vteřin nebude probíhat komunikace mezi serverem a klientem. V dalším kroku je pomocí funkce tcp_arg předán ukazatel na strukturu TCP_SERVER_STRUCT, která nese informace o aktuálním stavu TCP instance, strukturách pbuf, tcp_pcb a informaci o přijaté zprávě. Posledním krokem je provázání ukazatele tcp_server_pcb a metody server_accept pomocí funkce tcp_accept.

```
bool TCP_SERVER::init(const uint16_t port) {
        printf("TCP_SERVER::init\r\n");

        tcp_server_pcb = tcp_new();

        if(!tcp_server_pcb) {
                printf("Can not create \"tcp_server_pcb\"\r\n");

                return false;
        }

        err_t error;
        error = tcp_bind(tcp_server_pcb,IPADDR_ANY,port);

        if(error == ERR_OK) {

                tcp_server_pcb = tcp_listen(tcp_server_pcb);
                tcp_server_pcb->so_options |= SOF_KEEPALIVE;
                tss.state = TCP_NONE;
                tcp_arg(tcp_server_pcb,(void *)&tss);
                tcp_accept(tcp_server_pcb,server_accept);
                printf("TCP server ready\"\r\n");
                return true;
        }

        printf("Can not create \"tcp_bind\"\r\n");

        return false;

}
```

Metoda sendData jak je z názvu patrné slouží k odesílání zpráv. Metoda má jako parametr konstantní ukazatel na pole znaků. Odeslání zprávy je možné, pokud je aktuální stav TCP spojení ve stavu TCP_ACCEPTED nebo TCP_RECEIVED tedy pokud je klient připojen k dané instanci TCP spojení. V takovém případě se předá funkci tcp_sent ukazatel na tcp_pcb a ukazatel na metodu server_sent, která zajistí odeslání zprávy. Dalším krokem je předání přetypovaného ukazatele na zprávu a jeho délku.

```
void TCP_SERVER::sendData(const char *c){

        if((tss.state == TCP_ACCEPTED) ||(tss.state == TCP_RECEIVED )) {
                tcp_sent(tss.pcb, server_sent);

                tcp_write(tss.pcb, (void*)c, strlen(c), 1);

                tcp_recved(tss.pcb, strlen(c));
        }

}
```

Metody availableData a getData slouží k informování a získání přijaté zprávy.

Privátní část třídy obsahuje metody server_accept, server_recv, acknowledge, server_error, server_poll, server_sent, server_send a server_connection_close. Uvedené metody byly převzaty z ukázkového programu TCP_ECHOSERVER [14] a rozšířeny o metodu acknowledge. Metody z ukázkového programu umožňují zajistit TCP komunikaci avšak pouze odeslání poslední přijaté zprávy. Proto bylo potřeba vyřešit příjem dat a odesílání informace o přijmutí dat. Metody jsou vyvolány knihovními funkcemi lwIP od přerušení Ethernetového rozhraní. Z toho důvodu musí být všechny funkce statické. Takto požadované metody knihovny lwIP tvoří problém, že jednotlivé metody nemají přistup k proměnným dané instance třídy TCP_SERVER. Tento problém byl vyřešen tím, jak bylo uve-

deno v popisu metody init, že se pomocí funkce tcp_arg předá ukazatel na strukturu TCP_SERVER_STRUCT. Tímto způsobem je zajištěn přistup ze statických metod k proměnným každé instance třídy TCP_SERVER. Pro potřeby komunikační jednotky byly zapotřebí vytvořit a upravit metody server_recv a acknowledge.

Metoda server_recv z velké části vychází z ukázkového programu. Bylo zapotřebí upravit dva stavy TCP spojení a to stavy TCP_ACCEPTED a TCP_RECEIVED. Ve stavu TCP_ACCEPTED bylo zapotřebí dodělat potvrzení přijetí zprávy. Pro příjem zpráv slouží stav TCP_RECEIVED. V tomto stavu je potřeba získat ze struktury pbuf zprávu, která je označená jako payload. Získaný ukazatel je potřeby otestovat zda se nejedná o EOT. EOT je znak, který označuje konec spojení. Tento znak je vyvolán klávesovou zkratkou Ctrl + D. V případě že se nejedná o znak EOT je získaná zpráva překopírována do proměnné ukazující na strukturu TCP_SERVER_STRUCT. V dalším kroku je potřeba nahodit proměnou flagData, která slouží k informování o přijmutí zprávy. Poslední krok je zavolání metody acknowledge, která potvrdí přijmutí zprávy.

```
...
} else if(tss->state==TCP_ACCEPTED) {
        printf("State->ACCEPTED\r\n");

        tss->state = TCP_RECEIVED;
        tss->p = p;

        acknowledge(tpcb,tss);

        return ERR_OK;

} else if(tss->state == TCP_RECEIVED) {
        printf("State->RECEIVED\r\n");

        if(!tss->p) {
                tss->p = p;
                char *x;
                x = (char *)p->payload;
                if((int)(*x) != 0x04)
                {
                        strncpy(tss->data,(char*)p->payload,p->len);
                        tss->data[p->len] = '\0';
                        tss->flagData = true;
                }
                acknowledge(tpcb,tss);
        } else {
                struct pbuf *ptr;
                ptr = tss->p;
                pbuf_chain(ptr,p);
        }

        return ERR_OK;
}
...
```

Metoda acknowledge slouží k potvrzení zpráv od klienta. V prvním kroku se předá ukazatel na aktuální zprávu a délku této zprávy. V dalším kroku se provede předání ukazatele na následující zprávu, uvolnění paměti právě přečtené zprávy a zaslání potvrzení klientovi o obdržení zprávy.

```
void TCP_SERVER::acknowledge(struct tcp_pcb *tpcb, struct TCP_SERVER_STRUCT *tss){

        struct pbuf *ptr;
        ptr = tss->p;

        uint16_t p_len;
        p_len = ptr->len;

        tss->p = ptr->next;

        if(tss->p){
                pbuf_ref(tss->p);
        }

        pbuf_free(ptr);
        tcp_recved(tpcb,p_len);

}
```

4.2.2 Třída CAN_BUS

Třída CAN_BUS slouží ke komunikaci na sběrnici CAN. Třída CAN_BUS se skládá z veřejného konstruktoru a metod init, debug, run, setFilter, enableRXInterrupt, disableRXInterrupt, enableTXInterrupt, disableTXInterrupt, setTXBuffer, send, setRXBuffer, getRXBuffer a availableRXData. Privátní část třídy obsahuje proměnné canTxMsg pro zápis a canRxMsg pro čtení CAN zpráv z registrů. Pro signalizaci CAN zprávy slouží proměnná availableRxData. V proměnné message je zapsaná zpráva ve JSON formátu, která je posílaná přes TCP spojení.

```
class CAN_BUS {
public:
        CAN_BUS();
        bool init(uint32_t bitRate);
        void debug(uint32_t mode);
        void run();
        void setFilter();
        void enableRXInterrupt();
        void disableRXInterrupt();
        void enableTXInterrupt();
        void disableTXInterrupt();
        void setTXBuffer(char* message);
        void send();
        void setRXBuffer();
        const char* getRXBuffer();
        bool availableRXData();

private:
        CanTxMsg canTxMsg;
        CanRxMsg canRxMsg;
        bool availableRxData = false;
        char message[256];
};
```

Metoda init slouží k inicializaci komunikačnímu rozhraní CAN. Metoda je volána s parametrem bitRate, který stanovuje přenosovou rychlost. Při inicializaci je možné vybrat pět přenosových rychlostí 100 kbit/s, 125 kbit/s, 250 kbit/s, 500 kbit/s, 1 Mbit/s. Přenosová rychlost komunikačního rozhraní je možné měnit za běhu aplikace.

Metoda debug slouží k ladění komunikačnímu rozhraní CAN. Pomocí této metody je možné zapnout módy Silent mode nebo Loop back mode. Metoda run se slouží k zapnutí komunikačnímu rozhraní CAN. Tato metoda se může volat až po metodě init popřípadě

debug. Metoda setFilter má nastavená filtrační pravidla na příjem všech zpráv, která jsou posílaná přes komunikační sběrnici CAN. Metody enableRXInterrupt, disableRXInterrupt, enableTXInterrupt, disableTXInterrupt slouží k povolení nebo blokování přerušení od komunikačního rozhraní CAN.

Metoda setTXBuffer slouží k nastavení parametrů proměnné canTxMsg pro odeslání CAN zprávy. Parametrem této metody je znakový řetězec, který nese CAN zprávu ve formátu JSON. Zpráva ve formátu JSON je zaslána od TCP klienta. Pro zpracování JSON zprávy slouží funkce strtok, která vstupní zprávu převede na datovou strukturu [klíč] =>[hodnota]. Následně pomocí klíče je možné přiřadit hodnotu do požadované struktury CanTxMsg. Struktura CanTxMsg obsahuje proměnné pro identifikaci standardního nebo rozšířeného identifikátoru, dále délku zprávy a datové pole pro zprávu. Metoda send provádí přiřazení z datové struktury CanTxMsg do registru komunikačního rozhraní CAN.

Metoda setRXBuffer je volání od přerušení komunikačního rozhraní. Přijatá zpráva je zapsána do datové struktury CanRxMsg. Po skončení přerušení je z hlavní smyčky programu volána metoda getRXBuffer, která z přijaté CAN zprávy vytvoří řetězec ve formátu JSON, který je následně odeslán přes TCP spojení.

4.2.3 Třída USART

Třída USART slouží ke komunikaci na standardech RS-232 a RS-485. Třída USART se skládá z veřejného konstruktoru s parametrem a veřejných metod init, setBaudRate, enableRXInterrupt, disableRXInterrupt, enableTXInterrupt, disableTXInterrupt, enableUSART, disableUSART, sendTXBuffer, setTXBuffer, setRXBuffer, getRXBuffer a availableRXData. Privátní část třídy obsahuje konstruktor, který znemožňuje inicializovat instanci třídy USART bez parametru USART_TypeDef. Privátní část třídy dále obsahuje ukazatel na USART periferii a proměnné pro přístup k informacím o frekvenci dané periferie. Ve třídě se dále nachází struktura pro uložení odesílané, přijaté zprávy a informace o dostupnosti zprávy.

```
class USART {
public :
        USART(USART_TypeDef *usart);
        bool init();
        bool init(uint32_t baudRate);
        void setBaudRate(uint32_t baudRate);
        void enableRXInterrupt();
        void disableRXInterrupt();
        void enableTXInterrupt();
        void disableTXInterrupt();
        void enableUSART();
        void disableUSART();
        void sendTXBuffer();
        void setTXBuffer(const char *msg);
        void setRXBuffer();
        const char* getRXBuffer();
        bool availableRXData();

private:
        USART();
        USART_TypeDef *usart;
        RCC_ClocksTypeDef RCC_Clocks;
        uint32_t apb_clock;

        struct BUFFER {
                #define MSG_LEN 256
                char data[MSG_LEN];
                bool availableData = false;
        } RX_BUFFER, TX_BUFFER;

};
```

Konstruktor slouží k předání ukazatele na USART_TypeDef, který nese informaci o konkrétní periferii. Metody init slouží k inicializaci periferie. První metoda init bez parametru provádí inicializaci sériové komunikace na rychlost 38400 bit/s prostřednictvím volání druhé metody init. Druhá metoda init má parametr baudRate, který nese informaci o požadované přenosové rychlosti. Inicializace periferie probíhá inicializací konkrétních GPIO pinů, podle informace ze získaného ukazatele na USART_TypeDef přiřazeného pomocí konstruktoru. V další části metody se podle daného ukazatele zjistí, na které konkrétní APB sběrnici se daná periferie nachází. Následně se provede zapnutí, restartovaní periferie a získání frekvence APB sběrnice pro následné nastavení přenosové rychlosti pomocí metody setBaudRate.

Metoda setBaudRate slouží k nastavení přenosové rychlosti sériové komunikace. Tato metoda umožňuje přenastavit přenosovou rychlost za běhu aplikace pokud přijde požada-

vek od klientské aplikace. Metody enableRXInterrupt, disableRXInterrupt, enableTXInterrupt a disableTXInterrupt slouží k povolení a blokování přerušení.

Metoda setTXBuffer slouží k zapsání zprávy do struktury BUFFER pro následné odeslání zprávy pomocí metody sendTXBuffer. Metoda sendTXBuffer je volána od přerušení dané periferie. Při každém přerušení dojde k odeslání jednoho znaku dokud se nedostane na konec řetězce. V takovém případě dojde k zavolání metody disableTXInterrupt pro vypnutí přerušení pro odesílání.

Metoda setRXBuffer je volána od přerušení dané periferie. Při každém přerušení dojde k přečtení jednoho znaku, který je zapsán do struktury BUFFER. Tento postup se provádí dokud není přečtený znak \r nebo konec řetězce. Následně se nahodí příznak dostupné zprávy k odeslání přes TCP spojení. Metody availableRXData a getRXBuffer slouží k informování o dostupnosti zprávy a získání zprávy.

4.2.4 Třída GPIO

Třída GPIO slouží k inicializaci a ovládání pinů mikrokontroléru. Veřejná část třídy GPIO se skládá z konstruktoru a metod initPin, toggle, readPin, write, setAF. Privátní část třídy obsahu ukazatel na daný GPIO pin a číslo bitu daného GPIO bloku.

4.2.5 Třída IO_BLOCK

Tříd IO_BLOCK složí k ovládání řídících signálů a sledování stavových signálů na komunikační jednotce. Veřejná část třídy IO_BLOCK obsahuje konstruktor a metody write a readInputPins. Privátní část obsahuje dvojici proměnných datové struktury std::vector typu GPIO.

Konstruktor provádí inicializaci řídících a stavových GPIO pinů pomocí třídy GPIO. Inicializované GPIO piny jsou vloženy do struktury std::vector.

Metoda write slouží k ovládání řídících signálů komunikační jednotky. Metoda readInputPins slouží ke čtení stavových signálů. Metoda prochází všechny piny uložené ve struktuře std::vector, přečte jejich logickou hodnotu a zapíše jejich hodnotu do výstupního řetězce, který je následně odeslán do klientské aplikace k zobrazení aktuálního stavu.

4.2.6 Hlavní program komunikační jednotky

Hlavní program se skládá z globálních instancí tříd USART,GPIO,CAN a IO_BLOCK, dále přerušovacích funkcí od periferií, konfigurační funkce a main funkce. Pro obsluhu komunikačních standardů z přerušovacích funkcí a funkce main byly instance komunikačním standardů vytvořeny jako globální. Pro komunikační standard RS-232 jsou vytvořeny tři globální instance třídy USART. První instance slouží pro komunikaci mezi komunikační jednotku a externí jednotkou. Tato instance je spojena s periferii USART2. Druhá a třetí instance slouží k odposlechu komunikace mezi dvěma externími jednotkami. Tyto instance

jsou navázány na periferie UART4 a UART5. Pro komunikační standard RS-485 je vytvoření globální instance třídy USART navázána na periferii USART6, dále pro řízení přístupu ke sběrnici jsou vytvořeny instance DE a RE z třídy GPIO. Pro CAN slouží globální instance třídy CAN_BUS. Uvedené instance komunikačních standardů RS-232, RS-485 a CAN jsou po spuštění inicializovány na přenosové rychlosti:

- Přenosová rychlost pro RS-232 je 38400 bit/s.
- Přenosová rychlost pro RS-485 je 1 Mbit/s.
- Přenosová rychlost pro CAN je 100 kbit/s.

Po inicializaci komunikačních instancí se provede inicializace Ethernetového rozhraní. Po této inicializaci se provede inicializace pěti instancí třídy TCP_SERVER. Instance třídy TCP_SERVER slouží ke konfiguraci a komunikaci s instancemi komunikačních rozhraní.

Pro konfiguraci komunikační jednotky je dostupné spojení na portu 5000. Konfigurace obsahuje možnost měnit přenosové rychlosti komunikačních standardů RS-232, RS-485 a CAN, dále je možné ovládat řídící signály a restartovat komunikační jednotku. V intervalu 500 ms je odesílán stav stavových signálů. Pro konfiguraci komunikační jednotky slouží funkce config.

Pro odesílání a příjem zpráv od prvního kanálu RS-232 je dostupné spojení na portu 5001. Pro odposlech komunikace mezi dvěma externími jednotkami je dostupné spojení na portu 5002. Komunikační standard CAN je dostupný na portu 5003. Pro standard RS-485 je dostupné spojení na portu 5004.

4.3 Navázání spojení s komunikační jednotkou

Pro navázání spojení a komunikaci s komunikační jednotku přes TCP spojení je možné na distribucích operačních systémů Linux a macOS využít terminálový příkaz telnet. V případě operačního systému Windows je možné využit program Putty. Navázání spojení prostřednictvím terminálu se skládá z příkazu telnet, IP adresy a portu: *telnet 192.168.1.250 5000*. Konfigurace komunikační jednotky na portu 5000 probíhá prostřednictvím zpráv ve formátu JSON. Ukázkové zprávy obsahuji informace o nastavení přenosové rychlosti zvolené periférie a ovládání řídících pinů.

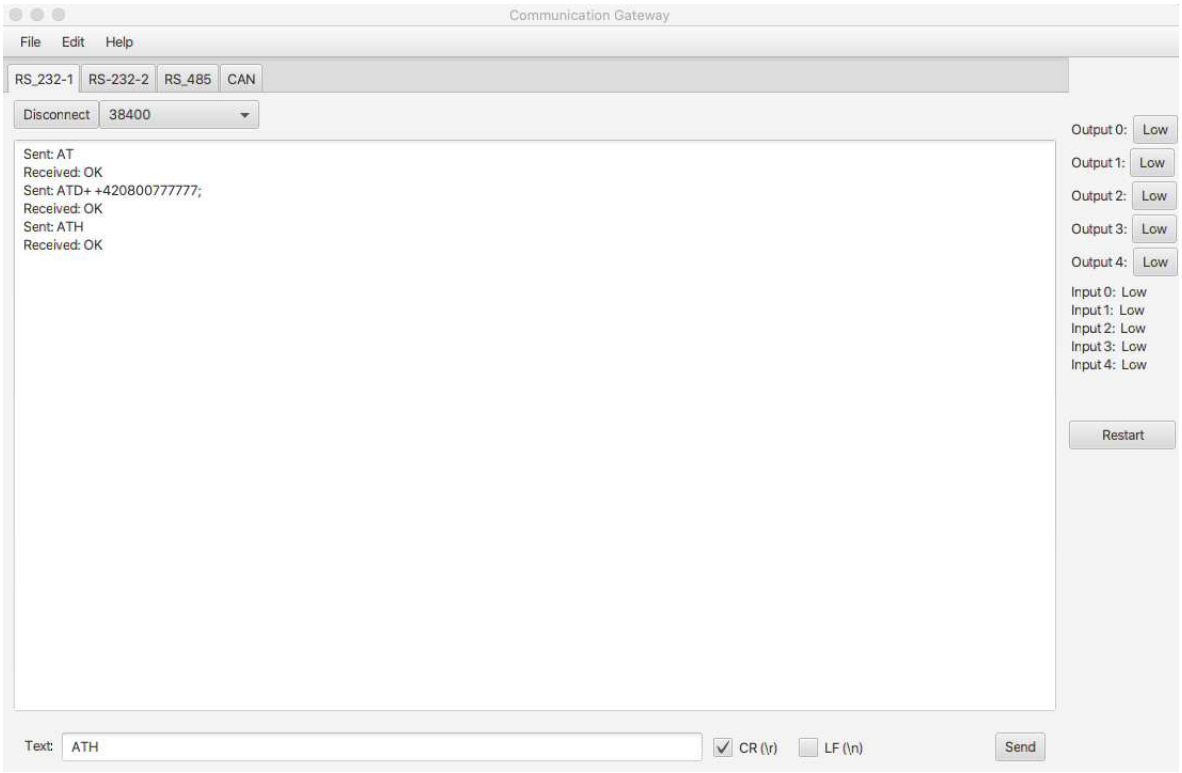
```
{"USART2":"38400"}
{"USART4":"38400","USART5":"38400"}
{"CANBUS":"250000"}
{"OUTPUT3":"1"}
```

Pro komunikační rozhraní RS-232 a RS-485 jsou zprávy odesílány a přijímány v „plain text“ formátu. Pro komunikační rozhraní CAN jsou zprávy odesílány a přijímány v textovém formátu JSON. Ukázková CAN zpráva ve formátu JSON obsahuje identifikátor, délku zprávy a pole bajtů.

```
{"StdId":"22","DLC":"3","D0":"88","D1":"33","D2":"22"}
```

4.4 Klientská aplikace

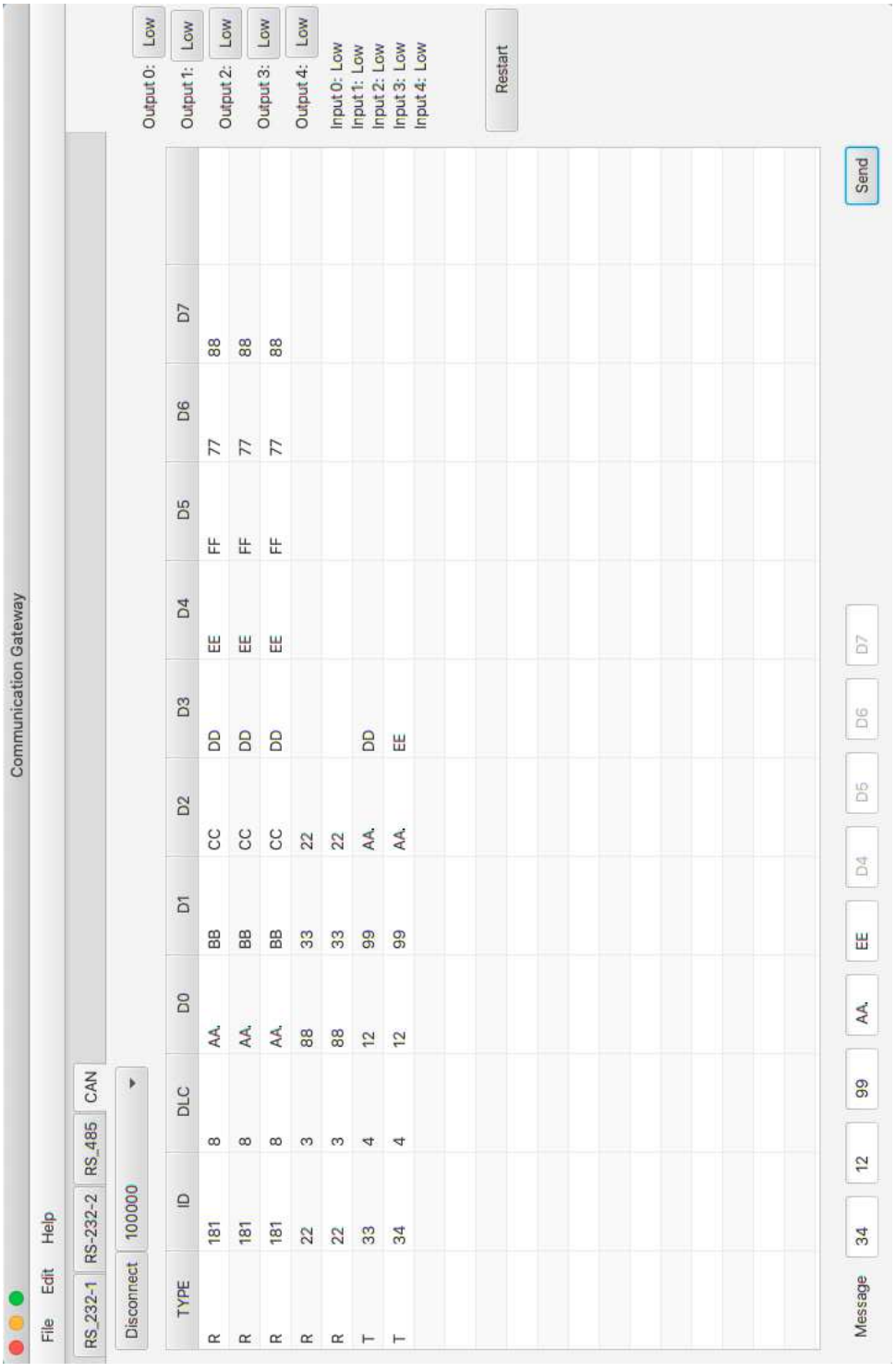
V kapitole 4.3 byl popsán způsob, jak navázat spojení s komunikační jednotkou, avšak zmíněny způsob je spíše vhodný pro automatizované zpracování dat, avšak složitý pro zadávání obsluhou. Za tímto účelem byla vytvořena multiplatformní aplikace v programovacím jazyce Java, která umožňuje jednoduchým způsobem konfigurovat a ovládat komunikační jednotku. Náhled aplikace je znázorněn na obr. 4.1.



Obr. 4.1: Klientská aplikace

Aplikace je rozdělena do čtyř záložek. Každá záložka umožnuje přístup ke konkrétnímu komunikačnímu rozhraní a provádět jednoduchým způsobem změnu přenosové rychlosti. Pro komunikační rozhraní RS-232 a RS-485 je k dispozici výpis komunikace na daném rozhraní, dále textový vstup s možnou volbou znaku \r a \n.

V záložce CAN se zobrazují zprávy od komunikačního rozhraní CAN. Zprávy zobrazené v tabulce jsou rozšířené o označení R což označuje přijatou zprávu a T odeslanou zprávu. Pro odeslání zprávy je k dispozici pole 9 vstupních prvků, které slouží k zadání identifikátoru a obsahu zprávy. Délka zprávy DLC se dopočítá automaticky při odesílání. V aplikaci je možné ovládat řídící signály označené jako „Output X:“ a číst stavové signály označené jako „Input X:“.



Obr. 4.2: Klientská aplikace - CAN zprávy

5

Měření elektromagnetické kompatibility

U vytvořené komunikační jednotky byly provedeny následující testy:

- Test odolnosti proti elektrostatickému výboji
  ČSN EN 61000-4-2
- Test odolnosti zařízení proti rychlým přechodovým jevům - skupině impulzů
  ČSN EN 61000-4-4
- Emise šířené vedením 150 kHz – 30 MHz
  ČSN EN 61000-6-3
- Vyzařované emise 30 – 1000 MHz
  ČSN EN 61000-6-3

Funkční kritéria

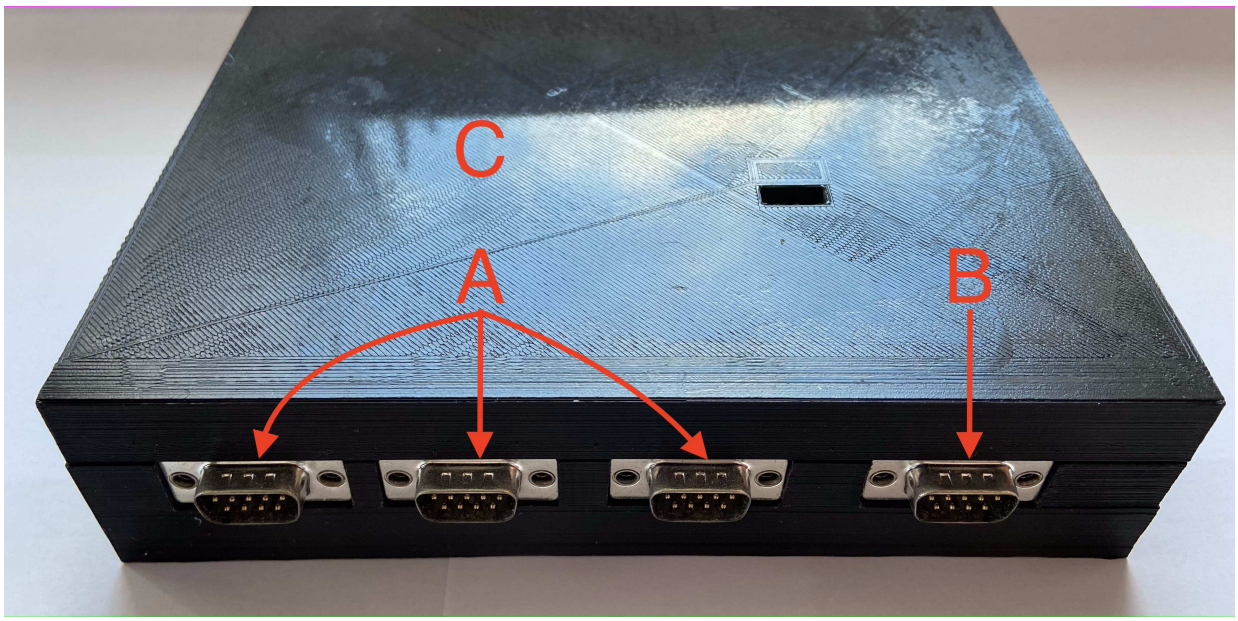
- **Funkční kritérium A**: Přístroj musí pracovat nepřetržitě během zkoušky i po ní dle svého určení. Není dovoleno žádné zhoršení činnosti nebo ztráta funkce pod úroveň stanovenou výrobcem.
- **Funkční kritérium B**: Přístroj musí po zkoušce pracovat nepřetržitě dle svého určení. Není dovoleno žádné zhoršení činnosti nebo ztráta funkce pod stanovenou výrobcem.
- **Funkční kritérium C**: Je dovolena dočasná ztráta funkce za předpokladu, že funkce je samoobnovitelná nebo může být obnovena řízením.

5.1 Test odolnosti proti elektrostatickému výboji

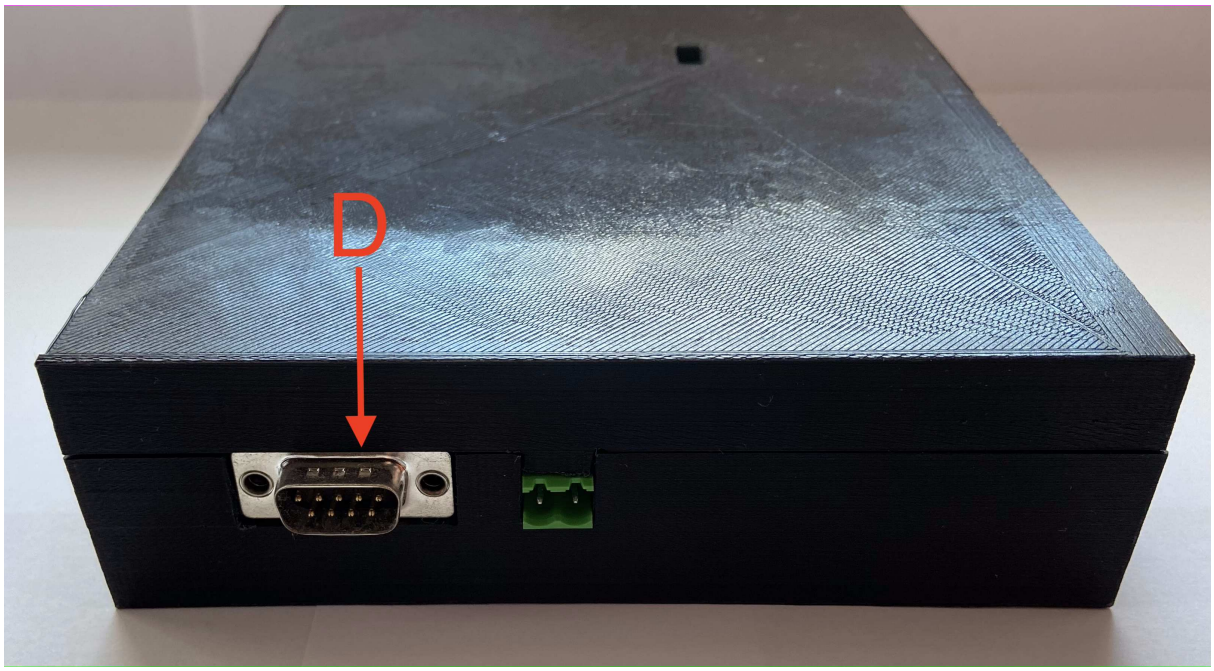
Kontaktní výboj		
Označení místa	Úroveň	Výsledek testu
CAN (viz obr. 5.1 bod B)	$+4kV$	A
CAN (viz obr. 5.1 bod B)	$-4kV$	A
RS-485 (viz obr. 5.2 bod D)	$+4kV$	A
RS-485 (viz obr. 5.2 bod D)	$-4kV$	A
RS-232 (viz obr. 5.1 body A)	$+4kV$	C
RS-232 (viz obr. 5.1 body A)	$-4kV$	C

Vzduchový výboj		
Označení místa	Úroveň	Výsledek testu
Kryt zařízení (viz obr. 5.1 bod C)	$+8kV$	A
Kryt zařízení (viz obr. 5.1 bod C)	$-8kV$	A

Tab. 5.1: Výsledky testů dle ČSN EN 61000-4-2 ed.2: 2009



Obr. 5.1: Místa aplikace ESD



Obr. 5.2: Místo aplikace ESD

5.2 Test odolnosti zařízení proti rychlým přechodovým jevům - skupině impulzů

AC, vazba L+N+PE, opakovací kmitočet 5 kHz	
Úroveň	Výsledek testu
$+1kV$	A
$-1kV$	A

DC, vazba + - -, opakovací kmitočet 5 kHz	
Úroveň	Výsledek testu
$+500V$	A
$-500V$	A

RS-485, kapacitní kleština, opakovací kmitočet 5 kHz	
Úroveň	Výsledek testu
$+500V$	A
$-500V$	A

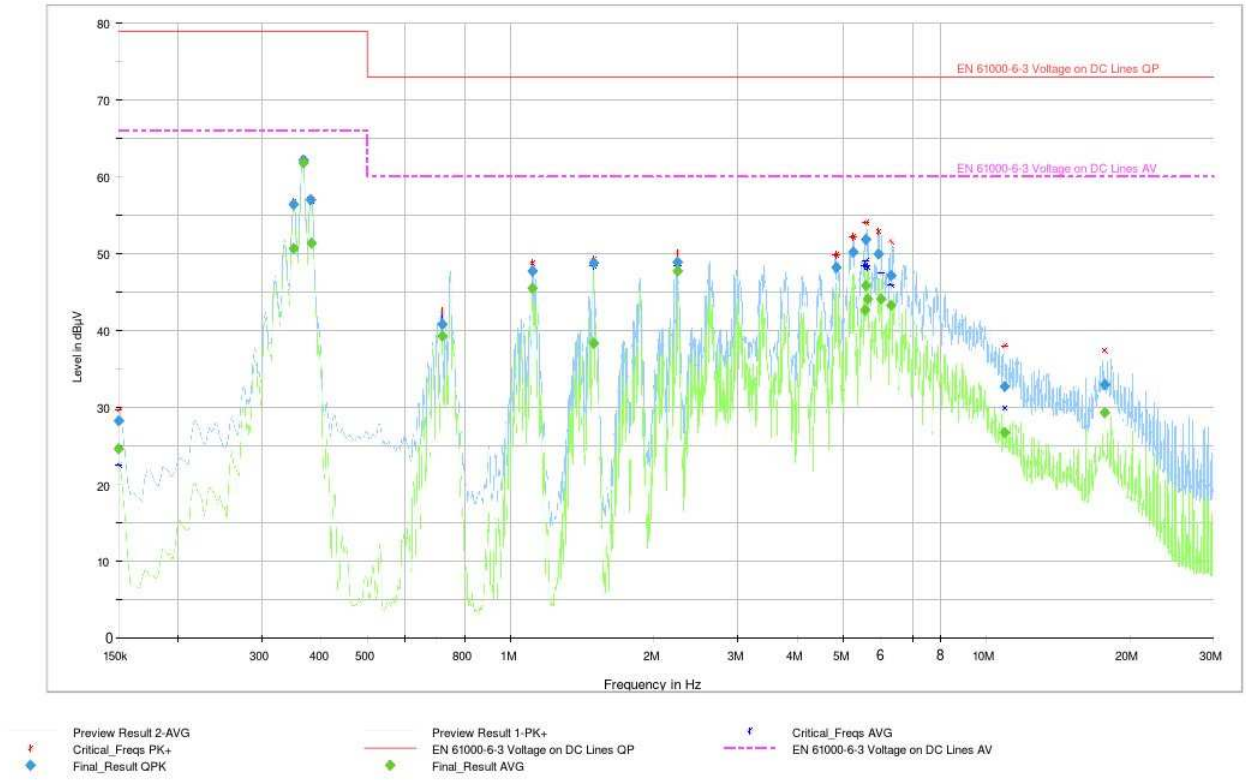
RS-232, kapacitní kleština, opakovací kmitočet 5 kHz	
Úroveň	Výsledek testu
$+500V$	A
$-500V$	A

CAN, kapacitní kleština, opakovací kmitočet 5 kHz	
Úroveň	Výsledek testu
$+500V$	A
$-500V$	A

Tab. 5.2: Výsledky testů dle ČSN EN 61000-4-4 ed.3: 2013

Obr. 5.3: Konfigurace zkoušeného zařízení při testu odolnosti proti rychlým přechodovým jevům / skupině impulzů

5.3 Emise šířené vedením 150 kHz – 30 MHz

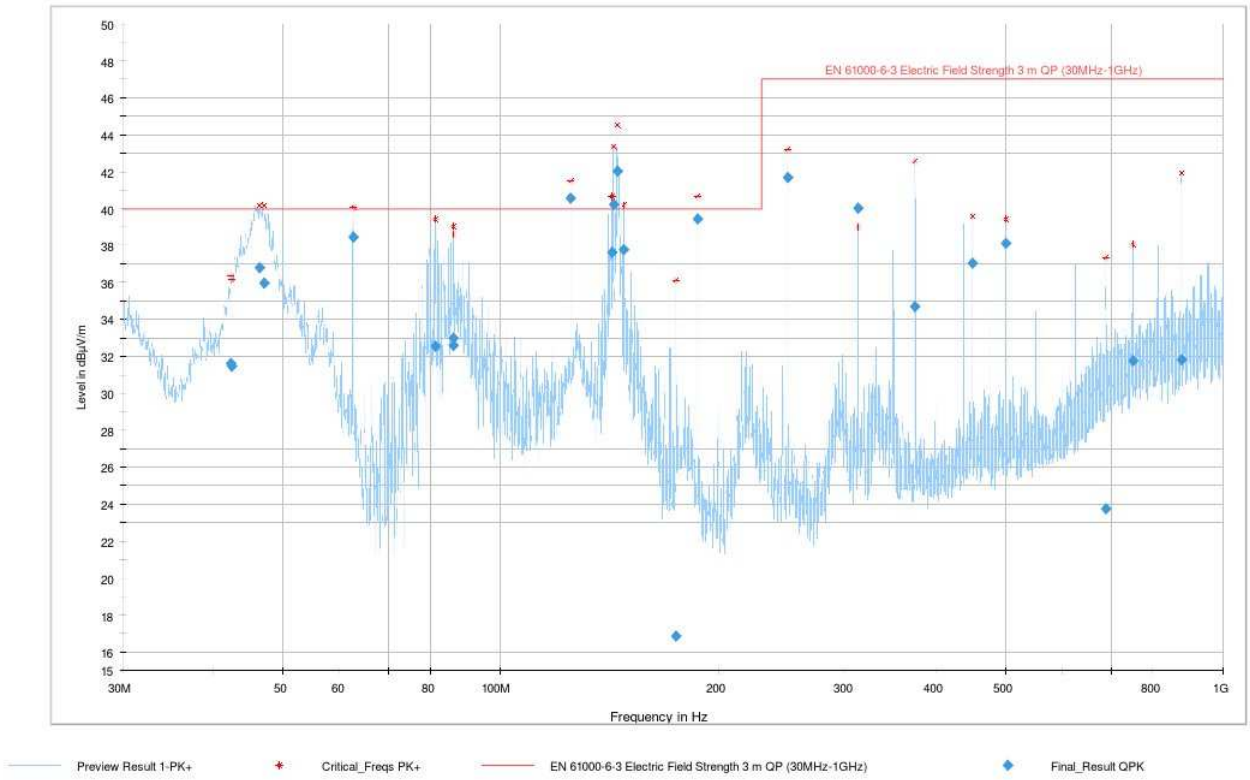


Obr. 5.4: Emise šířené po napájecím vedení 150 kHz – 30 MHz, LISN

Frequency	Process State	QuasiPeak	Average	Limit	Margin	Meas. Time	Bandwidth	Line	PE
[MHz]		[dBuV]	[dBuV]	[dBuV]	[dB]	[ms]	[kHz]		
0,150000	FINAL	—	24,61	66,00	41,39	1000,0	9,000	+	GND
0,150000	FINAL	28,32	—	79,00	50,68	1000,0	9,000	+	GND
0,350250	FINAL	—	50,69	66,00	15,31	1000,0	9,000	+	GND
0,350250	FINAL	56,38	—	79,00	22,62	1000,0	9,000	+	GND
0,366000	FINAL	—	61,81	66,00	4,19	1000,0	9,000	+	GND
0,366000	FINAL	62,14	—	79,00	16,86	1000,0	9,000	+	GND
0,379500	FINAL	57,02	—	79,00	21,98	1000,0	9,000	+	GND
0,381750	FINAL	—	51,37	66,00	14,63	1000,0	9,000	+	GND
0,717000	FINAL	40,78	—	73,00	32,22	1000,0	9,000	+	GND
0,717000	FINAL	—	39,33	60,00	20,67	1000,0	9,000	+	GND
1,110750	FINAL	—	45,53	60,00	14,47	1000,0	9,000	+	GND
1,110750	FINAL	47,77	—	73,00	25,23	1000,0	9,000	+	GND
1,493250	FINAL	48,74	—	73,00	24,26	1000,0	9,000	+	GND
1,493250	FINAL	—	38,3	60,00	21,70	1000,0	9,000	+	GND
2,238000	FINAL	—	47,76	60,00	12,24	1000,0	9,000	+	GND
2,238000	FINAL	48,88	—	73,00	24,12	1000,0	9,000	+	GND
4,825500	FINAL	48,17	—	73,00	24,83	1000,0	9,000	+	GND
5,253000	FINAL	50,21	—	73,00	22,79	1000,0	9,000	+	GND
5,543250	FINAL	—	42,71	60,00	17,29	1000,0	9,000	+	GND
5,574750	FINAL	51,79	—	73,00	21,21	1000,0	9,000	+	GND
5,574750	FINAL	—	45,88	60,00	14,12	1000,0	9,000	+	GND
5,619750	FINAL	—	44,14	60,00	15,86	1000,0	9,000	+	GND
5,934750	FINAL	49,93	—	73,00	23,07	1000,0	9,000	+	GND
6,000000	FINAL	—	44,05	60,00	15,95	1000,0	9,000	+	GND
6,288000	FINAL	47,12	—	73,00	25,88	1000,0	9,000	+	GND
6,290250	FINAL	—	43,27	60,00	16,73	1000,0	9,000	+	GND
10,909500	FINAL	32,73	—	73,00	40,27	1000,0	9,000	+	GND
10,909500	FINAL	—	26,75	60,00	33,25	1000,0	9,000	+	GND
17,693250	FINAL	—	29,27	60,00	30,73	1000,0	9,000	+	GND
17,695500	FINAL	32,95	—	73,00	40,05	1000,0	9,000	+	GND

Tab. 5.3: Emise šířené po napájecím vedení 150 kHz – 30 MHz, LISN

5.4 Vyzařované emise 30 – 1000 MHz

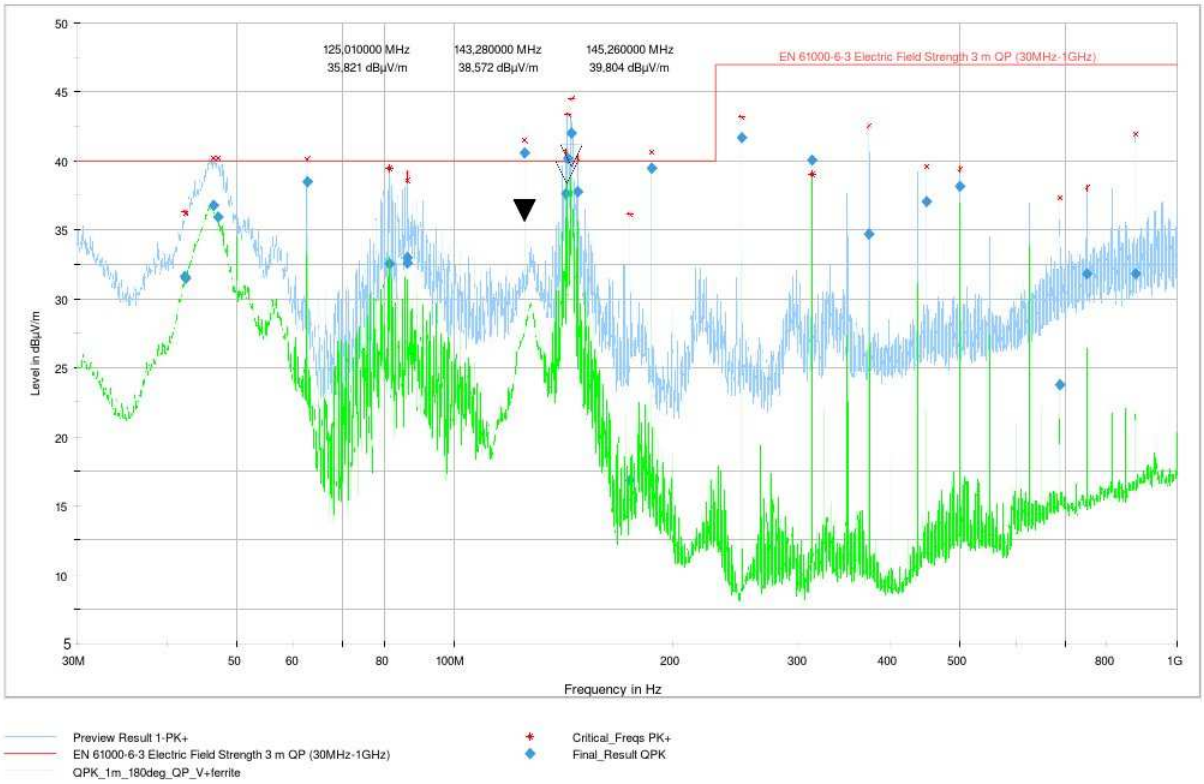


Obr. 5.5: Vyzařované emise 30 - 1000 MHz

Frequency	Process State	QuasiPeak	Limit	Margin	Meas. Time	Bandwidth	Height
42,360000	FINAL	31,60	40,00	8,40	1000,0	120,000	100,0
42,510000	FINAL	31,48	40,00	8,52	1000,0	120,000	100,0
46,410000	FINAL	36,82	40,00	3,18	1000,0	120,000	100,0
47,070000	FINAL	35,95	40,00	4,05	1000,0	120,000	100,0
62,490000	FINAL	38,47	40,00	1,53	1000,0	120,000	100,0
81,210000	FINAL	32,56	40,00	7,44	1000,0	120,000	100,0
86,040000	FINAL	33,00	40,00	7,00	1000,0	120,000	100,0
86,130000	FINAL	32,58	40,00	7,42	1000,0	120,000	100,0
125,010000	FINAL	40,59	40,00	-0,59	1000,0	120,000	100,0
142,710000	FINAL	37,65	40,00	2,35	1000,0	120,000	100,0
143,310000	FINAL	40,20	40,00	-0,20	1000,0	120,000	200,0
145,260000	FINAL	42,04	40,00	-2,04	1000,0	120,000	100,0
148,110000	FINAL	37,78	40,00	2,22	1000,0	120,000	100,0
174,930000	FINAL	16,85	40,00	23,15	1000,0	120,000	100,0
187,500000	FINAL	39,44	40,00	0,56	1000,0	120,000	200,0
249,990000	FINAL	41,69	47,00	5,31	1000,0	120,000	100,0
312,510000	FINAL	40,03	47,00	6,97	1000,0	120,000	100,0
375,000000	FINAL	34,69	47,00	12,31	1000,0	120,000	100,0
450,000000	FINAL	37,05	47,00	9,95	1000,0	120,000	200,0
500,010000	FINAL	38,14	47,00	8,86	1000,0	120,000	100,0
687,480000	FINAL	23,77	47,00	23,23	1000,0	120,000	100,0
750,000000	FINAL	31,79	47,00	15,21	1000,0	120,000	100,0
875,010000	FINAL	31,83	47,00	15,17	1000,0	120,000	100,0

Tab. 5.4: Vyzařované emise 30 – 1000 MHz

5.5 Vyzařované emise 30 - 1000 MHz s použitím feritového jádra na Ethernetu



Obr. 5.6: Vyzařované emise 30 – 1000 MHz s použitím feritového jádra na Ethernetu

5.6 Vyhodnocení výsledků měření

V rámci měření elektromagnetické kompatibility byly provedeny následující testy:

- ČSN EN 61000-4-2
- ČSN EN 61000-4-4
- ČSN EN 61000-6-3

Při testu RS-232 dle ČSN EN 61000-4-2 došlo k přerušení komunikace na RS-232 a byl nutný vzdálený restart testovaného zařízení. Zařízení dle testu 4-2 splňuje funkční kritérium C, čím není ve shodě s požadavky normy ČSN EN 61000-6-1.

Z výsledku prvního měření vyzářených emisí v rozsahu 30-1000 MHz, byl na 3 hodnotách překročen povolený limit. U druhého měření bylo použito feritové jádro na Ethernetovém kabelu, naměřené vyzářené emise byly pod stanoveným limitem. Rušení bylo způsobeno nevhodným kabelem UTP CAT5. Tento problém může být vyřešen kabelem FTP CAT5 nebo STP CAT5.

6

Závěr

Cílem této diplomové práce byl návrh a realizace komunikační jednotky pro vzdálenou správu zařízení připojených ke standardům RS-232 a CAN. Komunikační jednotka pro vzdálený přistup využívá připojení prostřednictvím rozhraní Ethernet.

V rámci diplomové práce byly splněny všechny body zadání. Navíc oproti zadání byla navržená komunikační jednotka rozšířena o standard RS-485, stavové a řídící signály. Pro implementaci komunikační jednotky byla zvolena vývojová deska NUCLEO-F429ZI s mikrokontrolérem STM32F429. Pří návrhu bylo zvoleno řešení umožňující v budoucnu vyměnit vývojovou desku NUCLEO-F429ZI za jiný model např. NUCLEO-F767ZI nebo ESP32, který disponuje Ethernet i WIFI rozhraním. Při návrhu komunikační jednotky byly všechny komunikační standardy galvanicky odděleny od mikrokontroléru.

Pro vzdálené ovládání komunikační jednotky byla vytvořena multiplatformní aplikace v programovacím jazyce Java. Vytvořená aplikace umožňuje konfigurovat komunikační jednotku dále umožňuje přístup ke zvolenému komunikačnímu rozhraní, provádět odesílání a příjem zpráv. U vytvořené komunikační jednotky bylo provedeno měření elektromagnetické kompatibility včetně závěrečného vyhodnocení výsledků. V příloze se nachází kompletní sada schémat zapojení a snímky vytvořené komunikační jednotky. V budoucnu bych se rád věnoval možnostem přenosu dat na RS-232, RS-485 a CAN pomocí dvou komunikačních jednotek propojené prostřednictvím rozhraní Ethernet.

Literatura

[1] AXELSON, Jan. *Serial port complete: COM ports, USB virtual COM ports, and ports for embedded systems. 2nd ed.* Madison, WI: Lakeview Research, 2007. ISBN 978-1-931-44806-2.

[2] KRIST, Petr. *Průmyslové sběrnice a elektromagnetická kompatibilita.* Plzeň, 2005. Rigorózní práce. Západočeská univerzita v Plzni, Fakulta elektrotechnická, Katedra aplikované elektroniky a telekomunikací. Vedoucí práce Prof. Ing. Jiří Pinker, CSc.

[3] KRIST, Petr. *Moderní principy průmyslových komunikací.* Plzeň, 2008. Disertační práce. Západočeská univerzita v Plzni, Fakulta elektrotechnická, Katedra aplikované elektroniky a telekomunikací. Vedoucí práce Prof. Ing. Jiří Pinker, CSc.

[4] KOSTURIK, Kamil. *Řídící a informační sběrnice: Úvod.* 2019.

[5] KOSTURIK, Kamil. *Řídící a informační sběrnice: Controller Area Network.* 2019.

[6] DI NATALE, Marco. *Understanding and using the controller area network communication protocol: theory and practice.* New York: Springer, 2012. ISBN 14-614-0313-8.

[7] ISO724x *High-Speed, Quad-Channel Digital Isolators [online]. Texas Instruments*, 2017 [cit. 2020-01-24]. Dostupné z: http://www.ti.com/lit/ds/slls868t/slls868t.pdf

[8] MAX3232 *3-V to 5.5-V Multichannel RS-232 Line Driver/Receiver [online]. Texas Instruments*, 2017 [cit. 2020-01-24]. Dostupné z: http://www.ti.com/lit/ds/symlink/max3232.pdf

[9] SN65HVD3086E *LOW-POWER RS-485 FULL-DUPLEX DRIVERS/RECEIVERS [online]. Texas Instruments*, 2012 [cit. 2020-01-24]. Dostupné z: http://www.ti.com/lit/ds/symlink/sn65hvd3086e.pdf

[10] ISO742x *Low-Power Dual-Channel Digital Isolators [online]. Texas Instruments*, 2015 [cit. 2020-01-24]. Dostupné z: http://www.ti.com/lit/ds/symlink/iso7421.pdf

[11] PCA82C251 *CAN transceiver for 24 V systems [online]. NXP Semiconductors*, 2011 [cit. 2020-01-24]. Dostupné z: https://www.tme.eu/Document/210dcdcea9559676b5329ae0aa33f1c0/PCA82C251T.pdf

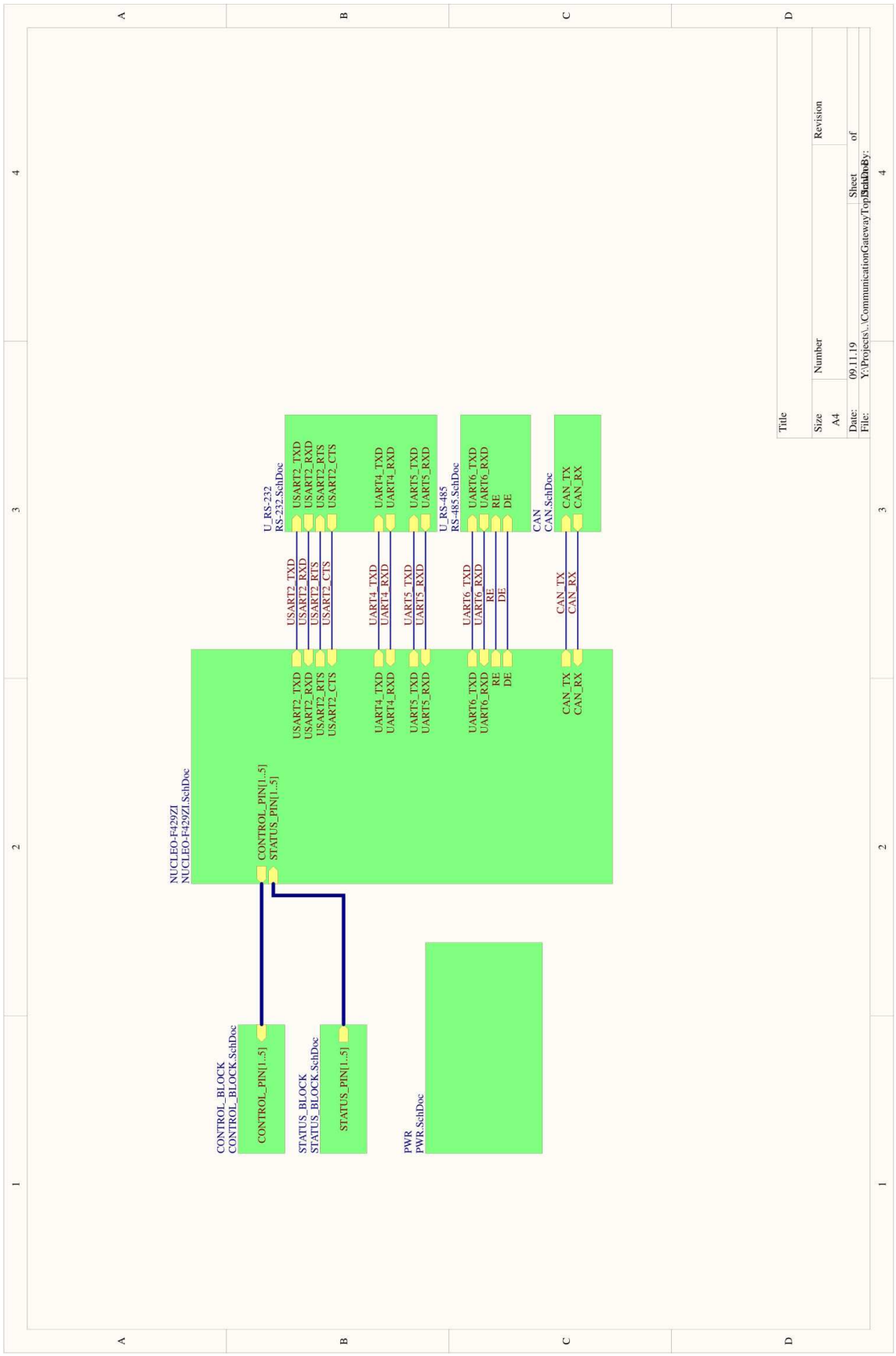
[12] LwIP - lightweight TCP/IP. *LwIP Wiki [online].* [cit. 2020-02-22]. Dostupné z: https://lwip.fandom.com/wiki/LwIP_Wiki

[13] LwIP Application Developers Manual. *LwIP Wiki [online].* [cit. 2020-02-22]. Dostupné z: https://lwip.fandom.com/wiki/LwIP_Application_Developers_Manual

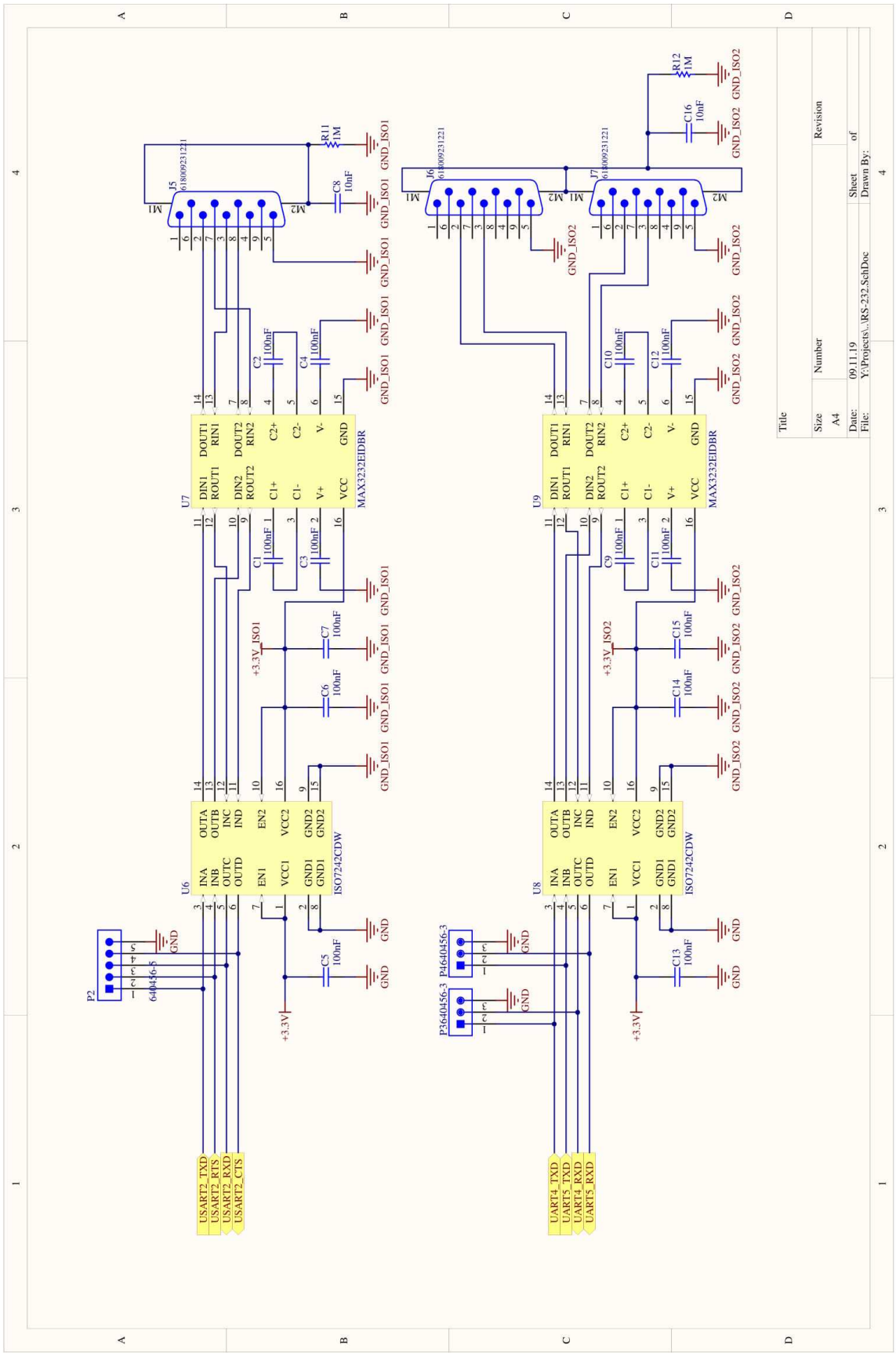
[14] LwIP_TCP_Echo_Server. *GitHub [online].* [cit. 2020-02-23]. Dostupné z: https://github.com/STMicroelectronics/STM32CubeF4/tree/master/Projects/STM324x9I_EVAL/Applications/LwIP/LwIP_TCP_Echo_Server

Příloha A

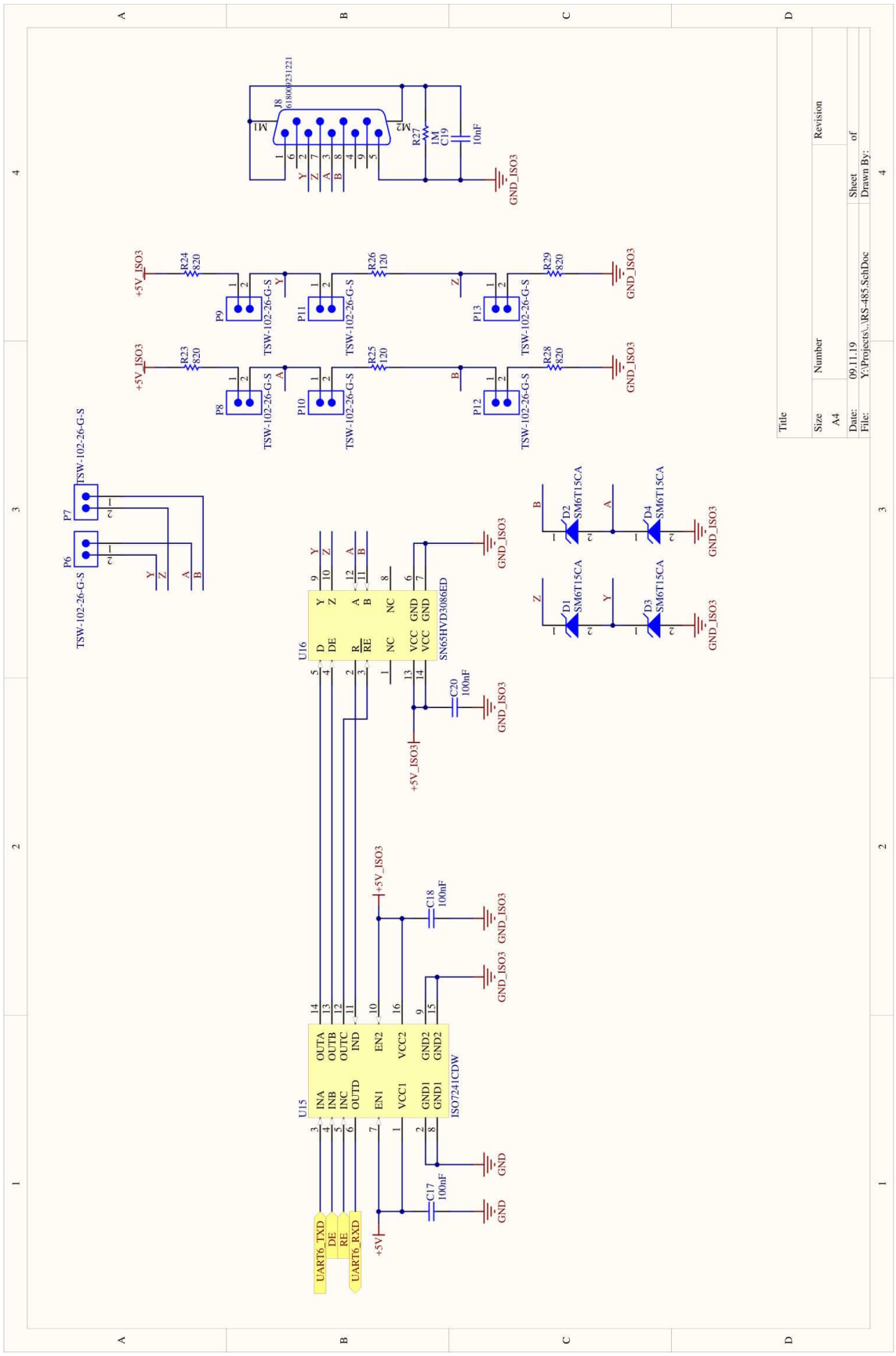
Komunikační jednotka



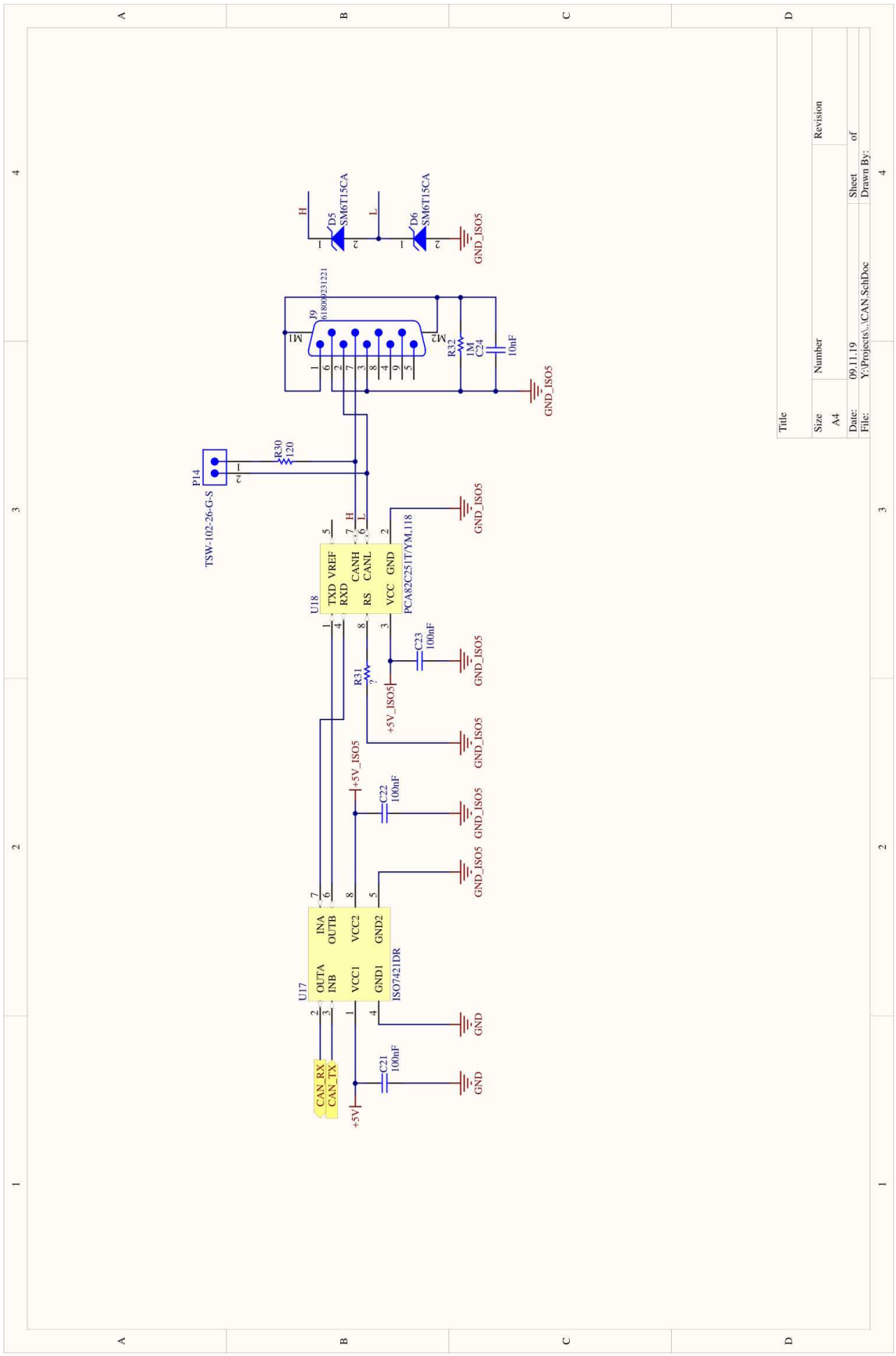
Obr. A.1: Blokové schéma komunikační jednotky



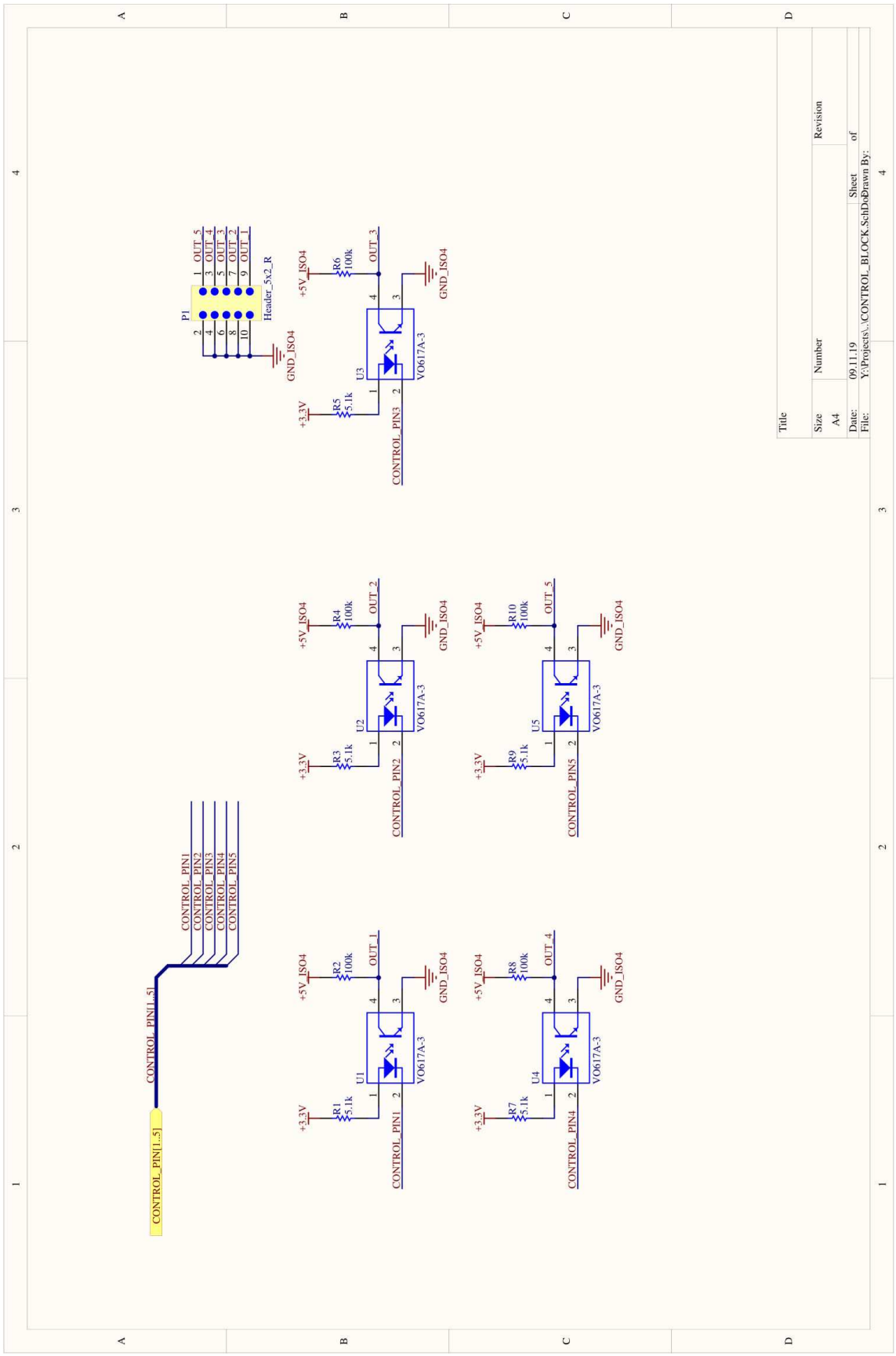
Obr. A.2: Zapojení RS-232



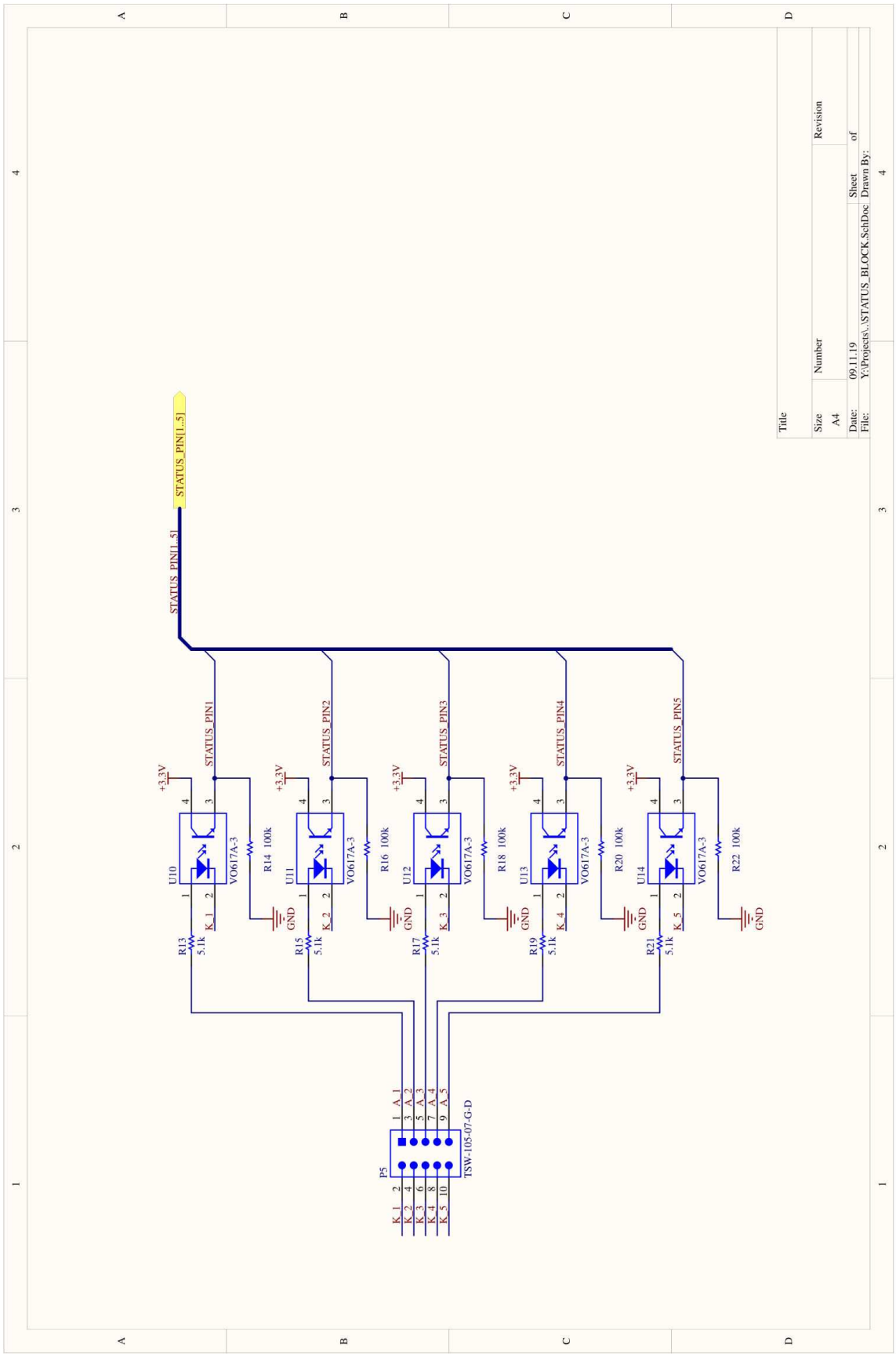
Obr. A.3: Zapojení RS-485



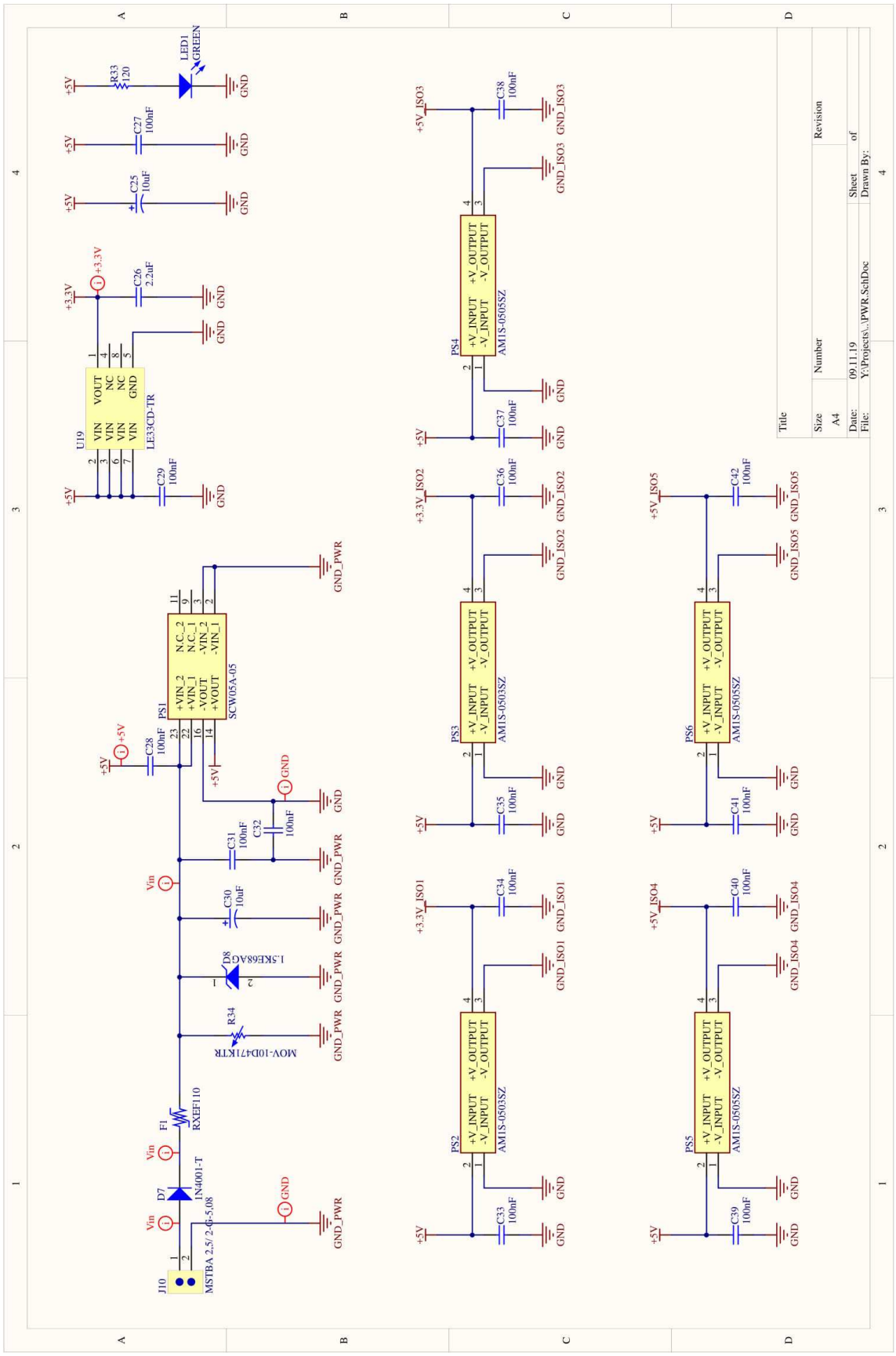
Obr. A.4: Zapojení CAN



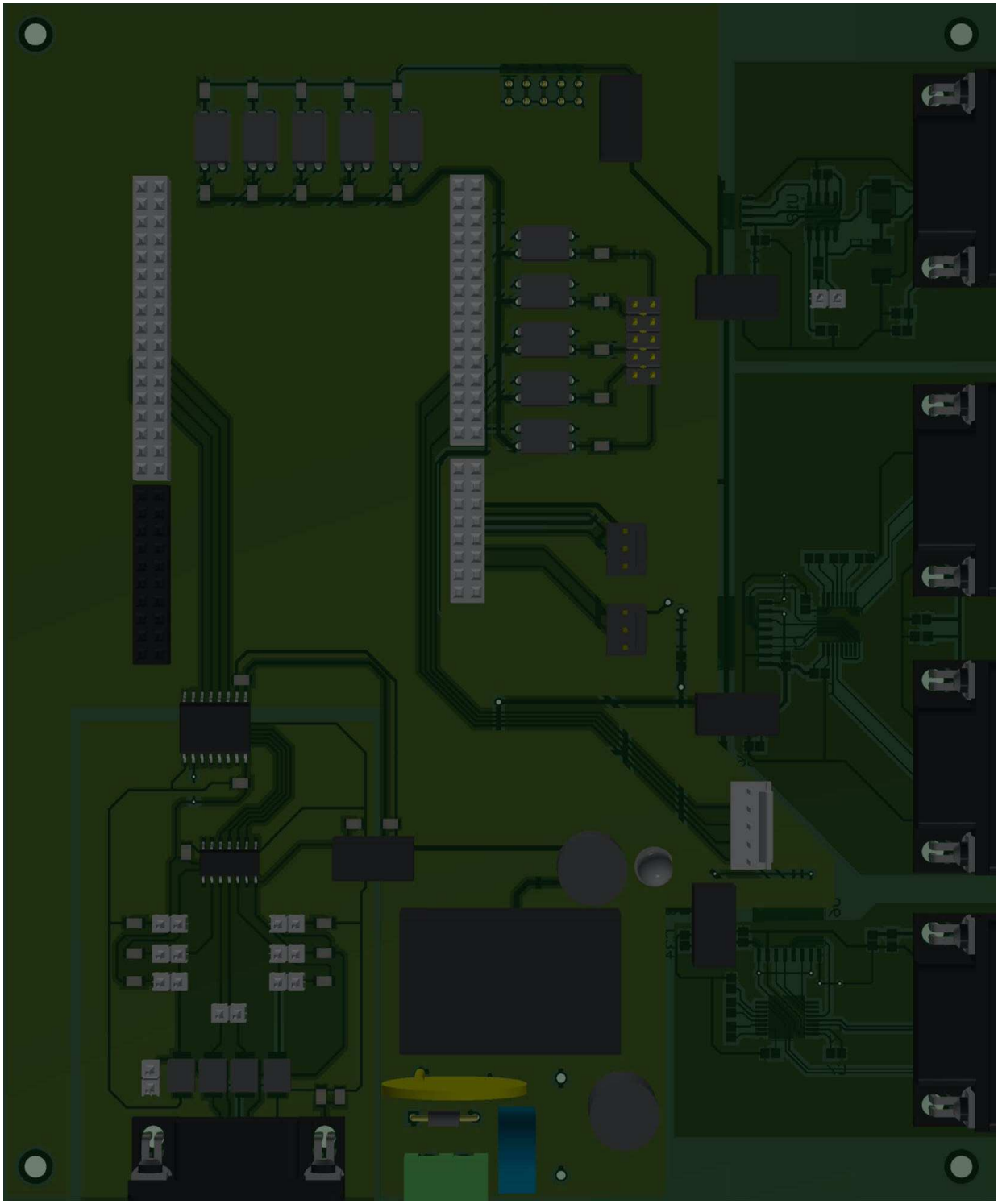
Obr. A.5: Zapojení řídících signálů



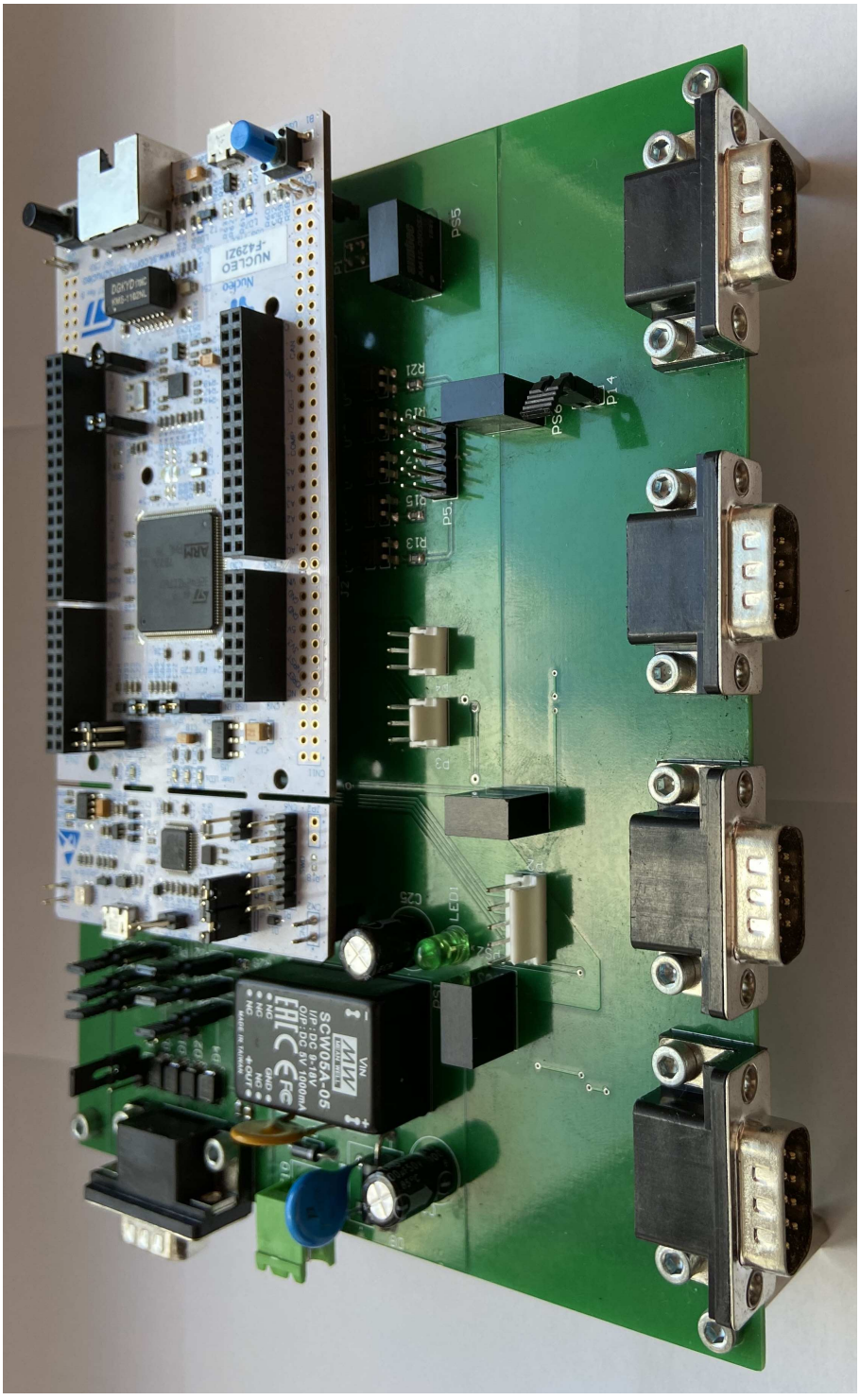
Obr. A.6: Zapojení stavových signálů



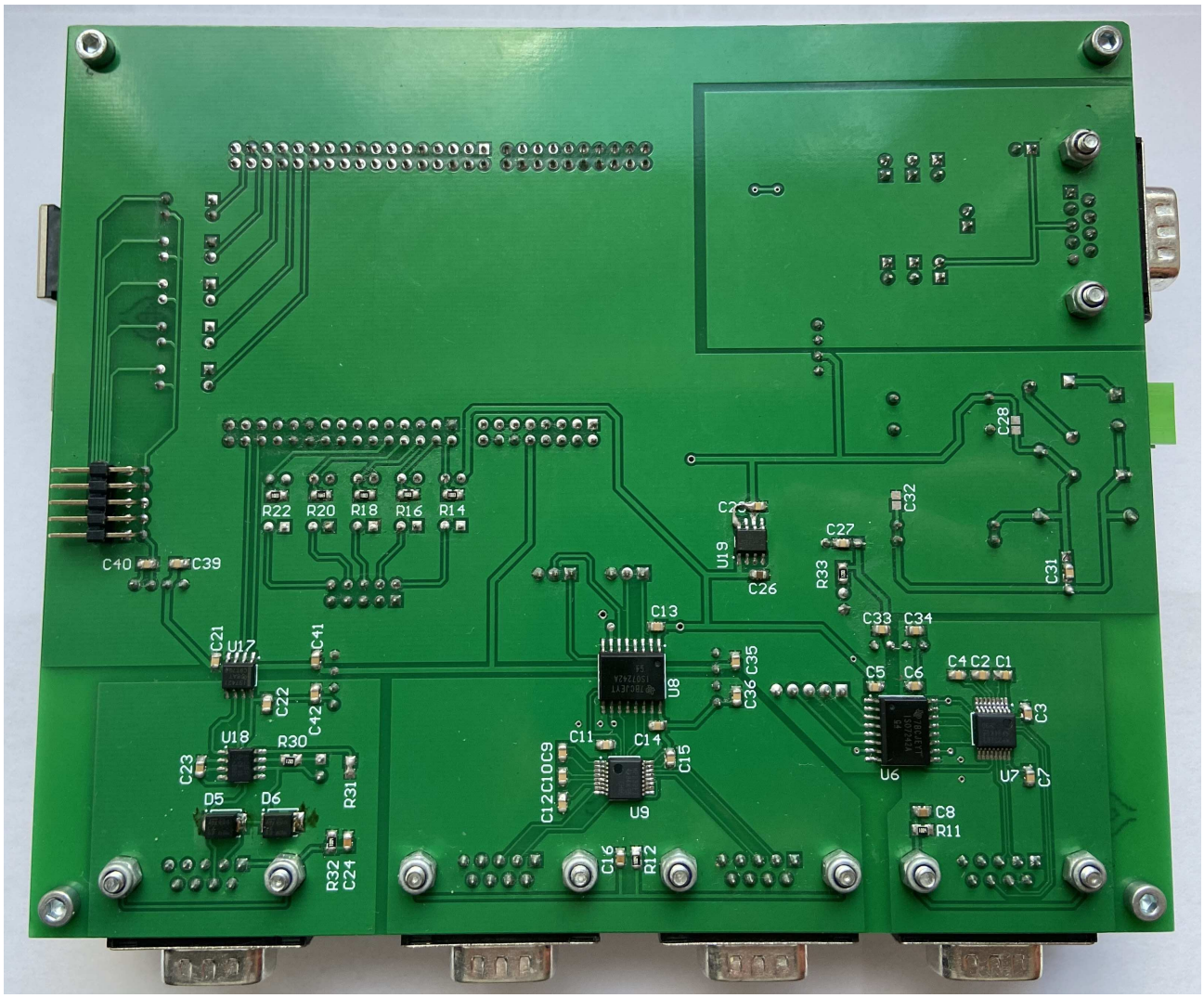
Obr. A.7: Zapojení napájecích obvodů



Obr. A.8: Navržená komunikační jednotka



Obr. A.9: Fotografie osazeného plošného spoje 1



Obr. A.10: Fotografie osazeného plošného spoje 2