FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Přístupový systém s využitím RFID karet
Student:	Bc. Petr Elexa
Vedoucí:	Ing. Matěj Bartík
Studijní program:	Informatika
Studijní obor:	Návrh a programování vestavných systémů
Katedra:	Katedra číslicového návrhu
Platnost zadání:	Do konce zimního semestru 2020/21

Pokyny pro vypracování

Analyzujte současné přístupové systémy využívající RFID a navrhněte, realizujte a otestujte prototyp zařízení, který:
- bude s uživatelem komunikovat prostřednictvím RFID čtečky s rozhraním Wiegand26, volitelně stavovým světlem nebo pípáním,
- bude komunikovat s nadřízenou jednotkou.

Návrh musí podporovat požadavek na real-time zpracování, zohledňovat nemožnost fyzického přístupu k finální verzi zařízení a jednoduchou fyzickou instalaci výsledného systému.
Součástí práce je vytvoření komunikačního protokolu zařízení s nadřízenou jednotkou, která bude poskytovat/odepírat přístup a bude využívat sběrnici CAN. Navržený protokol musí zohledňovat velký počet zařízení připojených do systému.
Na základě prototypu připravte schéma pro další verzi zařízení.
Nadřízenou jednotku, která bude používat vámi vytvořený protokol, navrhněte a případně vytvořte.
Stanovte doplňující požadavky na systém.

Seznam odborné literatury

Dodá vedoucí práce.

doc. Ing. Hana Kubátová, CSc.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 3. září 2019

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Přístupový systém s využitím RFID karet

Bc. Petr Elexa

Katedra číslicového návrhu
Vedoucí práce: Ing. Matěj Bartík

13. ledna 2020

Poděkování

Děkuji vedoucímu práce za vstřícný přístup a rady při tvorbě této práce. Dále svým blízkým za jejich podporu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principu při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisu. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisu, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programu, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli zpusobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelum). Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 13. ledna 2020 . . . . . .

Odkaz na tuto práci

Elexa, Petr. *Přístupový systém s využitím RFID karet*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

Abstrakt

V této práci je navržen a implementován základní systém pro řízení fyzického přístupu osob, založený na použití RFID karet. Tento systém podporuje velmi rozšířené RFID karty komunikující na frekvence 125 kHz.

Součástí jsou přístupová zařízení sloužících k ovládání elektrických zámků dveří a připojení čteček karet s rozhraním Wiegand 26. Dále je součástí systému nadřízená jednotka s databází, která rozhoduje o udělení či neudělení přístupu. Ke komunikace mezi zařízeními a nadřízenou jednotkou byl navržen a použit vlastní protokol založený na CAN sběrnici.

Pro propojení a napájení přístupových zařízení je použit jeden rozvod síťového kabelu CAT-5e k omezením počtu potřebné kabeláže a napájecích zdrojů a tím usnadnění instalace systému. Elektronické zámky je možné napájet z těchto přístupových zařízení.

Na základně prototypu byl vytvořen firmware a schéma hardwaru pro přístupová zařízení s rozhraním pro řízení dvou dveří na platformě NXP LPC11C24. Dále byl vytvořen software pro nadřízenou jednotku fungující na operační systém GNU/Linux.

Klíčová slova RFID, Wiegand, řízení fyzického přístupu, elektrický zámek

Abstract

This work contains a design and implementation of a basic physical access control system for persons, based on the use of RFID cards. This system supports very widespread RFID cards that communicate at 125 kHz.

The system contains access devices used to control electric door locks and to connect card readers with Wiegand 26 interface. The system includes a master unit with a database which grants access. An application protocol for CAN bus has been designed for communication between the devices and the master unit.

A single CAT-5e network cable is used to interconnect and power access devices to reduce the number of cabling and power supplies needed to make system installation easier. Electronic locks can be powered from these access devices.

A firmware and hardware schematics for two-door access control devices were created based on the access control device prototype on the NXP LPC11C24 platform. A master unit software was created for GNU/Linux operating system.

Keywords RFID, Wiegand, physical access control, electric door lock

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Systémy řízení přístupu jsou široce používané v objektech, kde je velký pohyb osob. Jejich nevýhodou je, že nabízená řešení jsou drahá a nevyplatí se je použít v nekomerčních prostředí.

Téma jsem si zvolil, protože každý den používám přístupový systémy v práci jako uživatel, když otevírám jakékoliv dveře a chci získat přehled o tom, jak to funguje uvnitř.

V této práci se zabývám návrhem a realizací systému řízení přístupu s využitím RFID karet za použití běžně dostupných čteček karet s Wiegand 26 rozhraním, který bude určen na základní řízení přístupu osob v objektu loděnice.

V první kapitole je rešerše existujících přístupových systémů a používaných technologií. V poslední části jsou sepsané požadavky na systém a stanoveny cíle s ohledem na zadání.

Druhá kapitola obsahuje na úvod seznámení s CAN sběrnicí, návrh struktury systému, výběr platformy a volba softwarové architektury pro řídící jednotku a přístupová zařízení. Jako poslední je část s volbou zapojení a napájení přístupových jednotek.

V třetí kapitole je popsáno vytvoření prototypu, firmwaru a elektrického schématu řídícího panelu s použitím mikrokontroleru NXP LPC11C24. Potom implementace softwaru řídícího serveru pro OS/Linux.

Poslední kapitola popisuje jak a co bylo otestováno.

Kapitola 1

Přístupové systémy

Přístupové systémy, konkrétně systémy pro *fyzické* řízení přístupu osob, určují kdo, kdy a kam může vstoupit. V dnešní době jsou velmi rozšířené. Používají se především tam, kde je potřeba řízení přístupu většího počtu osob do objektům nebo prostor přes dveře, výtahy, brány, závory a jiné bariéry.

Systémy se podle použitých kontrolních prostředků rozdělují do třech skupin: Jsou to lidské, mechanické a elektronické. Nadále se budou uvažovat pouze elektronické, které jsou ovládané počítačem a práva jsou nastavována přes software.

1.1 Elektronické přístupové systémy

V přístupovém systému [6][7] má každá osoba přiřazena jedinečný údaj (identifikátor), který jí umožní přístup ke všem objektům, prostorám a zařízením, na které má přidělená patřičná oprávnění.

Tyto systémy fungují typicky následovně, pokud je čtecímu zařízení předložen identifikátor, jsou jeho informace poslány řídícímu panelu. Tento panel komunikuje se serverem s databází, kde je uložen seznam oprávnění a další pravidla systému. Informace z identifikátoru jsou porovnány se seznamem a přístup je povolen nebo zamítnut. V případě povolení je umožněn vstup odemčením zámku. V opačném případě zámek zůstane zamknutý. Další používané funkce a schopnosti systémů jsou:

- Zaznamenávat dobu a místo přístupů;
- Určovat dobu, kdy je možné vstoupit;
- Vydávat dočasná oprávnění;
- Detekovat neoprávněné přístupy;
- Integrovat s dalšími systémy (např. bezpečnostní a požární).

Výhodou elektronických systémů oproti mechanickým je efektivní správa přístupových práv udělovaných jednotlivým osobám, větší komfort uživatelů a v případě ztráty nebo odebrání přístupu není nutno měnit vložku zámku. Je také možné zamezit předávání a kopírování klíčů mezi uživateli použitím vhodného druhu identifikátoru.

Nevýhodou jsou vyšší pořizovací náklady, potřeba systém spravovat vyškolenou osobou a nutnost zabudování komponent systému zejména při instalaci do již existujících objektů.

1.2 Identifikace osob

Osoby užívající příst. systém mají přidělenou vlastní identitu. Tu prokazují jedním i více identifikátory. Každý identifikátor musí být v rámci jednoho systému unikátní, aby šlo osoby a jejich identity odlišit. Identifikátory jsou biometrické údaje (např. otisk prstu, obličej nebo duhovka oka), znalost hesla a elektronicky čitelné identifikátory.

Použití pouze biometrický údajů nebo hesel není běžné, většinou se používají ve spojení s jiným identifikátorem [6, s. 58–59].

Biometrické údaje jsou osobní údaje a jejich sdílení není ze strany uživatele žádoucí. Hesla lze lehce sdělit někomu jinému, čímž se vlastně jako identifikátor duplikují. [7]

1.3 Komponenty přístupových systémů

Základní komponenty přístupových systémů jsou [8]:

- elektronický identifikátor,
- čtecí zařízení,
- řídící panel ovládající zámky,
- řídící server s databází oprávnění.

1.3.1 Elektronický identifikátor

Elektronický identifikátor může mít mnoho forem. Pro přístupové systémy jsou typické fyzické předměty jako karty, přívěšky a mobilní zařízení. Příklady jsou na obrázku 1.1.

Identifikátor je pasivní nebo aktivní [9]. Pasivní jsou napájeny z elektromagnetického pole čtecího zařízení a nepotřebují tedy vlastní zdroj. To znamená, že jsou malé a mají nízké výrobní náklady. Aktivní a polo-aktivní mají vlastní zdroj napájení – baterii a mohou přenášet data na větší vzdálenost (v řádu stovek metrů). Ale za cenu větší velikosti a výrobních nákladů.

Tabulka 1.1: RFID frekvence v přístupových systémech

název	frekvence	rychlost přenosu	velikost paměti
LF	120–135 kHz	$\mathrm{bit\,s^{-1}}$	stovky bitů
HF	13,56 MHz	$\mathrm{kbit\,s^{-1}}$	desítky až stovky kb

Moderní přístupové systémy pro osoby používají bezkontaktní identifikátory založené RFID a HID technologiích. Obě technologie mají standardizované frekvence na kterých se používají.

1.3.1.1 RFID – Radio Frequency IDentification

Technologie RFID [10][8] je způsob identifikace založený na přenosu dat elektromagnetickým polem. Data jsou uložena na čipu, který je společně s anténou součástí tzv. tagu. Tag (podle typu použití) může mít mnoho forem, proto dále bude pro zjednodušení označován pouze jako karta.

K obousměrné komunikaci s kartou se používá čtecí zařízení. Tím je kartu možné číst nebo do něj zapisovat. Dostupné jsou opakovaně a jednou zapisovatelné verze (OTP). Případně pouze čitelné, pokud byly zapsány při výrobě.

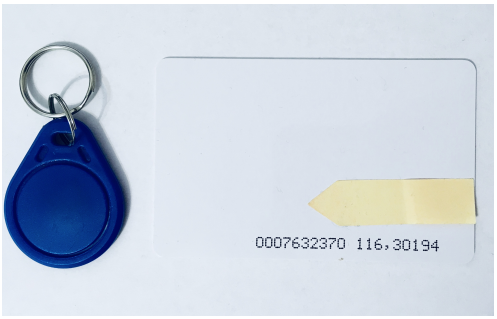
Základní schopnosti RFID karet se odvíjí od přenosové frekvence. Jsou jimi nízkofrekvenční LF a vysokofrekvenční HF. V tabulce 1.1 je jejich porovnání. Jiné frekvence se pro řízení přístupu osob nepoužívají. Výhodou HF karet oproti LF je rychlejší přenosová rychlost. Proto má větší smysl do čipu karty integrovat mikroprocesor a další obvody. Taktéž disponuje větší pamětí, je tedy možné HF karty použít pro více aplikací najednou.

Mikroprocesorové RFID karty jsou známé pod označeními smart card (v překladu chytrá karta) a čipová karta. Ty umí data zpracovávat a vykonávat více funkcí. Zejména používat šifrování a autentizaci. HF karty ale nutně nemusí obsahovat žádný procesor a potom se jedná pouze o paměťové karty. Stejně tak některé LF karty mohou obsahovat procesor.

Příklady rozšířených komerčních LF a HF pasivních karet:

- HID Proximity,
- EM Microelectronic EM4200 (nástupce EM4100),
- Atmel ATA5577 (nástupce Atmel T5557),
- Mifare Classic,
- Mifare DESfire,
- HID iCLASS.

Uvedené příklady karet jsou na trhu už delší dobu a bylo k nim vytvořeno mnoho kompatibilních klonů, které jsou široce dostupné.



Obrázek 1.1: Příklad generické RFID karty a přívěšku

Pro žádné LF karty používané pro přístup osob nebyl adoptován standardizovaný protokol[1] pro komunikaci se čtecím zařízením. To znamená, že takové karty a čtecí zařízení od různých výrobců jsou často vzájemně nekompatibilní. To se netýká HF karet, protože mnoho výrobců dodržuje alespoň jeden ze standardů pro typy karet [11]: ISO/IEC 14443 typ A[2], typ B[3] a případně ISO/IEC 15693.

1.3.1.2 Datové formáty a struktura RFID karet

Pro identifikaci v přístupovém systému, je na RFID kartě naprogramováno vždy alespoň číslo karty. Někdy se používají i další čísla, jako například kód zařízení (tzv. facility code). Tato čísla jsou uložena ve specifickém binárním formátu a na konkrétním místě.

Existuje více typů formátování čísel karet. Mezi nejpoužívanější patří otevřený 26-bitový Wiegand formát a jeho variace [12]. Tyto variace jsou ale většinou proprietární formáty, např. HID Corporate 1000, kde je přidělování čísel karet řízeno tvůrcem formátu. Formát není závislý na typu a frekvenci RFID karty. 26-bitový a 34-bitový Wiegand formát je popsán v části 1.3.2.3.

Každý typ karty má specifickou strukturu uložení dat. HF RFID karty většinou mají strukturu podle normy ISO/IEC 7816-4. Dostupné LF RFID karty, stejně jako komunikační protokol, mají strukturu danou každým výrobcem. Kromě dat pro přístupový systém mohou na kartě být uloženy další informace pro jiné aplikace.

1.3.1.3 NFC – Near Field Communication

NFC je sada komunikačních protokolů umožnující komunikaci dvěma zařízením na krátkou vzdálenost. Je spravováno NFC fórem, které harmonizuje a rozšiřuje existující bezkontaktní komunikační standardy.

[1] Standardizaci k použití ke správě věcí došlo v roce 2009 ISO/IEC 18000-2.
[2] Vyvinuto z technologie Mifare od firmy Philips (nově NXP).
[3] Vyvinuto z technologie Calypso od firmy Innovatron.

Význam pro RFID přístupové systémy je v tom, že NFC sada implementuje evoluci standardů vytvořených pro bezkontaktní HF RFID karty pomocí emulace karet na zařízení s NFC [13]. To umožňuje použít jakékoliv mobilní zařízení, které je vybavené podporou emulace karty, jako identifikátor v přístupovém systému.

1.3.2 Čtecí zařízení

Čtecí zařízení (čtečka karet[4]) je jediná část systému, se kterou interaguje uživatel. Jeho úkolem je vyčíst přístupové identifikační číslo z RFID karty a předat ho řídícímu panelu.

Podporované frekvence a typy karet jsou různé. Většinou je podporována pouze jedna frekvence. V případě LF je zpravidla podporován jeden typ karet, pokročilejší jich podporují více a jsou nastavitelné. Standardní HF RFID a NFC karty jednoho typu je možné číst jakoukoliv standardní HF čtečkou karet.

Nejpoužívanější druhy rozhraní pro propojení s řídícím panelem jsou:

- Wiegand rozhraní a Wiegand protokol;
- RS485 s různými protokoly nebo nově standardizovaný OSDP (Open Supervised Device Protocol);
- Ethernet s TCP/IP.

Zabezpečené RFID karty mohou vyžadovat pro vyčtení autentizaci (např. heslem) nebo šifrování přenosu. V tom případě je pro vyčtení čísla z karty nutná čtečka, která umí obousměrně komunikovat s kartou, případně i s řídícím panelem. Čtečky také bývají běžně vybaveny bzučákem a signalizačními LED.

Nominální napájecí napětí čteček karet bývá 12 V nebo 24 V průměrným proudovým odběrem kolem 100 mA. Popis elektrického zapojení přítomného na většině dostupných čtečkách LF karet je v tabulce 1.2.

1.3.2.1 Wiegand rozhraní

Wiegand rozhraní [12] je jednosměrné rozhraní od čtečky karet k řídícímu panelu. Skládá se ze tří vodičů označovaných `Data 0`, `Data 1` a `Data Return`. První dva přenášejí data, třetí je datová zem.

Datové signály

Signály používané pro Wiegand rozhraní [1] jsou `Data 0` a `Data 1`. Jak může být zřejmé z názvu, po signálu `Data 0` se přenáší binárních nuly a po `Data 1`

[4] Čtečkou karet se běžně označuje čtecí zařízení, které umí číst elektronický identifikátor.

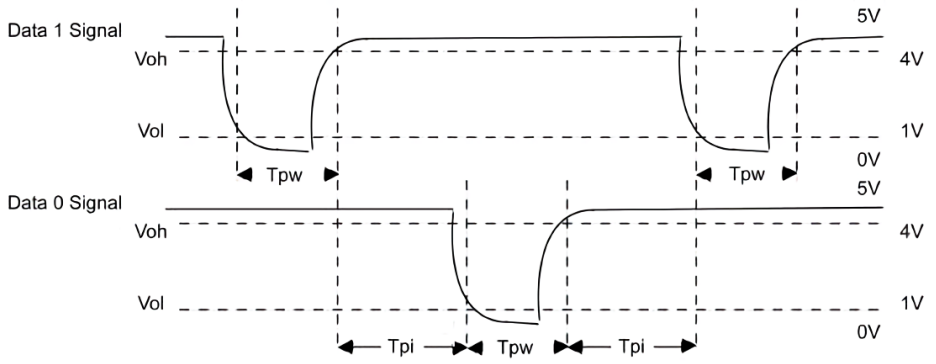
Tabulka 1.2: Běžné elektrické zapojení čtečky LF karet

označení vodiče	popis
napájení	vstup napájení pro čtečku (+)
zem	vstup napájení pro čtečku (−)
data 0	Wiegand datový signál
data 1	Wiegand datový signál
data return	zem pro Wiegand datové signály
první LED	ovládání světelného signálu
druhá LED	ovládání světelného signálu
bzučák	ovládání zvukového signálu

binární jedničky. V klidovém stavu, když se data nepřenášejí, mají oba signály napěťovou úroveň nad rozhodnou úrovní V_{oh}. Při přenosu binární hodnoty se vytvoří puls, tím že je korespondující signál stažen pod rozhodnou úroveň V_{ol} na dobu T_{pw}, přičemž druhý zůstává nad rozhodnou úrovní.

Jednotlivé transakce se rozlišují pouze pauzou v přenosu o délce větší než T_{pi}. Časová signálů a napětové úrovně jsou znázorněny na obrázku 1.2.

Podle doporučení v [14] má být perioda T_{pi} mezi 200 µs a 20 ms, a T_{pw} mezi 20 µs a 100 µs.



Obrázek 1.2: Časový průběh Wiegand signálů (převzato z [1])

1.3.2.3 Wiegand formát

Slovem Wiegand formát se většinou označuje původní 26-bitový formát odvozený od kontaktních Wiegand karet s Wiegand drátky. Ty měly do sebe zabudováno právě 26 drátků, které reprezentovaly jednotlivé bity.

26-bitový formát je otevřený standard, protože je veřejný a prodej karet v tomto formátu není nijak regulován. Díky tomu je s ním kompatibilní naprostá většina přístupových systémů. To ale způsobuje i to, že existují dupli-

citní karty [12]. Z důvodu omezeného počtu unikátních karet v tomto formátů, vzniklo mnoho dalších variant z rozšířenou bitovou šířkou a jinou strukturou. Populárním z nich je 34-bitový formát. Níže jsou uvedeny dvě nejpoužívanější varianty formátů.

26-bitový Wiegand formát

Tento binární formát [12] kóduje 16-bitové číslo karty a 8-bitový kód zařízení a je zabezpečen sudou a lichou paritou. Popsán je tabulkou 1.3. Sudou paritou se zabezpečuje prvních 13 bitů a lichou posledních 13 bitů. Počet možný unikátních karet v tomto formátu je $24^2 - 1$ (nulové číslo a kód se nepoužívá).

Tabulka 1.3: 26-bitový Wiegand formát

bit 0	bit 1–8 (8 je LSB)	bit 9–24 (24 je LSB)	bit 25
sudá parita	kód zařízení	číslo karty	lichá parita

34-bitový Wiegand formát

Verze 34-bitového formátu [12] se, na rozdíl od 24-bitového, může od každého výrobce lišit. Zde je popsána nejpoužívanější verze 34-bitového formátu.

Tento formát kóduje 16-bitové číslo karty a 16-bitový kód zařízení a je zabezpečen sudou nebo lichou paritou. Popsán je tabulkou 1.4. Paritou se zabezpečuje prvních 17 bitů a posledních 17 bitů.

Tabulka 1.4: 34-bitový Wiegand formát

bit 0	bit 1–16 (16 je LSB)	bit 17–32 (32 je LSB)	bit 33
parita	kód zařízení	číslo karty	parita

1.3.2.6 Wiegand protokol

Značí přenos binárních dat v jednom z Wiegand formátů po Wiegand rozhraní. Wiegand 26 je jednoznačné označení pro přenos po Wiegand rozhraní s 26-bitovým formátováním, díky tomu, že 26-bitový formát je jen jeden. Přenos binárních dat v tomto protokolu probíhá jednosměrně od čtečky karet k řídícímu panelu. Data se přenášejí o posledního (LSB) bitu k prvnímu (MSB).

1.3.3 Řídící panel

Obsluhuje jeden nebo více zámků pomocí relé a je přímo připojen ke čtečce karet. Většinou je pro čtečku ale i zámek zdrojem napětí. Volitelně má vstupy na monitorování stavu dveří, zámků, kliky apod.

Vyčtená čísla karet panel doplní informací o umístění čtečky (případně čas) a odešle je na server, který rozhodne o odemčení zámku. Některé panely

v případě výpadku spojení nebo pořád mohou rozhodovat samostatně a jen se synchronizují s řídícím serverem. S ním je panel propojen většinou přes RS-485, Ethernet nebo Wi-Fi. Protokol na těchto rozhraních bývá proprietární.

Jedná většinou se o dedikované zařízení, které ale může být integrován se čtečkou. Nevýhodou v tomto případě je, že k takové čtečce, musí být přivedeny všechny vodiče a celé zařízení je přístupná komukoliv, pokud je instalována u nezabezpečené strany dveří.

Panel nemusí obsluhovat jen zámky dveří, na kontakty relé může být připojeno jakékoliv elektricky ovladatelné zařízení např. závora, výtah, apod.

1.3.4 Řídící server

Server s databází obsahuje seznam oprávnění. Je jádrem systému a rozhoduje o (ne)udělení přístupu všech požadavků přicházejících z řídících panelů. Také zaznamenává veškeré události ohledně přístupu.

Řídící software většinou běží na serveru s OS Linux nebo Windows. Může být umístěn v lokální síti anebo internetu. Slouží k řízení celého přístupového systému. Většina umožňuje upravovat seznam oprávnění uživatelů, přístupu a zpracovávat záznamy událostí; Uživatele umí třídit do skupin a nastavovat jim přístup podle umístění a času.

Z důvodů decentralizace, redundance nebo většího počtu zařízení může v systému být více spolupracujících serverů a databází.

Funkci serveru může nahrazovat hlavní řídící panel, pokud je takový v systému.

1.3.5 Software pro správu

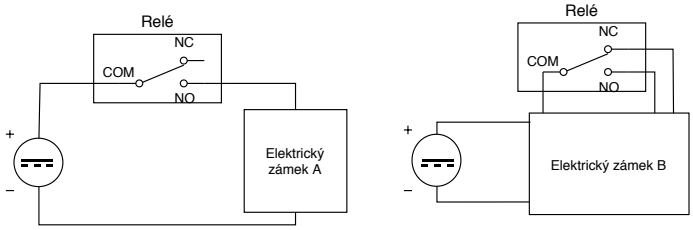
Umožňuje spravovat celý přístupový systém. Může běžet jako samostatná aplikace na dedikovaném zařízení, které se připojuje na řídící server s databází. Nebo může běžet přímo na něm, kdy je např. aplikační server integrován s řídícím serverem.

1.3.6 Elektrické zámky

Dveřní elektrické zámky používají na magnety, motory a solenoidy k elektrickému otevírání, zamykání a odemykání dveří. Ovládají se buď pouhým přivedením napájení (obr. 1.5 vlevo). Nebo jsou trvale napájené a mají další kontakty pro to vyhrazené (obr. 1.5 vpravo). Zámky se také odlišují tím jestli k odemčení/zamčení vyžadují přivedení napájení/signálu po celou dobu nebo stačí krátký impuls. Lepší zámky mají zabudované vyvedené monitorovací kontakty pro signalizování stavu zámku a dveří. Důležitým nastavením zámků je, jestli mají přivedením napájení se zamknou nebo odemknout. Kvůli bezpečnostním důvodům v případě výpadku proudu v systému.

Obrázek 1.3: Elektromechanický zámek Assa Abloy EL560 (převzato z [2])

Obrázek 1.4: Elektrický otvírač FAB (převzato z [3])

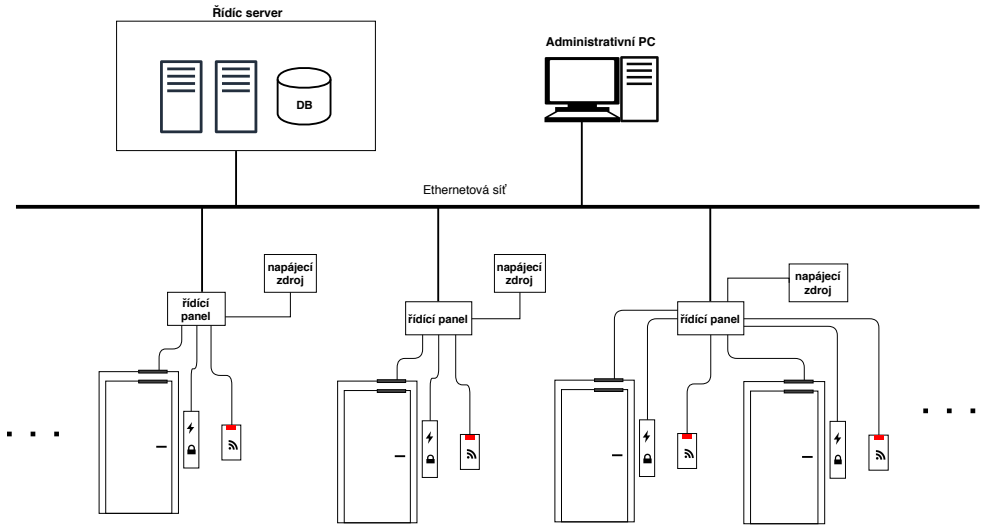


Obrázek 1.5: Zapojení zámků

Základní rozdělení elektrických dveřních zámků je na otvírače (obr. 1.4) a zámky (obr. 1.3). Otvírače na rozdíl od zámků jsou umístěny v rámu a odblokovávají pouze střelku dveří.

Napájecí napětí všech typů zámků bývá 12 V–24 V s proudovým odběrem do 1 A. Jsou napájeny střídavým nebo stejnosměrným proudem.

Přestože jsou elektronické zámky plně ovladatelné elektronicky, tak je lze většinou odemykat mechanickým klíčem, případně klikou z vnitřní strany.



Obrázek 1.6: Typická topologie přístupového systému

1.4 Autonomní přístupový systém

Je samostatné řešení pro jeden zámek (dveře). Řídící prvkem je čtečka karet, která zároveň supluje řídící panel. Někdy jsou dokonce kompletně integrovány včetně zámku do jednotky umístitelné na dveřích.

Fungují bez řídicího serveru, kde databáze je pouze lokální. Nelze propojit více zařízení dohromady. Jejich správa a konfigurace přístupu se provádí lokálně kartou nebo počítačem. Jsou určené pro velmi malý počet uživatelů.

1.5 Topologie přístupových systémů

Typická topologie systému je znázorněna na obrázku 1.6 a lze je rozdělit na více kategorií:

1.5.1 Centralizovaná

Rozhodujícím zařízením je server s databází. Odezva celého systému je na něm přímo závislá. V případě výpadku serveru systém přestane hned nebo po krátké době fungovat. Čtečky a panely totiž mohou po nějakou dobu pracovat samostatně.

1.5.2 Decentralizovaná

Čtečky anebo panely mají lokální databázi a rozhodují samostatně. Zároveň dochází k synchronizaci se serverem a hlavní databází. V případě výpadku serveru systém funguje dále ale s omezenou funkcí.

1.5.3 Distribuovaná

Čtečky anebo panely mají vlastní databázi a rozhodují samostatně. Mezi sebou komunikují stylem P2P (Peer to Peer). Zařízení musí být pokročilá, protože každé zastává celou funkci systému.

1.6 Požadavky na systém

Přístupový systém má podle zadavatele (vedoucí práce) být použit na základní řízení přístupu osob v objektu loděnice. Systém bude používán uživateli spíše nárazově, kdy do loděnice přijde skupina lidí. Přístup přes kartu má být nasazen na dvacet dveří, které jsou hustě rozmístěny (přibližně 3 m rozestup).

Rozšíří se s ním stávající možnost odemykání pomocí fyzického klíče a ulehčí se správa přístupu k jednotlivým místnostem. S tím, že systém bude ekonomičtější alternativou ke komerčním řešením, díky vlastnímu řídícímu panelu s malými výrobními náklady a použitím dostupných LF RFID čteček třetích stran.

Použitím karet místo klíčů se odstraní zejména nutnost replikace fyzických klíčů pro umožnění vstupu více osob. Fyzické klíče bude mít pouze omezené množství osob.

Níže jsem rozvedl typické použití systému a požadavky na řídící panel. Tyto věci vyplývají ze zadání nebo byli upřesněny zadavatelem, a poznatků z první kapitoly. Požadavky na řídící server nebyli nijak upřesněny, kromě toho, že má rozhodovat o přístupech.

1.6.1 Scénář použití z pohledu uživatele

Uživatel přiloží kartu k RFID čtečce karet, která je přes rozhraní Wiegand napojena na řídící panel (přístupové zařízení). To komunikuje protokolem postaveným nad CAN sběrnicí s řídícím serverem (nadřízenou jednotkou). Ten podle umístnění panelu a čísla karty udělí nebo odepře přístup a událost zaznamená. Čtečka karet informuje uživatele zvukem anebo stavovým světlem. Řídící panel připojený na dveřní zámek v případně udělení přístupu od nadřízené jednotky odemkne dveře na dostatečnou dobu.

1.6.2 Požadavky na řídící panel

Jeden panel má mít rozhraní pro až dvě čtečky a zámky, kvůli hustěji rozmístěným dveřím. Má také vhodně komunikovat s uživatelem přes signalizační LED a bzučák na čtečce karet. Navíc může být vhodná podpora monitorování stavu dveří přes spínač na rámu dveří.

Řídící panel bude zabudovaný. Musí pracovat spolehlivě v nepřetržitém provozu a být schopny se zotavit z poruch softwaru bez fyzického zásahu. Selhání jednoho z nich nesmí ovlivnit ostatní panely.

Jelikož panelů bude v systému nejvíce (kromě zámků a čteček), měly by jít lehce zabudovat, nakonfigurovat a propojit. Při zavádění panelu by mělo jít jednoduše nastavit jeho umístění, které bude rozhodující pro udělovaní přístupu.

1.7 Cíle práce

V prvním kroku navrhnu architekturu a propojení celého přístupového systému s ohledem na požadavky a existující řešení. Ten bude zaměřený na využití s větším počtem dveří, elektronicky odemykaných z jedné strany. V systému se zaměřím zejména na řídící panel, jakožto hlavní bod zadání.

Za druhé vytvořím hardwarový prototyp řídícího panelu pro dvoje dveře. K němu půjde připojit běžně dostupnou RFID čtečku přes rozhraní Wiegand 26 včetně světelné a zvukové signalizace. Bude vybavený CAN rozhraním pro komunikaci s řídícím serverem. Také mít standardně používané rozhraní pro ovládání elektronického zámku. Na tomto prototypu poté vyvinu firmware.

Dále vytvořím komunikační protokol nad CAN sběrnicí mezi řídícím serverem vhodný pro až řádově stovky podřízených panelů. Jelikož jeho definice je předpokladem pro implementaci komunikace mezi panelem a severem.

Pro prototyp vytvořím firmware s real-time zpracováním a zotavením z poruch, který rozumí 26-bitovému Wiegand formátu dat ze čtečky a implementuje vytvořený CAN protokol. Jeho hlavním úkolem bude přeposílání vyčtených čísel karet na server a podle jeho odpovědi odemknout nebo neodemknout dveře.

V této fázi by měl prototyp být finální a na jeho základě, vytvořím schéma zapojení hardwaru vhodné pro další verzi.

Na závěr vytvořím řídící server s přístupovou databází, který podle umístění dveří a čísla karty rozhodne o udělení přístupu. Bez něj by se nedalo moc dobře testovat systém jako celek, hlavně tedy komunikace.

Celý systém s ohledem požadavek real-time zpracování a nárazového používání, by měl umět reagovat na požadavky k odemčení dveří, a to průměrně do desítek milisekund. Občasné nedodržení reakčního času se nepovažuje za problém.

Kapitola 2

Analýza a návrh systému

V předchozí kapitole se probraly existující řešení a nepoužívanější technologie pro přístupové systémy. Z těchto informací vzešlo, že použití CAN sběrnice pro propojení řídících panelů se serverem, je celkem ojedinělé. Hardware řídícího panelu, bude tedy specifický. Komerční software pro server nebo panel bývá uzavřený. Našel jsem jediný otevřený systém [15], ten ale nepoužívá řídící server, a je založený na starším 8-bitovém MCU a přidání CAN komunikace by mohl být problém. Proto budu realizovat hardware i software řídícího panelu od začátku.

Po analýze požadované funkčnosti jsem došel k závěru, že jejich implementování v softwaru by nemělo být nikterak komplexní, pouze pracné. Zejména protože se jedná vlastně o centralizovanou topologii a komunikace s Wiegand čtečkami je pouze jednosměrná a zároveň jednoduchá. Problémem se může ukázat realizace hardwaru řídícího panelu, protože v této oblasti mám omezené zkušenosti. Zároveň se musím více seznámit s fungováním sběrnice CAN.

2.1 CAN – Controller Area Network

CAN je v průmyslu rozšířený asynchronní sériový protokol pro vysílání zpráv mezi všemi uzly na sběrnici. Původně byl vyvinut firmou Bosch[5], která uvolnila několik verzí specifikaci spojového protokolu [16][17]. Poté došlo ke standardizaci v sadě norem ISO 11898. Protokol (spojová vrstva) a část fyzické vrstvy je specifikován v normě [18].

2.1.1 CAN protokol

CAN sběrnice obecně nabývá jedné ze dvou logických hodnot: dominantní a recesivní. Pokud se zároveň vysílají obě hodnoty dominantní převládne. V době neaktivity má sběrnice recesivní hodnotu. Dominantní hodnota odpovídá logické hodnotě 0 a recesivní logické hodnotě 1.

[5]Rober Bosch GmbH

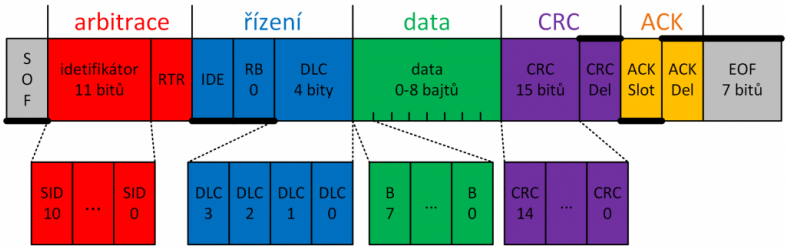
Princip CAN komunikace: Každý uzel může začít vysílat rámec[6], pokud je sběrnice volná. V případě, že dojde současnému pokusu o získání sběrnice dvěma a více uzly, uplatní se bitová arbitrace. Každý uzel při vysílání zároveň monitoruje zda vysílaná hodnota koresponduje s tou na sběrnici. Využívá se toho, že dominantní přepisuje recesivní hodnotu. Takže pokud uzel detekuje při posílání arbitračního pole nesoulad, okamžitě uvolní sběrnici, aby se nepoškodil přenášený rámec. Po vyhrání arbitrace jedním uzlem, kdy ostatní už uvolnily sběrnici, dojde k dokončení přenosu zbylých polí rámce. Následuje potvrzení korektního přijetí od všech uzlů. Při úspěšném vyslání je rámec přijat všemi uzly. Ty podle nastaveného filtru rámec zahodí nebo ho předají aplikaci.

V současné době existují dvě hlavní verze CAN protokolu označované: CAN 2.0 (Classical CAN) a CAN FD (CAN with Flexible Data-Rate). Oba protokoly implementují samostatné nebo vestavěné CAN řadiče a to převážně v hardwaru. Rozhraní a nástroje, které podporují CAN jsou široce dostupné.

2.1.1.1 CAN 2.0

CAN 2.0 [16][18] je plně kompatibilní rozšíření první verze protokolu. Ke standardnímu formátu rámce s 11-bitovým identifikátorem se přidal rozšířeného formátu rámce, který má navýšenou délky identifikátoru na 29 bitů. Uzly na jedné sběrnici mohou vysílat standardní i rozšířený formát rámce.

Standardní formát rámce (Standard frame format – SFF) je popsán na obrázku 2.1 a rozšířený (Extended frame format – EFF) na obrázku 2.2. Svislými čarami jsou odděleny jednotlivá pole rámce. Bit stuffing (vkládání bitů) k udržení synchronizace uzlů, kdy po pěti stejných bitech se vloží doplňkový bit, se v obrázcích neuvažuje. Po rámci v jednom z uvedených formátů následuje mezi-rámcový prostor, který dává čas na zpracování.



Obrázek 2.1: Standardní formát CAN rámce (převzato z [4])

Význam bitů v standardním formátu na obrázku 2.1 je:

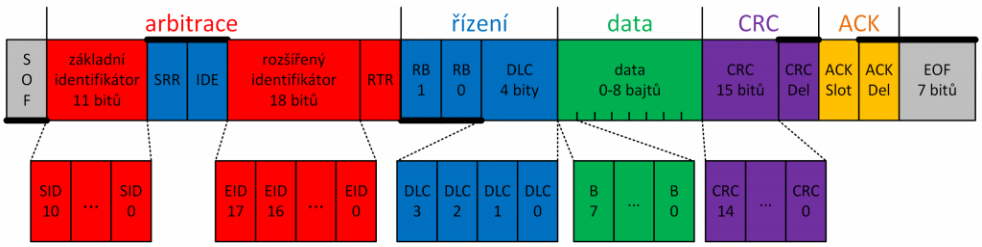
- SOF (start of frame – začátek rámce) je dominantní bit, který slouží k synchronizaci všech uzlů na začátku datového a vzdáleného rámce.

[6]Zpráva v CAN protokolu se nazývá rámec.

- Identifikátor (ID) je 11-bitová hodnota. Určuje prioritu rámce při arbitraci. Rámec s ID obsahujícím jen dominantní bity má nejvyšší prioritu. MSB je SID 10 a LSB je SID 0.
- RTR (remote transmission request – požadavek na vzdálený přenos) bit slouží k rozlišení dvou druhů rámců. Pro datové rámce má RTR bit dominantní a vzdálené rámce recesivní hodnotu.
- IDE (identifier extension – rozšíření identifikátoru) bit má dominantní hodnotu a značí, že se jedná o standardní formát bez rozšíření.
- RB0 (reserved bit 0) bit je vyhrazený pro budoucí rozšíření.
- DLC (data length code – kód datové délky) je 4-bitový kód, který udává délku v bytech následujícího pole dat. Délka 0–7 se kóduje jako dvojkové číslo (DLC0 je LSB). Délka 8 se kóduje nastavením DLC3 bitu na dominantní hodnotu a na ostatních bitech nezáleží.
- Data je bitové pole obsahující aplikační data. V každém bytu se přenáší první MSB. Délka je dána DLC bity, dovoleno je 0–8 bytů. Pokud je délka nula, datové pole není přenášeno.
- CRC (cyclic redundancy check – cyklický redundantní součet) je kontrolní součet, který zabezpečuje rámec od začátku až do konce datového pole (včetně). Slouží k detekci chyb vzniklých při přenosu.
- CRC Del (CRC delimiter – CRC oddělovač) je bit s recesivní hodnotou.
- ACK Slot (acknowledge slot – slot pro potvrzení) je místo pro potvrzení, že došlo k úspěšnému přijetí validního rámce. Vysílací uzel posílá recesivní hodnotu a čeká na příjemce, aby potvrdili příjem přepsáním na dominantní hodnotu. Pokud nedojde k potvrzení, vysílací uzel zopakuje rámec, po znovuzískání sběrnice.
- ACK Del (ACK delimiter - ACK oddělovač) je bit s recesivní hodnotou.
- EOF (end of frame – konec rámce) je 7 recesivních bitů. Slouží k detekci chyby při „bit stuffing“.

Význam bitů v rozšířeném formátu na obrázku 2.2 je stejný jako standardním. Navíc jsou ale přítomny:

- SRR (substitu remote request – zástupný vzdálený požadavek) bit má recesivní hodnotu. Nahrazuje RTR bit a dává se tím přednost standardním rámcům, které mají stejné ID jako SID rozšířeného rámce.
- IDE bit má v rozšířeném formátu recesivní hodnotu.



Obrázek 2.2: Rozšířený formát CAN rámce (převzato z [4])

- Rozšířený identifikátor (EID) je 18-bitová hodnota. Navazuje na základní identifikátor (SID), který je stejný jako ve standardním formátu, na celkových 29 bitů. MSB je SID 10 a LSB je EIB 0.

- RB1 (reserved bit 1) bit je vyhrazený pro budoucí rozšíření.

2.1.1.2 Druhy CAN rámců

- Data frame (Datový rámec) nese aplikační data a odpovídá standardnímu nebo rozšířenému formátu.

- Remote frame (Vzdálený rámec) slouží k vyžádání datového rámce se stejným identifikátorem od ostatních uzlů. Rámec je stejný jako datový, až na to, že RTR bit je recesivní a neobsahuje žádná data.

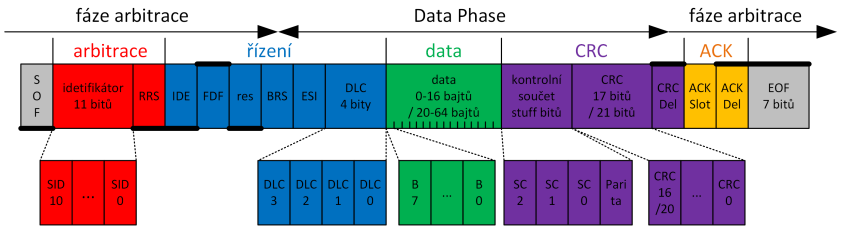
- Error frame (Chybový rámec) posílá uzel hned při detekování chyby v průběhu přenosu datového rámce. To vede k poškození tohoto rámce, a k detekci chyby ve všech ostatních uzlech. Tento rámec má speciální formát.

- Overload frame (Rámec přetížení) slouží k získání delší pauzy mezi datovými a vzdálenými rámci v mezi-rámcovém prostoru. Má také speciální formát.

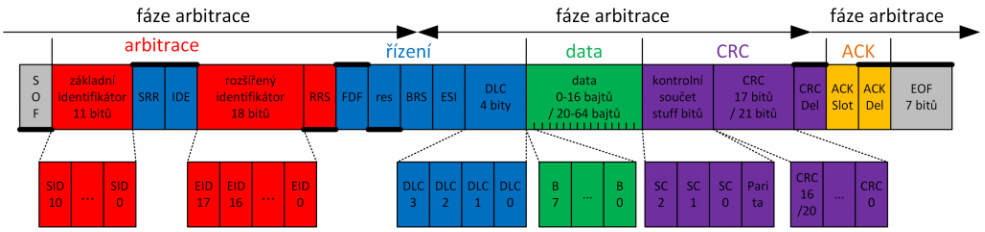
2.1.1.3 CAN FD

CAN FD [18] je nejnovější rozšířením protokolu (standardizován v roce 2015). Definuje nový FD formát rámce, který umožňuje přenášet až 64 bytů aplikačních dat. Datové pole je také umožněno přenášet až osmkrát vyšší rychlostí, než je rychlost pro arbitraci. Je hlavně určen pro aplikace, kde je potřeba rychlí přenos velkého objemu dat.

Tato verze je zpětně kompatibilní z klasickým CAN protokolem, ale kvůli proměnné rychlosti a navýšení dat musí při vysílání rámců v FD formátu podporovat CAN FD všechny uzly na sběrnici.



Obrázek 2.3: Standardní formát CAN FD rámce (převzato z [4])



Obrázek 2.4: Rozšířený formát CAN FD rámce (převzato z [4])

CAN FD specifikace je v ISO normě [18] odlišná od Bosch specifikace [17] a implementace jsou vzájemně nekompatibilní (došlo k vylepšení detekce chyb při přenosu) [4].

Používá se standardní i rozšířený identifikátor jako v CAN 2.0. Standardní formát FD rámce je popsán na obrázku 2.3 a rozšířený na obrázku 2.4. Oproti CAN 2.0 jsou ve formátech rámců tyto změny:

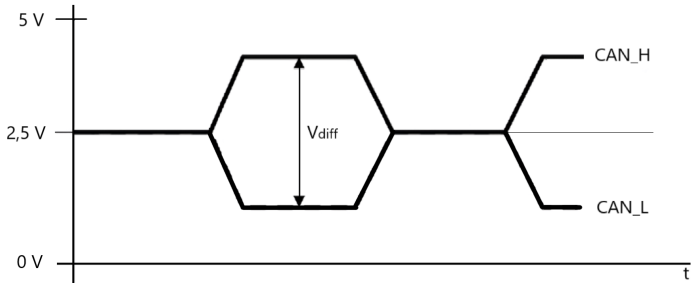
- RRS (remote request substitution) bit má dominantní hodnotu a nahrazuje RTR bit.
- FDF (flexible data rate format – FD formát) bit má recesivní hodnotu a značí použití FD formátu rámce.
- BRS (bit rate switch – přepínač bitové rychlosti) bit při recesivní hodnotě značí přepnutí na alternativní rychlost v datové fázi.
- ESI (error state indicator – indikátor chybového stavu) bit slouží k indikaci typu chyby. Recesivní hodnota odpovídá pasivní a dominantní hodnota aktivní chybě.
- DLC pole nově umožňuje kódovat bytové délky dat 12, 16, 20, 24, 32, 48 a 64. Pro tyto délky byli vyhrazeny popořadě kódy 1001_2 až 1111_2.
- Přibyl samostatný kontrolní součet vkládaných bitů a původní kontrolní součet byl navýšen na 17 až 20 bitů podle délky dat.

- Metoda vkládání bitů je stejná až do začátku CRC pole. Poté se po každém čtvrtém bitu vždy vloží doplňkový bit.
- Byla odebrána podpora pro vzdálené rámce.

2.1.2 CAN sběrnice (fyzická vrstva)

Následující popis se týká High-Speed (vysokorychlostní) CAN v normě [19]. Jiné varianty mohou používat pouze jeden datový vodič, jinou topologii, odlišné zakončení anebo nižší maximální rychlost.

Fyzická sběrnice se skládá ze dvou vodičů označované CAN_L a CAN_H, ve formě kroucené dvojlinky s impedancí 120 Ω. Jedná se o dva diferenciální signály. Rozhodování o stavu se provádí podle rozdílového napětí těchto signálů. Nominální napění v dominantním stavu je 1,5 V pro CAN_L a 3,5 V pro CAN_H. V recesivním stavu je 2,5 V pro oba signály. To dává nominální rozdílové napětí 2 V pro dominantní stav a 0 V pro recesivní.



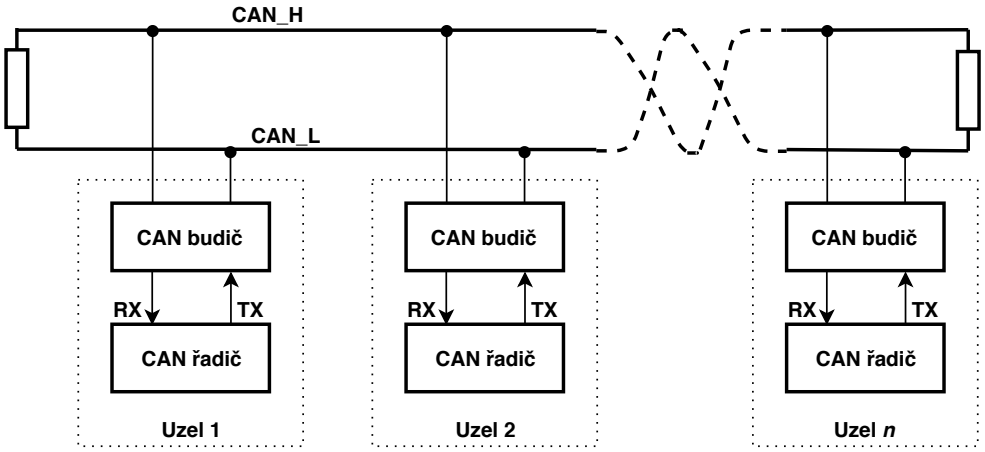
Obrázek 2.5: Průběh High-Speed CAN signálů

Vedení vodičů by mělo být co nejblíže lineární topologii pro zabránění odrazů signálů. Úsek od uzlu ke sběrnici by měl být co nejkratší. Na obou koncích je sběrnice zakončená odpory, které odpovídají impedanci kabelu. To zabraňuje odrazům signálu a zároveň dává do recesivního stavu, když stav není aktivně řízen žádným uzlem. Uzly na sběrnici se skládají z řadiče a budiče. Řadič implementuje spojový protokol. Budič zajišťuje propojení řadiče s fyzický médiem sběrnice. Viz znázornění na obrázku 2.6.

Rychlost přenosu je specifikována až do 1 Mbit s^{-1} při délce sběrnice do 40 m a úsecích mezi sběrnicí a uzlem do 30 cm. Délku je možné navýšit snížením rychlosti. V tabulce 2.1 jsou dosažitelné rychlosti podle délky.

2.1.2.1 Maximální počet uzlů

Maximální počet uzlů na sběrnici určují parametry CAN budiče a hodnota odporu sběrnice. Jde o to, že výstupní rozdílové napětí uzlu $U_{o(diff)}$ je dáno celkovým odporem sběrnice R_{bus} a výstupním proudem z budiče. Výstupní

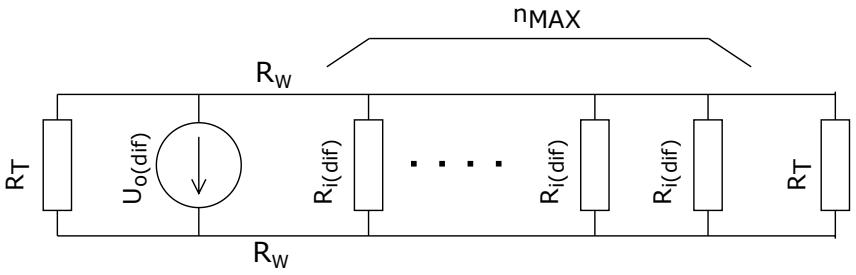


Obrázek 2.6: Zapojení High-Speed CAN s uzly

Tabulka 2.1: Orientační maximální rychlosti CAN sběrnice vzhledem k její délce

Délka sběrnice [m]	rychlost [kbit s^{-1}]
40	1000
100	500
200	250
500	100
1000	50

proud budiče je ale limitován, když je odpor sběrnice malý, úroveň výstupního rozdílové napětí je nedostatečná. Tím se při vysílání dominantního stavu nedosáhne minimální rozhodné napěťové úrovně, která tomuto stavu náleží.



Obrázek 2.7: Schéma obvodu odpovídající CAN sběrnici na obr. 2.6

Na obrázku 2.7 je schéma obvodu sběrnice, na kterém je vidět, že kromě zakončovacích odporů R_T je celkový odpor sběrnice určen i vstupním rozdílovým odporem $R_{i(diff)}$ každého budiče uzlu a odporem vodičů R_W. Každý budič má specifikovanou minimální hodnotu zátěžového odporu $R_{L(min)}$. Můžu

proto podle schéma s použitím vzorců pro sčítání odporů odvodit vzorec pro výpočet odporu sběrnice R_{bus} a sestavit nerovnici $R_{bus} \geq R_L$ rozepsanou v 2.1 (n značí počet uzlů na sběrnici). Úpravou této rovnice a dosazením $R_W = 0\,\Omega$ (v nejhorším případě je hodnota odporu vedení R_W je nula) získám rovnici 2.2 pro počet uzlů na sběrnici n a dosazením minimálních hodnot rovnici 2.3 pro výpočet maximálního počtu uzlů na sběrnici n_{max}.

$$\frac{R_T \cdot R_{i(diff)}}{(n-1) \cdot R_T + 2 \cdot R_{i(diff)}} + 2 \cdot R_W \geq R_L \tag{2.1}$$

$$n \leq 1 + R_{i(diff)} \cdot (\frac{1}{R_L} - \frac{2}{R_T}) \tag{2.2}$$

$$n_{max} = 1 + R_{i(diff)(min)} \cdot (\frac{1}{R_{L(min)}} - \frac{2}{R_{T(min)}}) \tag{2.3}$$

Běžné High-Speed CAN budiče mají parametry, které umožňují zapojení kolem 100 uzlů na sběrnici pokud R_T je $120\,\Omega$.

2.1.3 Závěr CAN

Použití CAN protokolu v přístupovém systému pro komunikaci panelů se serverem je vhodné díky jeho vlastnostem, zejména zabezpečení dat proti chybám a potvrzení správného doručení. Při použití široce implementovaného CAN 2.0, bude kapacita plně dostatečná pro přenos malého množství dat jako jsou různé číselné identifikátory.

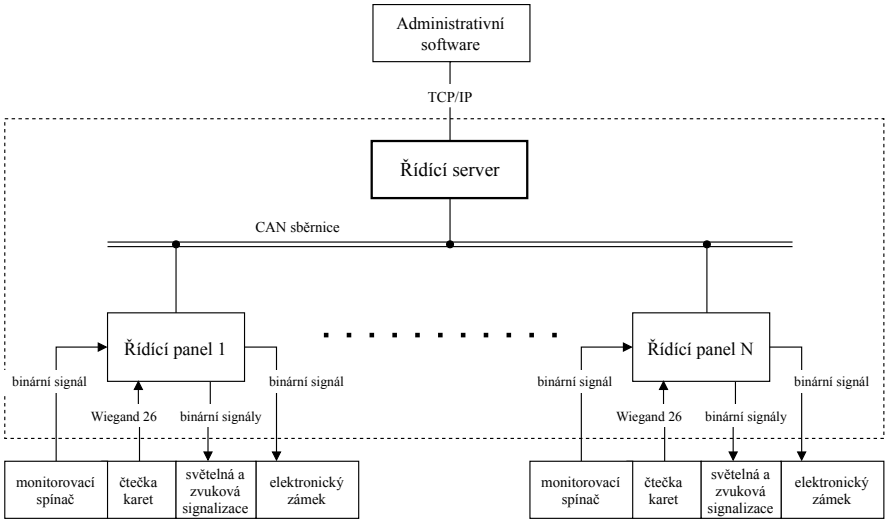
Jak jsem popsal, tak CAN má vlastnost, že maximální délka sběrnice závisí na rychlosti přenosu. Jelikož chci dosáhnou kompromisu, použiji lineární zapojení pro High-Speed CAN a rychlost zvolím do $250\,\mathrm{kbit\,s^{-1}}$. To dovolí použít vedení o délce až 200 m (v ideálním případě). Navíc oproti jiným topologiím je zakončování v lineárním zapojení pouze na koncích a není potřeba řešit zakončování každého CAN uzlu.

2.2 Struktura navrhovaného systému

Na obrázku 2.8 je vidět navržená vnitřní struktura systému, která odpovídá zadání. Jsou na něm vidět jednotlivé úrovně systému a použitá komunikační rozhraní. Realizované části jsou v přerušovaně ohraničeném obdélníku. Dále navrhnu architekturu pro řídící panel a server.

2.3 Architektura řídícího panelu

V této části vyberu platformu a další podpůrný software, který použiji pro realizaci řídícího panelu.



Obrázek 2.8: Vnitřní struktura přístupového systému

2.3.1 Výběr platformy

Nejdříve vyberu na jaké platformě bude implementován software. Na řídící panel jsou kladeny následující požadavky:

- Být vhodná pro vestavěné zařízení s real-time zpracováním;
- Zotavovat se z poruchy softwaru;
- Mít dostatek vstupně-výstupních pinů pro ovládání zámku, signalizaci na čtečce a monitorování stavu dveří;
- Podpora CAN 2.0 a Wiegand 26 rozhraní.

Pro vestavěné zařízení menšího rozsahu, zejména s real-time zpracováním jsou určeny MCU. Ty kromě procesoru integrují velké množství periférií a paměť do jednoho čipu.

Pro základní detekci a obnovu z poruchy programu se používá hardwarový časovač (tzv. Watchdog) s obvodem pro reset CPU. Watchdog je vyhrazený časovač, který po spuštění musí být periodicky obnovován z vykonávaného programu. Pokud tomu tak není vyvolá přerušení potažmo reset CPU.

CAN rozhraní lze přidat pomocí dedikovaného CAN řadiče a budiče. Jelikož chci zachovat nízké výrobní náklady a zjednodušit si návrh hardwaru, měla by vybraná platforma integrovat nejlépe obě tyto části.

Wiegand rozhraní je specifické a řadič pro něj není dostupný v MCU, jako tomu je u sériových rozhraní. Bude ho potřeba řešit softwarově.

Dále bych rád, aby se jednalo o mě známou platformu, pro níž jsou dostupné zdarma základní vývojové SW nástroje jako IDE a překladač GCC. Měl by k ní také existovat cenově dostupný ladící adaptér nebo vývojový kit.

Při výběru budu uvažovat jen MCU s procesorovým jádrem založeném na 32-bitové ARM architektuře ze skupiny Cortex-M. Kromě toho, že s ní mám zkušenosti, se jedná o populární volbu v nových návrzích, protože cena je srovnatelná z dříve používanými 8-bitovými a 16-bitovými MCU, a přitom nabízejí podstatně větší výkon. Díky popularitě je k nim dostupno mnoho HW i SW nástrojů. Z rodiny Cortex-M bude výkonnostně stačit i základní jádro Cortex-M0.

Vybíral jsem se mezi produkty od firem STMicroelectronics, NXP Semiconductors apod. Existuje několik kandidátů, které mají kromě CAN řadič zabudovaný i CAN budič. Jelikož zadavatel měl už koupené vývojové kity pro jednoho z nich, zvolil jsem právě ten. Je jím mikrokontrolér NXP LPC11C24 [20], který splňuje všechny požadavky. Jeho hlavní charakteristiky jsou: 50 MHz CPU, 8 kB operační paměti, 32 kB flash paměti, Watchdog časovač, C_CAN řadič [21] s ovladačem v ROM, CAN budič NXP TJF1051 [22], 36 GPIO pinů, nízká spotřeba, malá velikost a dostupná knihovna k periferiím. Pro představu, cena stanovená výrobcem je v přepočtu přibližně 39 Kč při odběru 10000 kusů [23].

2.3.2 Výběr OS

Vybrané MCU s 32-bitovým jádrem z rodiny ARM má dostatek výkonu a paměti pro běh operačního systému pro vestavěné zařízení. Proto jsem se rozhodl, v zájmu zjednodušení implementace softwaru a větší přenositelnosti firmwaru, se porozhlédnout po operačním systému reálného času (RTOS) pro vybranou platformu.

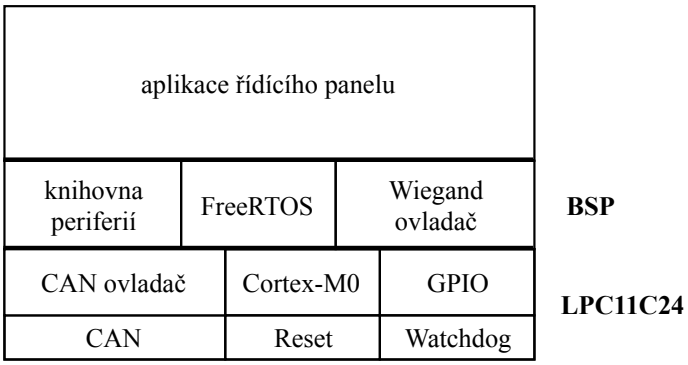
Dostupných RTOS, fungujících na vybrané platformě, s otevřenou licencí je mnoho. Vybral jsem ale systém FreeRTOS [24], se kterým mám nejvíce praktických zkušeností. Ten je také nativně podporován v IDE od výrobce pro zvolenou platformu.

2.3.3 Navržená architektura

Platformu pro řídící panel jsem vybral MCU LPC11C24, který má vše potřebné pro realizaci. Firmware implementuji s použitím FreeRTOS, integrovaného CAN ovladače a knihovny pro periferie od výrobce. Tato architektura je vidět na obrázku 2.9

2.4 Architektura řídícího serveru

Server bude dedikované zařízení s externím zdrojem připojené k panelů přes CAN a spravované přes Ethernet. Software bude potřeba realizovat od začátku, neboť budu používat vlastí aplikační protokol pro CAN. Musím proto vybrat vhodný typ platformy nejspíše plnohodnotný počítač a zvolit podpůrný software.



Obrázek 2.9: Architektura řídícího panelu

2.4.1 Výběr OS

V základě jsou dva přístupy jak realizovat řídící software, bez operačního systému nebo s ním. Bude potřeba komunikovat přes TCP/IP a CAN. Na serveru také bude běžet databázový software.

Bez použití existujícího OS by implementace byla velmi náročná. Proto jsem se ihned přiklonil k použití plnohodnotného moderního OS vhodného pro server. Nejpoužívanější jsou Microsoft Windows, GNU/Linux a BSD.

Pro vestavěné systémy je vhodný zejména GNU/Linux, neboť je ho možné konfigurovat podle potřeby aplikace. Je ho také možné přeložit s podporou pro real-time běh, pokud to bude potřeba.

2.4.1.1 SocketCAN

GNU/Linux podporuje CAN řadiče přes znakové a síťové ovladače zařízení.

Linux kernel má implementovaný standardní model pro CAN ovladače a CAN protokoly nazývaný SocketCAN [25]. Je možné používat znakové (jako virtuální) i síťové ovladače zařízení. Poskytované systémové API je téměř stejné jako pro klasické TCP/IP sokety, což je velkým přínosem, oproti specifickým API každého CAN ovladače nebo jiné implementace.

Zároveň s tím jsou dostupné ovladače pro vybrané CAN řadiče přímo v jádře. V současné době se jedná například o NXP SJA1000, Microchip MCP251x, ale je jich mnohem více [26]. Často je také kompatibilní ovladač dodáván přímo od výrobce.

Jsou implementované nízkoúrovňové protokoly jako `CAN_RAW` a `CAN_BCM`. První umožňuje surový přístup ke CAN ramcům a datům. Druhý implementuje periodické posílání zpráv přímo v jádře. Z aplikačních protokolů jsou obsaženy SAE J1939 (používá se k řízení motorů) a ISO-TP (pro přenos větších paketů, než umožňuje jeden CAN rámec).

2.4.2 Výběr databáze

V systému je potřeba databáze, kde budou uloženy čísla karet spárované s číslem, které reprezentuje umístění dveří. Odhaduji, že struktura bude jednoduchá. Implementovat vlastí databázi je zbytečné, existuje velké množství bezplatného databázového softwaru pro různé účely. Proto nějakou vhodnou pro použití v řídím serveru vyberu.

Po krátkém průzkumu jsem zjistil, že se používají dva základní druhy databází. Klasické relační a nerelační tzv. NoSQL, ty pro data místo tabulek a závislostí používají jiné jednoduší struktury. Jednou takovou, která mě zaujala je databáze typu klíč-hodnota, ta svoje data ukládá do asociativního pole. Datový typ dat uložených pod klíčem databáze nerozlišuje ani nekontroluje. Tato zodpovědnost je přenesena na aplikaci. Její výhody jsou rychlost a menší nároky na prostředky. Nejpopulárnější open source implementací pro tento typ je Redis (v roce 2019 podle online žebříčku [27]).

Po bližším prozkoumání dokumentace k Redis databázi [28] jsem zjistil: Běží jako samostatná aplikace, ke které se připojuje přes TCP spojení nebo soket; Data jsou umístěna v operační paměti, ale podporuje datovou persistenci na disku; Lze s ní pracovat jednoduše pomocí několika příkazů z klientů, které jsou dostupné pro mnoho programovacích jazyků; Má oficiálního konzolového klienta; Do hodnoty klíče lze uložit jakákoliv binární data; Je možné mít více instancí pro redundanci; Nepodporuje pokročilé zabezpečení přístupu, protože je určen pro běh v důvěrném prostředí.

Rychlostí zpracování je Redis na tom velmi dobře. Na oficiálních stránkách jsou dostupné testy [29] provedené na Intel Xeon E5520. Jedním z nich je test verze 2.6.14, kde se provedlo 100000 čtení položek o velkosti 3 B z 50 paralelních klientů na stejném stroji jako běží server. Latence příkazů pro čtení, který reprezentuje žádost o přístup, je v tomto testu do 3 ms.

Z uvedených skutečností jsem dospěl k závěru, že Redis jako databáze bude pro přístupový systém vhodná. V případě potřeby ji lze eventuálně použít jako cache ke klasické relační databázi.

2.4.3 Výběr platformy

Jakožto jediná řídící část systému musí mít dostateční výkon k: Vyřizování všech požadavků o přístup od řídících panelů; Komunikaci s administrativní aplikací; Dostatek paměti pro databázi; Umožňovat běh renomované distribuce GNU/Linux a Redis databáze; Zároveň musí disponovat CAN a Ethernet rozhraním.

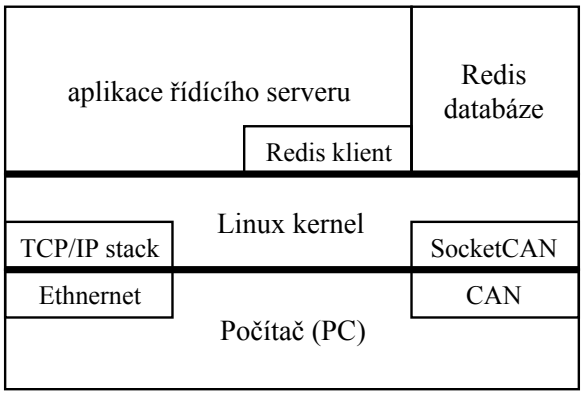
Výše uvedené věci splňují počítače s dostatkem operační paměti a výkoným procesorem. Vhodnými platformami tedy nejsou MCU. Rozhodl jsem se proto porozhlédnout po počítačích třídy PC. Založených na procesorové architektuře x86 nebo ARM.

Zásadní nevýhodou této volby je, že běžná PC nemají integrované rozhraní CAN. Přestože PC mají dostatek možností pro připojení dalších periférií, většinou se jedná o komplexní sběrnice jako USB a PCI Express. Běžně dostupné CAN adaptéry pro tyto sběrnice jsou drahé, neboť jsou cíleny na průmyslové zákazníky. Proto může být lepší vybrat PC s přístupnou jednoduchou sběrnicí jako SPI nebo I2C. Pro tyto sběrnice můžu levně vyrobit vlastí CAN adaptér například s řadičem MCP2515 [30] nebo SJA1000 [31], které mají ovladače v jádře Linuxu od verze 2.6.38. Hotová řešení s těmito čipy je možné i zakoupit.

Počítače s přístupnou SPI sběrnicí jsou třeba dnes rozšířené jednodeskové počítače. Přestože mají oproti PC omezené prostředky, k vývoji řídícího softwaru mi přijdou dostatečné. Ty které splňují i ostatní požadavky jsou mimo jiné Raspberry Pi a BeagleBone-X15. Proto jsem nakonec jako platformu pro počáteční vývoj vybral Raspberry Pi, který zároveň už vlastním. S tím, že pro finální verzi serveru přesto plánuji použít PC s USB–CAN adaptérem, neboť i ty nejlevnější nabízejí větší výkon něž jednodeskové počítače a k tomuto účelu lze použít téměř jakýkoliv počítač. Náklady na pořízení adaptéru, vzhledem ke zbytku systému, budou relativně malé, protože server bude nejspíše jeden. Zároveň takové řešení bude vzbuzovat větší důvěru.

2.4.4 Navržená architektura

Použiji počítač třídy PC rozšířený o CAN rozhraní s GNU/Linux. Řídící software implementuji s použitím Redis databáze a SocketCAN modulu. Tato architektura je vidět na obrázku 2.10



Obrázek 2.10: Architektura řídícího serveru

2.5 Napájení a propojení zařízení v systému

V systému budou desítky řídících panelů a k nim připojená doplňková zařízení, které vyžadují externí zdroj napětí: čtečky karet a elektrické zámky. Napájení

serveru bude samostatné, protože bude jeden a hardwarem je PC, které má vlastní napájecí zdroj. Musím tedy vyřešit jak budou ostatní zařízení napájeny a zapojeny.

2.5.1 Napájení panelů

Panely budou mít vstup pro externí napájení. To je možné řešit dvěma způsoby:

Samostatné napájení

Každý panel má vlastní hlavní zdroj. Tento způsob v zásadě neomezuje možný příkon a počet panelů. V blízkosti každého panelu musí být přístupná elektroinstalace budovy. Také to znamená pořízení řádově desítky zdrojů, které je třeba někam umístit.

V tomto případě je nutné izolovat CAN sběrnici u každého panelu a to kvůli možné neshodě potenciálů zemí jednotlivých panelů.

Společné napájení

Všechny panely jsou napájeny z jednoho společného hlavního zdroje. V takovém případě je potřeba instalovat vedení od tohoto zdroje k panelům. Toto řešení může omezit rozsah sítě panelů a přenášený výkon, kvůli případným ztrátám na dlouhém vedení.

2.5.2 Napájení doplňkových zařízení

Napájení doplňkových zařízení je možné řešit čtyřmi způsoby:

Samostatné napájení

Doplňková zařízení u každého panelu má vlastní zdroj. Každé zařízení může mít jiný vstupní rozsah napětí. V blízkosti musí být přístupná elektroinstalace budovy. Také to znamená pořízení řádově desítky zdrojů, které je třeba někam umístit.

Společné napájení

Panel a jeho doplňková zařízení mají jeden společný hlavní zdroj. V tomto případě musí mít tato skupina průnik v rozsahu vstupního napětí.

Napájení z panelu

Čtečka karet a případně zámek jsou napájeny přímo z panelu, který má vhodné napájecí výstupy. Tento způsob zvyšuje složitost panelu a jeho příkon, zařízení ale stačí zapojit jen do panelu a není nutné mít další zdroje.

Kombinace prvního a třetího

Jedno ze zařízení je napájeno přímo z panelu a druhé má vlastní nebo společný zdroj. Často jsou právě čtečky karet napájeny z panelu, protože mají malý příkon.

2.5.3 Volba napájení zařízení

Pro napájení všech zařízení jsem se rozhodl použít jeden hlavní zdroj. Doplňková zařízení budou napájena z panelů, jelikož stejně musejí být k němu zapojeny i když by z něho nebyly napájeny.

Důvodem pro toho rozhodnutí je ušetření na počtu zdrojů a zjednoduší instalace systému. Využiji také toho, že se stejně musí instalovat kabel pro komunikaci se serverem a kabel pro napájení povede společně s ním.

Mohl jsem tento způsob zvolit, protože dveře jsou hustě rozmístěny a je jich přibližně 20 a tedy panelů bude 10. Délka vedení bude do 100 m při průměrném rozestupu až 5 m a celkový přenášený výkon bude určitě menší než 310 W, když uvažuji v nejhorším případě 12 W zámek, 2 W čtečka a panel 2 W (předpokládám že příkon panelu bude podobný čtečce).

Použiji stejnosměrné napájení do 120 V z bezpečnostních důvodů, to je dostatečné pro požadovaný příkon a délku vedení.

2.5.4 Kabely pro propojení panelů

Panely budou zapojeny k serveru a ke společnému zdroji. Musím tedy vybrat vhodný kabel nebo kabely, s tím že je potřeba tyto vodiče: kroucená dvojlinka pro CAN sběrnici, společná zem (data a napájení) a jeden vodič pro napájecí napětí.

S ohledem na společné napájení a lineární topologii CAN sběrnice, napájecí a datové vodiče povedou od jednoho panelu k druhému, dále k serveru a zdroji.

Vlastnosti zvolené kroucené dvojlinky musí být vhodné pro CAN. Výkonové vodiče by měli být schopny přenést požadovaný příkon. Mám tedy dvě možnosti vybrat dva kabely nebo jeden, který má dostatek vodičů a je vhodný pro data i napájení. Po průzkumu cen vhodných kabelů jsem došel k závěru, že použití dvou kabelů by bylo značným navýšením ceny celého systému. Proto je vhodné použít jeden kabel.

V době tvorby práce jsou velmi dostupné síťové kabely a to zejména čtyřpatrové kroucené dvojlinky konkrétně Kategorie 5e (CAT-5e). Tento typ kabelu je standardizovaný v [32] a [33]. Jeho cena začíná už na 6 Kč/m. Přestože je určen zejména sítě jako Ethernet bývá v průmyslu používán pro CAN sběrnici. Zároveň existuje norma pro Ethernet [34], která specifikuje systém nazývaný Power over Ethernet (PoE), pro přenos energie v Ethernetové síti za použití právě kabelů CAT-5e nebo lepších. V tomto systému je možné přenášet až 90 W na 100 m při použití všech čtyř párů.

2.5.4.1 CAN s kabelem CAT-5e

V případě použití pro CAN je jediným omezením kratší délka sběrnice a menší počet uzlů.

Délka je omezena odporem tohoto kabelu, který je do 93,8 $\mathrm{m\Omega\,m^{-1}}$ [32]. To je více než nominální 70 $\mathrm{m\Omega\,m^{-1}}$ podle [19]. Avšak pro CAT-5e se skutečně používají měděné vodiče s průměrem 0,51 mm (AWG 24), které mají odpor nominálně 85 $\mathrm{m\Omega\,m^{-1}}$. Uvedené hodnoty jsou při 20 °C

Kabel CAT-5e má menší impedanci 100 Ω a tedy musí mít i menší hodnotu zakončovacích odporů. Tato hodnota je ale v dovoleném rozsahu pro High-Speed CAN. U zakončení je dovoleno se odchýlit od 120 Ω, ale zmenšením hodnoty zakončovacích odporů se sníží počet uzlů v síti [19].

Potřebuji vědět, jestli délka anebo počet uzlů nejsou limitující. Vypočítám proto maximální počet uzlů s použitím rovnice 2.3 v sekci 2.1.2.1. Dosazením parametrů z datového listu pro budič [22]: $R_{i(diff)(min)} = 19\,000\,\Omega$ $R_{L(min)} = 45\,\Omega$ a odporu zakončení $R_{T(min)} = 100\,\Omega$. Z toho vychází, že je možné připojit až 42 uzlů.

Pro vypočtení maximální délky sběrnice L_{max} [m] použiji rovnici 2.4 převzatou z [35], která závisí na:

- maximální rozhodné napětí dominantní úrovně budiče $U_{th(max)}$ [V],
- rezerva pro detekci dominantní úrovně k_{sm},
- maximální odpor vodiče na jednotku délky ρ_{max} [$\Omega\,\mathrm{m^{-1}}$],
- minimální rozdílové výstupní napětí budiče pro dominantní úroveň $U_{o(diff)(min)}$ [V],
- minimální vstupní rozdílový odpor budiče $R_{i(diff)(min)}$ [Ω],
- minimální zakončovací odpor $R_{T(min)}$ [Ω],
- maximální počet uzlů n_{max}.

$$L_{max} \leq \frac{1}{2 \cdot \rho_{max}} \cdot \left(\frac{U_{o(diff)(min)}}{U_{th(max)} + k_{sm} \cdot (U_{o(diff)(min)} - U_{th(max)})} - 1\right) \cdot \frac{R_{T(min)} \cdot R_{i(diff)(min)}}{R_{i(diff)(min)} + (n_{max} - 1) \cdot R_{T(min)}} \tag{2.4}$$

Maximální délka vychází 175 m při dosazení hodnot z datového listu pro budič [22] a $k_{sm} = 0,2$ $R_{T(min)} = 100\,\Omega$ $\rho_{max} = 0{,}0938\,\Omega\,\mathrm{m^{-1}}$. Vypočtené hodnoty jsou tedy dostačující.

2.5.4.2 Napájení přes kabel CAT-5e

Hlavním problémem je, jak velký výkon dokáže tento typ kabelu skutečně přenášet. PoE má stanovený limit 90 W, neboť je povoleno na jednom vodiči nejvýše 0,5 A [34]. To je ale hlavně kvůli splnění SELV (Safety Extra-Low Voltage), kde dostupný výkon nesmí překročit 100 W [36].

Maximální stálý proud pro každý vodič CAT-5e kabelu je přibližně 1,5 A při 25 °C. Budu uvažovat konzervativní 1 A na vodič, tím se získám rezervu pro vyšší teplotu prostředí. Kabel má čtyři páry, k napájení zbývají tři, celkový proud na kabel je tedy 3 A. Napětí uvažuji maximálně 60 V. Pro panel plánuji použít běžně dostupné obvody pro DC/DC měniče ke snížení na 12 V nebo 24 V a ty mají většinou vstupní limit právě kolem 60 V. Z toho vychází limit 180 W (bez přenosových ztrát). Dále zohledním ztráty na vedení včetně propojení.

Standardní konektory RJ-45 vhodné pro PoE včetně zástrčky mají přechodový odpor typicky 20 mΩ–50 mΩ [37]. Při splnění testů pro podporu PoE v normě [38], má změna tohoto odporu po 100 cyklech zapojení a odpojení pod zátěží být do 20 mΩ.

Odpor měděného vodiče v kabelu s průměrem 0,51 mm je 85 $\mathrm{m\Omega\,m^{-1}}$ při 20 °C. V zátěži bude odpor přibližně 95 $\mathrm{m\Omega\,m^{-1}}$ kvůli zahřívání vodičů ze ztrát. Předkládám nárůst o 15 °C při 1 A na vodič a prostředí 35 °C.

Každá panel bude muset mít pro snadné propojení do lineární sítě dvě zástrčky vedle sebe interně propojené. Panely budou rozmístěny průběžně, odpor vedení u každého z nich bude stoupat se vzdáleností od zdroje. Jelikož nevím jak přesně budou rozmístěny a chci zjednodušit výpočet, budu uvažovat zjednodušený model, kdy všechen výkon je přenášen až na konec vedení (nejhorší případ).

Pro výpočet maximálního výkonu dostupného u panelů uvažuji nejvýše 105 m kabelu s odporem 95 $\mathrm{m\Omega\,m^{-1}}$, 22 konektorů a zástrček (11 panelů pro 22 dveří) po 70 mΩ. Odpor interního propojení zástrček zanedbám. Při přenosu 180 W (60 V 3 A na výstupu zdroje) přes tři páry vychází, že ztráty na vedení budou $P = (2 \cdot R \cdot I^2) \cdot 3 = (2 \cdot (0,095 \cdot 105 + 0,07 \cdot 22) \cdot 1^2) \cdot 3 = 69\,\mathrm{W}$ a pro všechna zařízení je tedy dostupné až 111 W.

Závěrem je, že použití kabelu CAT-5e a RJ-45 znamená možný příkon jednoho panelu včetně doplňkových zařízení do 10 W. Avšak to je nejhorší případ v zjednodušeném modelu; Skutečně dostupný výkon bude vyšší a to zejména, při vhodném umístění zdroje například do půlky vedení – tím se totiž zmenší odpor vedení (včetně propojení) od zdroje k panelům o polovinu.

Přes zjištěná omezení volím k propojení kabel CAT-5e s RJ-45 konektory, a to kvůli nízkým nákladům na propojení a zjednodušení instalace. V realizaci budu počítat s omezím příkonu zámků napájených z panelu a vyhradím pro ně možnost použít vlastní zdroj, jelikož mají největší odběr. Pro napájení panelu a čteček je to dostačující.

2.5.5 Kabely pro připojení doplňkových zařízení

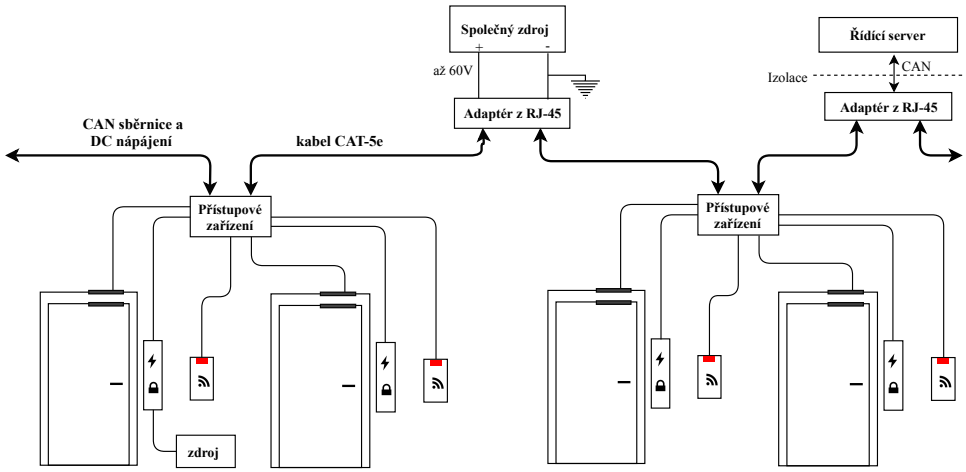
Doplňková zařízení mají vlastní propojovací kabely, které jsou dané výrobcem konkrétního produktu. Proto nebudu vybírat konkrétní datovou ani napájecí kabeláž. Zapojení do panelu budu řešit svorkovnicí, tím půjde připojit většinu vodičů používaných pro tyto zařízení.

2.5.6 Shrnutí napájení a propojení zařízení v systému

Panely budu napájet ze společného hlavního zdroje; Ty zapojím do sítě s řídícím serverem kabelem CAT-5e s RJ-45 konektory pro 60 V DC napájení a CAN sběrnici; Z toho plyne, že pro zdroj a server budu potřebovat adaptér pro připojení k této síti; Panely budou mít dvě RJ-45 zástrčky pro tento účel.

Počítám s délkou sítě do 105 m, kde může být nejvýše 11 panelů pro až 22 dveří. Dovolený příkon všech panelů je přibližně 111 W.

Doplňková zařízení budu napájet 12 V nebo 24 V DC přímo z panelu; Zámky s větším příkonem budou muset mít vlastní zdroj.



Obrázek 2.11: Zvolené napájení a propojení zařízení v systému

Kapitola 3

Realizace

Na základně analýzy a návrhu systému jsem vytvořil firmware pro řídící panel a software pro řídící server. Zároveň pro řídící panel jsem vytvořil prototyp a navrhl elektrické zapojení pro obsluhu dvou dveří k vytvoření první verze. Pro připojení napájecího zdroje a řídícího serveru jsem navrhl a vytvořil adaptér.

3.1 Vývojového zapojení řídícího panelu

V architektuře jsem pro řídící panel zvolil platformu s MCU NXP LPC11C24 s integrovaným CAN rozhraním. K započetí vývoje firmwaru jsem potřeboval sestavit minimální zapojení, které umožňovalo komunikaci se čtečkami a ovládání zámků. Dvě LF RFID čtečky a dvě vývojové desky LPCXpresso11C24 poskytl zadavatel.

3.1.1 Poskytnuté LF RFID čtečky

Jedná se o 125 kHz RFID čtečky kompatibilní pouze s EM4100 typem karet. Mají vstupní napětí 8–15V, Wiegand rozhraní, možnost přepínání mezi Wiegand 26 a Wiegand 34 protokolem, a bzučák. Signalizační LED si ovládají sami, nelze je ovládat z aplikace. RFID čtečka je vidět na obrázku 3.1 a v tabulce 3.1 je seznam vodičů.

3.1.2 Vývojová deska LPCXpresso11C24

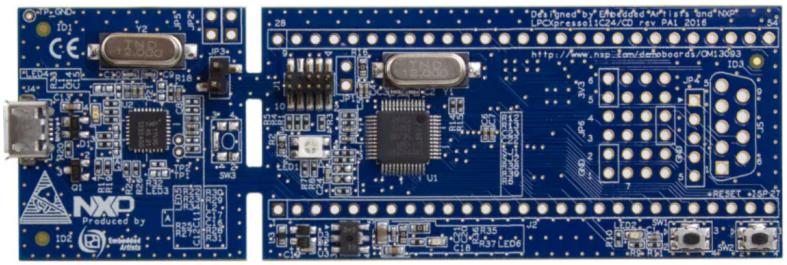
Jak jsem zmínil pro MCU, je dostupná vývojová deska LPCXpresso11C24 od Embedded Artist s odlučitelnou ladící sondou (debug probe) typu CMSIS-DAP. Deska je vyobrazena na fotce 3.2.

Tato vývojová deska má vyvedené všechny piny z MCU na pinové lišty a obsahuje součástky v tabulce 3.2. Napájet desku je možné 5 V z USB nebo

Obrázek 3.1: Použitá 125 kHz RFID čtečka

Tabulka 3.1: Vodiče na použité LF RFID čtečky

8 V–15 V DC
Zem (data a napájení)
Wiegand data 0
Wiegand data 1
Bzučák (aktivace uzemněním)
Wiegand 34 režim (aktivace uzemněním)



Obrázek 3.2: Vývojová deska LPCXpresso11C24 s CMSIS-DAP (vlevo) a MCU LPC11C24 (vpravo)

pinu pro externí napájení. Ladící sondu na desce je možné vypnout nebo odlomit od hlavní části.

3.1.3 Další komponenty

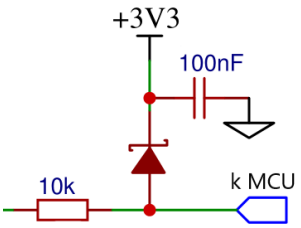
Kromě vývojové desky pro MCU a čteček byly potřeba další komponenty pro sestavení vývojového zapojení, které jsou popsány níže.

Tabulka 3.2: Součástky na vývojové desce LPCXpresso11C24

LPC11C24	hlavní MCU
LPC11U35	ladící sonda (CMSIS-DAP rozhraní)
USB micro-B	ovládání ladící sondy a napájení desky
SWD konektor	připojení externí ladící sondy
RGB LED	ovladatelné z hlavního MCU
Tlačítka	reset a ISP (přepnutí do programovacího režimu)
Oscilátory	dva 12 MHz krystaly pro obě MCU
Zdroj	3,3 V lineární regulátor pro obě MCU

3.1.3.1 Připojení RFID čteček

Signály rozhraní Wiegand mají 5 V logiku, kdežto MCU má 3,3 V. K připojení k MCU jsem nejdříve použil odpor v sérii pro omezení proudu a vodiče připojil na 5 V tolerantní piny MCU. Těchto pinů je ale nedostatek pro dvě rozhraní, proto jsem přidal diodu, jak je vidět na obrázku 3.3, která limituje napětí na vstupu MCU a je poté signály možné připojit na libovolné piny. Obě tyto řešení snižují rychlost změny signálu, ale to pro Wiegand rozhraní stačí. Pro vyfiltrování vysokofrekvenčního rušení jsem vstupy signálů dal feritové perly o impedanci 600 Ω při 100 Mhz, protože čtečky mohou být vzdálené několik desítek metrů.



Obrázek 3.3: 5 V/3,3 V logika zapojení s diodou

Wiegand signály data 0 a data 1 jsem přes uvedený obvod připojil v hardwaru na dva piny na MCU. Zvolil jsem piny, které jsou na stejném portu[7], protože obsluha přerušení na MCU je pro jednotlivé porty a ne piny. To jsem udělal pro obě čtečky.

Jelikož čtečky mají vstupní napětí 8 V–15 V DC, potřeboval jsem tedy zdroj 12 V. Pro potřeby vývoje jsem použil modul DC/DC měniče s obvodem XL6009, který jsem měl z jiného projektu a slouží ke zvýšení vstupního napětí. Na vstup tohoto modulu jsem napájel USB konektor a tím vytvořil levný zdroj se vstupem 5 V. Zdroj jsem připojil k nabíječce na telefon, která dodá proud až 1 A. Výstupní napětí je nastavitelné potenciometrem mezi 6 V–30 V. Napájení

[7]Port zastřešuje skupinu pinů.

čtečky s tímto zdrojem bylo k mému překvapení úspěšné, jelikož jsem čekal, že tento spínaný zdroj bude rušit čtení karet.

3.1.3.2 Světelná a zvuková signalizace

Některé RFID čtečky pro tento účel umožňují ovládat vestavěný bzučák a stavové LED. Tím, že je vyveden vodič pro každou barvu a bzučák. Aktivace signalizace probíhá uzemněním příslušného vodiče. Napětí na tomto vodiči může být různě velké, velikost proudu je také nejistá. K jejich aktivaci jsem proto použil osmi tranzistorové pole ULN2803A. Tyto tranzistory jsou páry bipolárních tranzistorů. To umožňuje spínat až 50 V a 500 mA. Řídící vstupy toho pole je možné přímo připojit k 3,3V pinům MCU.

Použitá čtečka nemá možnost řízení stavové LED. Běžně čtečky ale tuto možnost mají a podporu bude implementovat ve firmwaru. Pro simulaci jsem si připojil chybějící zelené a červené LED na tranzistorové pole. Celkem jsem připojil šest řídících vstupů pole k pinům MCU pro ovládání zelené barvy, červené barvy a bzučáku pro dvě čtečky.

3.1.3.3 Ovládání zámků

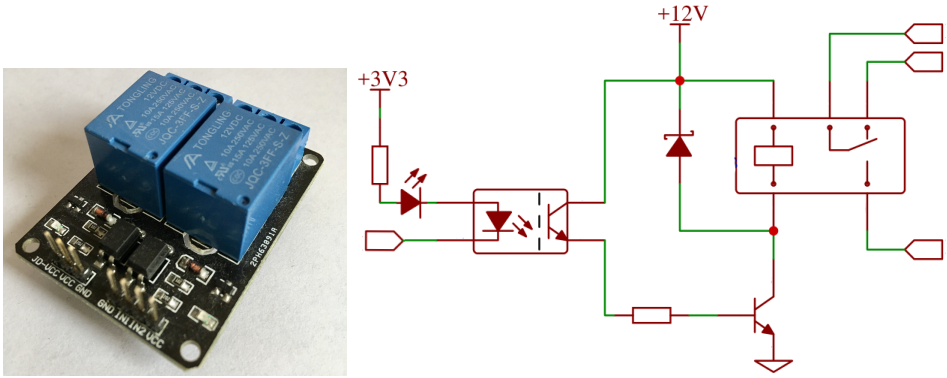
Elektrické zámky a otvírače jsou ovládané výkonovými kontakty nebo přivedením napětí. Nelze je řídit běžnými polovodičovými spínači, proto se musí ovládat přes relé. Standardně mají komerční řídící panely pro tento účel elektromagnetické relé s jedním přepínacím kontaktem (SPDT – Single Pole Double Throw). To znamená že má kontakty `NO` (normálně otevřeno), `NC` (normálně sepnuto) a `COM` (společný). Existují i polovodičová relé, ale ty nemají z principu konfiguraci přepínacím kontaktem.

Elektromagnet v relé se musí ovládat výkonovým tranzistorem, jelikož mají značný odběr. Z principu fungování, elektromagnetické relé dává přirozenou izolaci od spínaných kontaktů, ale řídící signál od MCU pro tranzistor musí být také izolován pro zabránění přenosu rušení při spínaní/rozepínání cívky v relé.

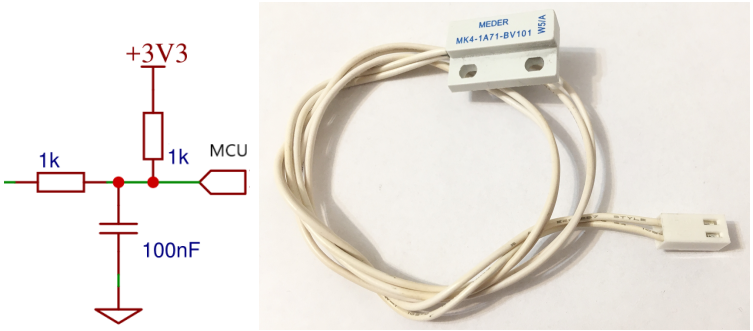
Na trhu jsou již dostupné kompletní relé moduly, které stačí připojit k napájení a jdou ovládat přímo z MCU. Proto jsem se pro toto zapojení rozhodl takový modul zakoupit. Module je na obrázku 3.4. Tento modul integruje dvě elektromagnetické relé spínané bipolárními NPN tranzistory s optočleny pro izolaci řídících signálů. Jedná se o 12 V modul, protože 12 V je hlavní napěťová větev kvůli 12 V vstupnímu napětí čteček, zároveň tuto větev potom použiji k napájení zámků. Příkon v aktivním stavu je přibližně 0,4 W (krátkodobě až 1 W). V neaktivním stavu nic neodebírá.

3.1.3.4 Monitorování stavu dveří

Pro přidání podpory pro detekci otevření dveří jsem použil magnetický spínač, který jsem přes obvod na obrázku 3.5 připojil přímo na vstupní pin MCU.



Obrázek 3.4: Modul s dvěma 12 V elektromagnetickými relé (vlevo) a zapojení jednoho relé (vpravo)



Obrázek 3.5: Magnetický spínač (vpravo) a vstupní odvod (vlevo)

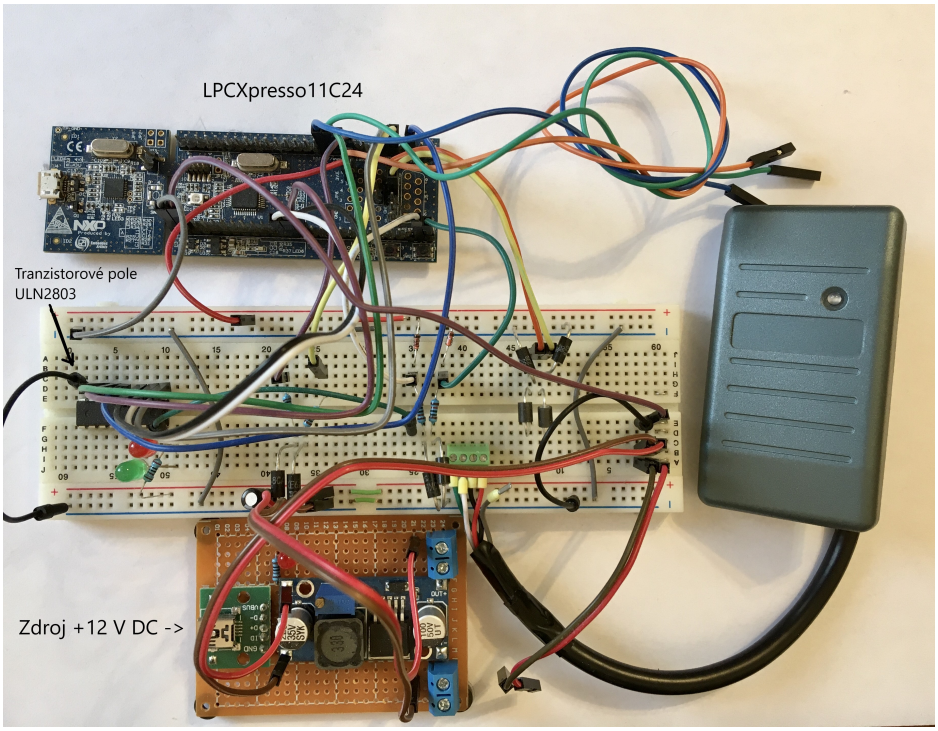
Sériové zapojený odpor s kondezátorem je dolní propust pro vyfiltrování případných zákmitů. Výtažný odpor určuje výchozí logickou hodnotu vstupu.

3.1.4 První vývojové zapojení hardwaru

První vývojového zapojení jsem sestavil na nepájeném poli z vývojové desky (rev. A), uvedeného relé modulu s dvěma relé, zdroje 12 V, dvou LF RFID čteček a dalších uvedených součástek. To jsem napájel ze dvou USB portů. Zapojení bylo velmi podobné jako na obrázku 3.6 v kterém akorát chybí relé modul, senzor a druhá čtečka. V tomto zapojení nebyl připojen žádný el. zámek, protože v té době jsem ještě žádný nepořídil a ani jsem k němu neměl dostatečný zdroj.

3.2 Počáteční firmware řídícího panelu

V počátečním vývoji řídícího panelu jsem používal první verzi vývojové zapojení, na kterém jsem implementoval čtení karet, ovládání zámků a stavových



Obrázek 3.6: První vývojové zapojení panelu (přibližné)

signálů (LED a bzučák). V této fázi nebyla implementována komunikace se serverem a panel pracoval v off-line režimu. K tomuto účelu jsem implementoval jednoduchou lokální databázi pro čísla karet.

3.2.1 Úvodní kroky

Jako podpůrný software jsem se v architektuře rozhodl použít knihovnu periférií od výrobce MCU a FreeRTOS. Zde jsou kroky, které jsem udělal před zahájením implementace firmwaru:

1. Nainstaloval jsem si IDE NXP MCUXpresso verze 10.3.1. To obsahuje GCC pro ARM a umožňuje ladění. Zároveň umí spolupracovat s FreeRTOS.

2. Prošel uživatelský manuál k LPC11C24.

3. Podrobně seznámil s knihovnou od výrobce, nazývanou LPCOpen, která má rozhraní pro ovládání periférií MCU.

4. Stáhl ukázkový projekt pro LPC11C24 s blikající LED. Tento projekt jsem nahrál do vývojové desky. Tím jsem otestoval korektnost nastavení IDE a funkčnost MCU na desce.

5. Z ukázkového projektu jsem odstranil nepotřebné části a nalinkoval ho na poslední verzí implementace knihovny LPCOpen kompatibilní s LPC11C24.

6. Do projektu jsem naimportoval FreeRTOS 10.1.1 a nastavil, tak aby zabíral co nejméně místa. Zakázal jsem dynamickou alokaci paměti.

7. Do projektu jsem přidal blikání LED z FreeRTOS task pro vyzkoušení správného nastavení.

8. Zprovoznil jsem UART komunikaci pro ladící výpisy.

9. V tomto stavu byl projekt připravený k implementaci firmware panelu.

Po těchto krocích jsem začal implementovat nižší vrstvy firmwaru. Rozhodl jsem se postupovat následovně: Nejdřív vytvořit modul Wiegand ovladače a nad ním potom postavit modul zastřešující funkce kolem čtečky a zámku. Tím bude možné podporovat více dveří. Aplikační vrstva potom bude používat rozhraní instance, která reprezentuje trojici dveře-zámek-čtečka.

3.2.2 Wiegand ovladač

Wiegand ovladač implementuje: Příjem dat po Wiegand rozhraní, ze kterých sestavuje jednotlivé rámce. Funkce pro dekódování těchto rámců na číslo karty, číslo zařízení a kontroly paritních bitů podle 26-bitového Wiegand formátu. Wiegand rozhraní jsem popsal v části 1.3.2.1 a Wiegand formát v části 1.3.2.3.

3.2.2.1 Příjem dat z Wiegand rozhraní

Wiegand transakce začíná sestupnou hranou. K detekci této události jsem nastavil na každém portu, kam jsou připojeny signály `data 0` a `data 1`, přerušení při sestupné hraně.

V obsluze přerušení zjistím z jakého pinu bylo vyvoláno a hodnotu tohoto pinu přečtu pro kontrolu, jestli je na něm logická nula – to znamená, že odpovídající signál je aktivní. V opačném případě událost ignoruji. Podle toho, jestli je zrovna aktivní signál `data 0` nebo `data 1` si uložím binární hodnotu 0 nebo 1. Hodnoty ukládám do paměti od vyšší adresy k nižší, protože je první posílán MSB a platforma je Little-endian. To se opakuje pro do té doby, než přijmu všech 26 bitů jednoho rámce.

Po nějaké době ladění se mi podařilo správně přijmou rámec od čtečky. V průběhu jsem narazil na jeden problém. Po několika úspěšných příjmů nebo restartu čtečky k docházelo nesprávnému přijímání. To jsem zjistil, že bylo způsobeno nedokončenou transakcí. Došlo k přijetí jen několika bitů a když přišel další rámec, tak se na navázalo tyto bity a došlo k jeho poškození. K tomu docházelo, protože jsem zapomněl implementovat expiraci transakce, pokud nepřijde žádný bit po uplynutí maximální periody mezi bity. To jsem

napravil přidáním FreeRTOS časovače, který spustím na začátku transakce a obnovuji při každém příchozím bitu. Když časovač vyprší, zahodím zatím přijatá data.

Celé přijímání tedy probíhá v obsluze přerušení. Všechny přijaté rámce se z něho ukládám s číslem zdrojového rozhraní do fronty s omezenou velikostí. Je na vyšší vrstvě, aby si rámec vyzvedla. Pokud je fronta plná, tak je zahazuji.

3.2.2.2 Dekódování Wiegand formátu

Data při přijímání ukládám do struktury, která odpovídá 26-bitovému Wiegand formátu. Jeden rámec je tedy reprezentován jednou instancí struktury. Pro dekódování jsem přesto do ovladače přidal zvlášť funkce pro získání čísla rámce a čísla zařízení ze surového rámce. V rámci přijímání nedělám kontrolu parity rámce, vyšší vrstvě jsem na to v rozhraní ovladače připravil funkci.

3.2.3 Ovladač čtečky karet

Ovladač čtečky karet jsem vytvořil za účelem sjednocení všech funkcí potřebných pro čtení karet, řízení červené a zelené LED a bzučáku, ovládání zámků a monitorování otevření dveří. Ovladač je postaven nad Wiegand ovladačem.

Nejdřív jsem vytvořil strukturu, která obsahuje přiřazení pinů a portů k jednotlivým signálům komponent. Potom strukturu s nastavením pro ovladač, konkrétně dobu trvání odemčení a svícení zelené LED. Obě tyto struktury se při inicializaci nastaví podle jednoho konfiguračního souboru.

3.2.3.1 Čtení karet

Čtení karet jsem implementoval funkcí, která vyzvedává přečtené rámce z fronty Wiegand ovladače. Parametrem funkce je doba čekání na kartu. Pokud je ve frontě dostupný Wiegand rámec provede se kontrola parity a vrátí se 24-bitové číslo, které obsahuje číslo karty v dolních bitech a číslo zařízení ve horních.

3.2.3.2 Funkce k odemykání

Funkci k odemykání zámku jsem spojil se světelnou a zvukovou signalizací. Při zavolání se aktivuje relé na dobu danou nastavením a podle parametrů bzučák a zelenou LED. Když rozsvítím zelenou LED, tak zhasnu červenou. Červená LED svítí vždy, když je zelená zhasnutá. Funkce neblokuje volajícího a vrací se okamžitě. Deaktivování totiž řeším dvěma FreeRTOS časovači, které spustím při aktivaci. První časovače zařídí zamknutí a druhý vypnutí zelené LED a bzučáku. V konfiguračním souboru jsem nastavil výchozí dobu odemčení na 5 s a dobu signalizace na 500 ms.

3.2.3.3 Monitorování dveří

Monitorování dveří jsem implementoval přerušením nastaveným na sestupnou a náběžnou hranu vstup, kde je připojen dveřní spínač. V obsluze přerušení přečtu logickou úroveň a podle ní aktualizuji stav dveří. Vyšší vrstva si tento poslední stav může vyčíst.

V konfiguračním souboru je možné pro každé dveře nastavit, jestli se jedná o spínač typu normálně rozepnutý nebo normálně sestupný (při otevřených dveří). To je totiž důležité k správnému vyhodnocení logické úrovně na vstupu spínače.

3.2.4 Lokální databáze

V raných fází vývoje nebyla hotová komunikace se serverem a proto jsem se rozhodl k implementaci dočasné lokální přístupové databáze. Tu jsem implementoval jako statické pole dat ve volné operační paměti. Tato databáze umožňuje uchovat až 128 čísel karet. Ke kartám je možné uložit jenom oprávnění k prvním, druhým nebo oběma dveřím.

Databázi má rozhraní pro získání hodnoty podle klíče, smazání klíče, přidání klíče a smazání celého obsahu.

Funguje obecně nad 4 B prvky, které se skládají z klíče a hodnoty. Prvky ukládám do čtyř stále seřazeným polí od velikosti 128 B. Hašovací funkce určuje, podle posledních tří bitů klíče, do kterého pole prvky patří. Klíče se hledají a vkládají půlením intervalů (binární vyhledávání).

3.2.5 Aplikační smyčka

Pro zprovoznění základních funkcí panelu jsem vytvořil jedno vlákno, ve kterém běží aplikační smyčka. V inicializace vytvořím instance ovladače čtečky pro každé dveře podle konfigurace. Poté ve smyčce čekám na číslo karty čtením z ovladače. Pokud číslo obdržím, kouknu se do lokální databáze a podle oprávnění zavolám funkci k odemčení dveří ze kterých pochází požadavek. Na konci smyčky také čtu stav dveří.

V této fázi jsem měl všechny části firmwaru pro fungování v samostatném režimu. Takže jsem do lokální databáze zapsal testovací data a vyzkoušel správnost chování firmwaru a vývojového zapojení. Vše fungovalo jak mělo, až na pár drobných chyb ve firmwaru, které jsem opravil.

3.3 Elektrické schéma řídícího panelu

V této fázi jsem začal s přípravou elektrického zapojení pro vytvoření prototypu řídícího panelu. Nejdříve jsem vývojové zapojení, kromě 12 V zdroje, zanesl do elektrického schéma. Při zakreslování jsem použil schéma vývojové desky LPCXpresso11C24 jako inspiraci.

Poté jsem schéma začal rozšiřovat o prvky pro napájení ze společného zdroje, paměť pro uložení konfigurace, ochranné prvky a konektory. Při zapojování jsem bral v potaz doporučení v datovém listu jednotlivých komponent a uživatelském manuálu (aplikační poznámky).

3.3.1 Chyba ve vývojovém zapojení

Při proměřování vývojového zapojení jsem zjistil, že řešení k vyrovnání napětí mezi 5 V a 3,3 V logikou na Wiegand signálech je chyba. Použil jsem totiž Schottkyho diodu místo Zenerovy. To způsobovalo zvýšení napětí na 3,3 V napájecí větvi už při připojení jenom jedné čtečky karet. Naštěstí nedošlo k poškození MCU, protože má velkou toleranci pro vstupní napětí.

Wiegand rozhraní má malou přenosovou rychlost, kde přenos jednoho bitu trvá 20 µs–100 µs. Proto jsem se rozhodl použít odporový dělič napětí místo zapojení s diodou, přestože může se vstupní kapacitou pinů a parazitní kapacitou prodlužovat dobu změny signálu.

Hodnoty odporu děliče jsem zvolil dostatečně velké pro zachování co nejmenších ztrát, tak aby doba náběžné a sestupné hrany nebyla moc prodloužená vzhledem době přenosu jednoho bitu. Nakonec jsem pro snížení napětí na 3,3 V zvolil hodnoty odporu 10 kΩ a 20 kΩ.

Doba náběžné a sestupné hrany t[s] se dá spočítat rovnicí 3.1 (převzato z [39]) s následujícími parametry:

- hodnota odporu v sérii se vstupem $R[\Omega]$,
- vstupní a parazitní kapacita C[F],
- počáteční napětí V_I[V],
- cílové napětí V_F[V],
- zdrojové napětí V_S[V].

$$t = -[0,66 \cdot R \cdot C \cdot \ln(\frac{V_F - 0,66 \cdot V_S}{V_I - 0,66 \cdot V_S})] \tag{3.1}$$

V mém případě vychází při $R = 10\,\text{k}\Omega$ $C = 50\,\text{pF}$ $V_S = 5\,\text{V}$ $V_F = 2{,}4\,\text{V}$ $V_I = 0{,}05\,\text{V}$ (2,4 V je rozhodná úroveň MCU pro log. 1), že vzestupná hrana bude trvat 0,423 µs. To je nejvýše 2 % doby přenosu bitu.

K vyzkoušení nového řešení a ověření zda jsem hodnoty odporů zvolil správně, jsem upravil vývojového zapojení. Čtení karet stále fungovalo spolehlivě beze změny a proto jsem tuto úpravu zanesl do schématu.

3.3.2 Komponenty pro fungování pro MCU

Při zapojování komponent potřebných pro fungování MCU LPC11C24 jsem se inspiroval elektrickým schématem vývojové desky [40]. Použil jsem stejné hodnoty vstupních keramických kondenzátorů na napájení, stejný 3,3 V lineární regulátor XC6227 a stejné zapojení krystalového oscilátoru.

3.3.3 Napájení ze společného zdroje

Jak jsem uvedl v návrhu systému (2.5), všechny panely budou napájeny ze společného zdroje. Maximální příkon každého panelu vyšel na 10 W při 60 V DC v nejhorším případě. Předpokládám, že půjde dosáhnout alespoň na 12 W. Vstupní napětí panelu by proto mělo být co nejblíže 60V. Toto napětí je potřeba efektivně převést na 12 V pro ostatní komponenty panelu a doplňková zařízení. Kromě 12V napájecí větve je ještě potřeba 5V větev pro lineární regulátor.

Jakou průzkum jsem v tabulce zapsal typický odběr komponent, které jsem naměřil ve vývojovém zapojení. Podle toho jsem alokovat dovolený příkon a zbytek z 12 W jsem alokoval na zámek. Hodnoty jsou přepočítány na příkon na napájecím vstupu panel, podle předpokládané nejhorší efektivity 12 V zdroje 82 % a 5 V zdroje 85 %.

Tabulka 3.3: Přibližná alokace odběru komponent

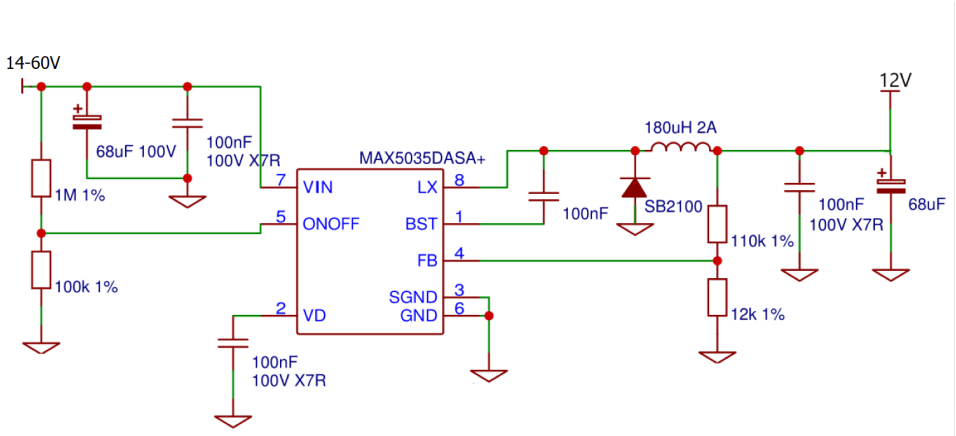
Komponenta	Dovolený příkon [W]
Jedno relé (včetně optočlenu)	0,7
MCU (včetně 3,3 V zdroje)	0,4
RFID čtečka	1,1
Zámek	4

Po průzkumu vhodných řešení, jsem se rozhodl, že navrhnu spínané zdroje za pomoci DC/DC měničů. Bude se jednat o 12 V 1 A hlavní zdroj a za ním připojený 5 V 500mA zdroj. Pro seznámení s návrhem spínaných zdrojů jsem si přečetl o základech v [41].

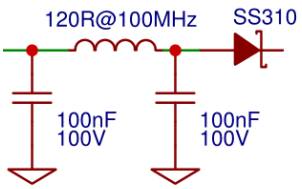
3.3.3.1 12 V zdroj panelu

Pro hlavní zdroj panelu ze společné sítě jsem vytvořil zapojení spínaného zdroje založením na DC/DC měniči MAX5035 od Maxim Integrated. Má spínací frekvenci 125 kHz a vstupní napětí až 76 V. Efektivita měnič je až 82,5 % při proudu 200 mA a vstupním napětím 60 V. Pouzdro měniče je 8-pinové 8PDIP nebo TSOT-23.

Vytvořené zapojení zdroje je na obrázku 3.7. Při tvorbě postupoval podle instrukcí, k výběru komponent a jejich hodnot v datovém listu měniče.



Obrázek 3.7: Zapojení 12 V zdroje (MAX5035)



Obrázek 3.8: Filtr vstupního napětí a ochrana proti otočené polaritě

Před zdroj jsem také předřadil filtr šumu a ochranu panelu proti zápornému napětí, použitím diody s malým úbytkem napětí jak je vidět na obrázku 3.8.

3.3.3.2 5 V zdroj panelu

Pro tento zdroj jsem vybral DC/DC měnič MP2315, který má spínací frekvenci 500 kHz. Jeho výhodou jsou malé rozměry potřebné cívky a kondenzátorů. Samotný obvod je také malý, má pouzdro 8-pinové TSOT-23. Efektivita měnič při napájení MCU, tzn. proud přibližně 80 mA, je až 89 %.

Pro sestavení zapojení zdroje pro panel jsem použil ukázkové zapojení a součástky v datovém listu. Na výstup zdroj jsem ještě přidal tlumivku ve formě feritové perly, která více vyhladí výstupní napětí.

3.3.4 Interní konektory

Interní konektory jsem použil pinové lišty. Na ty jsem vyvedl UART rozhraní – pro konfiguraci panelu a nahrávání firmwaru a SWD – ladící rozhraní. Pro přechod do programovacího režimu k nahraní firmwaru přes UART rozhraní jsem přidal ISP a reset propojky.

3.3.5 Paměť pro uložení konfigurace

Pro uložení konfigurace jsem přidal do schématu 1 Kb EEPROM paměť s I^2C rozhraním a připojil k MCU. Přidal jsem nastavitelnou propojku pro zakázání zápisu do EEPROM.

3.3.6 Rozhraní panelu

Přidal jsem šroubovací svorkovnice pro všechny vstupy a výstupy kromě společného napájení ze sítě a CAN rozhraní.

Na napájecí vstupy a výstupy jsem přidal vhodné transily na ochranu před napěťovými špičkami a proti přepětí (při přetětí se vyzkratuje). Pro Wiegand signály jsem vybral modul vhodný pro datové signály. Tyto součástky musí být umístěny co nejblíže konektoru.

Na oba napájecí vstupy jsem dal 1 A tavnou pojistku. Na napájecí výstupy pro každou RFID čtečku a zámek jsem dal vratnou pojistku. Pro čtečky má udržet 100 mA a pro zámky 250 mA. Hodnoty jsem zvolil na základě maximálního příkonu, který jsem rozděl pro jednotlivé komponenty v tabulce 3.3.

CAN rozhraní z MCU jsem vyvedl na RJ-45 konektory. Přidal jsem volitelné zakončovací odpory 100 Ω a 120 Ω, aby bylo možné sběrnici zakončit přímo na panelu. Na CAN signály jsem dal feritové perly pro odfiltrování vysokofrekvenčního rušení a diody určené na ochranu před elektrostatický výbojem přímo pro CAN sběrnici, tyto součástky musí být umístěny co nejblíže konektoru.

3.3.7 Informační LED panelu

Pro indikaci fungujícího napájení na řídícím panelu jsem přidal červenou LED na výstup 5 V zdroje.

Také jsem připojil k pinu MCU zelenou LED. Ta bude sloužit k indikaci stavu z firmwaru.

3.4 Prototyp řídícího panelu

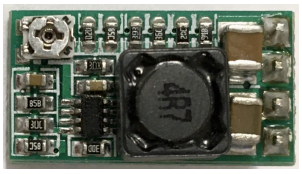
Prototyp jsem vyrobil na univerzální DPS, tak aby co nejpřesněji reprezentoval elektrické schéma panelu, které do té doby vzniklo. Liší se ale v těchto věcech:

- Některé součástky jsou nahrazeny nejbližšími podobnými co jsem měl v tu dobu dostupné.
- Nemá vyhrazené napájecí výstupy pro zámky.
- Nemá proudové pojistky.
- Má zvlášť konektor jen pro CAN rozhraní.

3.4.1 Moduly prototypu

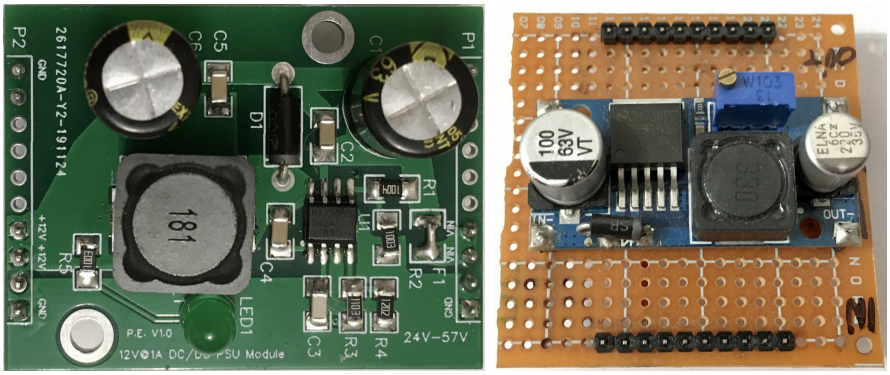
Prototyp je částečně složen z modulů: DC/DC měniče, relé a MCU jsou samostatné DPS zasazené do patice. To jsem udělal hlavně z toho důvodu, že tyto moduly vyžadují SMD součástky, které nelze na univerzální DPS připájet. Také proto, abych je mohl lehce nahradit.

Zdroj 5 V větve, DC/DC měnič MP2315, jsem zakoupil jako hotový modul a napájel na prototyp. Tento modul je vidět na obrázku 3.9. Pro relé jsem použil modul dvou relé z vývojového zapojení. Modul s MCU je vývojová deska LPCXpresso11C24.



Obrázek 3.9: Modul zdroje 5 V pro prototyp (MP2315)

Hlavní zdroj 12 V větve, založený na DC/DC měniči MAX5035, jsem navrhl a nechal vyrobit, protože nebyla dostupná hotová deska. Elektrické schéma je na obrázku B.3. Schéma zdroje neobsahuje vstupní filtr, diody a transily, protože jsou na desce prototypu. Vyrobená a osazená DPS je na obrázku 3.10, kde je zároveň vidět DC/DC měnič LM2596HV, který jsem krátce používal, než se vyrobila DPS zdroje.



Obrázek 3.10: Moduly zdrojů 12 V pro prototyp

3.4.2 Finální prototyp panelu

Blokové schéma prototypu je na obrázku B.4 a samotný prototyp na obrázku B.5 a B.6.

3.5 První verze schéma řídícího panelu

První verze elektrického schéma vychází z verze použité pro vytvoření prototypu.

3.5.1 Změny v první verzi

Ve schéma jsem nahradil DC/DC měnič MAX5035 za LMR16010 od Texas Instruments, abych ušetřil místo na DPS. K tomuto obvodu totiž stačí fyzicky menší cívky a kondenzátory, protože má vyšší spínací frekvenci než MAX5035. Tu jsem nastavil na 500 kHz jako kompromis mezi efektivitou a velikostí potřebných součástek. Tímto obvodem jsem zároveň nahradil DC/DC měnič MP2315 pro 5 V větev, abych zmenšil počet různých součástek. Efektivitu má velmi podobnou jako nahrazené měniče. Při zapojování jsem postupoval podle doporučení v datovém listu pro LMR16010.

Hlavní DC/DC měnič jsem nastavil, aby napájel řídící panel, když je vstupní napětí alespoň 35 V. Abych zabránil přetížení společné napájecí sítě, když nastane stav, kdy je v síti malé napětí. Nejvyšší vstupní napětí jsem omezil na 58 V, kvůli použití transilu na vstupu s hranicí 58 V a zároveň abych nechal dostatečnou rezervu od nejvyššího vstupního napětí měniče.

Místo feritových perel na CAN signálech, jsem dal tlumivku určenou pro lepší odrušení rozdílových signálů jako je CAN.

3.5.2 Varianty řídících panelů

Kromě hlavního el. schéma zapojení s rozhraním pro dvoje dveře, kterou označuji varianta 2D. Jsem ještě připravil el. schéma s podporou jen pro jedny dveře – varianta 1D. Ta neobsahuje vstup pro napájení z 12 V, jelikož to má být odlehčená varianta. Jinak je to podmnožina varianty 2D.

Každá z variant má své určení. První je vhodná pro hustě rozmístěné dveře, druhá pro řídce. Je to pro minimalizovaní počtu potřebných panelů na vedení a celkového příkonu. Schéma zapojení první verze přístupového zařízení varianty 2D je k nalezení v příloze C. Jinak je na přiloženém médiu, kde je také varianta 1D.

3.6 Aplikační CAN protokol

Aplikační protokol pro komunikaci řídící panelů s řídícím serverem jsem vytvořil tak, aby bylo možné mít v jedné síti mnoho řídících panelů, aby šlo každé zprávě nastavit prioritu a aby byl snadno rozšiřitelný.

Vlastnosti protokolu:

- Bezstavová komunikace (klient si nemusí nic pamatovat).
- Používá pouze 29-bitový rozšířený formát CAN rámce.

- Rychlost CAN sběrnice by ve všech zařízeních měla být 125 kbit s^{-1}.
- Až 64 různých zpráv.
- Adresace až 1019 dveří.
- Nastavitelná priorita zpráv.

3.6.1 Formát zpráv

Zpráva má hlavičku a datovou část. Hlavička zprávy je 29-bitová arbitrační část CAN rámce. Formát hlavičky je vidět v tabulce 3.4.

Tabulka 3.4: Formát hlavičky zprávy CAN protokolu

číslo bitu	28–26	25–20	19-10	9–0
název pole	Priorita	Kód funkce	Cílová adresa	Zdrojová adresa
velikost pole	3 bit	6 bit	10 bit	10 bit

3.6.2 Priorita zprávy

Priorita dává přednost zprávě před ostatními při získávání sběrnice. Rozsah priorit zprávy je 0–7. Nejvyšší priorita je 0.

3.6.3 Kód funkce

Kód funkce určuje význam zprávy a případně udává formát datové části, pokud je přítomna. V tabulce 3.5 je přehled alokovaných funkčních kódů včetně formátu datové části a jejich priority.

Tabulka 3.5: Přehled zpráv v protokolu

kód funkce	označení	délka dat [B]	odesílatel
0	RESERVED	-	-
1	USER_AUTH_REQ	4	panel
2	USER_AUTH_FAIL	4	server
3	USER_AUTH_OK	4	server
4	DOOR_CTRL	1	server
5	DOOR_STATUS	1	panel
6	MASTER_ALIVE	0	server

Význam funkčních kódů:

- RESERVED – vyhrazené nesmí se používat.

- USER_AUTH_REQ – žádost panelu o autorizaci uživatele k dveřím identifikovaným adresou odesílatele. V datové části se posílá číslo uživatele (první bajt je LSB).

- USER_AUTH_FAIL – odpověď serveru na USER_AUTH_REQ, že uživatel není autorizován k požadovaným dveřím. V datové části se posílá číslo uživatele, která není autorizován (první bajt je LSB).

- USER_AUTH_OK – odpověď serveru na USER_AUTH_REQ, že uživatel je autorizován k požadovaným dveřím. V datové části se posílá číslo uživatele, která je autorizován (první bajt je LSB).

- DOOR_CTRL – příkaz serveru pro panel. V datové části posílá jeden bajt s číslem příkazu.

  Příkaz k odemčení má číslo 1.

  Příkaz k vymazání lokální databáze má číslo 2.

- DOOR_STATUS – informace o stavu dveří posílaná panelem na server při změně stavu. Minimální perioda je 2 s. V datové části se posílá jeden bajt s číslem stavu.

  Stav zavření má číslo 1.

  Stav otevření má číslo 2.

- MASTER_ALIVE – zpráva posílaná serverem s periodou 5 s. Slouží k detekci výpadku komunikace se serverem.

3.6.4 Adresní prostor

Zdrojová adresa je adresa odesílatele zprávy. Cílová adresa je adresa příjemce.

Adresní prostor má 1024 unikátních adres. Adresa 0 je vyhrazená a nesmí se používat. Řídící servery mají alokovány adresy 1–3. Řídící panely mají alokovány adresy 4–1022.

Adresa 1023 je vyhrazena pro broadcast. To umožňuje zařízením snadné nastavení filtru přijímaných zpráv. Broadcast adresa se smí používat jen v poli cílové adresy.

3.7 Dokončení firmwaru řídícího panelu

Po vytvoření aplikačního protokolu jsem ho implementoval do firmwaru řídícího panelu. Potom jsem dokončil hlavní aplikační smyčku přidal čtení a zápis konfigurace z EEPROM.

3.7.1 Implementace komunikace po CAN sběrnici

Pro příjem a posílání CAN zpráv jsem použil ovladač CAN řadiče v ROM od připravený od výrobce.

Příjem a zpracování zpráv řeším v obsluze přerušení od CAN ovladače. Implementoval jsem zpracování všechny funkčních kódů posílaných serverem. Na obrázku je vidět vývojový digram obsluhy přerušení.

Při žádosti o přístup CAN zprávu hned sestavím a odešlu na sběrnici. Server je považován za nedostupný, když po uplynutí dvou period nepřijde notifikace, to potom přestanu odesílat veškeré další žádosti a zhasnu indikační zelenou LED.

Chybové zprávy sběrnice ignoruji, protože je nemůžu nijak řešit. Řeším ale fatální chybu „BUS OFF“. Ta nastane jen v případech poškození sběrnice, když počet chyb přesáhne určitou mez a CAN řadič se odpojí od sběrnice. Ten je řešitelný jenom resetem této periferie. Proto řídící panel po uplynutí dvou sekund restartuji a pokusím se obnovit komunikaci.

3.7.1.1 Adresa řídícího panelu v systému

Adresa panelu v CAN protokolu je uložena v externí EEPROM. Pro nastavení této adresy jsem implementoval možnost při startu panelu tuto adresu zapsat přes UART (Více v dokumentaci). Samozřejmě je možné tuto adresu do EEPROM zapsat přes externí programátor. Když adresa v EEPROM není nebo je neplatná řídící panel nenastartuje.

Každý panel má dvě adresy pro identifikaci umístění obou dveří. Obě adresy musí být přiřazeny hned vedle sebe, jedna je sudá a druhá je lichá. V panelu je totiž uložena jen jedna, ta sudá a tu lichou si firmware odvodí.

3.7.2 Aplikační smyčka

Aplikační smyčka běží v jednom vlákně a má na starosti zpracování a posílání žádostí o přístup a posílání aktualizací stavu dveří.

Hned na začátku se pokusím z fronty přijatých karet vyzvednou jedno číslo karty. Čekám 500 ms než přijde karta, pokud je fronta prázdná. Když získám číslo karty, sestavím zprávu se žádostí a odešlu ji serveru.

Potom čekám uplyne jedna sekunda od začátku smyčky. V závislosti na tom, jak dlouho trvalo vyzvednutí a poslání karty. Čekání je tu proto, abych omezil počet žádostí na nejvýše jednu za sekundu a ochránil se proti brute-force útoku přes Wiegand rozhraní.

Na konci smyčky posílám stav dveří, když se změnil. Perioda minimálně jedna sekunda, díky čekání po zpracování požadavku.

3.7.3 Zotavení z poruch

K zotavení z poruch panelu při startu firmwaru nastavuji Watchdog časovač. Obnovu tohoto časovače dělám po inicializace jenom z hlavní aplikační smyčky. Zvolil jsem periodu 2 s, která je dvojnásobkem maximální doby vykonávání hlavní smyčky.

Dále při startu nastavuji detekci podpětí, tak aby došlo k resetu MCU při poklesu napětí pod 2,63 V.

3.8 První verze firmwaru řídícího panelu

První verze implementuje všechny požadavky a komunikuje s řídícím serverem. Shrnutí schopností firmwaru:

- Řízení přístupu na až dvou dveřích s možností monitorování stavu dveří.
- Komunikace přes CAN sběrnici se serverem, který rozhoduje o všech žádostech o přístup.
- Možnost vzdáleného odemčení příkazem ze serveru.
- Nastavení umístění dveří přes UART rozhraní.
- Indikace stavu komunikace se serverem.
- Zotavení z poruch pomocí Watchdog časovače.
- Podpora Wiegand 26 protokolu.

Zapojení RFID čtečky, relé, senzoru k MCU a další funkční parametry jsou plně konfigurovatelné v jednom hlavičkovém souboru `terminal_config.h`. Hardwarové varianty 1D a 2D používají tedy stejný kód, kde 1D má vypnuté jedno rozhraní pro dveře. V hlavičkovém souboru je možné nakonfigurovat:

- Rychlost CAN sběrnice pro komunikaci se serverem.
- Rychlost I2C sběrnice a adresu pro EEPROM.
- Zapnout nebo vypnout zvukovou signalizaci.
- Typ dveřního senzoru (NO/NC).
- Dobu odemčení a signalizace pro každé dveře zvlášť.
- Vypnutí jednoho rozhraní pro dveře.
- Přiřazení pinů na které jsou připojeny hardwarové komponenty.

Firmware jsem naprogramoval stylem event-driven pro zajištění rychlé odezvy. To znamená, že vykonávání programu přímo závisí na příchodu zprávy od řídící serveru nebo zprávy od čtečky karet, ve zbylém čase se odesílá periodicky stav dveří a obnovuje Watchdog.

K implementaci jsem použil pouze programovací jazyk C a C knihovnu newlib-nano. Nepotřeboval jsem totiž použít žádné konstrukce z jazyka C++. Využití paměti MCU je v tabulce 3.6. Hodnoty jsou získané s použitím optimalizace překladu (-O3) a GCC 7.3.1 v balíku GNU Tools for ARM verze 7-2018-q2-update.

Tabulka 3.6: Využití paměti MCU LPC11C24 (varianta 2D)

Pamět	využitá velikost [B]	velikost [KB]	využití [%]
Flash	16544	32	50,49
RAM	4624	8	56,45

K první verzi jsem dodělat projekt pro systém CMake. Ten umožní komukoliv správně zkompilovat produkční firmware z příkazové řádky bez použití IDE MCUXpresso. Kompilaci přes CMake jsem zprovoznil jen na OS Windows, k tomuto účelu je u zdrojových kódu přiložen jeden spustitelný skript s krátkým návodem. Na OS GNU/Linux by měl jít firmware zkompilovat přes distribuci IDE MCUXpresso pro Linux (nezkoušel jsem).

Binární forma této verze firmwaru pro obě varianty hardwaru, zdrojové kódy s komentáři a projektové soubory jsou na přiloženém médiu. Návod k oživení hardwaru je v příloze A.

3.9 Software řídícího serveru

Software řídícího serveru jsem implementoval jako službu pro plaformu GNU/Linux. Hardwarem může být jakýkoliv počítač s CAN rozhraním s ovladači pro SocketCAN. Software jsem napsal v jazyce Python verze 3.5. Návod na instalaci je v příloze. Zdrojový kód je na přiloženém médiu.

Používám Redis databázi k uložení dat k rozhodování o přístupu, tak jak jsem si určil v architektuře. Pro připojení k Redis serveru používám oficiálního klienta pro Python.

Redis databázový server může být umístěn na stejném nebo jiném počítači. Při druhé možnosti musí být počítač s Redis serverem na stejné a důvěrné TCP/IP síti jako řídící server.

Komunikační protokol pro přístup administrativní aplikace k řídícímu serveru jsem neimplementoval. Ovládání systému administrativní aplikací je možné jenom skrze manipulace s obsahem Redis databáze. Aplikace by se k Redis serveru měla připojovat přes zabezpečený kanál (například SSH tunel), jelikož Redis server nepodporuje šifrování komunikace.

Hlavní aplikační smyčka serveru používá systémové volání „select“ pro čekání na zprávy od panelů. Při přijmutí okamžitě zprávu dekóduje. Pokud je to validní zpráva s žádostí o autorizaci, tak se podívá do databáze, jestli je uživatel oprávněn a hned odpoví. Změny stavu dveří server zapisuje do souboru a aktuální hodnoty uchovává v databázi. Zpracování je napsáno, tak aby při jakékoliv chybě pokusil server zotavit. Všechny chyby jsou zaznamenány.

3.9.1 Datový model databáze

Redis server jsem nakonfiguroval pro použití tří databází. První je pro vazby uživatel a skupina. V druhé jsou skupiny s přiřazenými adresami dveří. Třetí je seznam adres dveří, jejich stavů a režimů. Všechny Redis databáze obsahují pouze záznamy klíč-hodnota v binárním formátu.

3.9.1.1 Databáze uživatelů

Databáze uživatelů obsahuje číslo karty uživatele jako klíč a název skupiny jako hodnotu. Příklad Redis příkazu pro přidání záznamu do této databáze je `SET 7523010 Warehouse`.

3.9.1.2 Databáze skupin

Databáze skupin obsahuje název skupiny jako klíč a množinu adres dveří jako hodnotu. Vždy by měla existovat skupina `__all` a `__empty`. Příklad Redis příkazu pro přidání záznamu do této databáze je `SADD Warehouse 10 15 18`.

3.9.1.3 Databáze dveří

Databáze dveří obsahuje adresu dveří jako klíč a pole jako hodnotu. V poli jsou dvě položky. První je režim dveří a druhá je stav dveří. Režim dveří může nabývat hodnot `off`, `on` a `learn`. Stav dveří může nabývat hodnot `open` a `closed`. Příklad Redis příkazu pro přidání záznamu do této databáze je `RPUSH 10 on closed`.

3.9.2 Adaptér pro nadřízenou jednotku

K připojení serveru k síti panelů jsem vyrobil pasivní adaptér z RJ-45 odděluje napájení a CAN sběrnici. Také jsem vyrobil jsem aktivní adaptér s CAN řadičem a budičem pro připojení k Raspberry Pi přes SPI rozhraní. Pasivní adaptér je v příloze D, aktivní v příloze E.

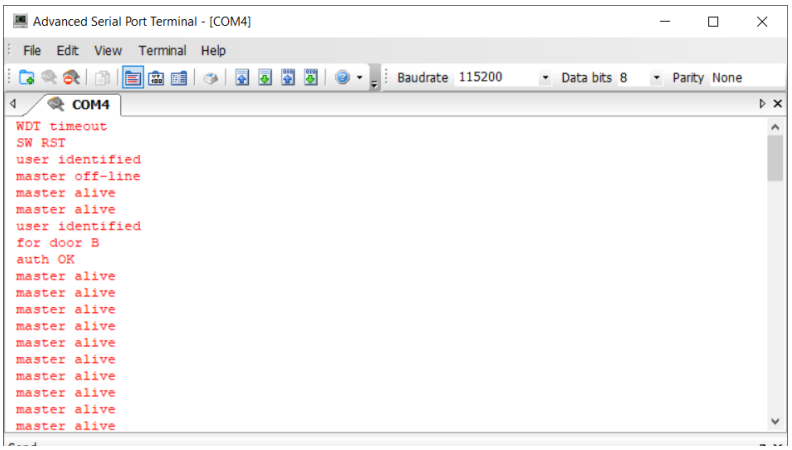
Kapitola 4

Ověření funkčnosti systému

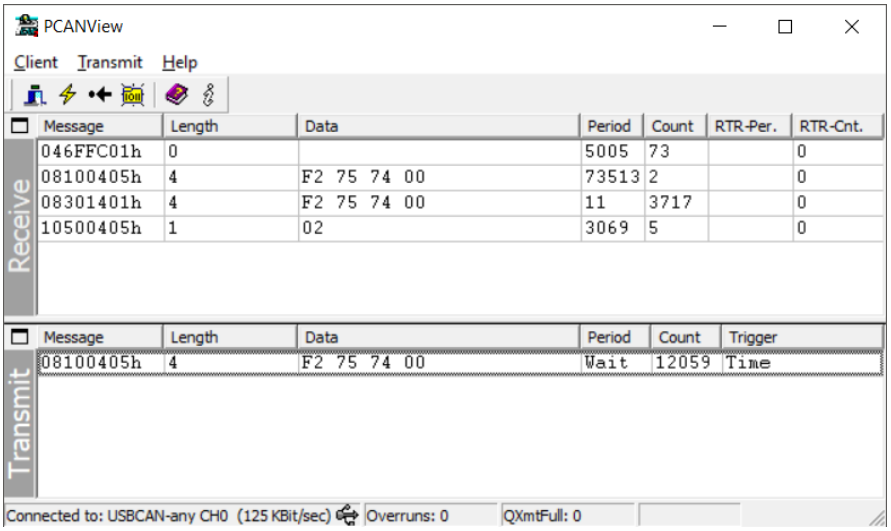
V průběhu vývoje jsem testoval systém s ladícím výpisem na sériovou linku a ladícími výpisy na straně serveru (obr. 4.1). Do systému jsem měl také připojený počítač s nástrojem pro zachytávání CAN komunikace pro kontrolu implementace všech funkcí protokolu na straně serveru a řídícího panelu (obr. 4.2).

4.1 Testování po dokončení systému

Po dokončení vývoje jsem otestoval zapojení na obrázku 4.3 s napájecím zdrojem 54 V 150 W přes kabel CAT-5e o délce 10 m. Vše fungovalo podle očekávání. Testoval jsem čtení karet, dveřní senzory, nastavování adresy a obnovení ztraceného připojení k serveru.



Obrázek 4.1: Ladící výpis z řídícího panelu



Obrázek 4.2: Nástroj PCANView

4.2 Zátěžový test řídícího serveru

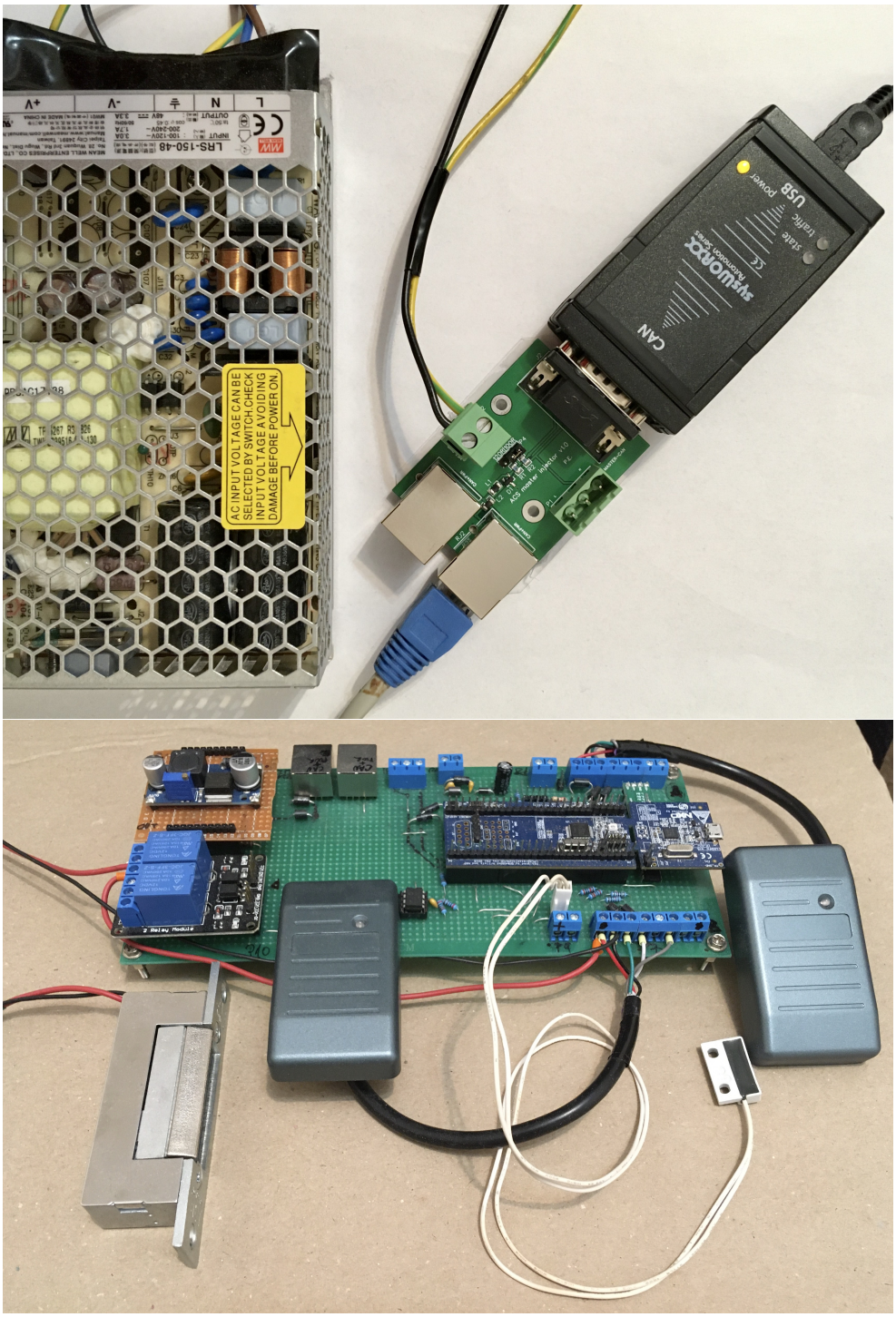
Na testovacím zapojení4.4 jsem provedl syntetický zátěžový test softwaru serveru na Raspberry Pi 1. K emulaci požadavků v protokolu jsem použil počítač s CAN-USB adaptérem. Dosáhl jsem propustnosti až 200 vyřízených požadavků za sekundu. Zároveň jsem objevil problém s špatně nastavenými filtry CAN zpráv, což jsem opravil v softwaru serveru i firmwaru panelu.

4.3 Příkon prototypu řídícího panelu

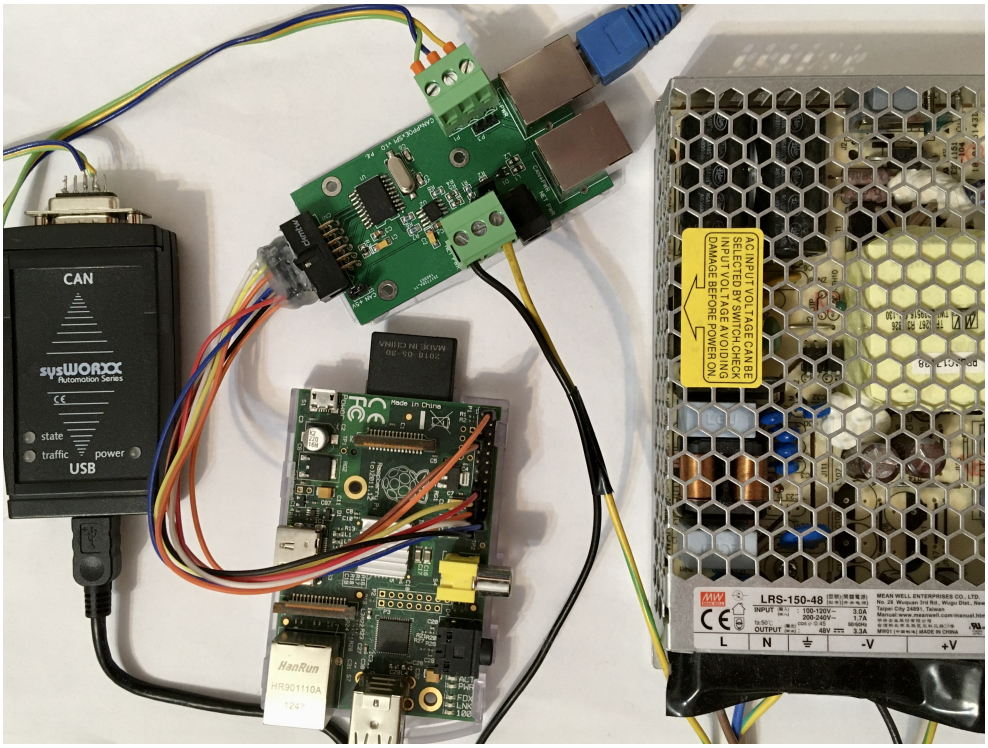
Na finálním prototypu jsem změřil skutečný příkon při různých stavech aktivity. Měření jsem provedl na zapojení prototypu přes 3 m CAT-5e kabel a přes adaptér na výstup společného zdroje. Do prototypu byl zapojen jeden dveřní otvírač s odběrem 250 mA a dvě RFID čtečky, každá s odběrem 40 mA–50 mA. Výstupní napětí zdroje bylo 54,9 V a proud jsem měřil na výstupu zdroje. Naměřené hodnoty jsou v tabulce 4.1 (Poslední stav je dopočítaný z ostatních měření). Výsledky jsou lepší než jsem očekával.

Tabulka 4.1: Příkon prototypu panelu v různým stavech aktivity

Stav panelu	Příkon na výstupu zdroje [W]
Klidový	0,78
Dvě čtečky aktivní	1,1
Dvě čtečky a jeden zámek aktivní	4,67
Dvě čtečky a dva zámky aktivní	8,24



Obrázek 4.3: Testovací zapojení prototypu, zdroje a serveru (PC)



Obrázek 4.4: Testovací zapojení zdroje a serveru (Raspberry Pi)

Závěr

V této závěrečné práci jsem na základě analýzy existujících systémů řízení fyzického přístupu osob úspěšně navrhl a vytvořil vlastní systém s použitím RFID karet podle stanovených požadavků.

Systém jsem navrhl pro ulehčení a zlevnění instalace tak, aby řídící panely byly propojeny běžně dostupným síťovým kabelem typu CAT-5e a napájeny z jednoho společného zdroje.

Vytvořil jsem prototyp řídícího panelu na univerzální desce plošných spojů a na základě něho elektrické zapojení pro první verzi. Řídící panel má konektory pro dvě RFID čtečky s rozhraním Wiegnad 26, dva zámky a vstupy na monitorování dveří. Lze z něho přímo napájet elektronické zámky a RFID čtečky.

Aplikační CAN protokol pro komunikaci řídícího serveru a řídících panelů jsem vytvořil a použil. Také jsem implementoval software pro řídící server na platformě GNU/Linux. Ten je možné nainstalovat na běžný počítač třídy PC. Vytvořil jsem adaptér pro připojení počítače do sítě panelů.

Výsledný systém podporuje řízení až dvaceti dvou dveří v jedné síti za použití jedenácti řídících panelů z jednoho napájecího zdroje a jednoho řídícího serveru. Jeden řídící server může obsluhovat jednu i více sítí. Celkem jeden systém může obsahovat až 500 přístupových zařízení.

Všechny implementované funkce systému jsem otestoval, ale jen na malém zapojení s jedním prototypem řídícího panelu a řídícím serverem na počítači. Na vytvoření více desek plošných spojů podle první verzi elektrického zapojení se stále pracuje.

Výhledově by bylo vhodné dodělat do přístupového systém administrativního klienta. Tím se stane systém kompletním a bude ho možné reálně nasadit.

Literatura

[1] Farpointe Data: *Pyramid series – Wiegand Data Format* [disk]. 2004. Dostupné z: text/references/Pyramid-Wiegand-Data-Format.pdf

[2] Assa Abloy: *ABLOY EL560, EL561, EL562, EL563* [online]. 2017, [cit. 9.1.2020]. Dostupné z: "https://www.abloy.com/Abloy/Abloy.com%20OW2/Products/Electric%20locks%20datasheets/EL560_EL561_EL562_EL563_DATASHEET.pdf"

[3] *FAB Profi* [online]. Assa Abloy, [cit. 9.1.2020]. Dostupné z: https://www.assaabloy.cz/cs/local/cz/produkty/elektromechanicke-produkty/elektricke-otvirace/elektricke-otvirace-fab/fab-profi/

[4] Macháček, M.: *CAN FD – nová verze CAN protokolu* [online]. MACH SYSTEMS s.r.o., 2015, [cit. 15.12.2019]. Dostupné z: https://www.machsystems.cz/novinky/2015/can-fd-nova-verze-can-protokolu

[5] *PCAN-USB.* CAN interface for USB [online]. PEAK-System Technik GmbH, 2019, [cit. 22.11.2019]. Dostupné z: https://www.peak-system.com/PCAN-USB.199.0.html

[6] Norman, T.: *Electronic Access Control.* Boston: Butterworth-Heinemann, první vydání, 2012, ISBN 978-0-12-382028-0, 448 s.

[7] Kisi: *Access Control and Physical Security Guide* [online]. 2019, [cit. 27.11.2019]. Dostupné z: https://pages.getkisi.com/access-control-systems-introduction

[8] *Understanding the confusing world of RFID tags and readers in access control* [online]. Nedap N.V., 2019, [cit. 8.12.2019]. Dostupné z: https://www.nedapidentification.com/insights/understanding-the-confusing-world-of-rfid-tags-and-readers-in-access-control

[9] *What is RFID?* [online]. Nedap N.V., 2019, [cit. 8.12.2019]. Dostupné z: https://www.nedapidentification.com/insights/what-is-rfid

[10] *Comparing RFID and NFC Access Control Systems* [online]. Kisi, 2019, [cit. 8.12.2019]. Dostupné z: https://www.getkisi.com/guides/rfid-access-control

[11] Wehr, J.: *IS THE DEBATE STILL RELEVANT?* An in-depth look at ISO 14443 and its competing interface types [online]. AVISIAN Publishing, 2003, [cit. 8.12.2019]. Dostupné z: https://www.secureidnews.com/news-item/is-the-debate-still-relevant-an-in-depth-look-at-iso-14443-and-its-competing-interface-types

[12] HID Corporation: *Understanding Card Data Formats* [online]. 2006, [cit. 15.10.2019]. Dostupné z: https://www.hidglobal.com/sites/default/files/hid-understanding_card_data_formats-wp-en.pdf

[13] Texas Instruments: *NFC card emulation using the TRF7970A* [online]. 2019, [cit. 8.12.2019]. Dostupné z: http://www.ti.com/lit/an/sloa208b/sloa208b.pdf

[14] SIA AC-01-1996.10. *Access Control Standard Protocol for the 26-bit Wiegand TM Reader Interface*. Maryland, USA: Security Industry Association (SIA), 1996.

[15] *Open Access 4.0* [online]. ACCX Products, 2015, [cit. 10.12.2019]. Dostupné z: http://www.accxproducts.com/wiki/index.php?title=Open_Access_4.0

[16] Robert Bosch GmbH: *CAN Specification* [online]. version 2.0, Sep. 1991, [cit. 20.5.2019]. Dostupné z: http://esd.cs.ucr.edu/webres/can20.pdf

[17] Robert Bosch GmbH: *CAN with Flexible Data-Rate Specification* [online]. version 1.0, Apr. 2012, [cit. 10.12.2019]. Dostupné z: https://can-newsletter.org/uploads/media/raw/e5740b7b5781b8960f55efcc2b93edf8.pdf

[18] ISO 11898-1. *Road vehicles – Controller area network (CAN)* – Part 1: Data link layer and physical signalling. ISO 11898-1:2015(E). Second Edition. Geneva: International Organization for Standardization, 2015.

[19] ISO 11898-2. *Road vehicles – Controller area network (CAN)* – Part 2: High-speed medium access unit. ISO 11898-2:2016. Second Edition. Geneva: International Organization for Standardization, 2016.

[20] NXP Semiconductors: *LPC11Cx2/Cx4.* Product data sheet [online]. Rev. 3.2, 2016, [cit. 9.12.2019]. Dostupné z: https://www.nxp.com/docs/en/data-sheet/LPC11CX2_CX4.pdf

[21] Robert Bosch GmbH: *C_CAN.* User's Manual. [online]. Rev. 1.2, 2000, [cit. 9.12.2019]. Dostupné z: http://www.keil.com/dd/docs/datashts/silabs/boschcan_ug.pdf

[22] NXP Semiconductors: *TJF1051.* High-speed CAN transceiver. [online]. Rev. 5, 2016, [cit. 9.12.2019]. Dostupné z: https://www.nxp.com/docs/en/data-sheet/TJF1051.pdf

[23] *LPC11C24FBD48* [online]. NXP Semiconductors, 2019, [cit. 9.12.2019]. Dostupné z: https://www.nxp.com/part/LPC11C24FBD48

[24] *The FreeRTOS^{TM} Kernel* [online]. Amazon Web Services, Inc. [cit. 28.12.2019]. Dostupné z: https://www.freertos.org/RTOS.html

[25] Hartkopp, O.; et. al.: *Controller Area Network Protocol Family (aka SocketCAN)* [online dokumentace]. Linux Kernel Organization, Inc., [cit. 15.4.2019]. Dostupné z: https://www.kernel.org/doc/Documentation/networking/can.txt

[26] *CAN Bus.* SocketCAN Supported Controllers [online]. ELinux, 2019, [cit. 24.11.2019]. Dostupné z: https://elinux.org/CAN_Bus

[27] *DB-Engines Ranking of Key-value Stores* [online]. Solid IT gmbh, 2019, [cit. 24.11.2019]. Dostupné z: https://db-engines.com/en/ranking/key-value+store

[28] *Redis.* Documentation [online]. Redis Labs, 2019, [cit. 9.12.2019]. Dostupné z: https://redis.io/documentation

[29] *Redis.* Benchmark results on different virtualized and bare-metal servers [online]. Redis Labs, 2019, [cit. 9.12.2019]. Dostupné z: https://redis.io/topics/benchmarks

[30] Microchip Technology: *MCP2515. Stand-Alone CAN Controller with SPI Interface* [online]. 2019, [cit. 10.12.2019]. Dostupné z: http://ww1.microchip.com/downloads/en/DeviceDoc/MCP2515-Stand-Alone-CAN-Controller-with-SPI-20001801J.pdf

[31] Philips Semiconductors: *SJA1000. Stand-Alone CAN Controller* [online]. 2000, [cit. 10.12.2019]. Dostupné z: https://www.nxp.com/docs/en/data-sheet/SJA1000.pdf

[32] ANSI/TIA-568-C.2. *Balanced Twisted-Pair Commercial Building Telecommunications Cabling Standard.* ANSI/TIA-568-C.2-2009. Arlington, Virginia: Telecommunications Industry Association (TIA), 2009.

[33] ISO/IEC 11801-1. *Information technology – Generic cabling for customer premises* – Part 1: General requirements. ISO/IEC 11801-1:2017(E). Geneva: International Organization for Standardization, 2017.

[34] IEEE Standard for Ethernet Amendment 2: Physical Layer and Management Parameters for Power over Ethernet over 4 pairs. *IEEE Std 802.3bt-2018 (Amendment to IEEE Std 802.3-2018 as amended by IEEE Std 802.3cb-2018)*, Jan 2019: s. 1–291, doi:10.1109/IEEESTD.2019.8632920.

[35] Philips Semiconductors: *PCA82C250/251 CAN Transceiver. AN96116* [online]. 1996, [cit. 28.12.2019]. Dostupné z: `https://www.nxp.com/docs/en/application-note/AN96116.pdf`

[36] ČSN EN 60950-1 ed. 2. *Zařízení informační technologie – Bezpečnost* – Část 1: Všeobecné požadavky. Praha: Úřad pro technickou normalizaci, metrologii a státní zkušebnictví, 2016.

[37] Belopolsky, Y.: *Durability of Connecting Hardware under Electrical Load for Power-over-Ethernet Applications* [online]. 2007, [cit. 7.1.2019]. Dostupné z: `http://grouper.ieee.org/groups/802/3/at/public/2007/09/belopolsky_1_0907.pdf`

[38] IEC 60512-99-002. *Connectors for electrical and electronic equipment - Tests and measurements - Part 99-002: Endurance test schedules - Test 99b: Test schedule for unmating under electrical load.* International Electrotechnical Commission, Edition 1.0. 2019.

[39] Microchip Technology Inc.: *3V Tips 'n Tricks* [online]. 2008, [cit. 10.1.2019]. Dostupné z: `http://ww1.microchip.com/downloads/en/DeviceDoc/chapter%208.pdf`

[40] NXP Semiconductors: *LPCXpresso11C24 with CMSIS-DAP – Schematic* [online]. Rev. A, 2016, [cit. 9.12.2019]. Dostupné z: `https://www.nxp.com/downloads/en/schematics/LPCXpresso11C24-with-CMSIS-DAP-SCH.pdf`

[41] Texas Instruments Inc.: SNVA559C. *Switching regulator fundamentals.* [online]. February 2019, [cit. 10.1.2019]. Dostupné z: `http://www.ti.com/lit/an/snva559c/snva559c.pdf`

[42] ANSI/TIA-568-C.0. *Generic Telecommunications Cabling for Customer Premises.* ANSI/TIA-568-C.0-2009. Arlington, Virginia: Telecommunications Industry Association (TIA), 2009.

[43] CiA 303-1. *CANopen Recommendation* Part 1: Cabling and connector pin assignment [online]. ver. 1.9.0. Nuremberg: CAN in Automation e. V., 2017. Dostupné z: `https://www.can-cia.org/groups/specifications`

Příloha A

Dokumentace systému

Následující dokumentace popisuje první verzi systému.

Tento základní fyzický přístupový systém je určen pro středně velké instalace, které nevyžadují zvýšené zabezpečení ani pokročilejší funkce. Systém je založen na použití RFID identifikátorů. Slouží primárně k řízení přístupu osob podle oprávnění založeném na lokaci dveří. Jeho výhodnou je snadná vestavba díky napájení většiny komponent ze společného zdroje a jejich propojení jedním kabelem, a nízké pořizovací náklady řídících panelů.

Systém má dva druhy řídících komponent: řídící panel pro ovládání zámku a připojení čtečky karet, a k nim nadřazený řídící server s databází, který rozhoduje o přístupech. Topologie zapojení systému je znázorněna na obrázku A.1.

Řídící panely podporují čtečky karet komunikující s panelem Wiegand 26 protokolem. Používají se karty naprogramované ve 26-bitovém Wiegand formátu. Panel má 12 V výstupy pro napájení zámku a čtečky karet. Dále umožňuje připojit dvě LED a bzučák na čtečce, a jeden monitorovací kontakt. Specifikace přístupového zařízení je v tabulce A.1, kde varianta 1D označuje panel s rozhraním pro jedny dveře a varianta 2D pro dvoje dveře.

Napájení řídících panelů a k nim připojeným zařízením je ve společné síti z jednoho zdroje. Napájení je přenášeno přes síťový kabel kat. 5e sloužící zároveň pro komunikaci s řídícím serverem. Přenos energie po kabelu není kompatibilní s aktivním ani pasivním PoE standardem.

Řídící software serveru rozhoduje o udělování všech přístupů podle skupin uživatelských oprávnění a umístění dveří. Jednotlivé přístupy systém zaznamenává. Stav dveří může být monitorován spínačem. Server má vlastní zdroj napětí.

Administrativní aplikace není součástí. Systém lze ale administrovat z demonstračního konzolového klienta připojením na databázi řídícího serveru. Ten má základní funkce jako přidávání a odebírání oprávnění uživatelů k jednotlivým kontrolovaným prostorů, a to podle čísla RFID karty vlastněné uživatelem.

Tabulka A.1: Specifikace první verze řídícího panelu

Indikátory	napájecí LED a stav komunikace LED
Komunikace	CAN sběrnice 125 kbit s^{-1}
Napájecí napětí	48 V DC (min. 38 V, max. 58 V)
Maximální příkon	9 W (jedny dveře), 15 W (dvoje dveře)
Příkon zamčeno	průměrně 1 W (var. 1D), 1,5 W (var. 2D)
Příkon odemčeno	průměrně 5 W (var. 1D), 9,5 W (var. 2D)
Počet zařízení v síti	nejméně až 20 (var. 1D), 11 (var. 2D)
Rozhraní pro doplňková zařízení	
RFID čtečka	12 V DC, trvale max. 75 mA každá, Wiegand 26
Elektronický zámek	12 V DC, trvale max. 250 mA každý
Kontakty pro zámek	NO/NC/COM, max. 250 V AC a 100 V DC
Monitorovací vstup	Vodiče + a − (3,3 V 1 mA)
Vedení pro vstupní napájení a komunikaci se serverem	
Délka vedení	nejméně až 100 m (celkem všechny úseky)
Typ kabelu	síťový CAT-5e nebo lepší, pevné vodiče
Typ zástrčky	RJ-45 pro CAT-5e

Každá síť systému standardně podporuje nejméně až dvacet řídících panelů na sto metrech vedení.[8] Teoreticky až pět set v jednom systému. A to díky možnosti snadného škálovaní:

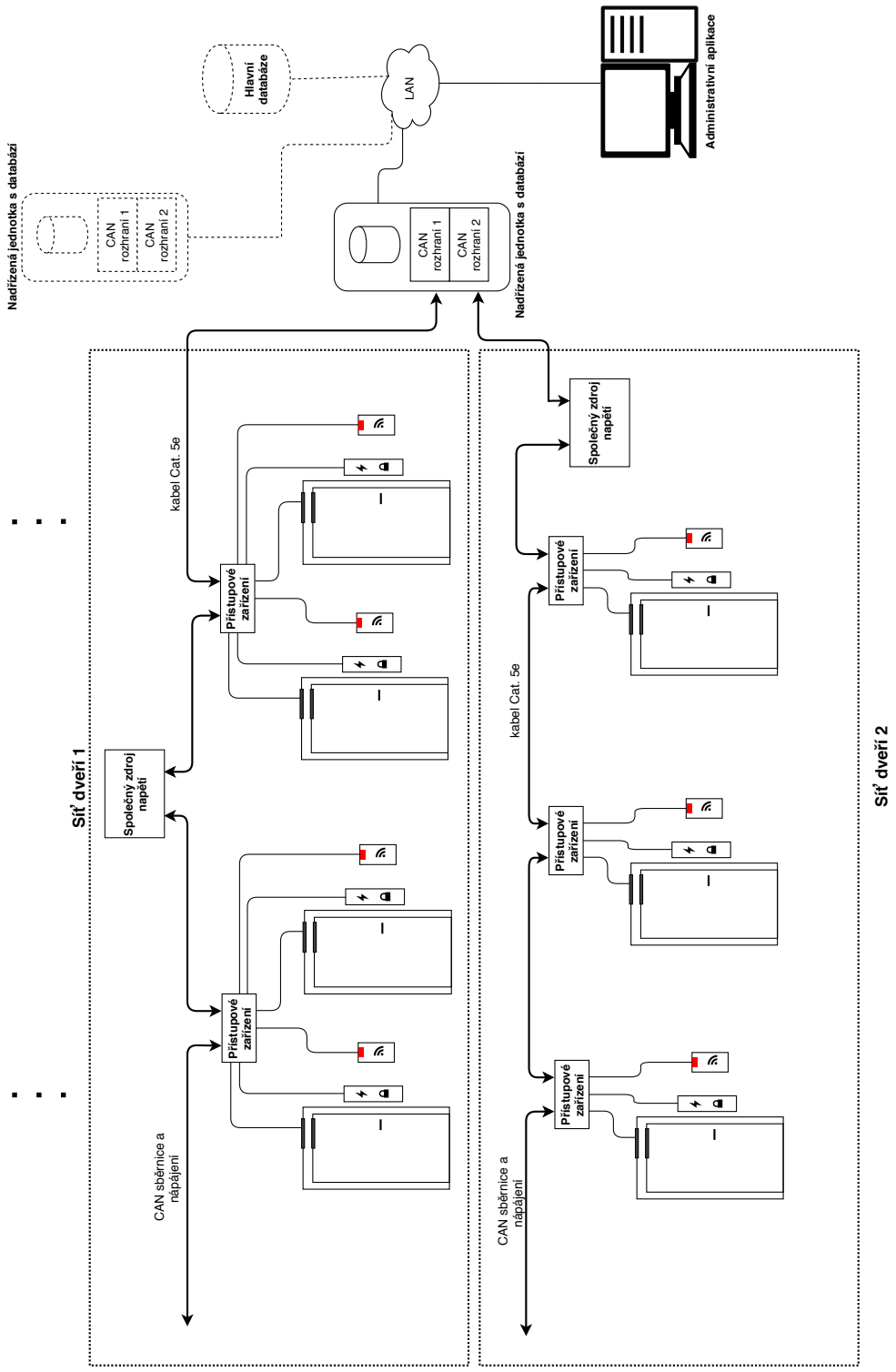
- Přidáním dalšího CAN rozhraní do řídícího serveru a tím vytvoření dalších sítím, jak je vidět na A.1;
- Prodloužit existující síť až na 500 m přidáním čtyř izolujících CAN opakovačů a dalších čtyř zdrojů. Tím se zároveň znásobí limit pro počet panelů na 100 (55 pro variantu 2D);
- Poslední možností je přidat více řídících serverů, které jsou propojeny skrze TCP/IP Ethernet. Při přidání více serverů je potřeba použít jedinou hlavní databázi pro všechny z nich. Nebo vytvořit distribuční mechanismus pro jednotlivé databáze v jednotkách a to z nadřazeného systému.

A.1 Návod k instalaci

A.1.1 Řídící panel

Pro zprovoznění panelu je nutné nahrát firmware do hardwaru a zapsat jeho unikátní adresu v systému do trvalé paměti EEPROM. Adresa panelu slouží jako identifikátor čtečky, podle které se uděluje přístup.

[8] Při použití verze s rozhraním pro jednu čtečku (0,5 W) a jeden zámek (3 W).



Obrázek A.1: Zapojení přístupového systému

Tabulka A.2: Rozložení dat v EEPROM řídícího panelu

0x00	Adresa [7:0]
0x01	Výplň [15:10], Adresa [9:8]

Nahrání firmwaru se provádí nejlépe nástrojem FlashMagic[9] přes sériovou linku (UART) z počítače. Postup je následující:

1. Panel musí napájený.
2. Počítač ze kterého se bude nahrávat firmware se připojí na UART piny. Jedná se 3,3 V UART s RX a TX signály. Pro správnou komunikace je třeba propojit datovou zem (pozor na rozdíly potenciálu počítače a panelu).
3. Na ISP piny umístěné na desce panelu se umístí propojka
4. Krátce se propojí reset piny nebo odpojí a připojí napájení. Tím se provede restart panelu to programovacího režimu.
5. Propojka pro z ISP pinů se odstraní.
6. V nástroji FlashMagic se vybere: „LPC11C24/301“, správný COM port, „Baud Rate: 57600“, „Interface: None (ISP)“, „Oscillator: 12 MHz“, „Erase all“, cesta k souboru firmwaru ve formátu HEX (přípona .hex) a „Verify after programming“.
7. Nahrávání se spustí tlačítkem „Start“. Po chvíli by měl nástroj indikovat výsledek.

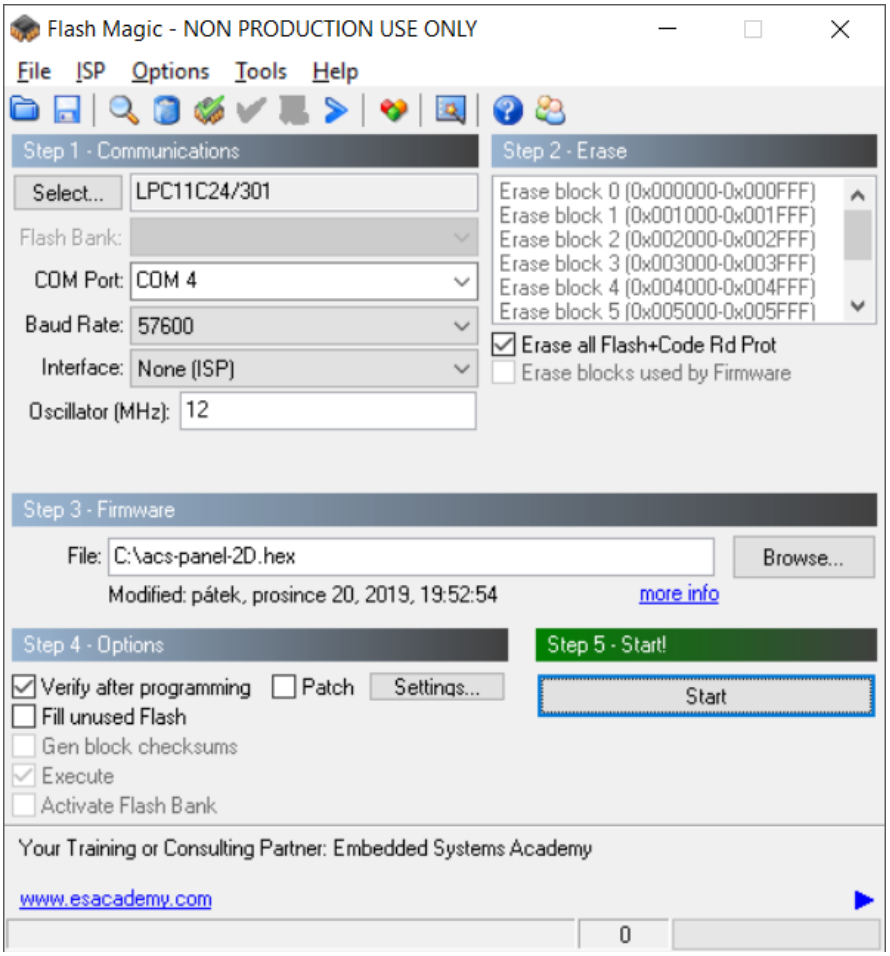
Správné nastavení je na obrázku A.2. Návod k nahrání firmwaru je zároveň v souboru `readme.txt` v `/bin` na přiloženém médiu.

Zapsání adresy je možné externím programátorem, pro tuto potřebu je struktura EEPROM paměti popsána v tabulce A.2 nebo přes sériovou linku (rychlost 115200 b/s, 8N1) při startu panelu. To se provádí zapsáním znaku 's' hned po startu, v případě úspěchu vrátí panel seznam možných operací s instrukcemi.

Každá dvojice čtečka a dveře má v systému mít unikátní adresu a to i mezi sítěmi. Rozsah adres platných pro přístupová zařízení je 4–1023. Adresní prostor dveří je rozdělen na sudé a liché adresy. Každý panel drží dvě pro každou čtečku. To i v případě, že je to varianta s připojením jedné čtečky. Tím se při použití pouze variant 1D snižuje počet unikátních adres na polovinu.

Firmware je konfigurovatelný jen při kompilaci – popis je v dokumentaci ke zdrojovému kódu a souboru `README` v `src/impl/acs-panel` na přiloženém médiu.

[9] `http://www.flashmagictool.com` (pouze OS Windows)



Obrázek A.2: Nastavení nástroje FlashMagic

Konektory panelu jsou na schématu C.3, kde je vidět jejich význam a zapojení jednotlivých pinů.

Propojení se čtečkou karet a zámkem může být dlouhé desítky metrů v závislosti na použitých vodičích.

Podporována je většina dostupných elektrických zámků a otvíračů, které jsou ovládané kontakty `NO`, `NC` a `COM` nebo přímo přivedením napětí a to po celou dobu trvání odemčení. Nelze použít zámky ovládané krátkými pulzy. Panel umožňuje přímo napájet pouze zámky s nízkým příkonem (podle specifikace), jinak musí mít vlastní napájecí zdroj.

Na vstup pro monitorování stavu dveří může být připojen pasivní magnetický spínač na rámu dveří nebo integrovaný v dveřním zámku. Lze nakonfigurovat zda se jedná o `NO` nebo `NC` variantu.

Na zařízení jsou dvě stavové LED. Červená indikuje funkční napájení. Zelená svítí při detekci periodických hlášení z nadřízené jednotky. Jakmile se

ztratí kontakt, do 5 s zelená zhasne.

A.1.2 Propojení panelů

Topologie propojovací sítě je lineární. Každý panel má dvě stejné RJ-45 zástrčky pro připojení do sítě. Kabel od jednoho panelu vede k dalšímu, jak je vidět na obrázku A.1.

Pro propojení panelů instalovaných u jednotlivých dveří v jedné síti se používá čtyř-párová kroucená dvoulinka s pevnými vodiči. Splňující požadavky pro kategorii 5e definovanou v [32] nebo ekvivalentní třídu D podle [33, kap. 9.3] nebo lepší. Průměr měděného vodiče musí být 0,511 mm (AWG 24) nebo větší.

Konektory RJ-45 (modulární 8P8C) propichující izolaci se zapojí na jednotlivé páry kabelu podle zapojení T568A nebo T568B definované v [42, 5.3.3]. Vždy se na obou stranách kabelu musí použít stejný typ. Konektory musí být určené pro použitý typ kabel a 4PPoE aplikaci, a mít specifikované trvalé zatížení proudem na každý kontakt alespoň 1,5 A při 25 °C.

První pár v kabelu je datový pro CAN sběrnici. Na obou koncích každé sítě, musí být zakončen odporem odpovídajícím impedanci kabelu. Doporučená hodnota je tedy 100 Ω pro kategorii 5e. Zakončování je integrované do hardwaru panelů a adaptérů. Ostatní páry jsou určené pro přenos energie.

Pozor na degradaci konektorů při zapojování a odpojování zařízení do sítě pod napětím z důvodu velkého výkonu přenášeného po kabelu. Je velmi vhodné při zásahu do instalace vypnout napájení sítě. Špatně nasazené nebo nevhodné konektory (např. řezné do vodiče nebo neodpovídající kategorie pro kabel) mohou snížit možný počet zařízení a zkrátit životnost vedení.

Pro lepší odvod tepla vzniklého ze ztrát se doporučuje použít stíněný kabel (obalený folií anebo stíněné vodiče) a konektory.

Teplota prostředí ve kterém se nachází vedení nesmí překročit 35 °C při použití kabelů CAT-5e.

A.1.3 Napájecí zdroj

Pro každou síť je nutný jeden společný izolovaný zdroj napětí s ochranou proti zkratu a výstupním napětím 58 V s výkonem alespoň 175 W pro plný počet řídících panelů (podle tabulky A.1). Požadovaný výkon může být nižší při nasazení méně panelů.

Pro zajištění nejmenších možných ztrát na vedení, by měl zdroj být zapojen co nejblíže k polovině délky sítě, tak aby polovina zařízení byla na jedné straně a zbytek na druhé. Doporučené je dodržet umístění mezi 1/4 a 3/4 délky, avšak systém bude funkční i při umístění zdroje na konec sítě. Toto doporučení je povinné v sítích, kde bude více 1D variant panelů než 2D.

Obrázek A.3: Příklad vhodného CAN rozhraní, převzato z [5]

A.1.4 Řídící server

Na řídícím serveru běží řídící software připojený k Redis databázi, pro uložení uživatelů a jejich oprávnění. Ta je standardně na tom samém serveru, ale může být na dedikovaném serveru připojeném přes TCP/IP protokol. K řídícím panelům je server připojen přes CAN sběrnici.

Software pro řídící server je možné nainstalovat na počítač, splňující tyto požadavky:

- má minimálně dvou jádrový procesor Intel Core třetí generace na 2 GHz nebo ekvivalent (je možné použít i některé výkonnější ARM procesory),
- je pro něj dostupná renomovaná distribuce OS GNU/Linux,
- velikosti operační pamětí je alespoň 1 GB (pro interní databázi),
- splňuje ostatní požadavky pro běh Redis databáze,
- má Ethernet rozhraní,
- je vybavený rozhraním pro CAN sběrnici, které splňující požadavky níže.

Použité CAN rozhraní musí mít galvanickou izolaci a odolnost proti elektrostatickému výboji z důvodu dlouhého vedení. Musí zároveň implementovat standard CAN 2.0 [16, část B]. Dále podporovat Linuxový balík SocketCAN, to znamená mít dostupný ovladač typu síťové zařízení (označované jako Network Device), ke kterému se dá přistupovat přes soketové rozhraní. Více informací k SocketCAN včetně podporovaných CAN kontrolérů je v jeho dokumentaci [25] a zdrojovém kódu Linux jádra. Příklad otestovaného CAN rozhraní do USB, které uvedené požadavky splňuje a má ovladač už zaintegrovaný v jádře, je produkt PCAN-USB [5] s kontrolérem NXP SJA1000.

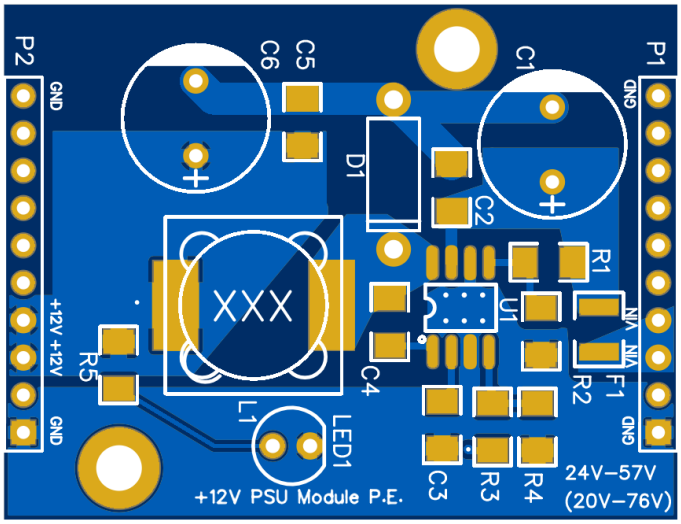
Návod k instalaci řídícího softwaru je popsán v souboru `readme.txt` u zdrojového kodu v adresáři `src/impl/acs-server` na přiloženém médiu.

Do sítě řídících panelů se server připojuje adaptérem, popsaným v příloze D, který odloučí pouze datové vodiče sběrnice CAN od napájení panelů. Adaptér má 9pinový D-Sub konektor, zapojení je kompatibilní s normou [43, kap. 6.1], pro připojení k serveru. Vzdálenost propojení mezi serverem a adaptérem musí byt co nejkratší pro zachování lineárnosti sítě. Proto je adaptér vhodné zapojit přímo do CAN rozhraní serveru.

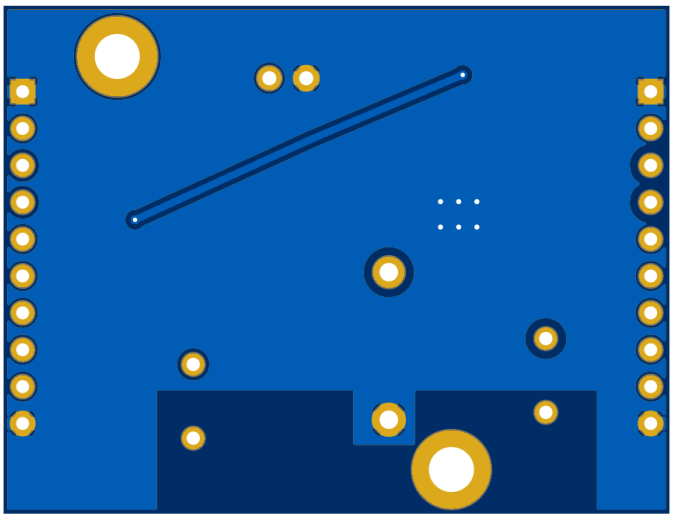
Doporučeno je zálohovat napájecí zdroj řídící serveru pro omezení možnosti ztráty funkce systému nebo dat z databáze.

Příloha B

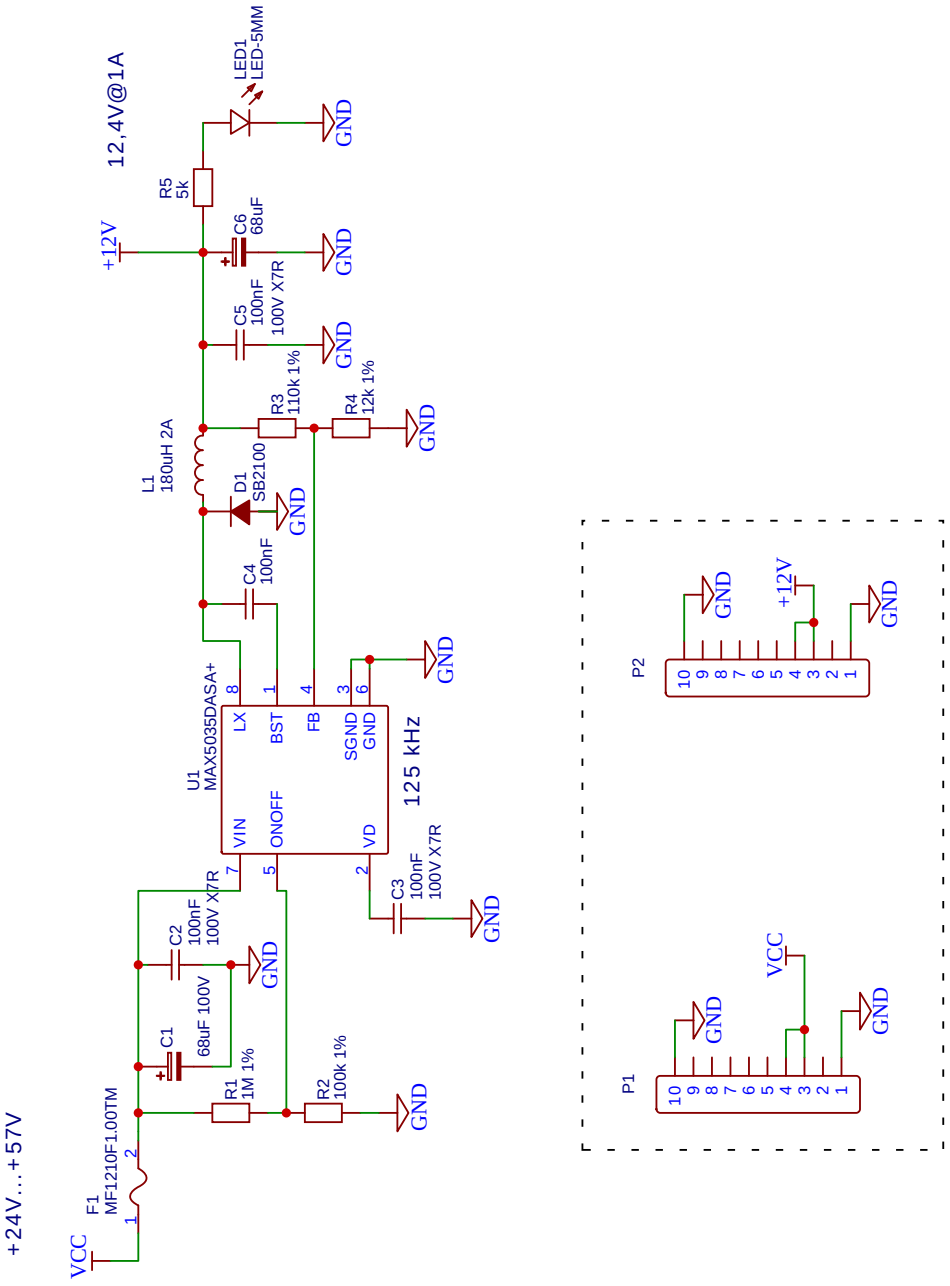
Prototyp přístupového zařízení



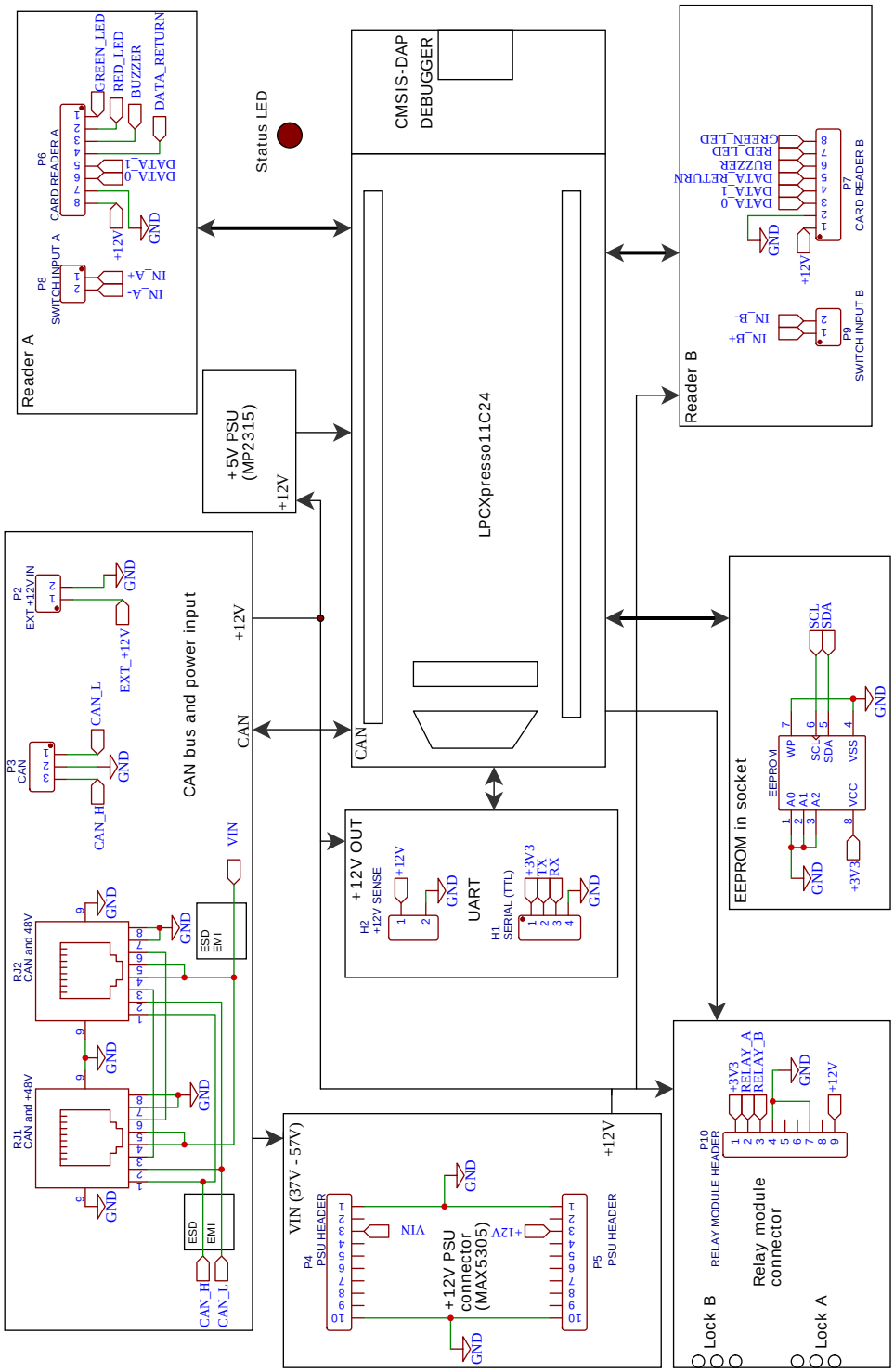
Obrázek B.1: Vrchní pohled na DPS hlavního spínaného zdroje prototypu



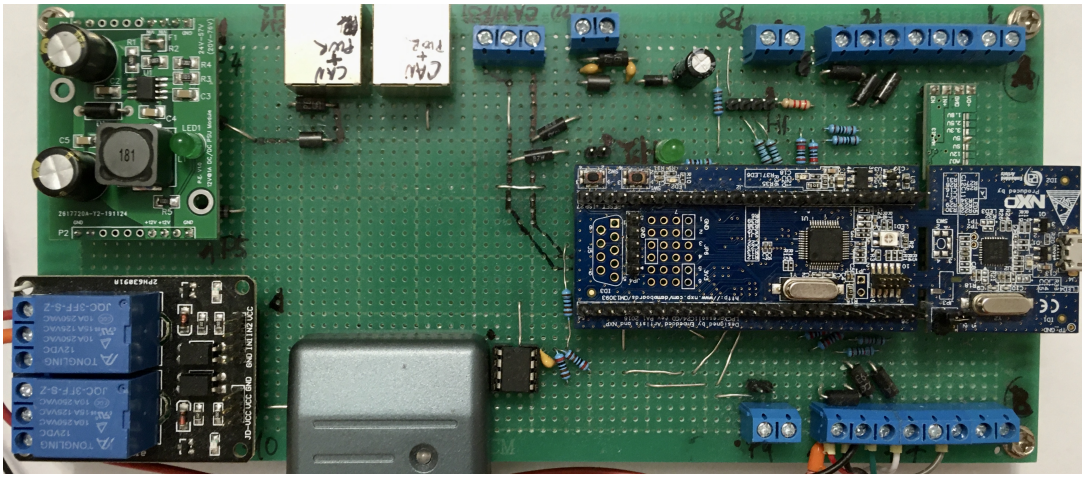
Obrázek B.2: Spodní pohled na DPS hlavního spínaného zdroje prototypu



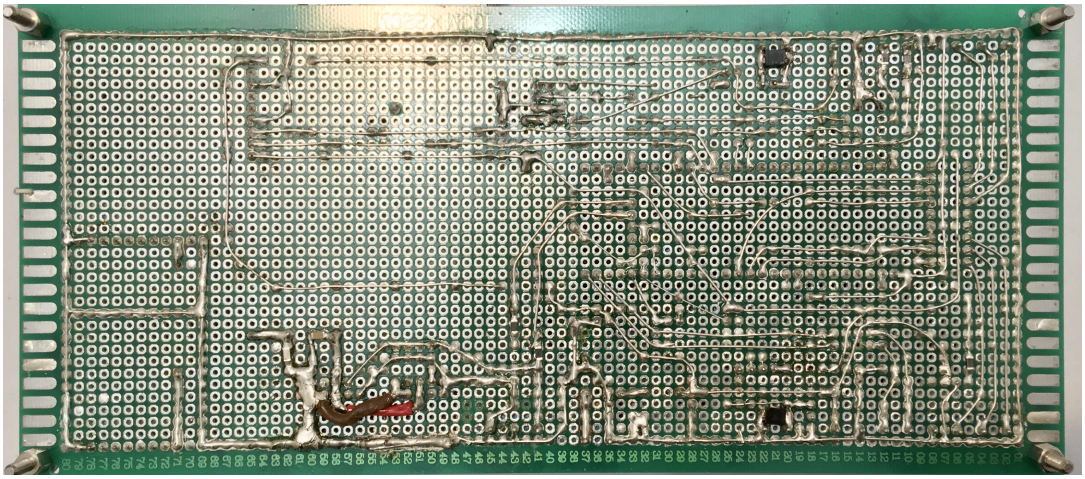
Obrázek B.3: Zapojení modulu hlavního spínaného zdroje prototypu



Obrázek B.4: Blokové schéma prototypu řídícího panelu



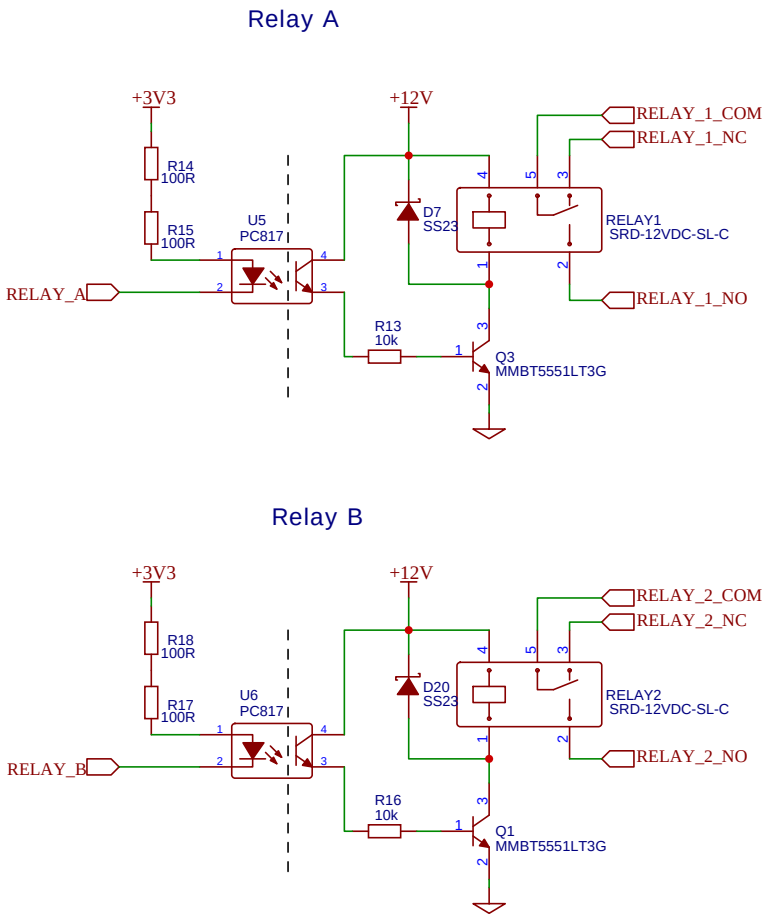
Obrázek B.5: Vrchní pohled na DPS prototypu řídícího panelu



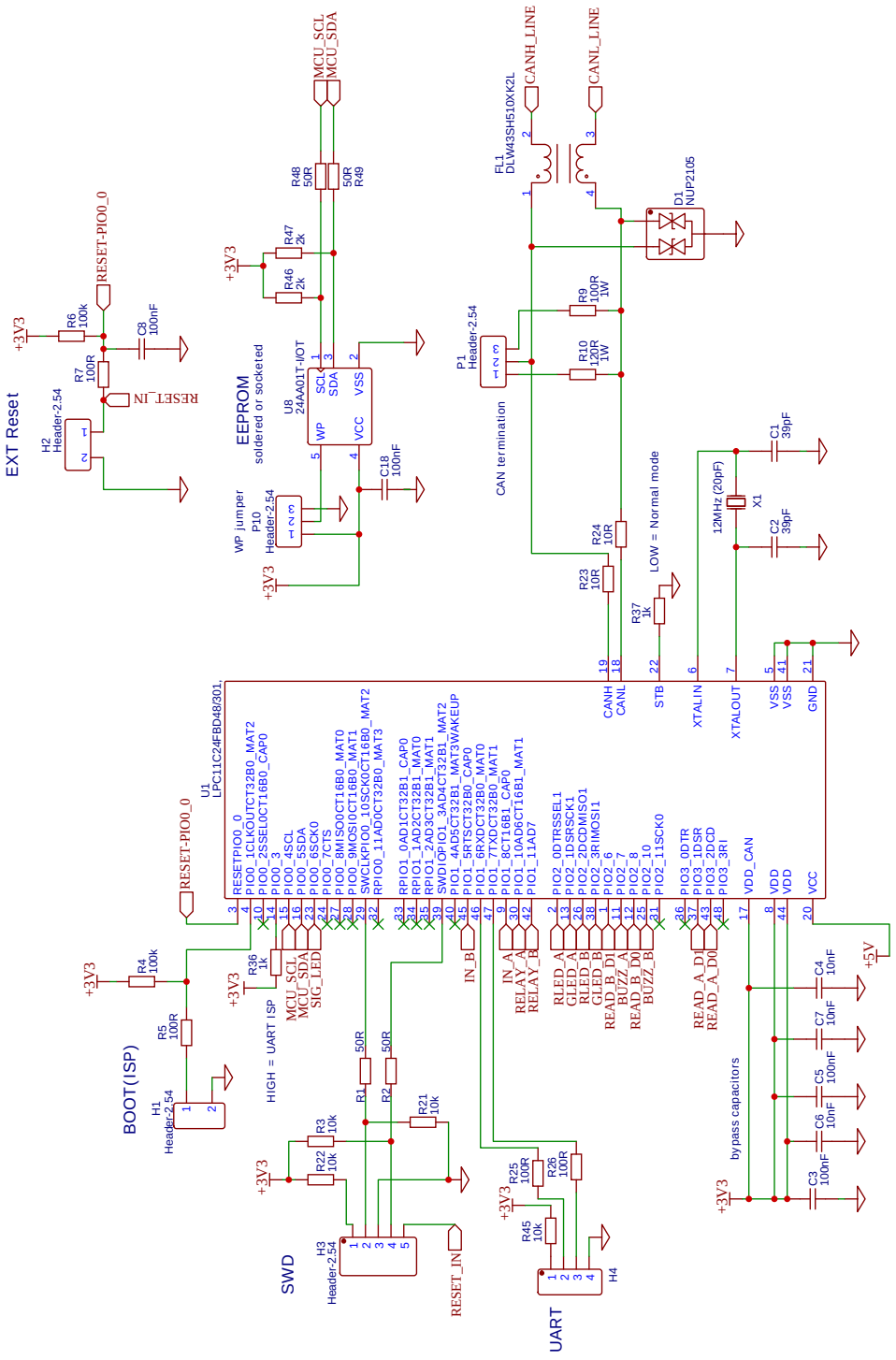
Obrázek B.6: Spodní pohled na DPS prototypu řídícího panelu

Příloha C

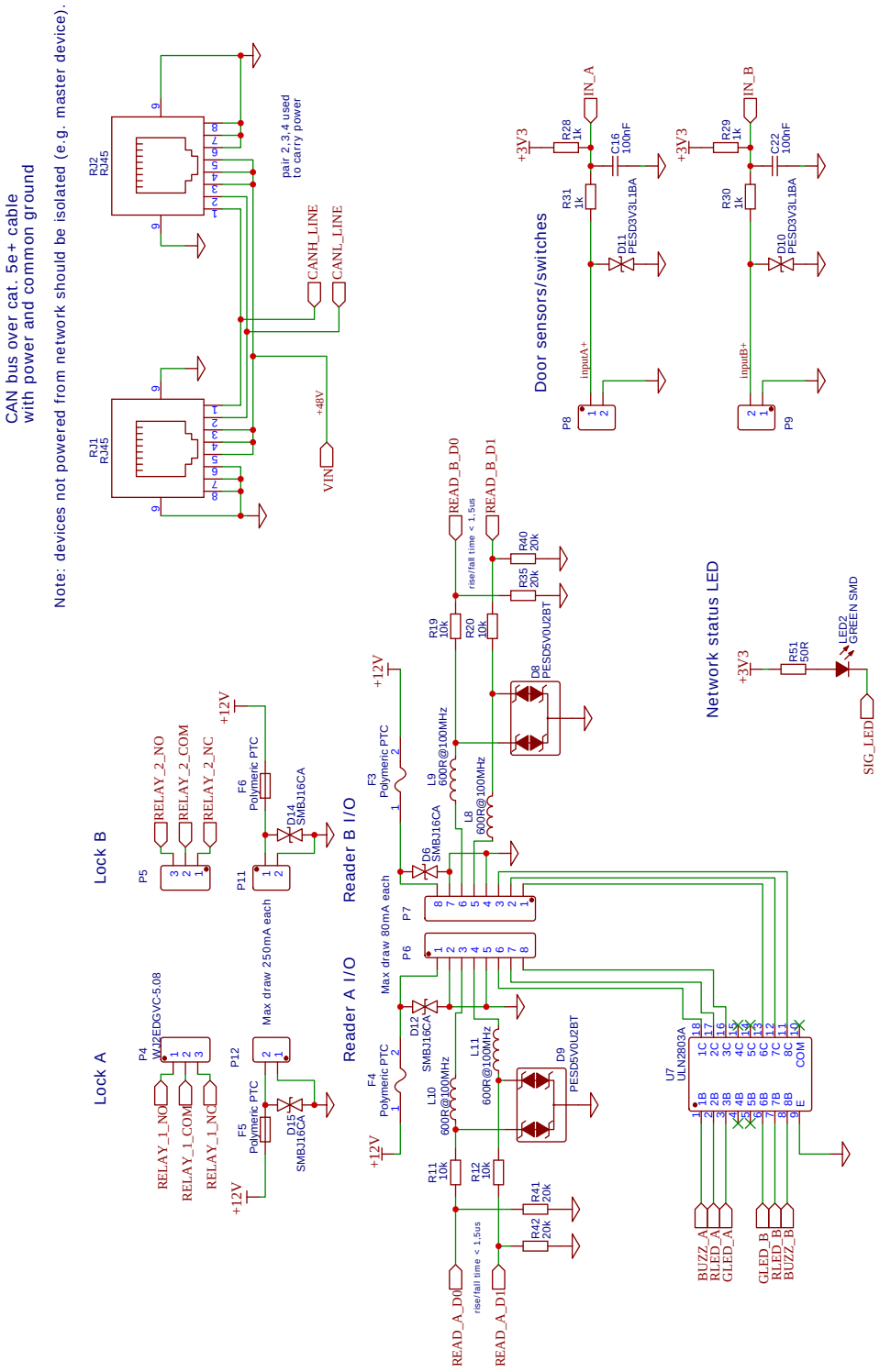
Schéma první verze panelu



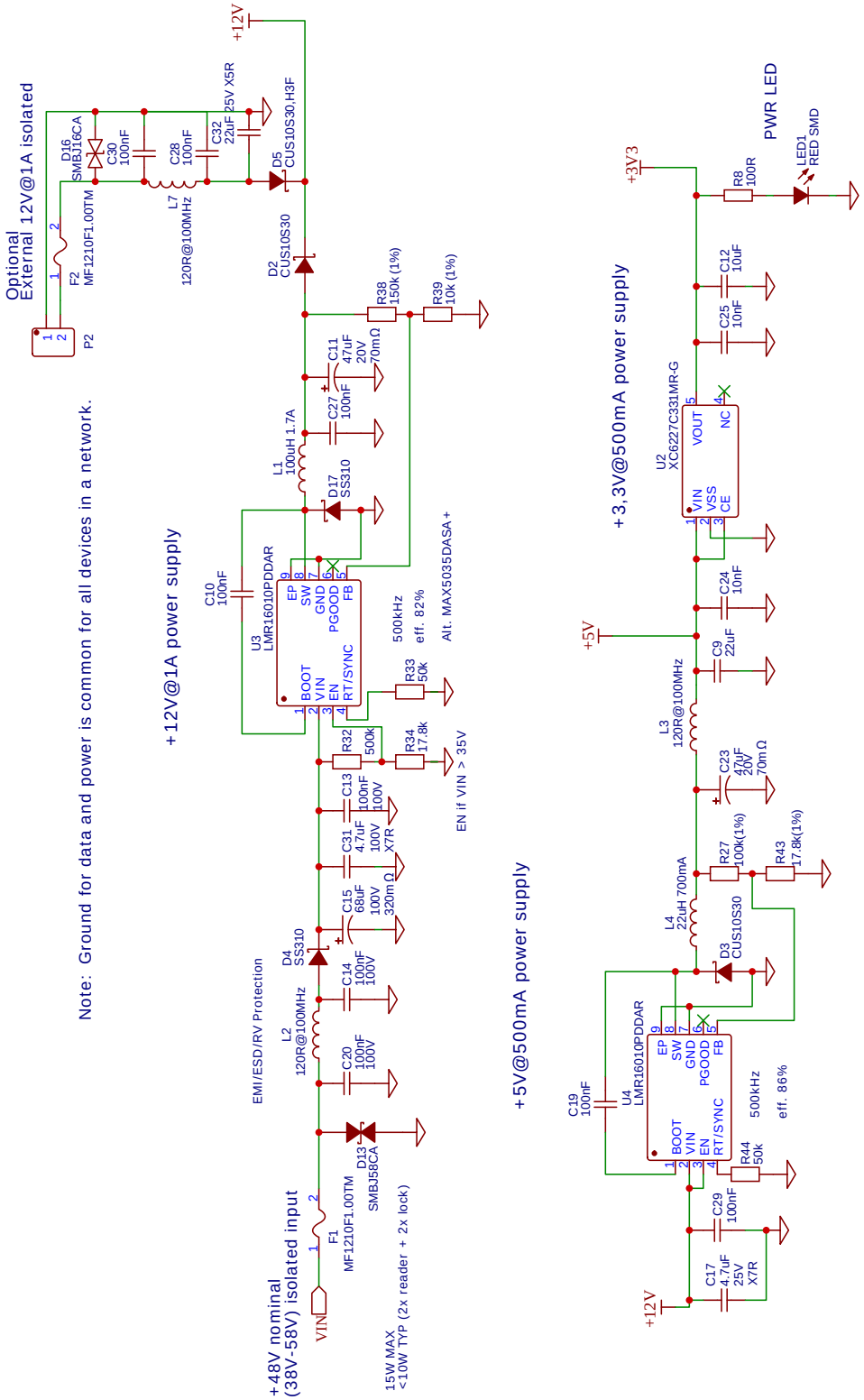
Obrázek C.1: Zapojení relé první verze řídícího panelu



Obrázek C.2: Zapojení procesoru první verze řídícího panelu



Obrázek C.3: Zapojení konektorů první verze řídícího panelu

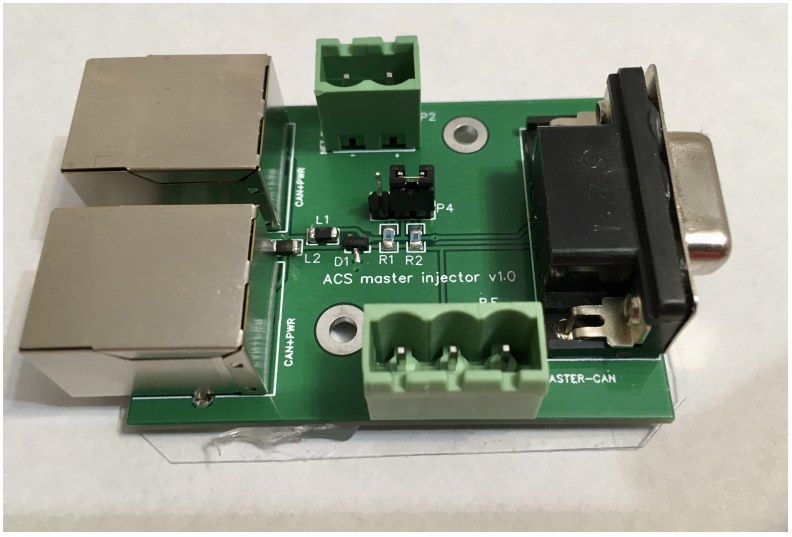


Obrázek C.4: Zapojení zdrojů první verze řídícího panelu

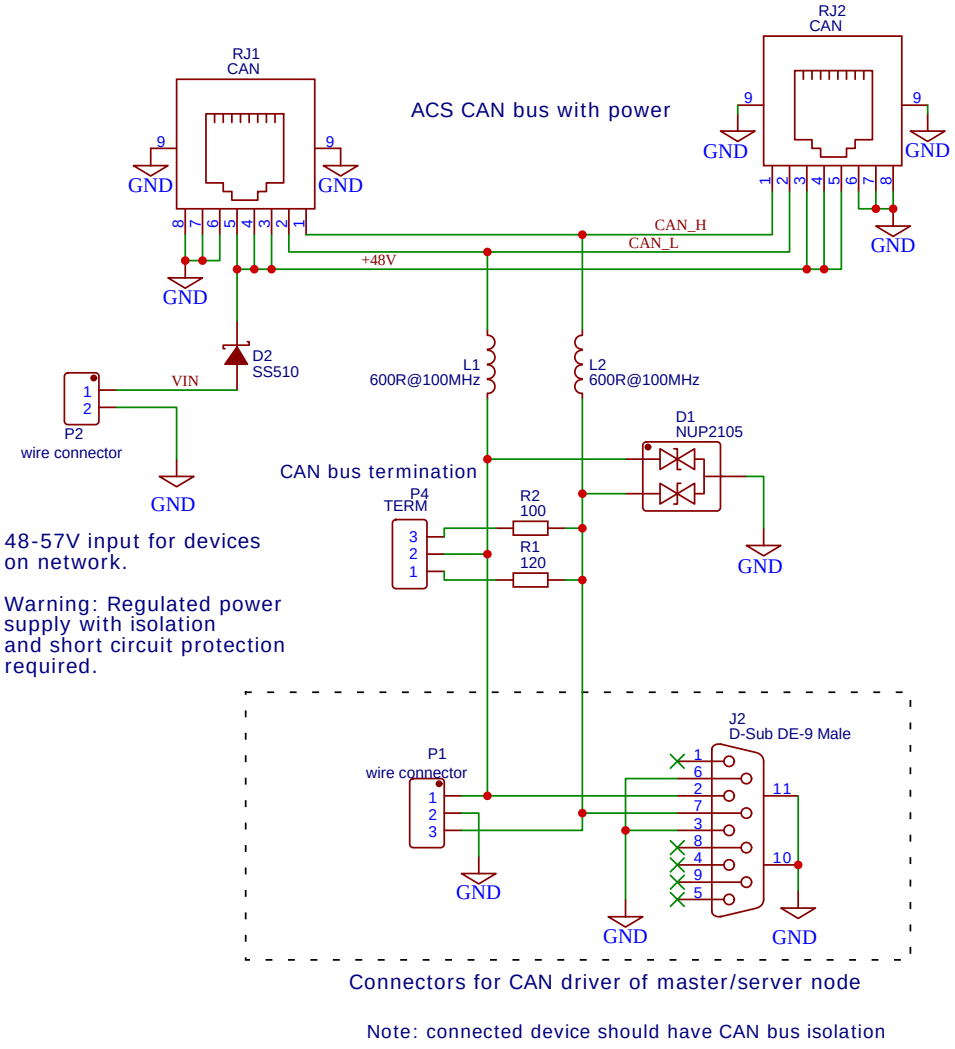
Příloha D

Adaptér pro přípojení CAN řadiče a zdroje

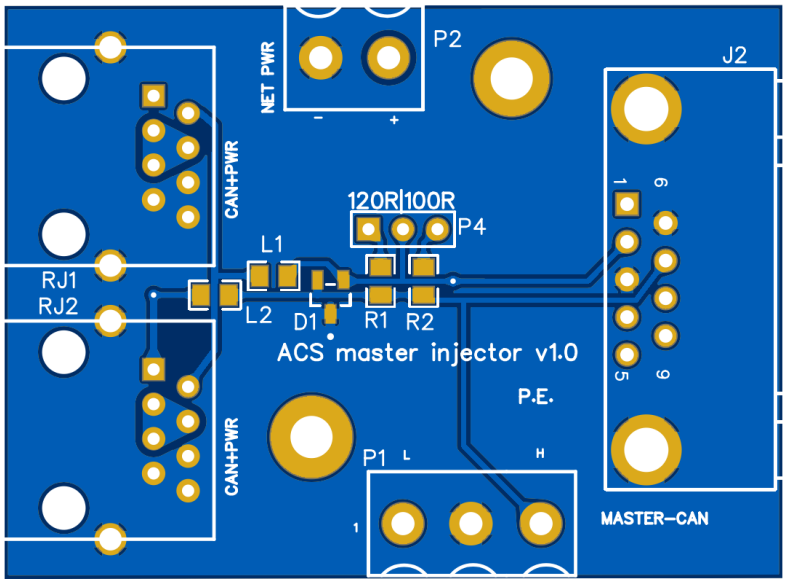
Adaptér slouží k připojení zdroje anebo CAN řadiče řídícího serveru do sítě panelů. Má integrované zakončovací odpory.



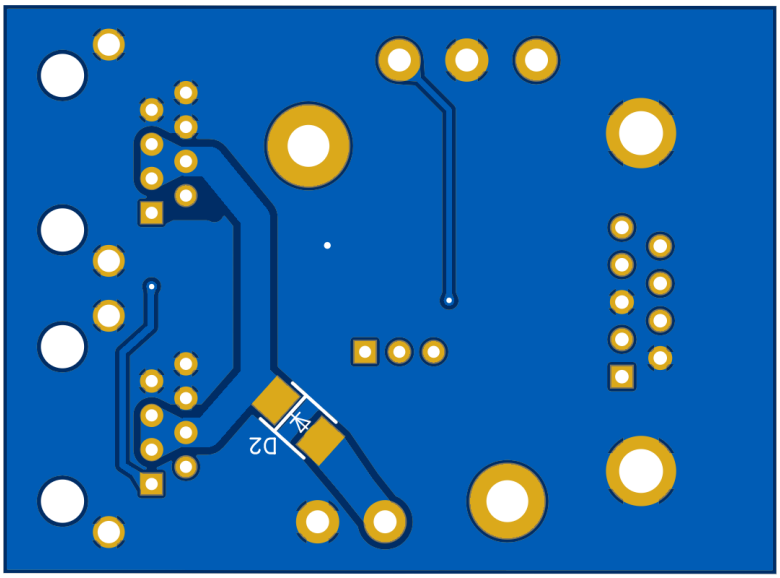
Obrázek D.1: První verze adaptéru pro server a zdroj



Obrázek D.2: Zapojení první verze adaptéru



Obrázek D.3: Vrchní pohled na DPS první verze adaptéru

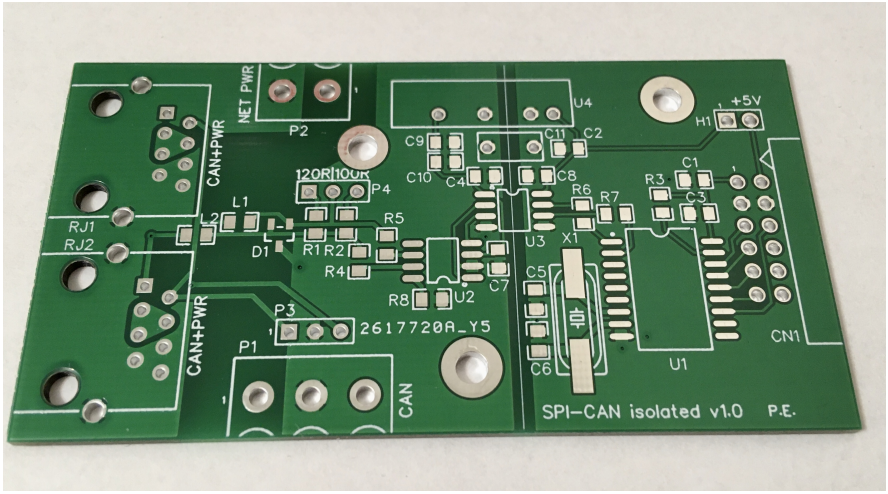


Obrázek D.4: Spodní pohled na DPS první verze adaptéru

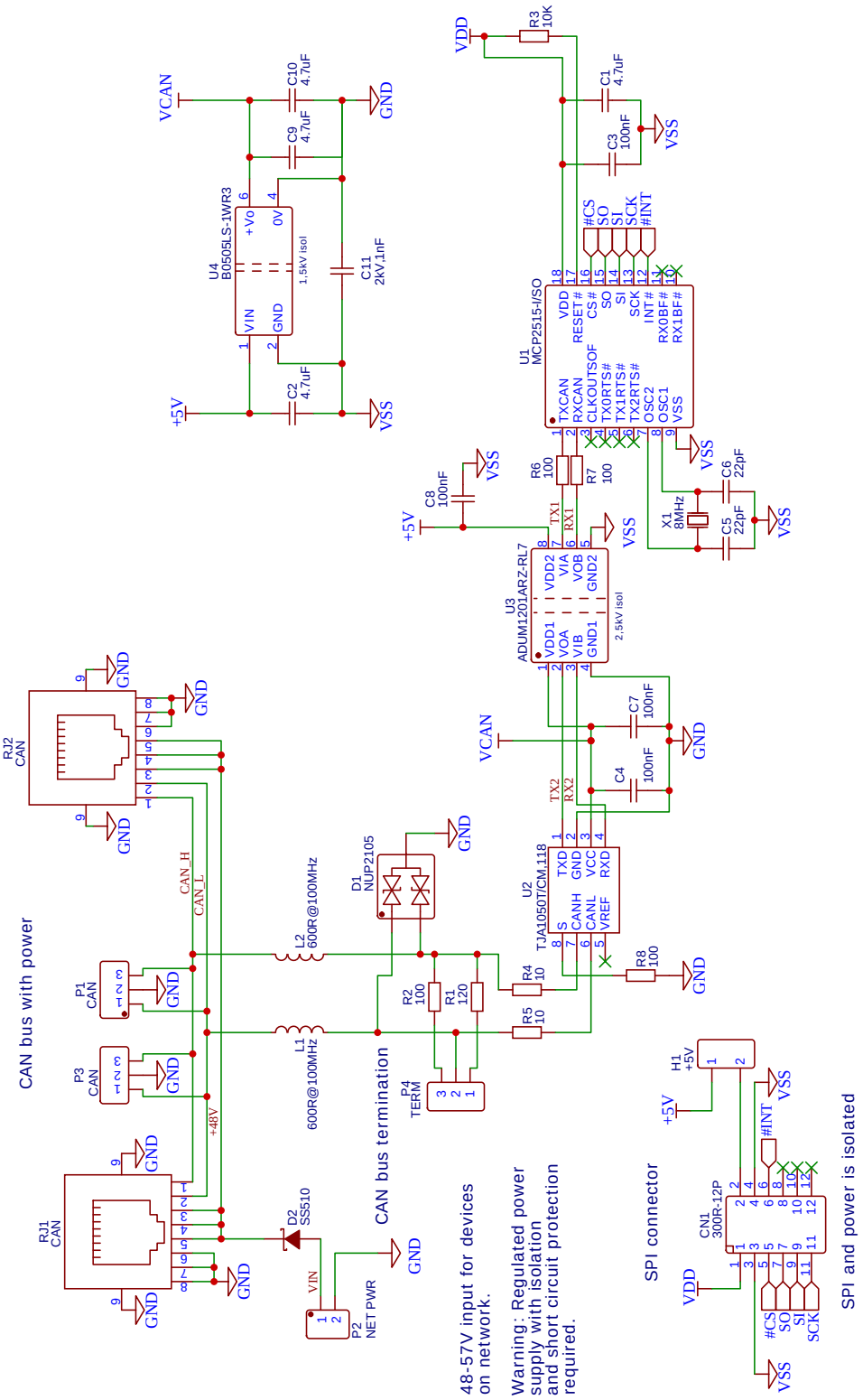
Příloha **E**

Izolovaný adaptér s CAN

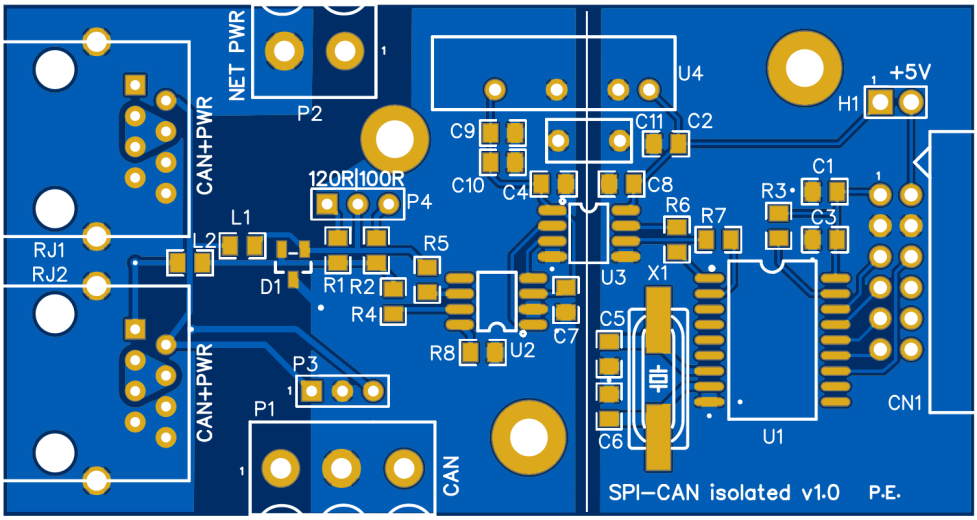
Adaptér je vhodný pro rozšíření řídícího serveru o CAN rozhraní pokud disponuje SPI rozhraním. Obsahuje CAN řadič a budič. Zároveň umožňuje připojení zdroje pro přístupová zařízení na síti.



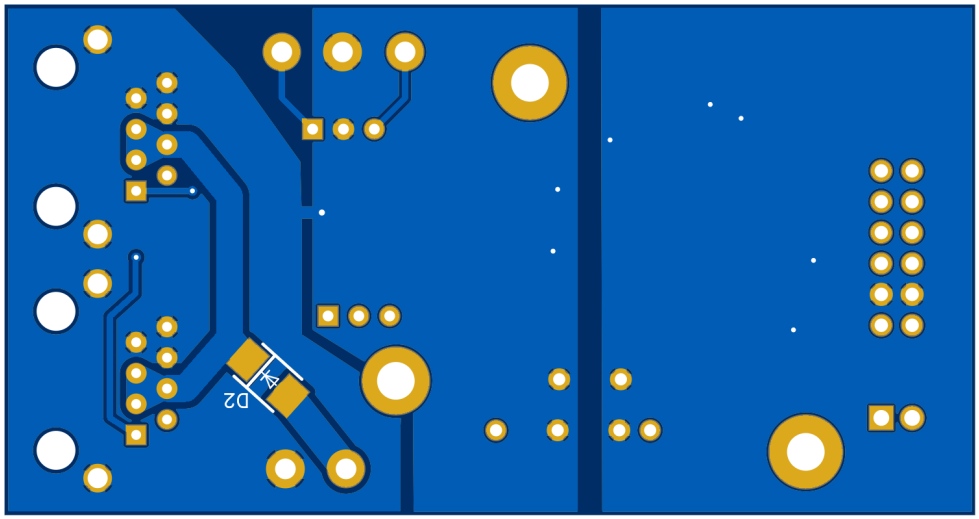
Obrázek E.1: První verze adaptéru s CAN



Obrázek E.2: Zapojení první verze adaptéru s CAN



Obrázek E.3: Vrchní pohled na DPS první verze adaptéru s CAN



Obrázek E.4: Spodní pohled na DPS první verze adaptéru s CAN

Příloha F

Seznam použitých zkratek

AC Alternating current

ACS Access control system

API Application programming interface

CAN Controller area network

CRC Cyclic redundancy check

DC Direct current

DPS Deska plošných spojů

EEPROM Electrically erasable programmable read-only memory

EFF Extended frame format

GCC GNU compiler collection

GPIO General-purpose input/output

GUI Graphical user interface

HW Hardware

I^{2}C Inter-integrated circuit

IDE Integrated development environment (integrované vývojové prostředí)

IP Internet protocol

ISP In-system programming

LED Light-emitting diode

LSB Least significant bit

MCU Microcontroller unit

MSB Most significant bit

NFC Near field communication

OS Operační systém

P2P Peer to peer

PC Personal computer

PCI Peripheral component interconnect

PoE Power over Ethernet

4PPoE Four-pair power over Ethernet

RAM Random-access memory

RFID Radio frequency identification

ROM Read-only memory

RTOS Real-time operating system

SW Software

SFF Standard frame format

SPI Serial peripheral interface

SSH Secure shell

SWD Serial wire debug

TCP Transmission control protocol

UART Universal asynchronous receiver/transmitter

UTP Unshielded twisted pair

USB Universal serial bus

V/V Vstupně-výstupní

Příloha **G**

Obsah přiloženého média

Zdrojové kódy implementace systému jsou také dostupné z repositáře na adrese: `https://gitlab.fit.cvut.cz/elexapet/acc-sys-door-term`

- `readme.txt` ... stručný popis obsahu média
- `bin` ... binární forma firmwaru panelu
- `schematics` ... schémata desek plošných spojů ve formátu PDF
- `src` ... zdrojové kódy
  - `3rd_party` ... zdrojové kódy třetí strany
  - `hw` ... projektové soubory k DPS pro EasyEDA
  - `impl` ... zdrojové kódy implementace
    - `acs-panel` ... zdrojový kód pro přístupová zařízení
    - `acs-server` ... zdrojový kód a instrukce pro server systému
    - `lpc_chip_11cxx_lib` ... projekt LPCOpen knihovny
  - `thesis` ... zdrojová forma práce ve formátu LaTeX
- `text` ... text práce
  - `references` ... některé použité dokumenty
  - `DP_Elexa_Petr_2020.pdf` ... text práce ve formátu PDF
- `tools` ... pomocné nástroje