VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

PREVODNÍK MEDZI PROTOKOLMI MODBUS-RTU A MODBUS-TCP

CONVERTER OF MODBUS-RTU AND MODBUS-TCP PROTOCOLS

SEMESTRÁLNÍ PROJEKT
TERM PROJECT

AUTOR PRÁCE **Bc. ERIK PITKO**
AUTHOR

VEDOUCÍ PRÁCE **Ing. VOJTĚCH MRÁZEK, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2020/2021

Zadání diplomové práce

Student: **Pitko Erik, Bc.**
Program: Informační technologie a umělá inteligence
Specializace: Inteligentní zařízení
Název: **Převodník mezi protokoly Modbus-RTU a Modbus-TCP**
Converter between Modbus-RTU and Modbus-TCP Protocols
Kategorie: Vestavěné systémy
Zadání:

1. Seznamte se problematikou průmyslových komunikačních protokolů, rozhraním RS-485.
2. Prozkoumejte možnosti implementace připojení vestavěných zařízeních k Ethernetu.
3. Zpracujte studii na výše uvedená témata.
4. Navrhněte vestavěné zařízení realizující převod mezi rozhraním Modbus-TCP a Modbus-RTU. V práci se zaměřte na jednoduchou konfigurovatelnost zařízení, spolehlivost a cenu zařízení.
5. Navržené zařízení implementujte.
6. Zhodnoťte naimplementované zařízení.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Mrázek Vojtěch, Ing., Ph.D.**
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 19. května 2021
Datum schválení: 30. října 2020

Abstrakt

Cieľom tejto práce je oboznámiť čitateľa s komunikačnými protokolmi Modbus TCP a Modbus RTU. Následne vytvoriť kompletné, cenovo dostupné vstavané zariadenie na báze mikroprocesoru STM32. Toto zariadenie by malo obsahovať jednoduché užívateľské rozhranie pre prvotnú konfiguráciu zariadenia. Po prvotnej konfigurácií by zariadenie malo byť schopné prevodu medzi spomenutými protokolmi.

Abstract

The main objective of this work is to inform the reader of the communication protocols Modbus TCP and Modbus RTU and create an embedded device based on the microprocessor STM32 capable of conversion between Modbus RTU and Modbus TCP protocols. The device should be capable of simple first run configuration with simple user interface.

Kľúčové slová

Prevodník, Modbus, Modbus RTU, Modbus TCP, STM32, Ethernet, MII, RMI, RS485

Keywords

Converter, Modbus, Modbus RTU, Modbus TCP, STM32, Ethernet, MII, RMII, RS485

Citácia

PITKO, Erik. *Prevodník medzi protokolmi Modbus-RTU a Modbus-TCP*. Brno, 2021. Semestrální projekt. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vojtěch Mrázek, Ph.D.

Prevodník medzi protokolmi Modbus-RTU a Modbus-TCP

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod odborným vedením Ing. Vojtěcha Mrázka, Ph.D. a s použitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

.....
Erik Pitko
19. mája 2021

Poďakovanie

Touto cestou by som chcel poďakovať Ing. Vojtěchovi Mrázkovi, Ph.D. za odbornú pomoc a cenné rady pri vedení tejto diplomovej práce.

Obsah

Kapitola 1

Úvod

Cieľom tejto práce je vytvoriť prevodník medzi priemyselnými protokolmi Modbus RTU a Modbus TCP ako vstavaný systém. Navrhnutý prevodník bude umožňovať prepojenie snímačov, čidiel a výkonných členov s rozhraním RS-485 a podporou komunikačného protokolu Modbus RTU do riadiacich alebo technologických informačných systémov s Ethernetovým rozhraním. Vďaka nízkej cene a vysokej dostupnosti zariadení s podporou Modbus RTU protokolu sa tento prevodník uplatní hlavne v procese riadenia výroby.

Prvá časť tejto práce má teoretický charakter a popisuje priemyselný Modbus TCP a Modbus RTU protokol, štandardné sériové zbernice (RS-232, RS-422, RS-485), rozhranie priemyselného Ethernetu a základné komunikačné pojmy. Druhá časť je venovaná samotnému návrhu vstavaného systému realizujúceho prevod medzi rozhraním Modbus TCP a Modbus RTU.

Vstavaný systém môže byť chápaný ako jednoúčelový počítač, v ktorom je zabudovaný celý riadiaci systém. Tieto zariadenia sa stávajú čoraz populárnejšie vďaka klesajúcej cene potrebných komponentov pre výstavbu takéhoto zariadenia. Zároveň sa aj služby pre vývoj a prototypovanie stávajú dostupnejšie.

V kapitole 2 sú zhrnuté informácie o prenose dát, vrátane rôznych topológii, možnosťami prepojenia zariadení a základnými typmi komunikácie.

Kapitola 3 je zameraná na priemyselný Ethernet, popis protokolu pre jeho využitie v zariadeniach a popis špecifikácie priemyselného Modbus TCP protokolu.

Kapitoly 4 a 5 sa venujú návrhu a implementácii celého vstavaného systému - od špecifikácie požiadaviek, cez návrh schémy a dosky plošných spojov (DPS), až po softvérovú časť zariadenia. Posledná kapitola je zameraná na popis funkčnosti a testovanie výsledného zariadenia.

Kapitola 2

Prenos dát

Prenos dát je proces prenášania správ alebo digitalizovaného signálu pomocou fyzického dvojbodového alebo viacbodového prenosového média. Dáta môžu byť prenesené buď v analógovej alebo digitálnej podobe. Tento pojem väčšinou vyjadruje prenos informácií na určitú diaľku, môže sa jednať o vzdialenosť niekoľkých metrov alebo až kilometrov.

V prípade analógového prenosu sa jedná o spojitý prenos signálu, ktorý sa mení v čase a amplitúde. Typicky je analógový signál prenášaný pomocou modulácie.

Pri digitálnom prenose sú správy reprezentované:

- signálom diskrétnom čase, ktorý ma diskrétny počet úrovní. Môže sa jednať napríklad o vzorkovaný alebo kvantifikovaný analógový signál.
- signálom v spojitom Čase, ktorý reprezentuje prúd bitov.

Základnou jednotkou pre dátovú komunikáciu je spoj, ktorý umožňuje výmenu informácií (užívateľských alebo riadiacich) medzi informačným zdrojom a informačným spotrebičom, t.j. medzi stanicami.

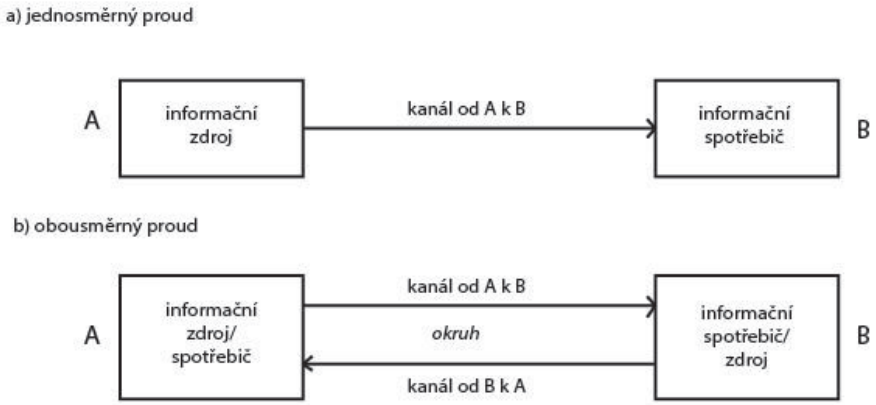
Dvojbodová komunikácia sa skladá zo dvoch staníc prepojených prenosovou cestou. Prenosová cesta sa realizuje pármi metalických vodičov, koaxiálnym vodičom[1], optickým vláknom alebo príslušným kmitočtovým pásmom pri bezdrôtovom prenose. Prenosová cesta slúži k prenosu signálov nesúcich informácie, ktoré môžu byť elektrické, optické alebo elektromagnetické.

Mnohobodová komunikácia (označovaná ako P2MP, PTMP alebo PMP) je komunikácia umožňujúca priame spojenie jedného zariadenia s viacerými zariadeniami. Tento spôsob prepojenia je najpopulárnejší v oblasti bezdrôtových sietí a telekomunikácií

Vlastný dátový spoj môže byť jednosmerný alebo obojsmerný. Prenosová cesta pre obojsmerný prenos signálov je označovaná ako tzv. okruh. Zatiaľ čo prenosová cesta pre jednosmerný prenos je tzv. kanál. Okruh je teda dvojica protismerných kanálov.

Pokiaľ medzi stanicami existuje kanál, jedná sa o simplexný (SX) prenos informácie od jednej stanice k druhej stanici. Pokiaľ medzi stanicami je spojenie typu okruh, dá sa po ňom prenášať informácie súčasne obojsmerne, jedná sa o plne duplexný prenos (FD). Pokiaľ sa prenos uskutočňuje striedavo, označujeme tento prenos ako polo-duplexný (HDX).

[1] Koaxiálny vodič je tvorený dvoma vodičmi, kde vonkajší obaľuje vnútorný (väčšinou medený, jednožilový), po ktorom sa prenášajú signály



Obr. 2.1: Dvojbodové spoje: jednosmerný a obojsmerný [28]

2.1 Topológie dátovej siete

Topológia dátovej siete vychádza z požiadavkov na aplikáciu prenosových prostriedkov, pripojovacích prostriedkov, prenosovej rýchlosti a prístupu k nim. Každý typ danej topológie má svoju tzv. fyzickú a logickú topológiu.

Fyzická topológia popisuje fyzické prepojenie vedenia prenosových prostriedkov. Logická topológia popisuje spôsob toku signálu.

LAN[2] môže mať fyzickú topológiu odpovedajúcej hviezde, napriek tomu môže spojovať jednotlivé uzly medzi sebou a fungovať ako kruh (napr. IBM Token Ring).

Topológia ma kľúčový význam v oblasti lokálnych sieti (LAN), kde s ňou úzko súvisí samotný spôsob komunikácie medzi jednotlivými uzlami.

Hviezda

Topológia hviezdy je analógiou starých terminálových sieti s centrálnym riadením. Centrálny uzol riadi smerovanie v sieti, zatiaľ čo ostatné uzly sa o smerovaní dát nestarajú. Tieto siete môžu preto byť veľmi jednoduché. Je vhodná v prípadoch, kde prevažuje v aplikácií komunikácia vedená medzi okrajovými a centrálnym uzlom. Pokiaľ vyžaduje aplikácia komunikáciu medzi okrajovými uzlami, kladú sa na centrálny uzol vysoké požiadavky (na výkon aj spoľahlivosť). Tieto siete sú potom menej spoľahlivé v porovnaní s ostatnými používanými topológiami. Prenos dát v týchto sieťach je možné riadiť veľmi jednoduchými protokolmi a dajú sa jednoducho monitorovať.

Výhodou topológie hviezdy je menšia náchylnosť k poruchám vodičov (len v rámci jednotlivých prepojení medzi koncovým uzlom) a súvisiacim výpadkom siete (pri poruche jedného spoja nie je postihnutá celá sieť, ale len príslušná stanica, okrem prípadu, kedy je výpadkom postihnutý centrálny uzol). Nevýhodou je väčšia spotreba káblových vedení.

[2]LAN - Lokálna sieť (Local Area Network)

Zbernica

Topológia zbernice nemá centrálny uzol a všetky uzly sú pripojené k zdielanému prenosovému prostriedku, ktorý umožňuje komunikáciu každý s každým. Vyžaduje zložitejšie riadenie prístupu k zdielanému prostriedku a komplikovanejšie protokoly pre riadenie prenosu dát po zbernici. Informačný signál nesúci správu sa šíri zbernicou všetkými smermi a všetky stanice majú prístup ku všetkým správam na zbernici. Skutočne príjmu iba takú, ktorá im je podľa cieľovej adresy určená.

Ku zbernici sa dajú ľahko pridávať alebo odoberať uzly bez toho aby sa tým porušil informačný tok. Ako prenosový prostriedok sa typicky používa koaxiálny kábel.

Výhodou zbernice je použitie jedného vedenia, jednotný spôsob zapojenia a jednoduché pridanie alebo odstránenie stanice zo siete. Nevýhodou oproti tomu je však vysoký počet odbočiek, ktorý môže spôsobiť problémy v sieti (rozpojenie konektoru môže spôsobiť zlyhanie celej siete) a vysoký počet pripojených staníc môže značne obmedziť využitie zbernice (nárast kolízií).

Kruh

Topológia kruh, tak ako zbernica, nemá centrálny uzol. Spojuje každé zariadenie s predchádzajúcim a nasledujúcim zariadením v sieti, s ostatnými uzlami v sieti prebieha komunikácia nepriamo, cez jeden alebo viac ďalších uzlov. Správy obiehajú uzavrenou cestou jedným smerom medzi uzlami, preto nie je potrebné riešiť žiadne smerovanie toku. Každý uzol prevezme správu od predchádzajúceho a pokiaľ sám nie je adresátom správy, pošle ju ďalej svojmu následníkovi. Každá stanica na kruhu slúži ako opakovač signálu a tiež ako bezpečnostná poistka proti chybným správam.

Výhodou kruhovej topológie je jednoduchý spôsob predávania dátových správ bez existencie kolízií medzi stanicami. Najväčšou nevýhodou je, že pri výpadku stanice dôjde k prerušeniu činnosti siete. Jednoduchý kruh sa ale so zdvojením prenosového prostriedku dá spraviť obojsmerným. Týmto spôsobom sa stáva odolnejší voči výpadku (záložným kruhom sa komunikačná cesta medzi aktívnymi prepojeniami stanice uzatvorí).

Strom

Stromová topológia používa hierarchické zoskupovanie uzlov s viacnásobnými vetvami, a to prepojením viacero sieti s topológiou hviezda. Zariadenia nižšej hierarchickej úrovne komunikujú s uzlami vyššej úrovne až ku koreňu stromu. Koreň má čiste technický význam, pretože nerozdeľuje správy a nie je zodpovedný za určenie cieľovej stanice. Správy dochádzajú ku všetkým staniciam v sieti a tie si potom vyberú iba tie, ktoré sú pre ne skutočne určené. Stromová topológia môže byť iba aktívna. Výhody a nevýhody sú obdobné ako v prípade topológie hviezda.

Zmiešaná

Zmiešaná topológia siete (mesh) ponúka viacero možných spojov medzi uzlami. Buď sa môže jednať o plne prepojenú sieť (fullmesh), kde sú všetky uzly prepojené každý s každým (v sieti celkom n * (n-1)/2 spojov), alebo o čiastočne prepojenú sieť (partialmesh), kde sa v sieti používa menej spojov (uzly, ktoré niesu priamo prepojené so všetkými ostatnými používajú ku komunikácií viacero skokov cez ostatné uzly v sieti). V tejto topológii sú uzly rovnocenné, neexistuje medzi nimi žiadny centrálny uzol.

Výhodou tejto topológie je vysoká spoľahlivosť vďaka redundancií spojov medzi uzlami. V prípade výpadku niektorého zo spojov môže existovať možnosť komunikácie po inej trase. Nevýhodou je ale nákladnosť siete práve kvôli množstvu spojov medzi uzlami.

2.2 Sériová a Paralelná komunikácia

Komunikácia, ktorá zahŕňa odosielanie dát cez sériovú komunikáciu (odosiela po jednotlivých bitoch) organizuje jednotlivé bity do určitej postupnosti. Táto postupnosť je veľmi dôležitá, pretože určuje ako sú dáta odosielané a prijímané. Označuje sa za spoľahlivú pretože dátový bit je odoslaný iba vtedy, ak už predchádzajúci bit bol odoslaný. [11]

Sériová komunikácia sa rozdeľuje na dva typy:

1. Asynchrónna
2. Synchrónna

Asynchrónna komunikácia

Dátové bity môžu byť odoslané v ktoromkoľvek časovom úseku. Pre riadenie komunikácie pred a po odosielaní samotných dát využívame tzv. „Start“ a „Stop“ bity. Tieto bity sa používajú pre synchronizáciu medzi vysielačom a prijímačom aby sa zaistilo korektné odoslanie/prijatie dát. Čas medzi odoslaním a prijímaním dátových bitov nie je konštantný, preto sú medzery v komunikácií využívané pre oddelenie rôznych sérií dat.

Výhodou takejto komunikácie je fakt, že okrem start a stop bitov nepotrebujeme žiadnu synchronizáciu medzi odosielateľom a prijímateľom. Naopak nevýhodou takejto komunikácie môže byť nižšia rýchlosť, no nemusí to byť vždy pravidlom.

Synchrónna komunikácia

Proces odosielania jednotlivých bitov v synchrónnej komunikácií je narozdiel od asynchrónnej komunikácií riadený zdielanými hodinami medzi odosielateľom a prijímateľom. Aj odosielateľ aj prijímateľ musia mať tieto hodiny synchronizované, aby nedošlo k strate paketov. Takáto forma komunikácie nám dovoľuje zvýšiť celkovú rýchlosť komunikácie pretože nie je potrebné používať start, stop bity ani pauzy medzi jednotlivými správami. Pri tejto komunikácii väčšinou využívame drahší hardware, kvôli nutnej synchronizácií medzi odosielateľom a prijímateľom.

Paralelná komunikácia

V paralelnej komunikácií využívame odosielania viacerých bitov cez viacero kanálov naraz. To znamená, že vieme dáta odosielať omnoho rýchlejšie ako pri sériovej zbernici, no kladú sa väčšie nároky na odosielanie a spracovanie takýchto dát. Tým, že sa jednotlivé bity odosielajú po rozdielnych kanáloch, môžu byť prijaté druhou stranou v rôznom poradí. Konkrétne poradie v ktorom tieto dáta prijímač prijme záleží na viacerých podmienkach, ako napríklad vzdialenosť od zdroja, vyťaženie konkrétneho kanálu apod. [11]

2.3 Rozhrania pre prenos dát

Keďže elektrický signál v drôtoch je analógový, musí byť pre funkčnú komunikáciu konvertovaný. Túto funkciu spĺňa zariadenie ukončujúce dátový okuh (ang. data circuitterminating equipment - DCE). Toto zariadenie je umiestnené medzi koncovým zariadením prenosu dát a telekomunikačným okruhom. Cieľom tohto zariadenia je premeniť telekomunikačný okruh (určený k prenosu signálu) na dátový okruh, ktorým je možné prenášať dáta. Koncové zariadenia priemyslovej siete alebo systému sa nazývajú DTE (ang. data terminal equipment). Pre komunikáciu medzi DTE a DCE sa väčšinou využíva RS (recommended standards) séria rozhraní, medzi ktoré patria protokoly ako RS-232, RS-422, RS-485.

Recommended Standards - RS

Recommended Standards protokoly patria pod sériovú komunikáciu. To vedie k tomu, že bity daného slova sú vysielané postupne za sebou a prijímač ich musí zase prekódovať na dané slovo.

Pre úspešnú komunikáciu je nutné na oboch stranách definovať:

- prenosovú rýchlosť
- chybovú kontrolu
- potvrdzovanie (handshake)
- formát rámca (start a stop bity)

Pomocou Recommended Standards môže byť definovaný synchrónny aj asynchrónny prenos. RS typicky definuje logické úrovne, maximálnu prenosovú rýchlosť, konektor a jeho zapojenie, podporované signály pre potvrdzovanie, požadované elektrické zaťaženie linky a elektrické vlastnosti sériových prenosov.

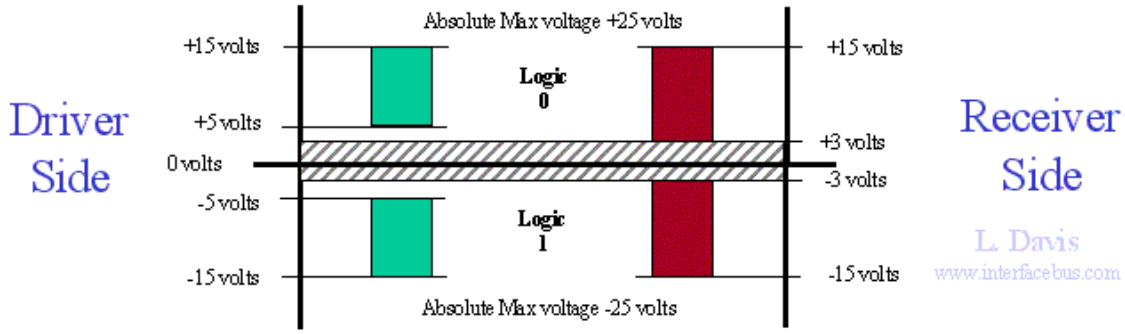
Protokol RS-232

Rozhranie RS-232 je štandard, ktorý bol prvý krát použitý v roku 1960. Toto rozhranie bolo často používané aj v rámci osobných počítačov. V dnešnej dobe je toto rozhranie využívane takmer výhradne v priemysle. V porovnaní s neskoršími štandardmi ako RS-422, RS-485 a Ethernetom má RS-232 nižšiu prenosovú rýchlosť, nižšiu maximálnu dĺžku káblového prepojenia, väčšie napäťové úrovne, väčšie štandardné konektory a nemá možnosť viacbodovej komunikácie. V moderných osobných počítačoch USB nahradilo väčšinu využití pre toto rozhranie.

Napriek všetkým nedostatkom rozhrania RS-232 je vďaka jeho jednoduchosti naďalej využívaný v prostredí v ktorom nám nezáleží na vysokej rýchlosti prenosu dát a ani dĺžke spojení. Jedná sa hlavne o priemyselné prostredie, sieťové zariadenia a vedecké prístroje.

RS-232 podporuje synchrónny aj asynchrónny prenos dat a keďže pre prenos dát využívame oddelené obvody pre každý smer prenosu môžeme dosiahnuť plne duplexnej komunikácie. Napäťové úrovne sa pohybujú od 3-5 do 15 voltov a od -3/-5 do -15 voltov (obr. 2.2) [8][3].

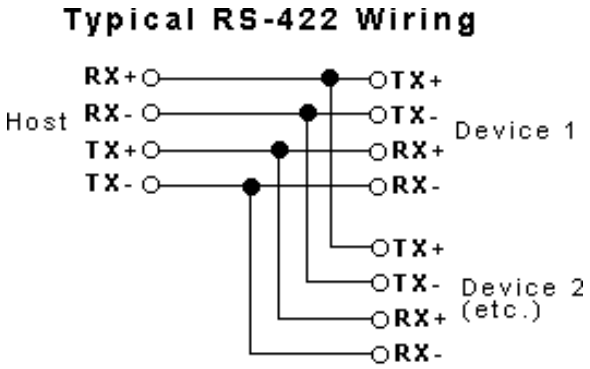
Maximálna dĺžka káblov nie je definovaná v rámci štandardu, no typicky sa dostávame na maximálnu vzdialenosť iba 15 metrov. S využitím nízkokapacitných káblov vieme dosiahnuť vzdialenosť až 300 metov. V prípade potreby využitia väčšej vzdialenosti je nutné využiť iné štandardy.



Obr. 2.2: RS-232 napäťové úrovne [8].

Protokol RS-422

Protokol RS-422 je veľmi podobný protokolu RS-232 a môžu byť programované rovnakým spôsobom. Narozdiel od RS-232 používa protokol RS-422 diferenciálne vodiče. Vďaka tomu dokáže rapídne predĺžiť komunikačnú vzdialenosť medzi zariadeniami a poskytuje vyššiu odolnosť voči rušeniu a napäťovým špičkám. Prenosová vzdialenosť sa tým pádom predlžuje až na 1.5 km a rýchlosť až 10 Mbps. Množstvo zariadení, ktoré je možné pripojiť k zbernici naraz vzrástlo na 32. Ďalším rozdielom oproti RS-232 sú napäťové úrovne, ktoré sa v tomto prípade pohybujú od 0 do ±5 V. Norma RS-422 nedefinuje fyzický konektor.



Obr. 2.3: Typické prepojenie zariadení RS-422 [15].

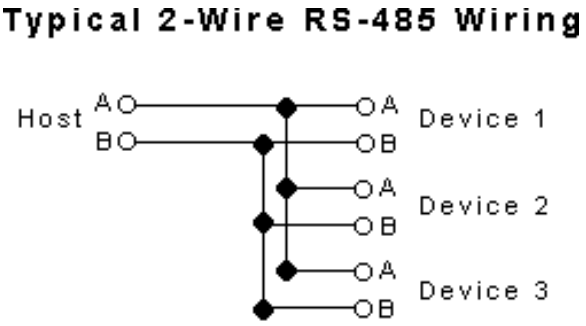
Rovnako ako pre protokol RS-485 platí, že informácia o adresácií zariadenia sa vyskytuje v hlavičke paketu. To znamená, že každé pripojené zariadenie dostane správu a je na zariadení aby rozpoznalo svoju adresu a tak rozhodnúť, či správu zahodiť alebo spracovať.

Protokol RS-485

RS-485 je jedna z najjednoduchších sériových diferenciálnych zberníc. Je natoľko podobná RS-422, že často dochádza k ich zámene. Rovnako ako RS-422 využíva jednu či dve krútené dvoj-linky pre prenos logických úrovní, vďaka tomu je zbernica odolnejšia voči elektromagnetickému rušeniu, ktoré sa indukuje na oboch vodičoch v rovnakej miere a teda nemení rozdiel napätí. Táto vlastnosť je vzhľadom k prúdovým špičkám a využitiu komunikácie pre dlhšie vzdialenosti veľmi dôležitá a zaisťuje prenos s minimom chyb a rušení.

Narozdiel od RS-422 sa na RS-485 môžu nachádzať viaceré zariadenia v pozícií mástra. Zatiaľ čo RS-422 umožňuje iba jedného mástra a zvyšné načúvacie zariadenia. RS-485 je vo veľkej väčšine prepojené iba jedným krúteným párom čo uľahčuje zapojenie. Programovanie RS-485 je zložitejšie než RS-422, pretože vysielanie aj prijímanie väčšinou prebieha po rovnakých kábloch a treba komunikáciu správne načasovať.

Signály RS-485 sú z pravidla označené A a B, niekedy tiež + a - či hot a cold. RS-485, na rozdiel od EIA-422 dovoľuje na spoločnú zbernicu pripojiť viac než 2 zariadenia, podľa štandardu je maximum 32, avšak na trhu sú dostupné radiče, ktoré zaťažia zbernice iba jednou osminou nominálneho zaťaženia a tým umožní navýšenie celkového počtu zariadení pripojených ku zbernici na viac než 200 [2]. Zbernica môže byť implementovaná buď ako plne duplexná. V tom prípade sa využívajú dva krútené páry, alebo ako polo-duplexná, kedy sa využije iba jeden krútený pár a je na zodpovednosti protokolu zaistiť, kedy majú zariadenia vysielať a kedy prijímať. Pri protokole Modbus RTU je využitý polo-duplexný variant. Protokol je založený na master-slave prístupe, to znamená, že všetky transakcie na zbernici sú iniciované zariadením typu master a zariadenia typu slave poslúchajú, čo master pošle a iba pokiaľ sú vyzvané tak odpovedajú [6]. Popisované chovanie odpovedá protokolu popísanému v kapitole 1.6.3. Pre komunikáciu po sériových zberniciach sa u mikropočítačoch obecne využíva periférie U(S)ART umožňujúca (a)synchrónnou komunikáciu s prípadným hardvérovým riadením toku.



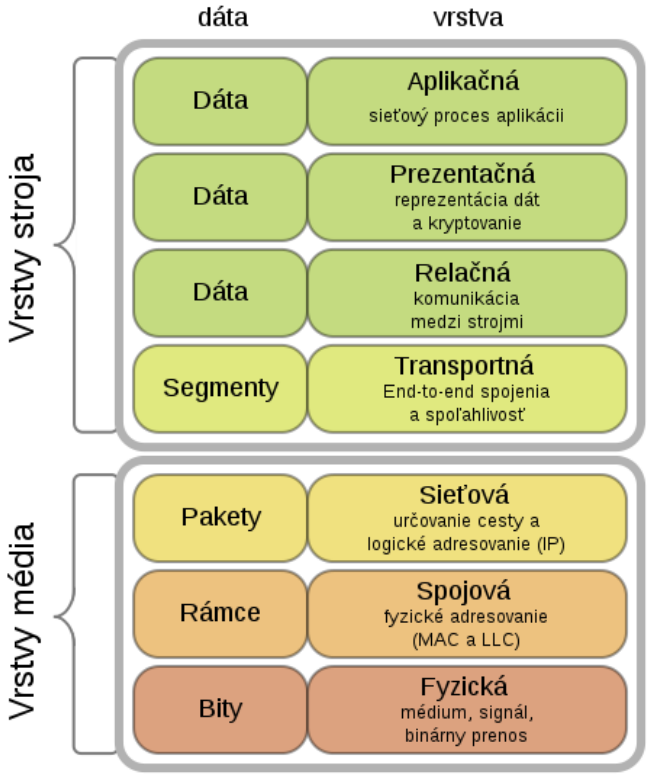
Obr. 2.4: Typické prepojenie zariadení RS-485 [15].

2.4 Model OSI

Referenčný model ISO/OSI vypracovala organizácia s názvom International Organization for Standardization (ISO) ako hlavnú časť snahy o štandardizáciu počítačových sietí. Všetky potrebné sieťové komponenty v zariadení môžu byť zoskupené do OSI modelu. Úlohou referenčného modelu je poskytnúť základ pre vypracovanie noriem pre účely prepojenia systémov. Otvorený systém podľa tohto modelu je abstraktným modelom reálneho otvoreného systému. Táto norma teda nešpecifikuje implementáciu systému, ale uvádza všeobecné princípy sedemvrstvovej sieťovej architektúry.

Model reprezentuje potrebné komponenty hardvéru a softvéru v rámci sieťového zariadenia vo forme siedmich vrstiev: fyzická, spojová, sieťová, transportná, relačná, prezentačná a aplikačná (obr. 2.5). Každá zo siedmich vrstiev vykonáva skupinu jasne definovaných funkcií potrebných ku komunikácií. Pre svoju činnosť využíva služieb susednej nižšej vrstvy. Svoje služby potom poskytuje susednej vyššej vrstve. Podľa referenčného modelu nie je do-

volené vrstvy vynechávať, ale niektoré vrstvy nemusia byť aktívne. Takej vrstve sa potom hovorí nulová alebo transparentná.



Obr. 2.5: Referenčný model ISO/OSI[26]

Komunikáciu potom tvorí:

- komunikácia medzi vrstvami systému, riadi sa pravidlami nazývanými aj rozhranie (ang. interface)
- komunikácia medzi rovnakými vrstvami rôznych systémov. Riadi sa protokolmi.

V závislosti na modeli vstavaných zariadení [[**model?**]] sa fyzická vrstva OSI modelu vzťahuje na hardvérovú vrstvu. Aplikačná, prezentačná a relačná vrstva OSI modelu sa vzťahuje na aplikačnú (softvérovú vrstvu) modelu vstavaných zariadení a zvyšné vrstvy OSI modelu sa najčastejšie vzťahujú na systém softvérovú vrstvu modelu vstavaných zariadení.

Kľúčom k pochopeniu účelu každej vrstvy OSI modelu je to, že tvorenie sieti zariadení nie je iba o prepojení zariadení medzi sebou. Ale tvorenie sieti je primárne o posielaní dát medzi zariadeniami a medzi rôznymi vrstvami každého zariadenia.

Na počiatku teda vznikne požiadavok niektorého procesu v aplikačnej vrstve. Príslušný podsystém požiada o vytvorenie spojenia na prezentačnú vrstvu. V rámci aplikačnej vrstvy je komunikácia s iným systémom riadená aplikačným protokolom. Podsystémy v prezentačnej vrstve sa dorozumievajú prezentačným protokolom. Takto sa dostávame postupne až k fyzickej vrstve, kde sa použije pre spojenie prenosové prostredie. Súčasne sa pri prechode z vyššej vrstvy k nižšej pridávajú k užívateľským (aplikačným) dátam hlavičky jednotlivých vrstiev. Tak dochádza k postupnému zapúzdrovaniu pôvodnej informácie. U príjemcu sa

<table>
<tr><th>OSI</th><th>TCP/IP</th><th colspan="7">Aplikácie a protokoly</th></tr>
<tr><td>7. Aplikačná
6. Prezentačná
5. Relačná</td><td>Aplikačná vrstva</td><td>telnet</td><td>FTP</td><td>TFTP</td><td>SMTP</td><td>RIP</td><td>DNS</td><td>Ostatné</td></tr>
<tr><td>4. Transportná</td><td>Transportná vrstva</td><td colspan="3">TCP</td><td></td><td colspan="3">UDP</td></tr>
<tr><td rowspan="2">3. Sieťová</td><td rowspan="2">Sieťová vrstva</td><td colspan="3" rowspan="2">IP</td><td>ICMP</td><td colspan="3"></td></tr>
<tr><td>ARP</td><td colspan="3">RARP</td></tr>
<tr><td>2. Spojová</td><td rowspan="2">Vrstva sieťového rozhrania</td><td colspan="7" rowspan="2">token ring ethernet iné typy protokolov</td></tr>
<tr><td>1. Fyzická</td></tr>
</table>

Tabuľka 2.1: Prehľad architektúry TCP/IP

potom postupne spracovávajú riadiace informácie jednotlivých vrstiev a vykonávajú sa ich funkcie. [14]

Model OSI v reálnom prostredí

Model OSI je iba odporúčaná pomôcka pre pochopenie sieťových protokolov implementovaných nie len vo vstavaných zariadeniach. Tým pádom nie je vždy pravidlom, že zariadenie implementuje všetkých 7 vrstiev. Tak isto nemusí byť pravdivé to, že máme iba jeden protokol na jednu vrstvu. Funkčnosť jednej vrstvy v OSI modeli môže byť implementovaná v jednom protokole alebo môže byť implementovaná pomocou niekoľkých protokolov a môže zahŕňať niekoľko vrstiev. Jeden protokol môže tiež implementovať funkcionalitu niekoľko vrstiev OSI modelu. Zatiaľ čo poznať model OSI je veľmi nápomocné, niekedy sa namiesto toho využíva skupina protokolov so spoločným názvom v samostatných vrstvách. Napríklad rodina protokolov TCP/IP je tvorená štyrmi vrstvami: aplikačná, transportná, sieťová vrstva a vrstva sieťového rozhrania. Vrstva sieťového rozhrania v sebe zahŕňa dve vrstvi modelu OSI (fyzickú a spojovú). Sieťová vrstva v modeli TCP/IP odpovedá sieťovej vrstve v modeli OSI a transportné vrstvy v oboch modeloch sú identické.

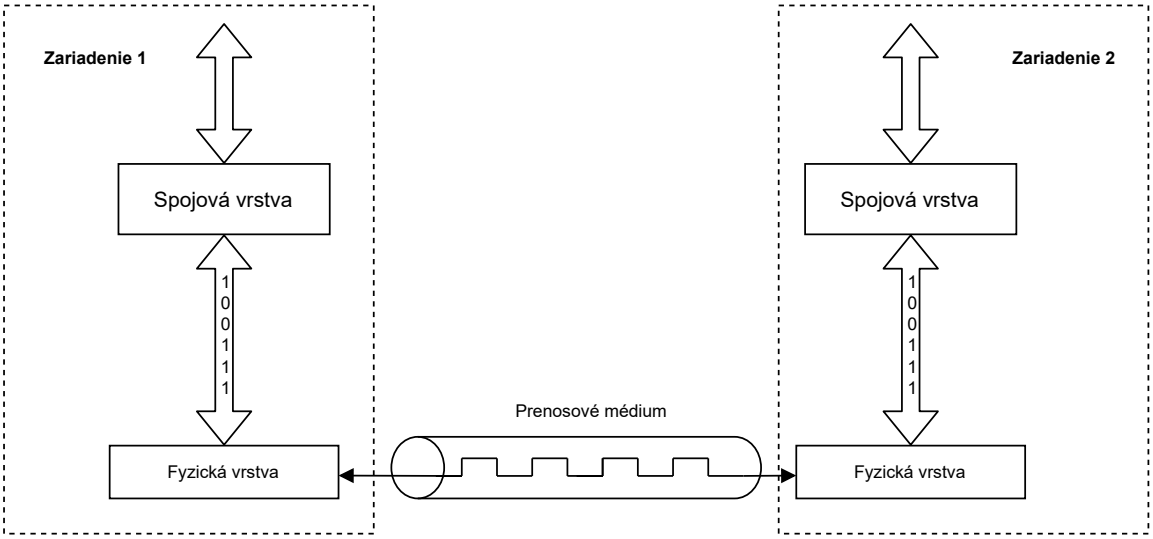
Fyzická vrstva

Fyzická vrstva (ang. Physical - PHY) reprezentuje všetok sieťový hardvér fyzicky prítomný vo vstavanom zariadení. Hardvérové komponenty na fyzickej vrstve pripájajú vstavané zariadenie k nejakej forme fyzického prenášača dát. Na tejto vrstve je dôležitá architektúra a vzdialenosť jednotlivých zariadení.

Fyzická vrstva definuje, spravuje a spracováva dátové signály cez hardvér. Tieto signály sú reprezentované prostredníctvom 1 a 0 prenášaných cez konkrétne komunikačné médium. Fyzická vrstva je zodpovedná za fyzický prenos dátových bitoch po sieťovom médií, ktoré prijíma z vyšších vrstiev vstavaného zariadenia. Ako aj ich prijímanie a odosielanie vyšším vrstvám v rámci vstavaného systému (obr. 2.6).

Spojová vrstva

Spojová vrstva je softvér, ktorý je najbližšie k hardvéru (fyzickej vrstve). Tým pádom obsahuje, mimo iné, akýkoľvek softvér potrebný pre kontrolu hardvéru. Vytváranie mostov (ang. bridging) taktiež prebieha na tejto vrstve pre prepojenie iných sieti k rôznym fyzickým vrstvám.



Obr. 2.6: Diagram toku dát na fyzickej vrstve.

Protokoly na spojovej vrstve, ktoré sa spoliehajú na istú fyzickú vrstvu môžu byť limitované na prepojenie s určitým prenosovým médiom (napr. PPP cez RS-232 alebo PPP cez Bluetooth). Protokoly na spojovej vrstve môžu byť prepojené s veľmi rôznymi médiami, ak existuje vrstva, ktorá simuluje originálne médium pre ktoré bol protokol vytvorený.

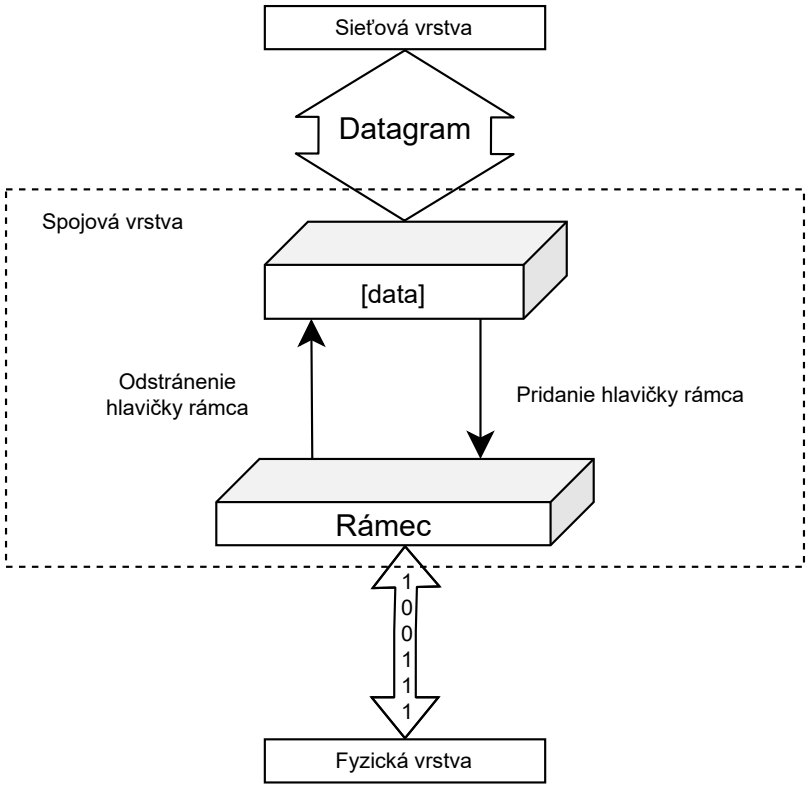
Spojová vrstva je zodpovedná za prijímanie dátových bitov z fyzickej vrstvy a formátovanie ich do skupín nazývaných rámce (ang. frames). Rôzne spojové štandardy majú rôzne formáty rámcov a rôzne definície. Spoločne majú ale to, že čítajú pole bitov týchto rámcov a zaisťujú aby boli prijaté celé rámce a nie iba nejaká ich časť. Zároveň kontrolujú rámce a zaisťujú že tieto rámce sú bez chýb a že sú mierené na toto zariadenie (zhoduje sa fyzická adresa zariadenia). Ak je rámec myslený pre toto zariadenie, potom sú všetky spojové hlavičky odstránené a zvyšné dáta (ang. datagram) sú odoslané na sieťovú vrstvu (obr 2.7).

Sieťová vrstva

Protokoly na sieťovej vrstve sú, ako aj protokolu na spojovej vrstve, implementované na vrstve systémového softvéru. Narozdiel od protokolov na spojovej vrstve sú tieto protokoly po väčšine nezávisle na hardvéri na ktorom bežia. Sú ale závislé na implementácií spojovej vrstvy.

V OSI sieťových vrstvách sa siete dajú rozdeliť na menšie podsiete, nazývané segmenty. Zariadenia môžu komunikovať v rámci segmentu pomocou ich príslušných fyzických adries. Narozdiel od toho zariadenia, ktoré sú v rozdielnych segmentoch, musia komunikovať pomocou inej adresy. Táto adresa sa nazýva sieťová adresa (ang. network address). Zatiaľ čo konverzia medzi fyzickou adresou a sieťovou adresou môže prebiehať v protokoloch na spojovej vrstve (ARP, RARP, atď.), sieťová vrstva tiež dokáže konvertovať tieto adresy. Sieťová vrstva zároveň dokáže prideliť sieťové adresy. Spojová vrstva dokáže cez schému adries riadiť tok datagramov a akékoľvek smerovanie datagramov z aktuálneho zariadenia na iné zariadenie (obr. 2.8).

Podobne ako spojová vrstva, ak sú dáta mierené na aktuálne zariadenie, tak sa odstránia všetky hlavičky z datagramu a ostávajúce dáta, nazývané paket (ang. packet). Tento paket je potom odoslaný na vyššiu vrstvu (transportnú). Rovnaký je aj postup opačným smerom



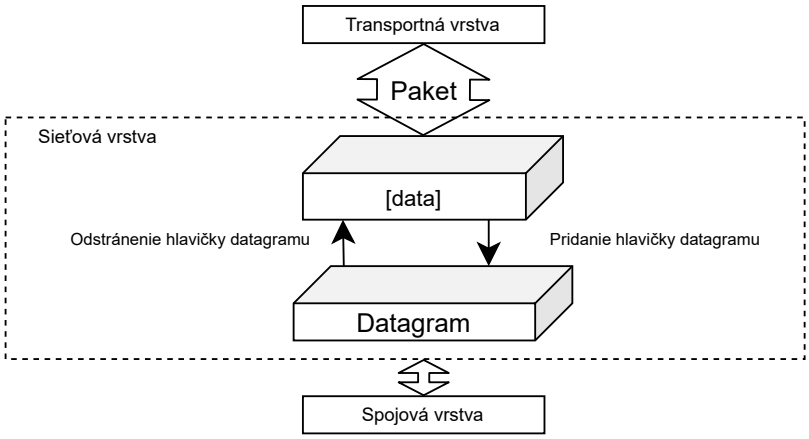
Obr. 2.7: Diagram toku dát na spojovej vrstve.

kedy, ak zariadenie odosiela paket, sieťová vrstva k nemu pridá hlavičky sieťovej vrstvy a odosiela datagram nižšej vrstve (spojovej).

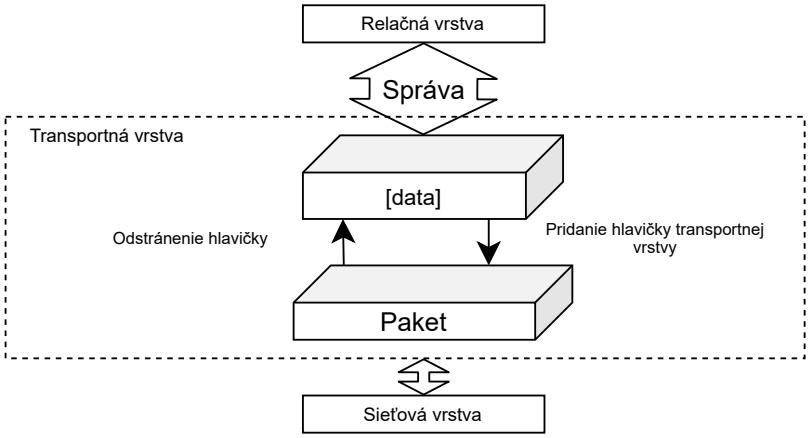
Transportná vrstva

Protokoly transportnej vrstvy sú špecifické pre protokoly na sieťovej vrstve. Typicky sú zodpovedné za vytvorenie a zahodenie spojenia medzi dvoma špecifickými zariadeniami. Tento typ komunikácie sa nazýva dvojbodová komunikácia (ang. point-to-point communication). Protokoly na tejto vrstve umožňujú viacero vysokoúrovňových aplikácií, ktoré bežia na zariadení, spojenie pomocou dvojbodovej komunikácie na iné zariadenia. Niektoré transportné vrstvy zaisťujú aj spoľahlivé spojenie medzi zariadeniami tým. Toto spoľahlivé spojenie je dosiahnuté tým, že zaisťujú odoslanie a prijímanie paketov v správnom poradí, sú odosielané určitou rýchlosťou (ang. flow control) a že dáta v paketoch nie sú poškodené. Protokoly na transportnej vrstve môžu zaistiť aj potvrdenie pre iné zariadenia po prevzatí paketu. Naopak ak došlo ku chybe pri prevzatí, môžu zažiadať o znovu-odoslanie paketu.

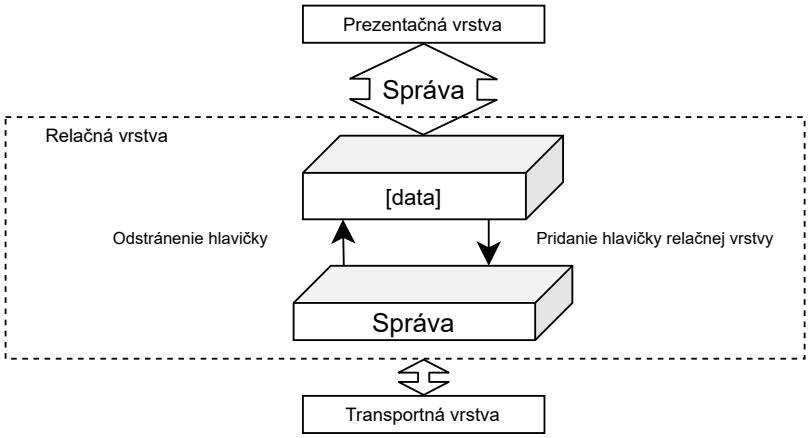
Tak isto ako pri predchádzajúcich vrstvách, aj pri transportnej vrstve dochádza k odstráneniu transportných hlavičiek z paketov. Zvyšné dátové polia z jedného alebo viacerých paketov sú potom poskladané do iných paketov tiež nazývaných aj správy (ang. messages). Tieto správy sú potom odoslané do vyšších vrstiev. Správy, ktoré prídu z vyšších vrstiev sú naopak rozdelené do viacerých paketov (ak sú príliš dlhé) a po pridaní patričných hlavičiek sú odoslané do nižších vrstiev pre ďalšie spracovanie (obr. 2.9).



Obr. 2.8: Diagram toku dát na sietovej vrstve.



Obr. 2.9: Diagram toku dát na transportnej vrstve.



Obr. 2.10: Diagram toku dát na relačnej vrstve.

Relačná vrstva

Spojenie medzi dvoma sieťovými aplikáciami na dvoch rôznych zariadeniach sa nazýva relácia (ang. session). Zatiaľ čo transportná vrstva riadi dvojbodovú komunikáciu medzi zariadeniami s viacerými aplikáciami, management relácií je riadený na relačnej vrstve. Spravidla sú reláciám pridelené porty a protokol na relačnej vrstve musí rozdeliť a riadiť relačné dáta, regulovať tok dát pre každú reláciu a ošetriť hocijakú chybu, ktorá mohla nastať v aplikácií, ktorá s reláciou spolupracuje. Relačná vrstva musí taktiež zaistiť bezpečnosť relácie. A to napríklad či s reláciou komunikuje správna aplikácia.

V prípade, že relačná vrstva prijme správu z nižšej vrstvy, potom sú všetky relačné hlavičky odstránené a zvyšné dáta sú odoslané do vyššej vrstvy. K správam, ktoré prídu z vyššej vrstvy naopak pridá relačnú hlavičku a správu následne odošle do nižšej vrstvy pre ďalšie spracovanie (obr. 2.10).

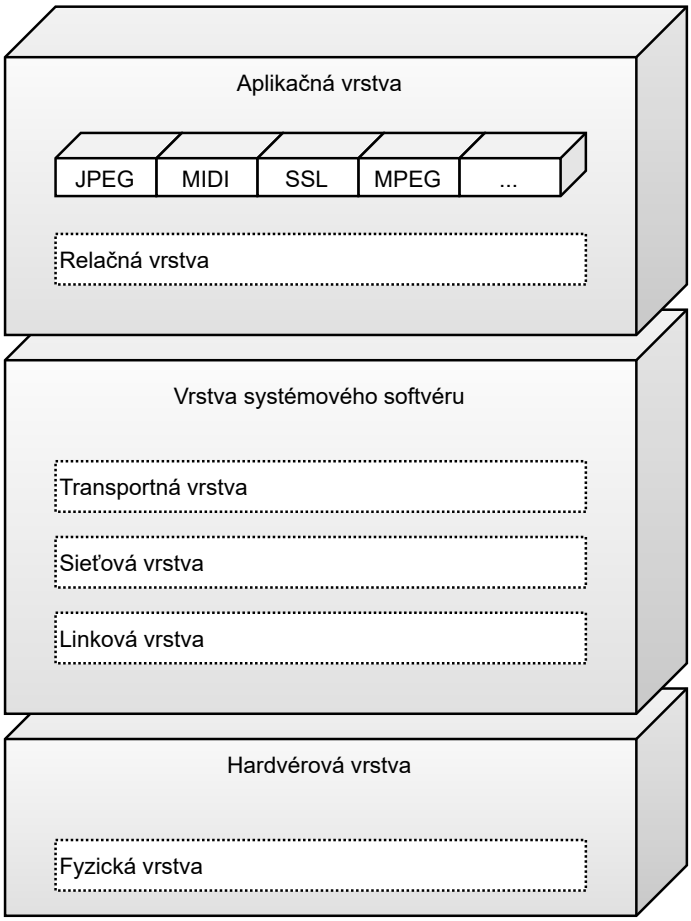
Prezentačná vrstva

Protokoly na prezentačnej vrstve sú zodpovedné za preklad dát na formát, ktorý aplikácie na vyššej úrovni vedia úspešne spracovať. V prípade odoslania dát pre iné zariadenie dochádza na tejto vrstve k prekladu aplikačného formátu na generický pre odoslanie a ďalšie spracovanie. Spravidla dochádza ku kompresií/dekompresii, šifrovaní/dešifrovaní dát a prevod protokolu/znakov práve na tejto vrstve. Vzhľadom ku modelu vstavaných systémov sú protokoly na prezentačnej vrstve implementované v sieťových aplikáciach, ktoré nájdeme v aplikačnej vrstve (obr. 2.11).

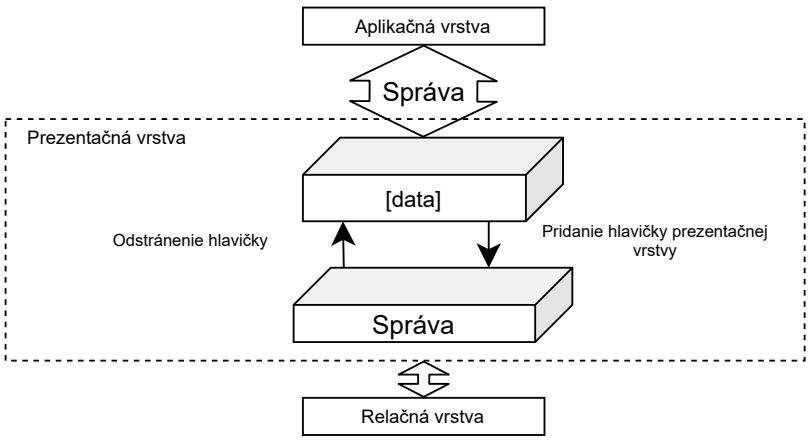
Prezentačná vrstva v prípade prijatia správy z nižšej vrstvy odstráni všetky prezentačné hlavičky a zvyšné dátové polia odošle so vyššej vrstvy. K správam, ktoré prijme z vyšších vrstiev pripojí prezentačnú hlavičku a odošle ich do nižšej vrstvy pre ďalšie spracovanie (obr. 2.12).

Aplikačná vrstva

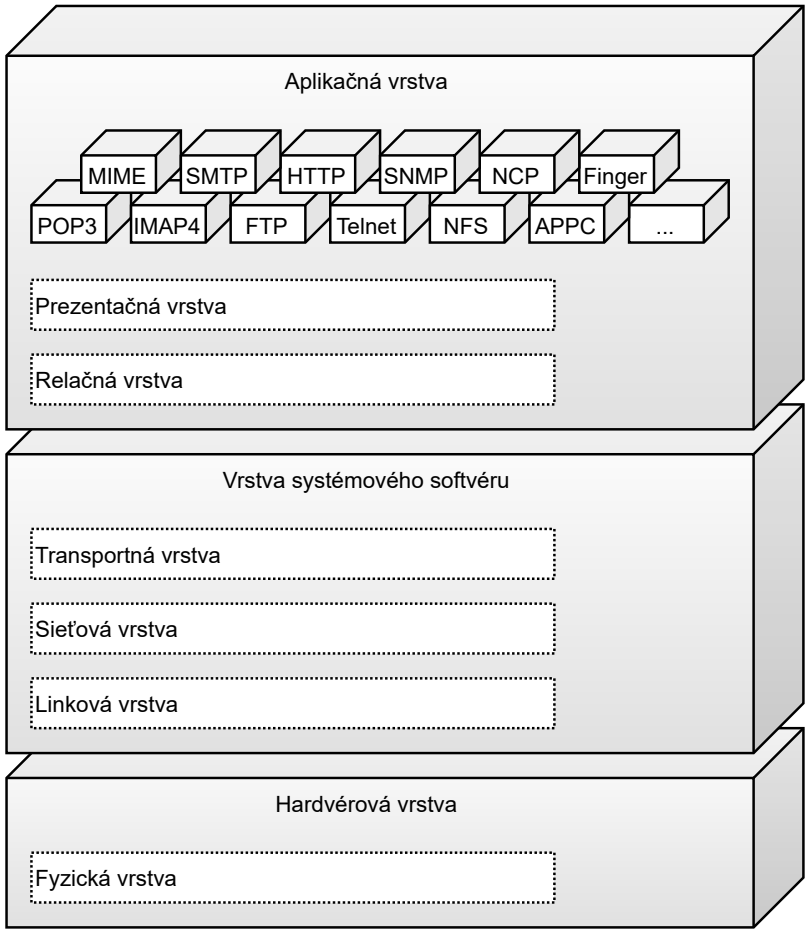
Zariadenie vytvára spojenie s iným zariadením na aplikačnej vrstve prostredníctvom niektorého z aplikačných protokolov (obr. 2.13). Protokoly na aplikačnej vrstve sú buď používané priamo používateľmi ako sieťové aplikácie alebo sú implementované v nejakej aplikácií.



Obr. 2.11: Protokoly prezentačnej vrstvy v modeli vstavaných systémoch.



Obr. 2.12: Diagram toku dát na prezentačnej vrstve.



Obr. 2.13: Protokoly aplikačnej vrstvy v modeli vstavaných systémoch.

Jeden z industriálne najpoužívanejších aplikačných protokolov používaných v rôznych zariadeniach výrobných procesov je protokol Modbus.

2.5 Aplikačný protokol Modbus

Modbus je jeden z najviac používaných industriálnych komunikačných protokolov v industriálnej automatizácií a kontrolných systémoch nízkej až strednej náročnosti [22]. Dôvodom je, že tieto systémy nemajú veľké nároky na komunikáciu čo sa týka prenosu dát ani na množstvo dát [22]. Umožňuje to teda vytvárať jednoduché a ekonomicky prijateľné hardwarové či softwarové prevedenie.

Tým že modbus je s nami už dlhú dobu je dobre otestovaný a zdokumentovaný, čo prináša veľa výhod. Obzvlášť čo sa týka rôznych jeho vylepšení a nadstavieb ako je napríklad jeho implementácia na bezdrôtových sieťach [7] alebo TCP/IP sieťach [9].

Modbus-RTU siete sú implementované na sériovej zbernici, čo prináša niekoľko hlavných problémov: (1) viac prenesených informácií po zbernici produkovaných z nových zariadení a rastúcim množstvom dát, ktoré sú požadované pre presnejšie a detailnejšie informácie o systéme produkuje viac chýb po takejto zbernici a (2) vyššie elektromagnetické rušenie.

Riešenie pre prvý problém je celkom jednoduché. Treba viac optimalizovať množstvo prenesených informácií z jednotlivých zariadení. Druhý problém nie je tak jednoducho riešiteľný, keďže je priamym následkom zvyšovania počtu zariadení v priestore ako aj voľbou architektov ťahať komunikačne káble blízko vysokonapäťových alebo zhlukovať veľké množstvo komunikačných liniek vedľa seba v jednej lište. Keďže jediná možnosť ako opraviť komunikačný šum v týchto sieťach je znovu odoslanie celej správy, dostávame sa k ďalšiemu problému preťaženia siete. Riešenie takéhoto problému potom spočíva iba vo vylepšení komunikačných kanálov alebo nahradenia celého Modbus-RTU protokolu niečim úplne iným.

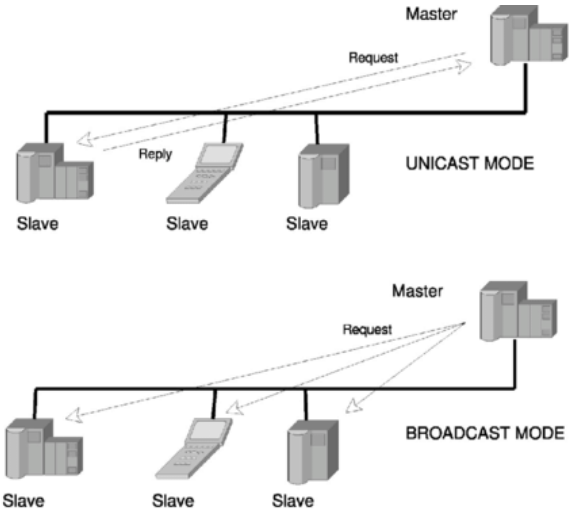
V dnešnej dobe je tento protokol nahradzovaný Modbus TCP/IP protokolom, ktorý je takmer bez komunikačných chýb vďaka využitiu TCP/IP [18]. Tento protokol ale vyžaduje špecializované nástroje, ktoré už v sebe zahŕňajú TCP/IP a je teda nekompatibilný s Modbus-RTU protokolom [25]. Oproti Modbus RTU je Modbus TCP oveľa náročnejší na spracovanie a vyžaduje zložitejší hardvér a teda využíva sa často skôr ku prepojeniu zariadení typu brána (ang. gateway), ku ktorej je pripojených niekoľko zariadení po sériovej zbernici. Práve v tomto smere zohráva veľkú rolu zariadenie ako prevodník medzi týmito protokolmi.

Modbus RTU

Modbus-RTU je najviac používaná forma Modbus komunikačného protokolu pre prenášanie industriálnych dát. Modbus-RTU špecifikáciu sprístupňuje Modbus organizácia: the MODBUS application protocol, ktorá špecifikuje master-slave konfiguráciu pre dátovú vrstvu, ASCII (American Standard Code for Information Interchange) a RTU (Remote Terminal Unit) pre komunikačné módy. Naviac špecifikuje ich implementáciu prostredníctvom asynchrónnej zbernice EIA/TIA-485 (RS485) a EIA/TIA-232 (RS232) zbernice. Najviac používaná metóda komunikácie prebieha cez EIA/TIA-485 dvojlinku.

Master-slave architektúra dovoľuje iba jedného mástra na sieti. Pre identifikáciu slave zariadenia v odosielanom pakete využíva jeden bajt, pričom adresy od 248 do 255 sú rezervované. Čo znamená, že dovoľuje komunikovať medzi 1 až 247 slave zariadeniami. Iba master môže zahájiť komunikáciu. A to tak, že odošle správu na slave zariadenia formou unicastu alebo broadcastu ako je ukázané na obrázku 2.14

Dáta sú po sériovej zbernici odosielané vo forme ôsmich bitoch alebo jedného bajtu. Rýchlosť tejto komunikácie musí byť rovnaká pre všetky zariadenia na sieti, no nie je v protokole špecifikovaná a neexistuje žiadny spôsob automatickej výmeny informácie o komunikačnej rýchlosti. Môže sa pohybovať od 1200 až po 115200 bps[3]. No väčšina zariadení nepodporuje rýchlosť vyššiu ako 38400 bps. Preto, aby sa predišlo chybám na zbernici v dôsledku nepodporovanej rýchlosti alebo zahltenia komunikácie v dôsledku príliš nízkej rýchlosti, sa typicky využíva na zbernici RS485 rýchlosť 9600 až 19200 bps [20]



Obr. 2.14: Master/slave komunikácia formou unicastu a broadcastu [25]

Reprezentácia dat

V Modbus-RTU protokole sú definované iba 2 dátové typy tzv. cievky (ang. coils) a registre (ang. registers). Cievky sú reprezentované jedno-bitovými hodnotami. Cievky môžu nadobúdať hodnoty ON (1) a OFF(0). Niektoré cievky reprezentujú digitálne vstupy, čo môže byť napríklad jedno bitový vstup zo senzora. Iné reprezentujú digitálny výstup, čo môže byť napríklad nejaký akčný člen (ang. actuator). Registre sú reprezentované 16 bitovými hodnotami bez znamienka, a teda môžu obsahovať hodnoty od 0 do 65535 (0 až FFFF v šestnástkovej sústave). Neexistuje žiadnu možnosť ako odoslať záporné hodnoty, hodnoty vyššie ako 65535 ani reálne hodnoty ako 200.25. Takéto hodnoty je nutné zakódovať do 16 bitového čísla.

Tak ako cievky, registre sú rozdelené do dvoch skupín. Vstupné a uschovávacie registre. Vstupné registre majú podobnú funkciu ako vstupné cievky, a to reprezentovať stav vstupného zariadenia(senzora), ktorý nie je možne reprezentovať iba jedným bitom. Jedná sa napríklad o termické senzory. V dnešnej dobe sa modbus zariadenia nepoužívajú pre priame napojenie na vstupno-výstupné zariadenia, preto rozdelenie registrov do dvoch skupín stráca význam a väčšinou vstupné a uschovávacie registre brané ako jedna skupina.

Pôvodne boli uschovávacie registre zamýšľané ako dočasné úschovne dát pre zariadenia ako modbus kontroléry. V dnešnej dobe fungujú skôr pre uschovávanie programových dát.

[3]bitov za sekundu

Modbus pakety majú iba funkciu odosielania dát, nepodporujú žiadnu komunikáciu ako preposielanie metadát o jednotlivých uložených dátach ako napríklad jednotky či popis. Na čo treba myslieť pri návrhu systému s využitím modbus protokolu.

Adresácia zariadení a identifikácia staníc

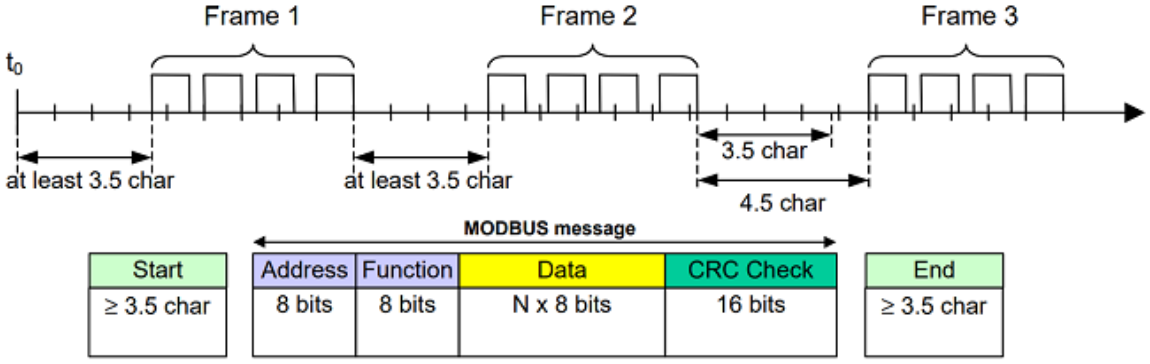
Štandardný Modbus-RTU protokol adresuje zariadenia v rozmedzí od 1 po 255 s tým, že 0 je obsadená pre broadcast na všetky zariadenia a s výhradne iba pre zápis. No broadcast sa využíva minimálne, pretože neexistuje žiaden mechanizmus pre potvrdenie správy jednotlivými zariadeniami na sieti. Klasická implementácia RS485 dovoľuje pripojenie iba 32 zariadení, no niektoré implementácie nám dovoľujú toto číslo značne rozšíriť.

Jednotlivé zariadenia na sériovej zbernici sú identifikované podľa čísla stanice, ktoré predchádza základnú štruktúru správy (počet staníc nie je softwarovo limitovaný). Následne sú jednotlivé slave zariadenia za stanicou identifikované číslom od 1 po 255

Kódovanie dát

Kódovací mechanizmus bližšie špecifikuje usporiadanie bitov v kontrolných a dátových hodnotách. Odosielateľ aj prijímateľ musí komunikovať s využitím rovnakého kódovania. Modbus-RTU špecifikácia zahŕňa dva rôzne spôsoby kódovania: ASCII a RTU. [20]

RTU kódovanie je oveľa častejšie, popisuje iba štandardné kódovanie tzv. "Big-endian". Čo znamená, že jednotlivé 16 bitové hodnoty sú kódovane s najvýznamnejším bitom ako prvým. Decimálna 8-bitová hodnota ako napríklad 41 by bola teda zakódovaná v binárnej podobe ako 0010 1001. 16-bitová hodnota ako napríklad 300 by mala podobu 0000 0001 0010 1100.



Obr. 2.15: Modbus RTU rámec [17].

Kapitola 3

Priemyselný Ethernet

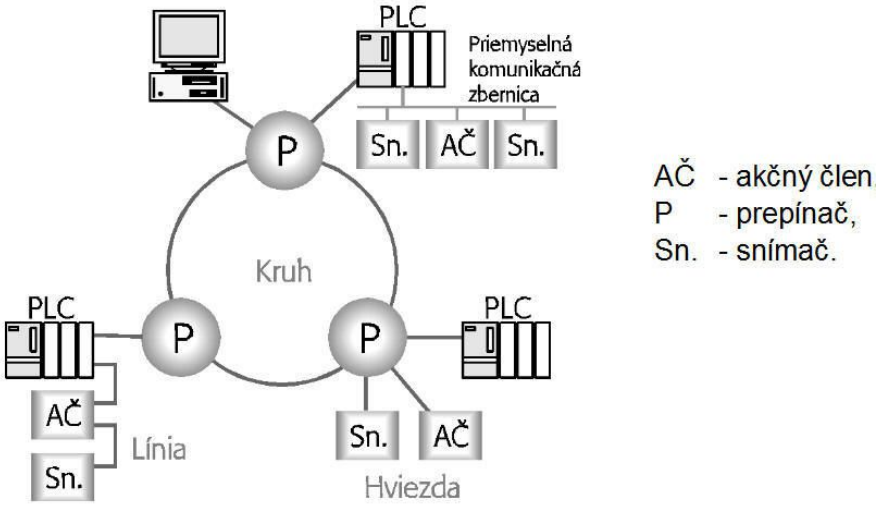
Štandard IEEE 802.3 špecifikuje lokálnu sieť (LAN) Ethernet. Pojem „lokálna sieť“ je možné interpretovať ako počítačovú sieť vytvorenú z obmedzeného počtu zariadení, ktoré sa nachádzajú v ohraničenom priestore. štandardy lokálnych sietí špecifikujú len fyzickú a linkovú vrstvu vo vrstvovom modeli OSI. Z toho vyplýva, že IEEE 802.3 (Ethernet) určuje vlastnosti použitých komunikačných rozhraní (napr. elektrických alebo optických) a metódu riadenia prístupu na prenosové médium. Ethernet je v súčasnosti jedna z najpoužívanejších LAN technológií. Prepája viac ako 85 percent počítačov prepojených cez LAN [19].

Štandard lokálnej siete Ethernet nebol navrhnutý pre používanie v priemysle na spodných úrovniach riadenia a nezohľadňuje potrebu prenosu údajov v reálnom čase. Ak zariadenie na sieti chce odoslať údaje inému zariadeniu, musí sledovať či momentálne nevysiela údaje iné zariadenie a ak nie, potom začne vysielať. Môže sa stať, že niekoľko zariadení začne vysielať údaje súčasne. Vznikne kolízia, ktorú zariadenia zaregistrujú, prestanú vysielať a začnú znova vysielať po zvolenom časovom intervale. Ku kolízii principiálne môže dôjsť aj potom. V zásade nie je možné predvídať ako dlho budú jednotlivé zariadenia čakať na prístup na prenosové médium (kým začnú úspešne vysielať údaje) čo znamená, že prístupová metóda nie je deterministická.

Zariadenia na sieti môžu byť prepojené na základe rôznych topológií. Na obrázku 3.1 je príklad prepojenia zariadení v sieti s priemyselným Ethernetom.

V prvých Ethernetovských sieťach sa používala topológia zbernica, v ktorej bol ako prenosové médium použitý koaxiálny kábel. V takomto prípade sa jednotlivé zariadenia pripájajú paralelne ku spoločným vodičom. Pri prenose údajov medzi prepojenými zariadeniami môže dochádzať ku kolíziám. Takéto riešenie sa stále používa v niektorých sieťach Priemyselného Ethernetu.

Ďalšou, v súčasnosti najrozšírenejšou topológiou v oblasti lokálnych sietí, je hviezda – respektíve strom, kedy sú komunikujúce zariadenia prepojené prepojovacím zariadením, ktoré môže byť implementované na úrovni fyzickej, alebo linkovej vrstvy. V prípade, že je toto prepojovacie zariadenie implementované na úrovni fyzickej vrstvy, označuje sa ako rozbočovač (anglicky „hub“) a slúži na prenos signálov (elektrických alebo optických) medzi odosielateľom a potenciálnymi príjemcami správy. Ak sú komunikujúce zariadenia prepojené na to určeným zariadením pracujúcim na úrovni linkovej vrstvy, tak medzi týmito zariadeniami nemôže dochádzať ku kolíziám. Prepojovacie zariadenie pracujúce na úrovni linkovej vrstvy sa označuje prepínač (anglicky „switch“). Prepínače je možné použiť nielen na prepojenie jednotlivých zariadení, ale aj na prepojenie častí siete, ktoré nemusia mať topológiu hviezda, ale môžu mať inú topológiu - napríklad zbernica. [5]



Obr. 3.1: Príklad prepojenia zariadení v sieti s priemyselným Ethernetom [5]

Výhoda priemyselného Ethernetu spočíva v tom, že jednotlivé organizácie môžu naďalej používať všetky tradičné nástroje, ktoré používali doteraz. No môžu fungovať na oveľa komunikovať po oveľa efektívnejšej infraštruktúre.

Aj keď technológia priemyselného Ethernetu pochádza z tradičného Ethernetu, nájdu sa tu určité rozdiely. Priemyselný Ethernet je stavaný na použítie v oveľa náročnejších okolitých podmienkach. Musí sa vedieť vyporiadať s flexibilným počtom zariadení, rôznym typom komunikácie, predvídateľnou komunikáciou v reálnom čase atď. [19].

3.1 Metódy prenosu údajov

Používané metódy prenosu údajov nie sú závislé na fyzickej a logickej topológii siete. Používajú sa tri metódy: Klient/Server, Publisher/Subscriber a Producent/Konzument. Každá z týchto metód má svoje výhody aj nevýhody a takisto aj svoju oblasť použitia

Klient/Server

Metóda klient/server umožňuje prenos medzi dvoma komunikačnými partnermi, kedy prenášané údaje nepožaduje žiadne iné zariadenie. Pri tejto metóde musí zariadenie požadujúce údaje, t.j. klient vyslať požiadavku zariadeniu, ktoré je zdrojom týchto údajov, čo je server. Po tejto požiadavke server odošle údaje zariadeniu, ktoré vystupuje vo funkcii klienta. Následne sa spojenie medzi zariadením klient a server zruší a pri opätovnom prenose údajov je ho nutné znova nadviazať.

Táto metóda je vhodná na prenos údajov medzi dvomi zariadeniami, napríklad riadiacimi jednotkami (PLC a iné). Nie je príliš vhodná na prenos údajov zo snímačov, pretože pri prenose každej hodnoty – v každom komunikačnom cykle – je nutné aby zariadenie vo funcii klienta (napr. PLC) vyslalo požiadavku zariadeniu vystupujúcemu vo funkcii servera (t.j. snímaču). Ak by hodnotu z daného snímača bolo potrebné preniesť do viacerých zariadení, tak by zaťaženie siete ešte vzrástlo.

Publisher/Subscriber

Metóda publisher/subscriber umožňuje príjem prenášaných údajov viacerými partnermi komunikácie súčasne. Zariadenie vysielajúce údaje obsahuje zoznamy zariadení, ktorým sú tieto údaje určené. Každé zariadenie, ktoré chce prijímať údaje od zariadenia publisher, ho musí jednorazovo o to požiadať a až v dôsledku tejto žiadosti si ho zariadenie publisher zaradí do svojho zoznamu.

Metóda je vhodná na cyklický prenos časovo kritických údajov v regulačných slučkách.

Pri odosielaní správ sa môže použiť skupinová adresácia, na základe ktorej sú správy prijímané len tými účastníkmi komunikácie, ktorým je správa určená, alebo je správa opakovane odosielaná všetkým zariadeniam, ktoré o údaje v správe požiadali.

Producent/Konzument

V metóde producent/konzument sa používa pri prenose správ skupinová adresácia, podobne ako v metóde pubslisher/subscriber. Rozdiel je však v tom, že žiadny partner komunikácie nemá vytvorený zoznam komunikačných partnerov, ktorým vysiela, alebo od ktorých prijíma údaje.

Pri prenose sú údaje označené špeciálnym identifikátorom. Prvý konzument daných údajov odošle žiadosť o tieto údaje ich producentovi. Producent a konzument sa dohodnú na skupinovej adrese a identifikátore pre tieto údaje. Producent bude odteraz odosielať dané údaje na dohodnutú skupinovú adresu. V prípade, že o tieto údaje má záujem aj iný konzument, tento požiada o pridelenie skupinovej adresy a identifikátora údajov od producenta, alebo iného konzumenta.

Metódy publisher/subscriber a producent/konzument znižujú zaťaženie komunikačného systému vychádzajúceho z Ethernetu, avšak niekedy môžu klásť zvýšené nároky na pripojené zariadenia. Ide predovšetkým o vysielanie a príjem správ s globálnou a skupinovou adresáciou, ktoré nemusia byť schopné spracovať všetky zariadenia na sieti.

3.2 Rozhranie pre prepojenie fyzickej vrstvy Ethernetu

Pre prepojenie fyzickej vrstvy zariadenia s jeho MAC (Medium Access Control) rozhraním je potrebné využiť vysoko-rýchlostného prepojenia, ktoré je štandardizované v rôznych rozhraniach. MAC na zariadení je zodpovedné za správu hardvéru, ktorý komunikuje po komunikačnom médiu. Takéto komunikačné médium môže byť dneska už veľmi zaužívané metalické prepojenie, optické alebo dokonca aj bezdrôtové prepojenie. Pre takéto prepojenie využívame zaužívané štandardy medzi ktoré patrí napríklad MII (Media Independent Interface), RMII (Reduced Media Independent Interface) alebo ich varianty GMII (Gigabit Media Intependent Interface) a RGMII (Reduced Gigabit Media Independent Interface).

3.2.1 Media Independent Interface

Media independent interface (MII) bolo pôvodne definované ako štandardné rozhranie pre pripojenie k Fast Ethernet (100Mbps), riadeniu prístupu k médiám (MAC[1]), bloku s PHY[2] čipom. MII je štandardizovaný IEEE 802.3u a pripojuje rôzne typy PHY k MAC. Byť nezávislý na médiu znamená, že sa dajú použiť rôzne zariadenia PHY k pripojeniu k rôznym

[1] MAC - pod vrstva druhej vrstvy modelu OSI - linková
[2] PHY - prvá vrstva modelu OSI - fyzická

mediam (krútená dvojlinka, optické vlákno), bez toho aby bolo nutné prepracovať alebo vymeniť hardware MAC. Dá sa teda použiť akékoľvek MAC s akýmkoľvek PHY, nezávisle na médiu pre prenos sieťového signálu.

Sieťové dáta na rozhraní sú orámcované štandardom IEEE Ethernet. Skladajú sa z preambule, oddelovača počiatočných rámcov, Ethernetových hlavičiek, dát špecifických pre protokol a kontroly cyklickej redundancie (CRC). Pôvodná MII prenáša sieťové dáta pomocou 4bitových nibbles[3] v každom smere (4 prenosové dátové bity, 4 prijímajúce dátové bity). Dáta sú taktované na 25MHz, aby sa dosiahlo priepustnosti 100 Mbps.

Pôvodný dizajn MII bol rozšírený, aby podporoval znížené signály a zvýsené rýchlosti. Medzi súčasne používané varianty patrí aj napríklad Reduced Media-independent interface (RMII).

Toto rozhranie vyžaduje 18 signálov, z ktorých je možné zdielať medzi viacero PHY iba dva. To predstavuje veľký problém v prípade ak jedno zariadenie má obsahovať viacero PHY modulov.

Vysielacie signály

Pre vysielanie signálov po MII sa využíva sedem signálov zobrazených v tabuľke 3.1. Vysielacie hodiny sú volne bežiace hodiny generované PHY na základe rýchlosti linky (25MHz pre 100Mbit). Zvyšné vysielacie signály sú riadené MAC synchrónne na nábežnej hrane TX_CLK. Toto usporiadanie umožňuje fungovanie MAC bez nutnosti poznania konkrétnej rýchlosti spojenia. Signál povolenia prenosu je udržovaný vysoko počas behu prenosu rámca a nízky v prípade, že je vysielač nečinný.

Chyba prenosu môže byť vyvolaná po dobu jedného alebo viacerých časových období behom prenosu rámca. To môže byť (v prípade zistenia chyby pri vysielaní) použité k prerušeniu rámca po zahájení prenosu. Tento signál sa môže použiť aj ako vyžiadanie prepnutia PHY do režimu zníženej spotreby (nízky signál na TX_EN a vysoký na TX_ER).

Signál	Popis
TX_CLK	Vysielacie hodiny
TXD0	Prenos dátového bitu 0
TXD1	Prenos dátového bitu 0
TXD2	Prenos dátového bitu 0
TXD3	Prenos dátového bitu 0
TX_EN	Povolenie prenosu
TX_ER	Chyba prenosu

Tabuľka 3.1: Vysielacie signály pre MII

Prijímacie signály

Pre prijímanie signálov po MII sa využíva deväť signálov zobrazených v tabuľke 3.2. Prvých sedem signálov je analogických signálom vysielača, okrem signálu RX_ER, ten už nie je voliteľný, ale slúži k označeniu situácie, kedy nejde prijímaný signál dekódovať na platné dáta. Podobne ako v prípade vysielača sa používa signál na RX_ER a RX_DV ku indikácií, že partner linky je v režime nízkej spotreby EEE.

[3]nibble - štvorbitová agregácia alebo polovica oktetu.

Signály CRS a COL sú asynchrónne s hodinami príjmu a majú zmysel iba v poloduplexnom režime. Nosný signál je v stave „vysoký“ v prípade, že sa vysiela, prijíma alebo je médium vnímané ako používané. Pokiaľ je zistená kolízia, COL sa zvýši po dobu trvania kolízie.

Signály riadenia

Okrem signálov vysielania a prijímania MII obsahuje aj dva signály pre riadenie toku. Sú to signály MDIO a MDC. Signál MDIO v sebe nesie údaje o správe zatiaľ čo MDC označuje hodiny pre správu. Tieto dva signály tvoria synchrónne sériove dátové rozhranie podobne I2C[4]. Zároveň sú to jediné dva signály, ktoré je možné zdielať medzi viacero PHY.

Signál	Popis
RX_CLK	Prijímacie hodiny
RXD0	Prenos dátového bitu 0
RXD1	Prenos dátového bitu 1
RXD2	Prenos dátového bitu 2
RXD3	Prenos dátového bitu 3
RX_DV	Platné dáta príjmu
RX_ER	Chyba prenosu
CRS	Nosný signál
COL	Detekcia kolízie

Tabuľka 3.2: Prijímacie signály pre MII

3.2.2 Reduced Media Independent Interface

Redukované MII je štandard, ktorý bol vyvinutý za účelom zníženia počtu signálov potrebných k prepojeniu PHY k MAC. Zníženie počtu pinov znižuje náklady a zložitosť sieťového hardwaru. Hlavne v kontexte mikrokontrolérov s integrovaným MAC, FPGA, multiportovými prepínačmi alebo opakovačmi a čipovými sadami základnych dosiek PC. K dosiahnutiu tohto cieľa boli v porovnaní zo štandardom MII zmenené štyri veci. Tieto zmeny znamenajú, že RMII používa približne polovicu signálov v porovnaní s MII.

1. Dva hodinové signály TX_CLK a RX_CLK boli nahradené jedným. Tieto hodiny sú skôr vstupom do PHY než výstupom, ktorý umožňuje zdieľanie hodinového signálu medzi všetkými PHY v multiportovom zariadení.

2. Taktovacia frekvencia sa zdojnásobila z 25MHz na 50MHz, zatiaľ čo dátove cesty sa zúžili z 4 bitových na 2 bitové.

3. Signály RXDV a CRS sú multiplexované do jedného signálu

4. Signál COL je odstránený

Signál MDC a MDIO sa dá stále zdieľať medzi viaceré PHY. Dokopy toto rozhranie vyžaduje 9 signálov oproti MII, ktoré ich vyžaduje 18. Z týchto 9 ich môžu byť dokopy

[4]I2C - sériová komunikačná zbernica

Signál	Popis
REF_CLK	Prijímacie hodiny
TXD0	Prenos dátového bitu 0
TXD1	Prenos dátového bitu 1
TX_EN	Ak je hodnota vysoká, hodinové dáta na TXD0 a TXD1 do vysielača
RXD0	Príjem dátového bitu 0
RX_D1	Príjem dátového bitu 1
CRS_DV	Multiplexované CRS a RX_DV
RX_ER	Chyba prenosu
MDIO	Údaje o správe
MDC	Hodiny pre správu dát

Tabuľka 3.3: Signály RMII [24]

zdieľané tri: MDIO, MDC a REF_CLK. Zvyšných 6 alebo 7 pinov musí byť na každom porte zvlášť.

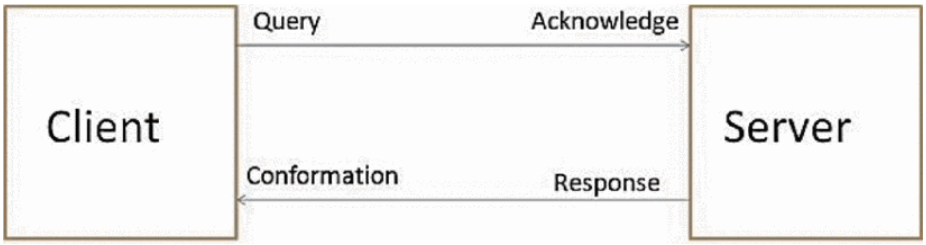
Odstránením niektorých pinov došlo ale k niektorým obmedzeniam. Neexistuje žiadny signál, ktorý by určoval, či je zariadenie v režime plného alebo polovičného duplexu. Rovnako neexistuje signál pre určenie rýchlosti 10 alebo 100 Mbps. Tieto údaje musia byť pre MAC aj PHY rovnaké, preto musia byť komunikované prostredníctvom MDIO/MDC.

Vstupný logický vysoký prah je určený na 2.0V, zatiaľ čo nízky je určený na 0.8V. Špecifikácia uvádza, že by zariadenia mali byť tolerantné k 5V, no niektoré čipy nie sú s 5V logikou kompatibilné.

3.3 Modbus TCP

Modbus-TCP komunikuje prostredníctvom klient/server architektúry po Ethernet TCP/IP sieti. Tento protokol zaobaľuje celé Modbus-RTU pakety do TCP obálky, ktorá sa potom prenáša štandardným Ethernetovým rozhraním. Je teda určitou ekvivalentnou formou Modbus-RTU protokolu. Identifikácia stanice je stále zahrnutá, no jej sémantický význam závisí na konkrétnej aplikácií. Adresácia jednotlivého zariadenia prebieha prostredníctvom IP adresy. Štandardný port pre Modbus-TCP protokol je 502, no ak je to potrebné, môže byť zmenený na ľubovolný iný port.

Komunikácia po Modbus-TCP prebieha v štyroch krokoch zobrazených na obrázku 3.2. V prvom kroku odošle žiadosť na pripojenie (ang. „Connection request“) na server, v druhom kroku je žiadosť prijatá resp. zamietnutá, v treťom kroku server odošle odpoveď na funkčný kód. V poslednom kroku klient odosiela potvrdenie o prijatí. Posledný krok nemusí byť na strane servera implementovaný, spojenie môže byť predčasne prerušené.



Obr. 3.2: Komunikáčný cyklus po Modbus-TCP [23]

Pole	Dĺžka (B)	Popis
Transaction Identifier	2	Identifikácia transakcie Modbus žiadosti/odpovede
Protocol Identifier	2	0 = Modbus protokol
Length	2	Dĺžka nasledujúcich bajtov
Unit Identifier	1	Identifikácia vzdialeného klienta pripojeného cez sériovu alebo inú zbernicu

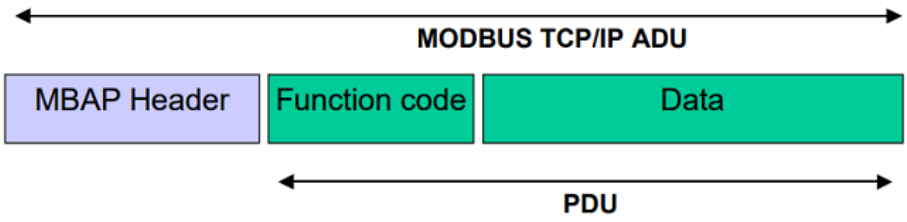
Tabuľka 3.4: Formát MBAP hlavičky [16]

Formát odosielanej správy sa skladá z MBAP[5] (zložený zo siedmych bajtov popísaných v tabuľke 3.4), funkčného kódu a dát (obr. 3.3).

Tento formát má určité odlišnosti oproti Modbus-RTU:

- Pole pre „slave address“, ktoré identifikuje ID slave zariadenia v Modbus-RTU je nahradené za jedno-bajtové pole „Unit Identifier“. Toto pole je použité pri komunikácií so zariadeniami typu router, bridge alebo gateway, ktoré používajú jednu IP adresu pre podporu viacerých nezávislých Modbus zariadení.
- Všetky Modbus požiadavky a odpovede sú navrhnuté tak, aby bolo možné overiť obsah správy. Pre funkčný kód, ktorý má fixnú dĺžku to nie je nutné. Pre políčka, ktoré môžu obsahovať dynamickú dĺžku je vždy uvedená veľkosť.
- Keď sa Modbus prenáša cez TCP, pridávajú sa extra údaje o veľkosti do MBAP hlavičky aby bolo možné na strane príjemcu zistiť kedy prišla celá správa, aj keď je rozložená na viacero časti pri prenášaní.

[5] MBAP - Modbus application header



Obr. 3.3: Komunikáčný cyklus po Modbus-TCP [16]

Kapitola 4

Návrh a implementácia hárdverovej časti vstavaného systému

Nasledujúca kapitola uvádza čitateľa do problematiky vstavaných systémov a ich významu pre dnešný svet. Popisuje nutné predpoklady pre hardvérové riešenie vstavaného systému pre prevod protokolov Modbus RTU a Modbus TCP, samotný návrh hardvéru a následne aj softvéru tohto vstavaného systému ako celku.

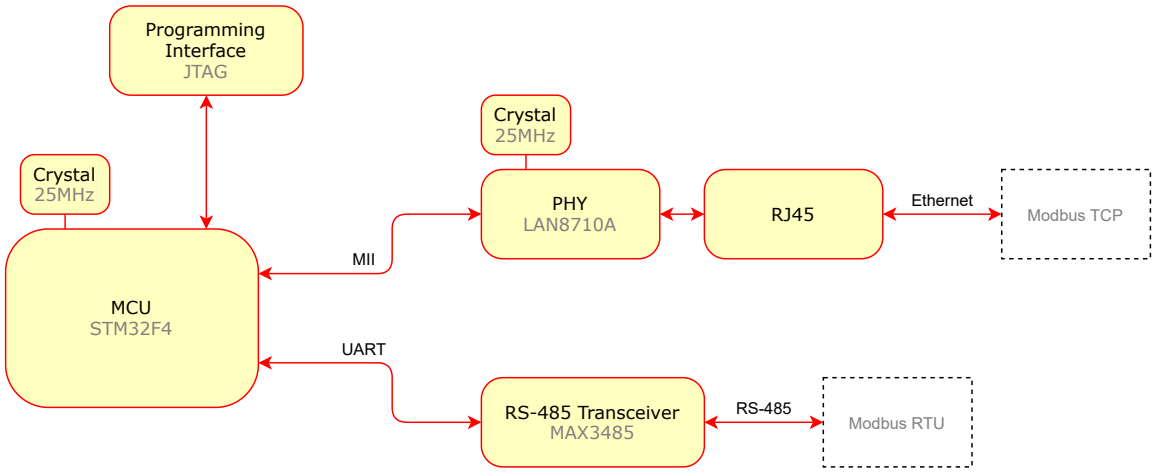
4.1 Špecifikácia požiadaviek pre vstavaný systém

Prevodník protokolov je vhodný príklad pre implementáciu vstavaného systému, pretože po prvotnom nastavení zariadenia nebude vyžadovaná žiadna interakcia s užívateľom. Pre spoľahlivé fungovanie systému je nutné, aby toto zariadenia spĺňalo tieto definované aspekty:

- **Spoľahlivosť** - očakáva sa nepretržitá a bezúdržbová prevádzka počas niekoľkých rokov, preto musí byť systém navrhnutý robustne.
- **Bezúdržbovosť** - systém by nemal vyžadovať zásah užívateľa, mimo prvotnej konfigurácie, či nutnej zmeny konfigurácie v priebehu fungovania systému.
- **Sériovosť** - predpokladá sa nasadenie tohto systému vo vyšších počtoch a teda systém musí obsahovať formu konfigurácie, ktorá umožňuje jednoduché nastavenie prostredníctvom aplikačného rozhrania.
- **Jednoduché sprevádzkovanie** - pre fungovanie systému má byť vyžadovaný iba minimálny počet parametrov, ostatné parametre by mali mať predom definované hodnoty ktoré užívateľ môže dodatočne zmeniť podľa potreby
- **Bezpečnosť** - predpokladá sa odolnosť voči predvídateľným poruchám ako napríklad výpadok napájania
- **Efektívnosť** - od systému je požadovaná nepretržitá funkčnosť a je preto potrebné aby systém nebol energeticky náročný.
- **Cena** - predpokladané nasadenie systému vo vyšších počtoch súvisí s potrebou nízkej jednotkovej ceny zariadenia.

- **Modbus-RTU** - je dôležité aby systém podporoval základne nastavenia sériovej zbernice (rýchlosť (ang. baudrate), paritu, dĺžka slova, stop bity), zároveň musí byť schopné pokračovať v komunikácií v prípade, keď od zariadenia nepríde žiadna odpoveď (napr. timeout)
- **Modbus-TCP** - zariadenie musí obsahovať pripojenie na Ethernet s minimálnou rýchlosťou 1 Mbps.

Podľa vyššie uvedených požiadaviek na vstavaný systém bolo navrhnuté blokové schéma systému. Toto schéma je zobrazené na obrázku 4.1 a poskytuje pohľad na systém vo vyššej abstraktnej forme. Výber špecifických komponentov a ich následne zapojenie je popísané v následujúcich sekciách.



Obr. 4.1: Blokové schéma vstavaného systému

4.2 Výber mikroprocesora

Pri výbere mikroprocesora musí byť daný dôraz na dostupnosť a cenu. Procesor by mal obsahovať aspoň periférii pre Ethernetové rozhranie a periférii pre komunikáciu prostredníctvom UART rozhrania. Tiež mal spĺňať základne požiadavky na výkon, úložný (FLASH) priestor (aspoň 1MiB), dostatočne veľkú pamäť RAM pre obsluhu Ethernetových softvérových súčastí ako je webový server.

Samotný výkon procesora nie je vysokou prioritou, keďže obsluha požiadaviek tohto systému bude výrazne obmedzená rýchlosťou Modbus RTU zbernice. Preto bola vybratá nižšia rada STM32 procesorov a to F4. Konkrétny model STM32F4 procesora bol vybraný na základe požiadaviek pre Ethernetovú a UART periférii a dostatočné množstvo ako FLASH tak RAM pamäte. Za konkrétny model bol zvolený procesor STM32F427VGT6, ktorý spĺňa všetky potrebné požiadavky a umožňuje zníženie spotreby procesora tým, že sa zníži násobič hodín vstupného kryštálu. Čo nám umožní dynamicky regulovať spotrebu energie v závislosti na zaťažení systému.

Samotný procesor umožňuje činnosť na frekvencií až 180MHz a mimo iné obsahuje:

- až 2 MiB flash pamäte

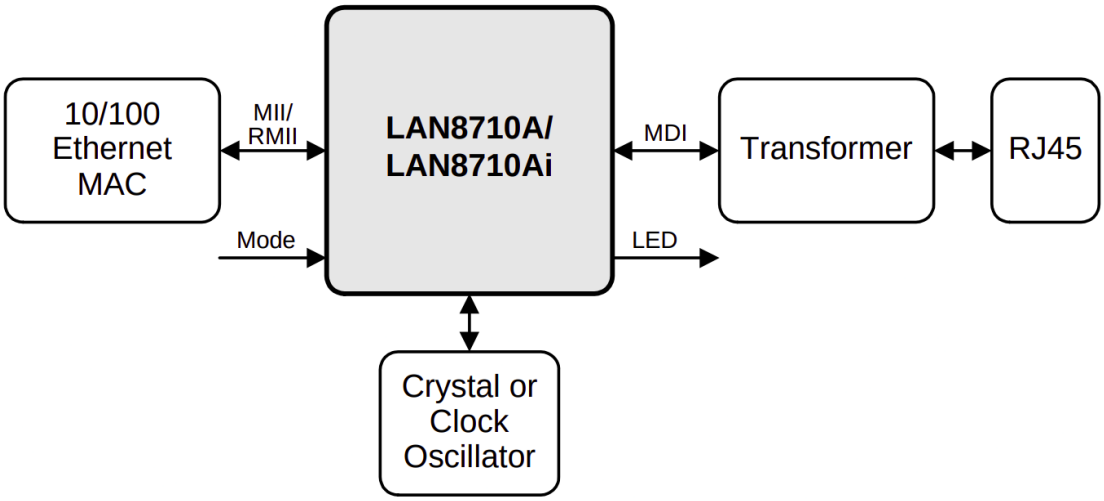
- 256 KiB RAM pamäte
- 3 x I2C
- 6 x SPI
- 2 x CAN
- 4 x USART/UART umožňujúcich až 11.25 Mbit/s
- 10/100 Ethernet MAC s dedikovaným DMA[1] s podporou ako MII tak RMII rozhrania.

4.3 Fyzická vrstva Ethernetového rozhrania

Fyzická vrstva je prvá vrstva modelu vrstvové sieťové architektúry (OSI). Má za úlohu prevod bitov na signál a opačný prevod zo signálu na prúd bitov. Keďže samotný procesor STM32F427VGT6 neobsahuje fyzickú vrstvu, ponecháva tak návrh konkrétneho rozhrania na užívateľovi. To umožňuje istú formu nezávislosti v implementácií rôznych druhov fyzickej vrstvy. Podmienkou pre fyzickú vrstvu ale tvorí implementácia MII alebo RMII rozhrania. Medzi tieto fyzické vrstvy patrí napríklad aj optické pripojenie. No v tomto prípade postačí obyčajné medené pripojenie.

Pre implementáciu tohto rozhrania bol vybraný PHY čip LAN8710A. Tento čip podporuje ako 10 tak aj 100 Mibps Ethernet rozhranie. Implementuje dokonca aj automatické vyjednávanie rýchlosti pripojenia (ang. auto-negotiation), automatickú detekciu polarity aj jej nápravu a podporuje MII aj RMII rozhranie.

Základné blokové schéma zapojenia tohto čipu je na obrázku 4.2. V tomto prípade ale transformér nebude implementovaný ako zvlášť súčiastka, ale bude priamo integrovaný vo vybranom Ethernetovom RJ45 konektore „MC001524“. Takáto implementácia nám umožňuje o niečo zjednodušiť návrh pričom nezvyšuje výslednú cenu dosky.



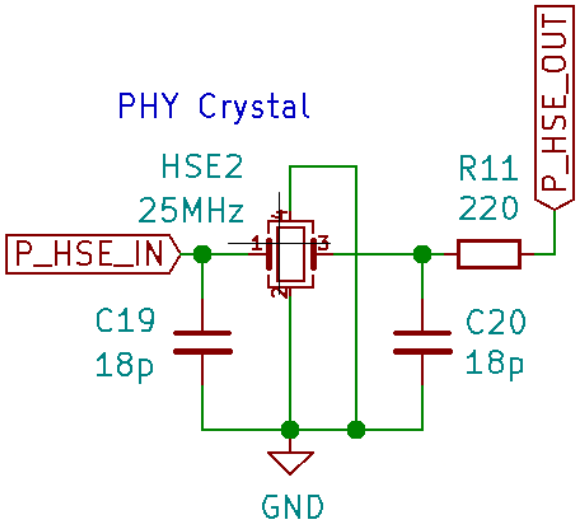
Obr. 4.2: Blokové schéma zapojenia PHY čipu LAN8710A [12]

Pre úspešné fungovanie integrovaného obvodu LAN8710A je nutné počítať s externým oscilátorom o frekvencií 25MHz. Schéma zapojenia oscilátoru vychádza z dokumentácie

[1] DMA - Direct Memory Access

integrovaného obvodu LAN8710A [12] na strane 66. Výsledné zapojenie oscilátoru je na obrázku 4.3. Kryštálový oscilátor, ktorý bol vybraný pre tento účel je MCSJK-7U-25.00-12-10-60-B-10. Tento Oscilátor vyžaduje kapacitu zátaže 12pF pre stabilné udržanie 25MHz frekvencie. Podľa toho boli vybrané kondenzátory C19 a C20. Z pohľadu kryštálu sa jedná o sériové zapojenie kapacity, čo podľa vzorca 4.1 vychádza na 9pF. Táto hodnota je zámerne nižšia, pretože počíta s parazitickými kapacitami na výslednej DPS[2]. Táto parazitická kapacita sa môže pohybovať v rozmedzí 2-4 pF v závislosti na dĺžke medených plôšok na DPS. Výsledná hodnota jedného kondenzátora sa teda vypočíta podľa vzorca $C_l = 2 * (C_{krys.zataz} - C_{parazit})$

$$C_{parallel} = \frac{1}{\frac{1}{C_1} + \frac{1}{C_2} + ...} \tag{4.1}$$



Obr. 4.3: Zapojenie externého oscilátoru pre LAN8710A.

Konektor RJ45

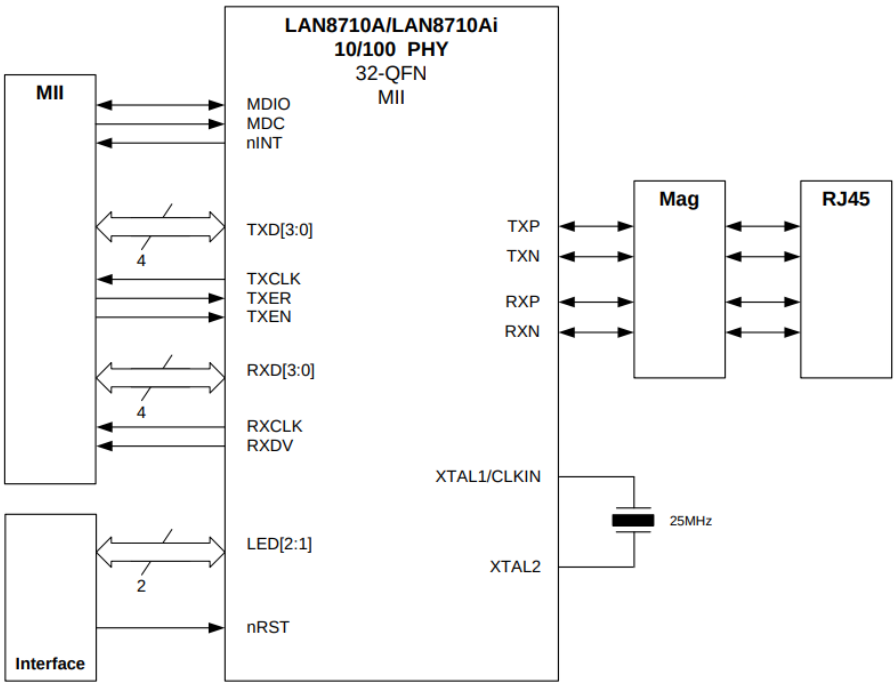
Z obrázku 4.4 vychádza zapojenie konektoru RJ45 k integrovanému čipu LAN8710A. Tento čip sa dá využiť ako s rýchlosťou 10 Mbps tak s rýchlosťou 100 Mbps. K funkčnému zapojeniu plnej rýchlosti 100 Mbps je nutné priviesť dva diferenciálne spoje:

- TXP, TXN
- RXP, RXN

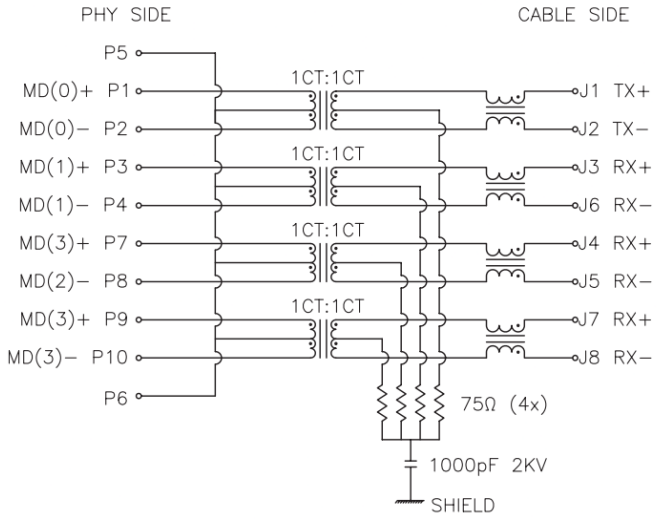
Pre minimalizáciu rušenia je nutné na týchto spojoch udržať rovnakú impedanciu ako v prenosovom médiu. V našom prípade sa jedná o 100 ohm impedanciu. Typ konektoru RJ45 bol vybraný model MC001524. Tento konektor v sebe obsahuje magnetickú časť zapojenia (obr. 4.5). Konektor je navrhnutý pre využitie v režime gigabitového ethernetu. V našom prípade to znamená to, že dátové piny 7, 8, 9 a 10 nebudú využité.

Výsledná schéma zapojenia konektora RJ45 je na obrázku **??**. Vychádza hlavne z pokynov pre zapojenie v dokumentácií PHY obvodu [12].

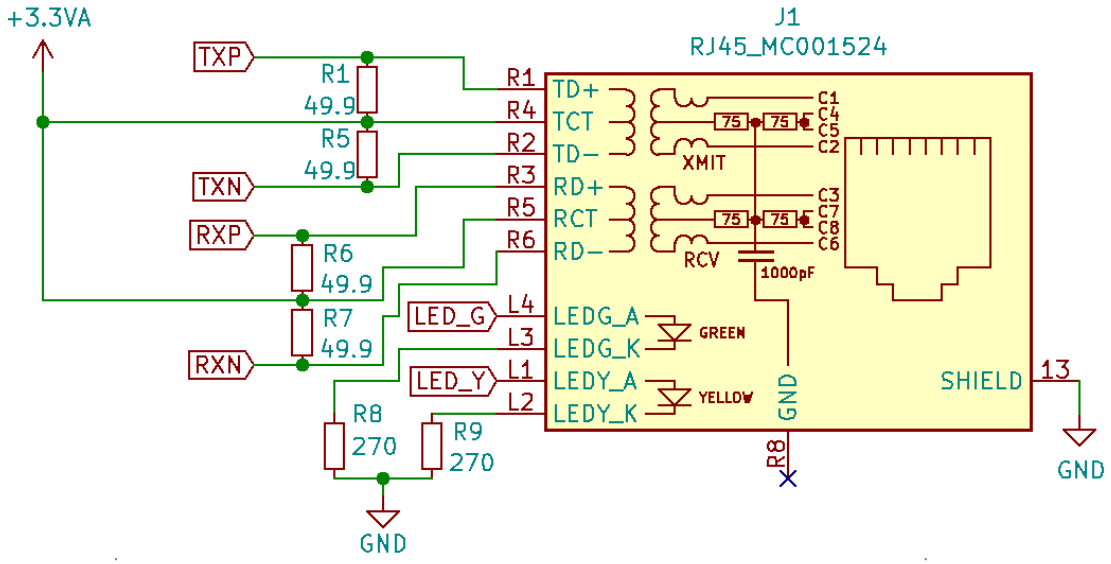
[2]DPS - doska plošných spojov



Obr. 4.4: Zjednodušené schéma zapojenia LAN8710A [12]



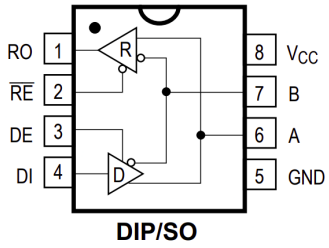
Obr. 4.5: Interná schéma zapojenia konektora MC001524 [13].



Obr. 4.6: Výsledná schéma zapojenia konektora RJ45 na DPS.

4.4 Rozhranie RS485

Keďže samotný mikrokontrolér neobsahuje periférium pre komunikáciu cez zbernicu RS485, musí byť táto zbernica implementovaná pomocou špecializovaného integrovaného obvodu. Pre tento účel bol vybraný integrovaný obvod „MAX3485“. Tento obvod bol vybraný z dôvodu nutnosti využitia 3.3V úrovní pri komunikácií s mikrokontrolérom a zároveň spĺňa všetky požiadavky pre komunikáciu po Modbus RTU protokole. MAX3485 umožňuje komunikáciu až s 32 zariadeniami na zbernici s rýchlosťou až 10Mbps, obsahuje integrovanú ochranu proti prehriatiu a umožňuje prepnutie do módu s nízkou spotrebou energie (2 nA) [10].



Obr. 4.7: Pohľad zhora na čip MAX3485 [10]

Obsluha integrovaného obvodu MAX3485 (obr. 4.7) prebieha prostredníctvom 4 pinov. Pin RO je tzv. „Receiver Output“, čo znamená, že ak je potenciál na pine A väčší ako na pine B o 200 mV, RO je nastavené na logickú jednotku. V opačnom prípade je v logickej nule. Pin DI je tzv. „Driver Input“. To znamená, že logická nula na tomto pine spôsobí, že na výstupe A bude logická nula a na výstupe B bude logická jednička. Zatiaľ čo logická jednička na pine DI spôsobí opačnú situáciu. A to logickú jedničku na výstupe A a logickú nulu na výstupe B.

Zvyšné piny $\overline{RE}$ a DE slúžia pre určenie či je čip nastavený do módu čítania alebo módu zápisu na zbernicu. Pokial su piny RE a DE nastavené do logickej nuly, tak je čip uvedený do módu čítania na zbernici, v prípade nastavenia oboch pinov na logickú jedničku je čip uvedený do stavu zápisu. Takéto chovanie jasne naznačuje, že sa nejedná o full duplex komunikáciu. Takisto je nutné zabrániť odosielanie na zbernici pokiaľ nepríde odpoveď od nejakého slave zariadenia.

Pokiaľ je pin $\overline{RE}$ nastavený na logickú jedničku a zároveň pin DE nastavený na logickú nulu, je toto zariadenie uvedené do stavu nízkej spotreby energie. Pre zjednodušenie návrhu s touto možnosťou nebude počítane a tieto dva piny budú prepojené. Zhrnutie funkcií pinov na integrovanom obvode MAX3485 je vyznačené v tabuľke 4.1.

Takáto obsluha čipu nie je náhodná a pripomína komunikáciu cez UART zbernicu. To umožňuje využiť vstavanú UART periférium vo vybranom mikrokontroléri, ktorú bude treba iba správne nastaviť.

4.5 Napájanie modulu

Zatiaľ čo mikrokontrolér STM32F427 a PHY čip LAN8710A oba umožňujú pracovať v napäťovom rozmedzí 1.7 V až 3.6 V pre integrovaný obvod MAX3485 už je toto napätie obmedzené na 3 V až 3.6 V. Pre zvýšenie rozmedzia napájania je teda nutné použiť regulátor

Číslo pinu	Názov	Funkcia
1	RO	Výstup prijímača. Ak je A >B o viac ako 200 mV, RO je nastavené na log. 1, v opačnom prípade je na log. 0
2	$\overline{RE}$	Aktivácia výstupu prijímača
3	DE	Aktivácia výstupu ovládača
4	DI	Vstup ovládača
5	GND	Uzemnenie
6	A	Neinvertujúci vstup prijímača a neinvertujúci výstup ovládača
7	B	Invertujúci vstup prijímača a neinvertujúci výstup ovládača
8	V_{CC}	Napájanie: 3.0 V $\leq V_{CC} \leq$ 3.6 V

Tabuľka 4.1: Popis funkcií pinov MAX3485

Názov	**Výstupné napätie (V)**
AMS1117-1.5	1.5
AMS1117-1.8	1.8
AMS1117-2.5	2.5
AMS1117-2.85	2.85
AMS1117-3.3	3.3
AMS1117-5.0	5.0

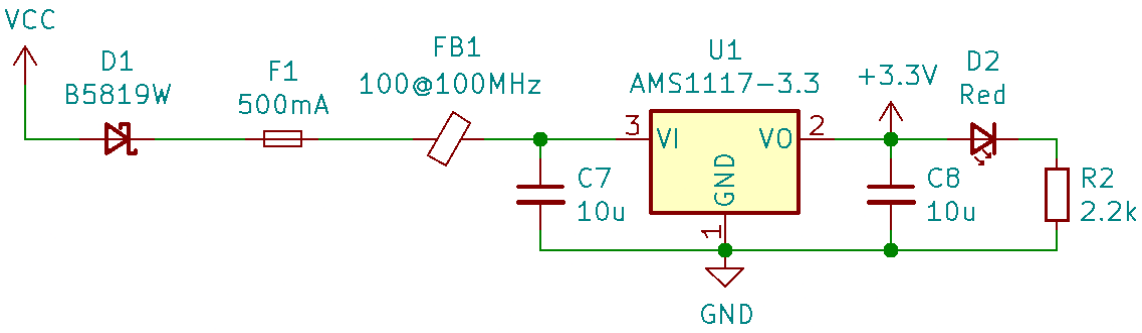
Tabuľka 4.2: Varianty regulátora napätia AMS1117.

napájania. Podľa špecifikácie by mal byť modul schopný pracovať s napájacím zdrojom v rozmedzí 5 až 12 V. Pre naše účely bol vybraný regulátor AMS1117. Tento regulátor je globálne dostupný a vyrába sa v niekoľkých variantoch rozdelených podľa výstupného napätia zobrazených v tabuľke 4.2.

Pre naše účely je teda vhodný regulátor AMS1117-3.3. Tento regulátor sa vyznačuje nízkym parazitickým prúdom, čo ho umožňuje použiť aj v prípade napájania z batérie. Pre tieto účely je ale dôležitejšie rozmedzie napájacieho napätia. V prípade AMS1117-3.3 sa jedná o rozmedzie od 4 do 15V. Keďže sa jedná o lineárny regulátor tak vyššie napájacie napätie bude znamenať nutnosť pridať nejakú formu chladenia. V našom prípade sa ale jedná o relatívne nízku spotrebu elektrickej energie, čo nám dovoľuje tento krok vynechať.

Keďže napojenie modulu v dôsledku nesprávneho zapojenia polarity je relatívne častá chyba ktorá by mohla viesť k úplnej nefunkčnosti modulu je pre dodatočné zabezpečenie modulu voči takémuto javu využitá tzv. Schottkyho dióda. Narozdiel od obyčajnej diódy má totiž výhodu v oveľa miernejšom poklese napätia a zároveň reaguje oveľa rýchlejšie. Naopak ma nevýhodu vo vyššom prúde, ktorý prepustí v zapojení v opačnom smere. Stále sa ale jedná o jednotky mA čo modul nijako neohrozí. Pre túto funkciu bola vybraná dióda B5819W, jedná sa o vysoko dostupnú diódu, ktorá sa bežne používa (nie len) v rámci ochrany pred reverznou polaritou.

Finálna podoba napájania modulu je na obrázku 4.8. Okrem ochrany pred reverznou polaritou je doplnená o poistku, ktorá by v prípade skratu mohla ochrániť komponenty na module. Taktiež bolo pridané filtrovanie ako vstupného, tak výstupného napätia regulátora. Medzi posledné úpravy patrí pridanie červenej LED diódy. Táto dióda má za úlohu oznámiť užívateľovi, či je na doske prítomné napájanie alebo nie.



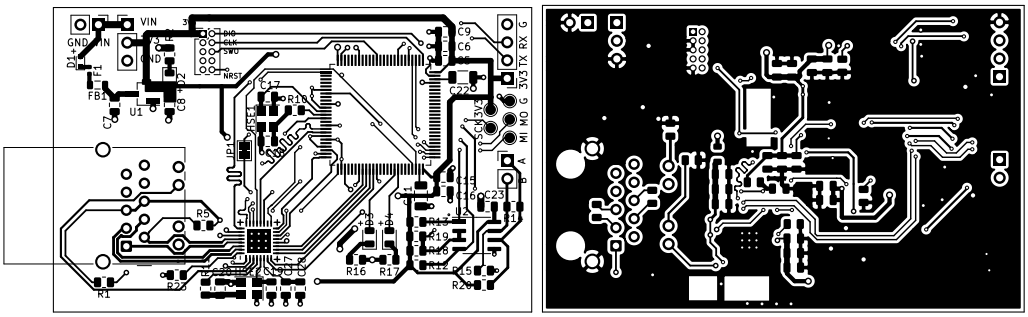
Obr. 4.8: Schéma zapojenia napájania modulu.

4.6 Návrh dosky plošných spojov

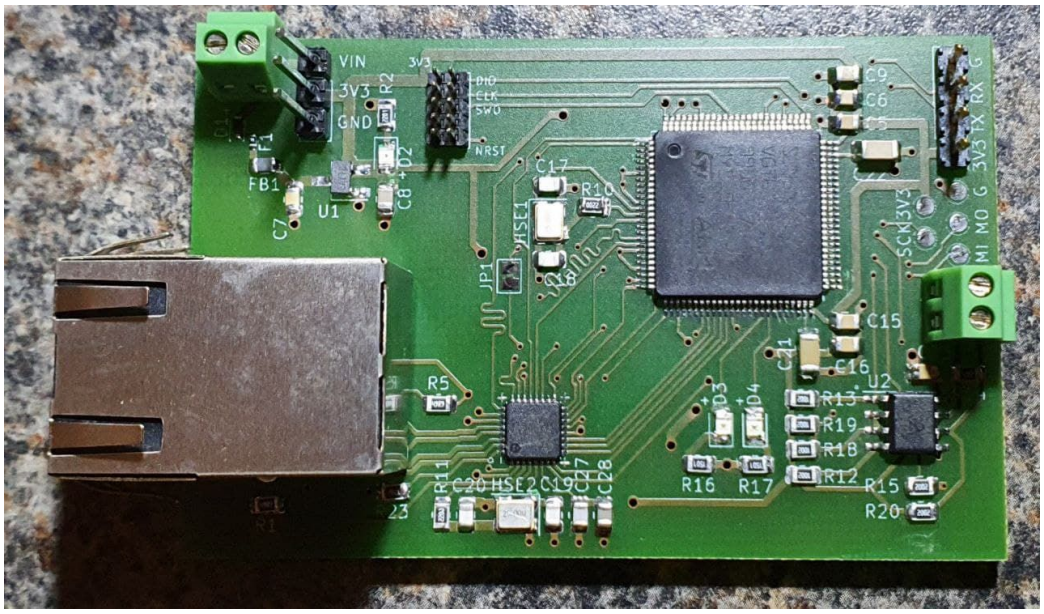
Po zhotovení schémy zariadenia je na rade modul vo forme dosky plošných spojov (ďalej už len DPS). Ten bol navrhnutý tak, aby ho bolo možné osadiť aj ručne. Z tohto dôvodu boli vybraté väčšie jednotlivé pasívne súčiastky než by bolo v inom prípade nutné. V rámci špecifikácie nebola veľkosť výslednej DPS obmedzená. Tým pádom väčšie súčiastky nespôsobovali žiaden problém. Výsledná DPS je na obrázku 4.10.

Ako hlavné limitné faktory výroby DPS boli použité nasledujúce obmedzenia:

- Najmenšia veľkosť vrtáka pre prepojky (ang. via) - 0.3 mm
- Najmenšia vzdialenosť od vodičov - 0.19 mm (0.15 mm pre diferenciálny pár)
- Najmenšia šírka vodiča - 0.15 mm
- Najmenšia vzdialenosť medzi vŕtanými dierami - 0.25 mm
- Najmenšia veľkosť súčiastky kladenej na DPS - 0805 (2012 metrických) - jedinú výnimku tvorí poistka na vstupe, ktorá nebola dostupná vo väčšom balení ako 0402 (1005 metrických)



Obr. 4.9: Nakreslená horná vrstva (vľavo) a spodná vrstva (vpravo) DPS. Exportované z programu KiCad.



Obr. 4.10: Hotová DPS po osadení všetkými súčiastkami.

Kapitola 5

Návrh a implementácia softvérového vybavenia

Návrh firmvéru pre mikrokontrolér bol vytvorený vo vývojovom prostredí STM32CubeIDE v jazyku C. Toto vývojové prostredie poskytnuté zadarmo od výrobcu STMicroelectronics. Pri vytváraní firmvéru bola využitá implementácia operačného systému reálneho času (FreeRTOS) a nizkoúrovňová implementácia TCP/IP vrstvy (lwIP). FreeRTOS rovnako ako lwIP sú popísané v tejto kapitole. Keďže prostredie STM32CubeIDE, v dobe písania tejto práce, nedokáže vytvoriť projekt s funkčnou Softvérovou inicializáciou Ethernetového rozhrania, vychádza tento softvér z existujúceho príkladového projektu od STMicroelectronics, ktorý implementuje základnú inicializáciu Ethernetového rozhrania.

V následujúcich podkapitolách je popísaný softvér implementovaný vo vstavanom zariadení. Kód je obohatený komentármi a vysvetlením pre jednoduchšie pochopenie.

5.1 Operačný systém reálneho času RTOS

Operačný systém reálneho času RTOS (ang. Real-time operating system) je systém, ktorý zabezpečuje použitie v aplikáciach, kde je potrebné vykonávať úlohy v pevne stanovených časových intervaloch. Takéto systémy si nájdu využitie najmä v priemysle, napríklad v regulačnej a riadiacej technike. Typickým príkladom je riadiaci systém lietadla, kde je potrebné zaručiť odozvu na riadiaci impulz v krátkom a pevne definovanom maximálnom čase. Nemusí sa ale vždy jednať len o veľké a vysoko komplexné systémy. Operačný systém reálneho času nájde využitie aj v často používaných zariadeniach ako sú novodobé práčky alebo umývačky.

Aplikácie reálneho času sú z pravidla riešené metódou súbežného spracovania jednotlivých úloh (ang. multitasking). Ide napríklad o aplikácie spracovania dát v reálnom čase, kedy je nutné zaistiť kontinuálne meranie a spracovanie dát, dodržať definovanú dĺžku a odozvu meraného signálu a súčasne zaistiť komunikáciu s nadradenými systémami. V takomto prípade sa bez paralelného spracovania nezaobídeme. Pri návrhu aplikácie je ale nutné počítať s problémami, ktoré jedno-úlohové programovanie nepozná [27].

Systém pracujúci v reálnom čase, je taký systém, ktorý vždy musí dať odpoveď na vnútorný signál v konečnom, predom stanovenom intervale. To znamená, že musia byť predom zaistené maximálne doby opozdenia reakcie na dané podnety [27].

Operačné systémy reálneho času vieme rozdeliť nasledovne:

- **Hard real time** - systém, v ktorom je bezpodmienečne nutné dodržať časový interval. Nedodržanie môže viesť k vážnejším nehodám.
- **Firm real time** - systém, ktorý môže pripustiť občasné nedodržanie časového intervalu.
- **Soft real time** - systém, kde nie je nutné dodržať časový interval, oneskorenie niektorej aplikácie nevedie k žiadnym alebo minimálnym následkom.

Základný princíp operačných systémov reálneho času je rovnaký ako princíp činnosti bežných operačných systémov prítomných napríklad na osobných počítačoch. Po väčšine je hlavný rozdiel v periodickom striedaní jednotlivých úloh podľa priradenej priority.

Úlohy operačného systému môžu nadobudnúť nasledujúce stavy (obr. 5.1):

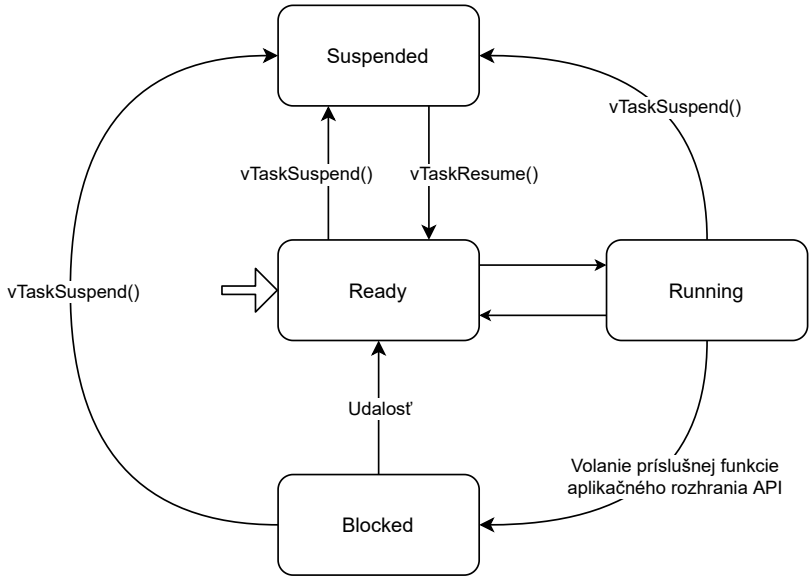
- **Running** - stav systému, v ktorom je úloha momentálne vykonávaná a na procesore má pridelené časové okno. Úloha opustí tento stav v prípade, ak o to sama požiada, alebo v nútenom opustení pomocou plánovača (ang. scheduler).
- **Ready** - stav, v ktorom je úloha pripravená na vykonávanie a čaká vo fronte na pridelenie procesorového času pre prepnutie do stavu „Running“. Jednotlivé úlohy môžu byť zoradené vo fronte napríklad podľa priority (ang. priority scheduling). V tomto prípade sa najskôr dostane ku procesorovému času aplikácia s najvyššou prioritou.
- **Blocked** - stav, kedy je úloha dočasne v režime blokovania a čaká na vstup dát. V tomto stave môže byť úloha len po istý čas. Po uplynutí tohto časového bloku je automaticky presunutá do stavu „Ready“.
- **Suspended** - stav, ktorý môže nastať iba zavolaním príslušnej funkcie aplikačného rozhrania API.

Semafory

Synchronizácia jednotlivých aplikácií je realizovaná prostredníctvom semaforov. Tieto semafory sa rozdeľujú na tri druhy. Môžu to byť jednoduché binárne semafory. Binárne semafory nadobúdajú iba hodnotu 0 alebo 1. Tieto hodnoty indikujú dostupnosť úlohy. Binárne semafory sú často riadené pomocou hlavnej úlohy. Druhou kategóriou sú semafory vo forme počítadla. Tieto semafory môžu nadobúdať viacero hodnôt a môžu byť viacnásobne prečítané. Takýto prístup nám umožňuje vstup viacerých k viacerým úlohám v kritickej sekcii naraz. Posledným druhom sú semafory so vzájomným vylúčením tzv. mutexy, ktoré sa najčastejšie využívajú pri rekurzívnych volaniach.

Časovače

Okrem semaforov môžeme využiť aj napríklad časovače. Tak isto ako semafory sú časovače implementované v jadre systému. Časovače rozdeľujeme na dva druhy. Prvým je jednorázový časovač, ktorý po uplynutí časového intervalu zavolá požadovanú úlohu práve jedenkrát. Druhý tvorí časovač, ktorý sa po uplynutí času automaticky reštartuje a môže byť použitý na periodické volanie úloh.



Obr. 5.1: Validné prechody stavov operačného systému reálneho času [4].

Fronty

Komunikácia medzi úlohami je realizovaná pomocou zdielanej pamäte a využívajú sa prevažne fronty. Pokiaľ je potrebná výmena dát medzi jednotlivými úlohami je potrebné alokovať potrebné množstvo pamäti. Tento prístup prináša nevýhodu v tom, že pre prenos takýchto dát musí byť dopredu známa (a nemenná) veľkosť dát pre odoslanie. Dáta sa medzi úlohami vymieňajú pomocou správ, ktoré sú vkladané do fronty FIFO (ang. First In First Out). Odosielané môžu byť jednoduché dátové typy ale aj celé štruktúry, záleží iba na veľkosti dostupnej pamäti. Pokiaľ ale potrebujeme prenášať dáta s rozličnou veľkosťou dát, ktorú dopredu nevieme určiť, je nutné priestor alokovať a vo fronte už prenášať iba odkaz na tento alokovaný priestor. V druhej úlohe je potom potrebné zaistiť uvoľnenie takto alokovanej pamäte a zároveň treba dať pozor na to, aby sme ju zatiaľ neprepísali v inej úlohe.

Existuje niekoľko variant dostupných implementácií systémov reálneho času a stále sa pracuje na ich rozvoji. Pre použitie s mikrokontrolérom s obmedzenou programovacou pamäťou a nárokmi na open source bola zvolená implementácia s názvom „FreeRTOS“. Tento systém je volne šíriteľný na komerčné účely pod licenciou MIT vďaka čomu si našiel široké využitie v oblasti open-source projektov. Operačný systém umožňuje vytvárať neobmedzene množstvo úloh (obmedzené iba veľkosťou RAM) a obsahuje ochranu pamäte MPU (ang. Memory Protection Unit), čo zabezpečuje ochranu systému pred alokáciou viac pamäte než je na systéme dostupnej a aktívne tak zabráni prípadnemu pádu aplikácie.

5.2 lwIP

LwIP (ang. lightweightIP) je odľahčená, nízkoúrovňová verzia TCP/IP vytvorená pre použitie vo vstavaných zariadeniach s nízkymi nárokmi na výpočtový výkon a pamäť. Táto implementácia bola napísaná švédskym programátorom Adamom Dunkelsom. Celý TCP/IP model je napísaný v jazyku C.

Model obsahuje implementáciu nasledujúcich protokolov:

- **IP vrstva:**
  - **IP** - Internet Protocol (obsahuje IPv4 aj IPv6)
  - **ICMP** - Internet Control Message Protocol
  - **IGMP** - Internet Group Management Protocol
- **Sieťová vrstva:**
  - **TCP** - Transmission Control Protocol
  - **UDP** - User Datagram Protocol
- **Aplikačná vrstva:**
  - **DNS** - Domain Name System
  - **SNMP** - Simple Network Management Protocol
  - **DHCP** - Dynamic Host Configuration Protocol
  - **HTTP** - Hypertext Transfer Protocol
  - **MQTT** - Message Queuing Telemetry Transport
  - **TFTP** - Trivial File Transfer Protocol
  - **SMTP** - Simple Mail Transfer Protocol
  - **SNTP** - Simple Network Time Protocol

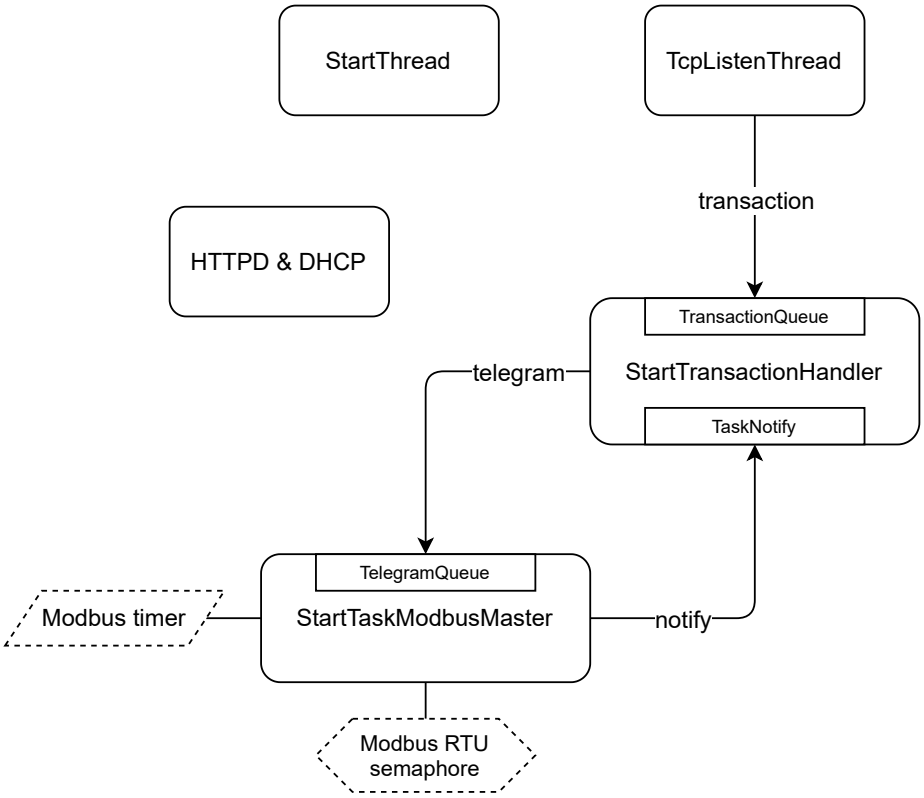
LwIP poskytuje tri aplikačné rozhrania pre prístup ku komunikácií po TCP/IP. Najnižšia úroveň komunikácie tzv. „RAW“. Táto úroveň používa mechanizmus spätného volania (ang. callback) pri príchode nových dát. Takéto správanie je nutné príslušne v programe ošetriť, čo môže byť zložitejšie na implementáciu. Mimo najnižšej úrovne poskytuje lwIP aj dve možnosti komunikácie na vyššej úrovni. Prvou takouto možnosťou je pomocou tzv. „netconn API“. Druhá možnosť sa označuje názvom „socket API“. Socket API sa narozdiel od natconn API snaží zachovať prenositelnosť medzi operačnými systémami tým, že implementuje štandard BSD (ang. Berkeley sockets).

5.3 Návrh firmvéru

Pri vývoji firmvéru bola použitá nízkoúrovňová nadstavba knižníc CMSIS (ang. Cortex Microcontroller Software Interface Standard) od výrobcu mikrokontroléru s názvom HAL (ang. Hardware Abstraction Layer). Knižnice HAL predstavujú aplikačnú úroveň riadenia mikrokontroléra a nie je potrebné pristupovať k registrom ako je to napríklad pri programovaní mikrokontrolérov platformy ARM.

Samotný firmvér je rozdelený do niekoľkých hlavných úloh (obr. 5.2), ktoré sa spolu synchronizujú a posielajú si medzi sebou dáta aby bola umožnená nepretržitá a paralelná obsluha jednotlivých klientov. Hlavná funkcia main() spustí úlohu StartThread, ktorá nastaví a inicializuje Ethernetové spojenie pomocou lwIP, prípadne spustí aj DHCP úlohu, ktorá zaisťuje načítanie konfigurácie z existujúceho DHCP servera. StartThread potom spustí HTTPD úlohu. HTTPD úloha potom spracováva HTTP požiadavky.

Hlavná funkcia main() okrem úlohy StartThread inicializuje aj ModbusMaster, TcpListenThread a TransactionHandler úlohy. Tieto úlohy sa všetky týkajú samotného prekladu



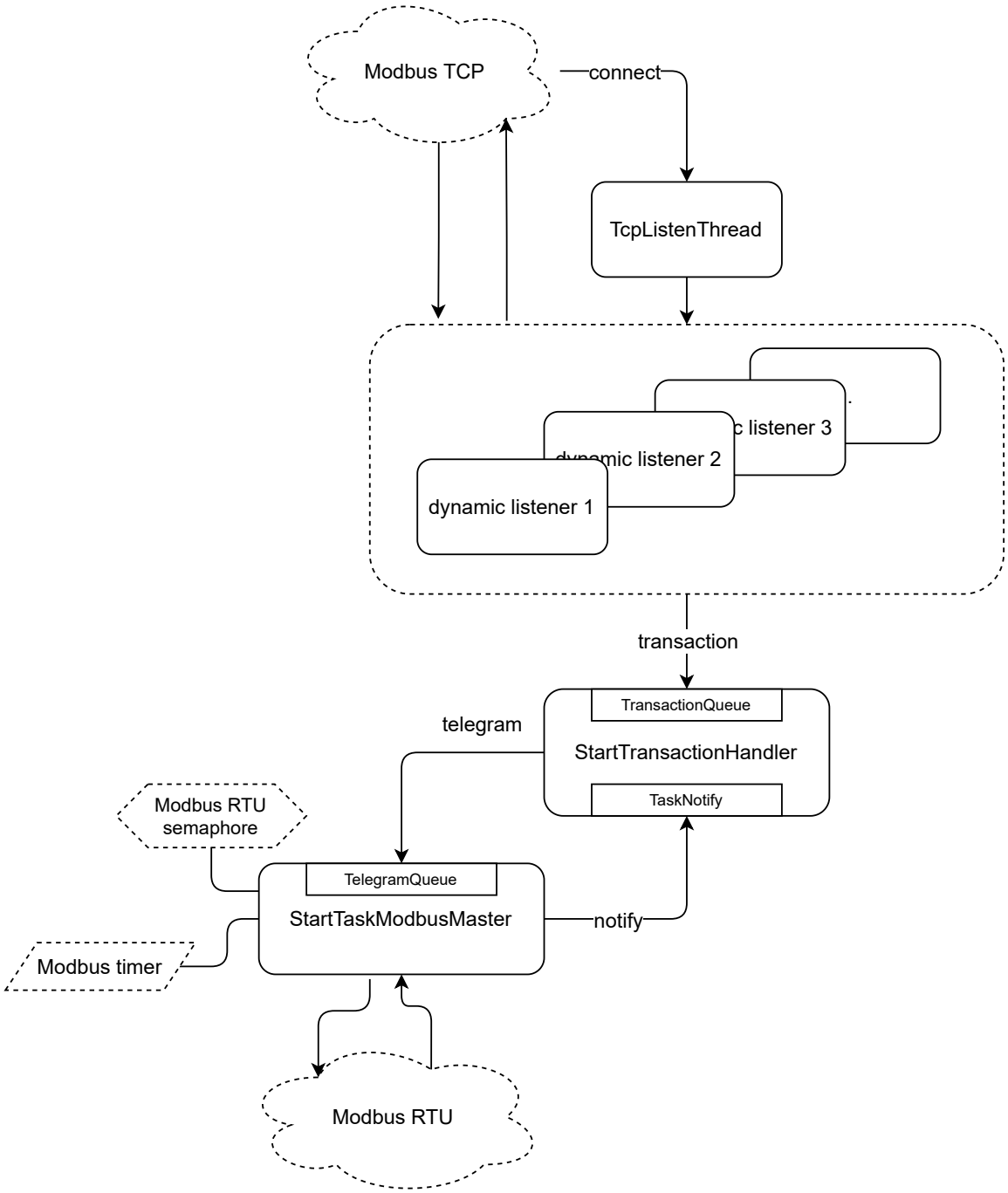
Obr. 5.2: Štruktúra RTOS úloh vo firmvéri.

Modbus TCP na Modbus RTU protokol a opačne. TCPListenThread načúva na prichádzajúce Modbus TCP pakety na porte definovanom v konfigurácií. Pri každom platnom spojení vytvorí dynamickú úlohu ktorá následne spracováva prichádzajúce pakety. Tým pádom môže TCPListenThread ďalej načúvať na prichádzajúce spojenia a modul tak môže obsluhovať viacero klientov naraz. Výsledná komunikácia medzi Modbus TCP a Modbus RTU zariadením je zobrazená na obrázku 5.3.

5.4 Inicializácia systému a periférií zariadenia

Inicializácia systému prebieha hneď pri napojení modulu. Tento proces začína inicializáciou systémového násobiča externých hodín. Hodnoty nastavenia tohto násobiča sú nastavené explicitne tak, aby jadro mikrokontroléra fungovalo na frekvencií 81.25MHz. Oproti najvyššej rýchlosti, ktorú mikrokontrolér dovoľuje (180MHz) je to značný skok dole, no pre samotnú komunikáciu po protokole Modbus RTU nie je vysoká rýchlosť potrebná. Pripojené zariadenia totiž často fungujú na rýchlosti menšej ako 19500 bps a samotné rozhranie RS485 nám nedovoľuje komunikovať s viacerými zariadeniami naraz.

V druhom kroku prebieha inicializácia periférií. Ak je to prvý štart zariadenia, načítajú sa predom nastavené hodnoty. Pre pripojenie do siete sa v prednastavených hodnotách počíta s využitím DHCP protokolu pre zistenie IP adresy, masky siete a adresy brány, s ktorou bude zariadenia ďalej komunikovať. Inicializácia Ethernetového rozhrania rovnako ako pripojenie do siete prebieha pomocou súborov funkcií z knižnice lwIP.



Obr. 5.3: Prepojenie Modbus TCP a Modbus RTU zariadení.

Po inicializácií systému a načítaných nastavení potrebných pre chod zariadenia prichádza na rad inicializácia potrebných komponent pre komunikáciu po Modbus RTU. A to hlavne UART periférie a vytvorenie obslužnej úlohy (ang. task) v systéme FreeRTOS. pre obsluhu Modbus RTU požiadavkov.

Ako posledná časť inicializácie prebieha vytvorenie a spustenie freeRTOS úlohy, ktorá ma za úlohu samotné pripojenie do siete a inicializáciu HTTP servera pre konfiguráciu zariadenia.

UART

Pre debugovací výstup a komunikáciu po RS485 pomocou integrovaného obvodu MAX3485 potrebujeme komunikovať po UART rozhraní. Využitý mikrokontrolér STM32F427VGT v sebe obsahuje niekoľko takýchto rozhraní, ktoré vieme využiť. Pre naše účely využijeme UART4 a UART5 periférie na mikrokontroléri. Tieto rozhrania boli vybrane vďaka pozícií ich príslušných vyvedených pinov na DPS.

UART5 bude využitý ako debuggovací výstup, to umožňuje inicializovať túto periférium hneď pri spustení mikrokontroléru z dôvodu, že konfigurácia tejto periférie je dopredu daná a nemení sa. Z tohto dôvodu vieme nastaviť UART5 periférium na konfiguráciu zobrazenú v tabuľke 5.1. Táto konfigurácia umožňuje komunikáciu prostredníctvom jednoduchého TTL na USB modulu ako je napríklad modul s integrovaným obvodom CH340.

Periféria	**BaudRate**	**WordLength**	**StopBits**	**Parity**
UART4	19200	9 B (8 B + parita)	1	párna
UART5	57600	8 B	1	bez parity

Tabuľka 5.1: Počiatočná konfigurácia periférií UART.

Inicializácia UART4 periférie prebehne obdobne ako UART5 s tým rozdielom, že konfiguráciu pre UART4 načítavame z FLASH pamäte pri štarte modulu, pokiaľ táto konfigurácia nie je dostupná (prvý štart zariadenia), je nastavená počiatočná konfigurácia podľa tabuľky 5.1. Túto konfiguráciu je potom možné zmeniť v rámci webového rozhrania. Pre správne fungovanie UART periférií je ešte nutné previesť konfiguráciu špecifickú pre konkrétny mikrokontrolér pomocou tzv. MSP[1]. V rámci tohto procesu sa inicializujú hodiny pre UART rozhranie, vstupno-výstupne piny a pre UART4 aj systém prerušenia (ang. interrupt).

5.5 Ukladanie konfigurácie

Najčastejšie používaná forma pre ukladanie dát je tzv. EEPROM (ang. Electrically Erasable Programmable Read-Only Memory). Jedná sa o typ pamäte do ktorej je možné zapisovať aj ju vymazať elektronicky. Umožňuje to ukladanie malého množstva dát na nevolatilnú pamäť, z ktorej sa potom dajú tieto dáta načítať aj po výpadku napájania.

Ak je naším cieľom vytvoriť cenovo dostupnejšie zariadenie, je možné túto pamäť nahradiť niektorým zo zabudovaných riešení v mikrokontroléri STM32F427xx/STM32F429xx:

- 4 KB záložná pamäť SRAM

[1]MSP - MCU Specific Package - Súbor balíčkov pre konfiguráciu periférií pre konkrétny mikrokontrolér

- zabudovaná pamäť Flash so špecifickým algoritmom pre zápis/čítanie.

Zabudovaná záložná pamäť SRAM uchováva dáta, pokiaľ je dostupné napájanie na VBAT pine mikrokontroléra. V prípade výpadku napájania a dostupnosti malej batérie pre udržanie dát je teda táto pamäť vhodná na použitie pre ukladanie dočasných dát, ktoré je nutné uchovať aj v prípade výpadku hlavného napájania VDD. Zároveň nám takáto pamäť dovoľuje vysoko-rýchlostný prístup zápisu/čítania závislej na frekvencií jadra mikrokontroléra.

V prípade ak nie je dostupné napájanie z batérie alebo sa záložná pamäť SRAM využíva na iné účely je možné využiť pre ukladanie dát aj zabudovanú Flash pamäť mikrokontroléra pre emuláciu EEPROM pamäte. Keďže v našom prípade nemáme záložné napájanie a dáta, ktoré chceme uložiť majú skôr charakter konfigurácie systému o ktoré nechceme prísť ani v prípade vybitia záložného zdroja, je pre nás ďaleko výhodnejšie využiť zabudovanú Flash pamäť.

Pre emuláciu EEPROM pamäte na Flash pamäť mikrokontroléra je využitá dostupná knižnica od výrobcu STMicroelectronics. Emulácia je dosiahnutá využitím posledných dvoch sektorov Flash pamäte, softvér potom prepína medzi týmito sektormi tak, aby to bolo pre užívateľa co najtransparentnejšie. Poskytnutá knižnica sa zároveň snaží o čo najjednoduchší prístup pre používateľa. Dá sa s ňou pracovať využitím iba troch funkcií. Prvá je funkcia pre inicializáciu pamäte, v ktorej prebieha aj počiatočné vymazanie a základný dáta management v prípade poškodenia dát. Potom máme dostupné dve funkcie pre čítanie a zápis hodnôt do pamäte.

Rozdiel medzi externou a emulovanou EEPROM pamäťou

EEPROM je komponenta, ktorú používa veľa vstavaných zariadení. Umožňuje uložiť bajty alebo slová do nevolatilnej pamäte.

Mikrokontroléry často používajú vstavanú Flash pamäť pre tento účel. Pre zníženie počtu komponentov, potrebného miesta a výrobnej ceny výslednej DPS je možné využiť internú Flash pamäť namiesto externej EEPROM pre súčasný prístup ku zdrojovému kódu programu a dátam uložených v pamäti.

Základné rozdiely medzi internou Flash pamäťou a externou sériovou EEPROM pamäťou sú rovnaké pre všetky mikrokontroléry, ktoré používajú technológiu Flash pamäte. Najväčšie rozdiely sú popísané v tabuľke 5.2.

Princíp emulovanej EEPROM

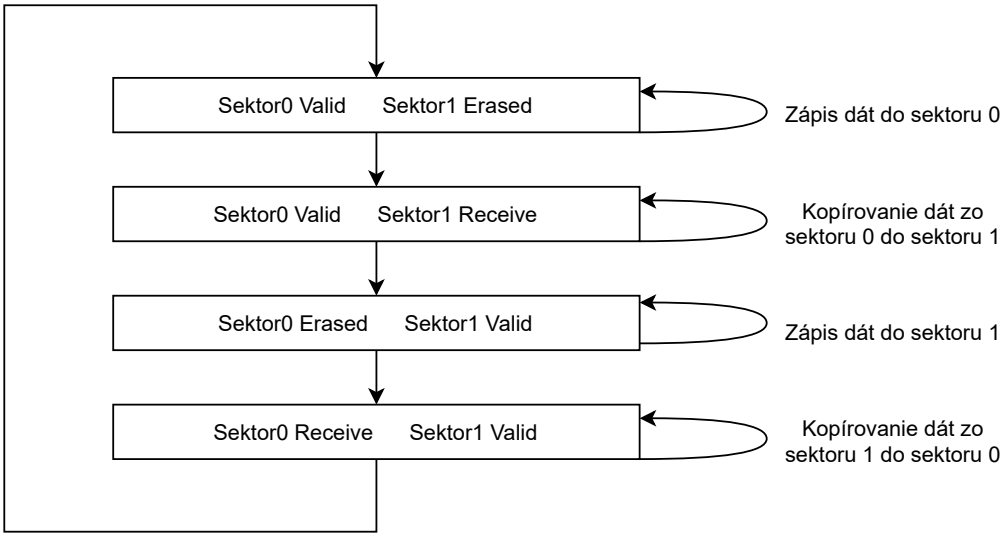
Emulácia EEPROM môže prebiehať rôznymi spôsobmi, treba brať hlavne na vedomie veľkosť Flash pamäte zariadenia a požiadavky na výsledný vstavaný systém. Postup implementovaný v knižnici od výrobcu zahŕňa využitie dvoch sektorov identickej veľkosti pre ukladanie nevolatilných dát. Prvý sektor je na začiatku premazaný a umožňuje zápis bajt po bajte a druhý, ktorý ho nahradí v prípade, ak je prvý nutné premazať. Hlavička, ktorá sa nachádza na prvom polslove (16 bitov) každého sektora nesie informáciu o stave sektora.

Hlavička, ktorá sa nachádza na začiatku sektora môže niesť nasledujúce informácie o stave sektora:

- **ERASED** - sektor neobsahuje žiadne dáta.
- **RECEIVE_DATA** - sektor dostáva dáta z iného plného sektora.

Funkcia	**Externá EEPROM**	**Emulovaná EEPROM s Flash pamäťou**	**Emulovaná EEPROM s SRAM pamäťou**
Čas zápisu	- náhodný zápis bajtu do 5ms, slova do 20ms - zápis strany (32 bajtov) do 5ms, zápis slova do 625 μs	zápis polslova od 30 μs do 237.25 ms	rýchlosť CPU s 0 čakacími stavmi
Vymazanie		vymazanie sektoru od 1 s do 3 s podľa veľkosti sektoru	
Zápis	po spustení nie je závislá na CPU	po spustení je závislá na CPU. Ak je zápis prerušený resetom na CPU, algoritmus pre emuláciu EEPROM je zastavený, no operácia zápisu do Flash pamäte nie je zastavená.	operácia môže byť prerušená CPU resetom
Čítanie	- sériové aj náhodne čítanie trvá okolo 100 μs - čítanie strany trvá 22 μs za každý bajt	paralelné čítanie (168 MHz) polslova trvá od 0.68 μs do 251 μs	rýchlosť CPU s 1 čakacím stavom
Cykly zápisu/čítania	1 milión cyklov zápisu	10 tisís zápisov na sektor	bez limitu

Tabuľka 5.2: Rozdiely medzi externou a emulovanou EEPROM pamäťou [21].



Obr. 5.4: Údaje v hlavičke na emulovanej EEPROM pri prepínaní medzi sektormi [21].

- **VALID_PAGE** - sektor obsahuje validné dáta a tento stav sa nemení pokial sa všetky dáta nie su premiestnené do vymazaného sektora.

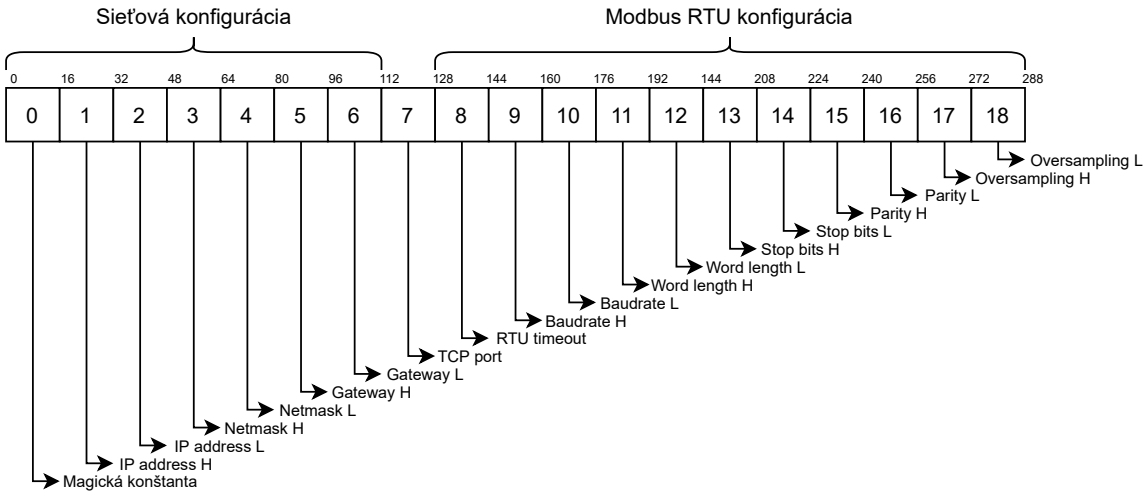
Na obrázku 5.4 je zobrazený postup menenia stavu sektorov.

Formát konfiguračných dát ukladaných do EEPROM

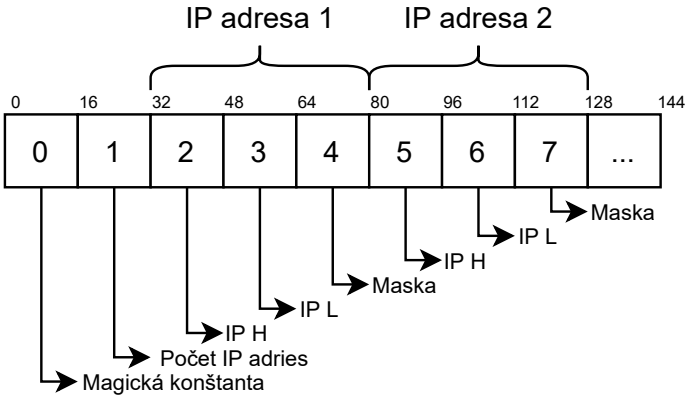
Dáta musia byť pred uložením do emulovanej EEPROM pamäte serializované do veľkostne rovnakých slov. Šírka slova vo využitej knižnici je šestnásť bitov. To znamená, že pri serializácií 32 bitových hodnôt je nutné túto hodnotu rozdeliť na dve šestnásť-bitové čísla. Ako prvé je uložených horných šestnásť bitov a na druhé miesto je vždy uložených spodných šestnásť bitov ako je ukázané napríklad v uložení konfiguračných dát na obrázku 5.5. Pod veľkým písmenom „H“ resp. „L“ je myslených horných resp. spodných šestnásť bitov hodnoty.

Magická konštanta v prvom slove na obrázku 5.5 má dve dôležité vlastnosti. V prvom rade pomáha identifikovať, či sa v pamäti nachádzajú validné dáta. V druhom rade drží informáciu o tom, či je využitá funkcia DHCP protokolu. V prípade ak je táto funkcia využitá, je táto magická konštanta nastavená na hodnotu 0x43 v šestnástkovej sústave a zvyšné sieťové informácie ignorované. V opačnom prípade je táto hodnota nastavená na hodnotu 0x42 a sieť je nakonfigurovaná podľa zvyšnej uloženej konfigurácie. Hodnota „TCP port“ v konfigurácií (obr. 5.5) označuje port, ktorý je neskôr využitý pre načúvanie na Modbus TCP pakety. Hodnoty pre Modbus RTU konfiguráciu sú uložené v presnom formáte ako sa zadávajú do HAL knižnice, no ak by bola nutnosť zmenšiť veľkosť uložených dát, je možné tieto dáta konfigurácie zmenšiť vďaka faktu, že väčšina týchto nastavení je reprezentovaných iba vo forme niekoľkých bitov. Pre zjednodušenie sú ale tieto hodnoty ukladané v plnej veľkosti.

Na obrázku 5.6 je znázornené ukladanie povolených IP adries zapísaných prostredníctvom webového rozhrania. Iba z daných adries (podsietí) je možné prijímať modbus TCP pakety. Slúži to ako istá základná forma bezpečnosti pred možným zahltením zariadenia prípadným útočníkom. Na prvom mieste opäť vyskytuje magická konštanta, v tomto prí-



Obr. 5.5: Formát konfiguračných dát uložených v emulovanej EEPROM pamäti.



Obr. 5.6: Formát povolených IP adries uložených v emulovanej EEPROM pamäti.

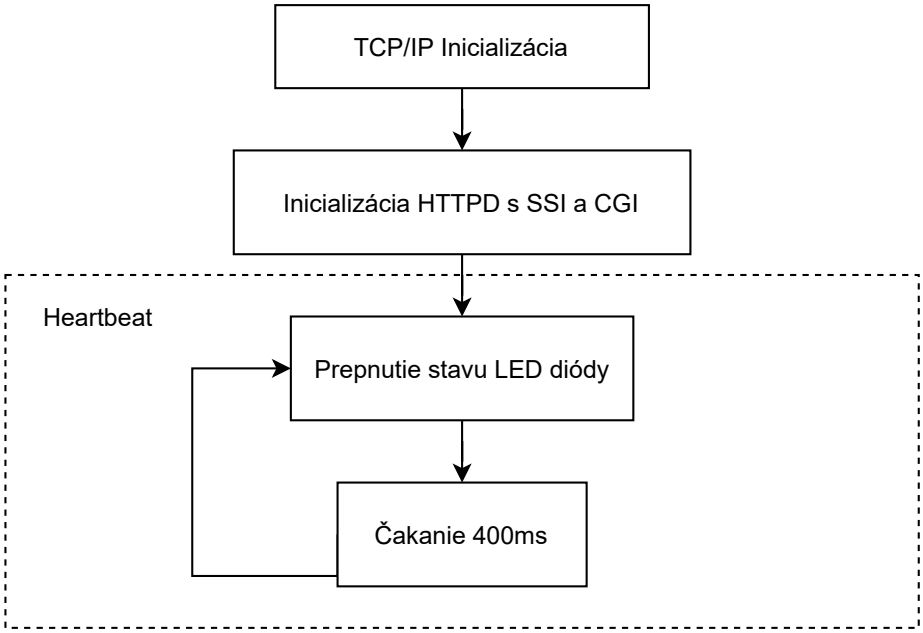
pade sa jedná iba o jednu hodnotu a to je 0x42. Táto hodnota má iba jedno opodstatnenie, slúži k identifikácií, či sú na danom mieste zapísané validné dáta. Pre lepšiu kontrolu dát je nutné implementovať nejakú formu výpočtu nad uloženými dátami. V tomto prípade sa ale jedná iba o konkrétnu konštantu, keďže prípad výskytu jednej konkrétnej hodnoty je dostatočne nízka pre tento účel. Na druhom mieste sa nachádza počet IP adries ktoré nasledujú za týmito dvoma údajmi. Tento údaj je veľmi dôležitý, pretože určuje ako ďaleko v pamäti sa vyskytujú údaje, zároveň v prípade nesprávnej hodnoty môže dôjsť ku nedostatku dostupného miesta vo vnútornej pamäti mikrokontroléra. Preto musí byť táto hodnota kontrolovaná proti príliš vysokej hodnote. Maximálna veľkosť tejto hodnoty je určená globálnou konštantou v zariadení kvôli bezpečnosti. Pre každú uloženú IP adresu (podsieť) je využitá ďalšia trojica slov. V prvej dvojici je uložená hodnota samotnej IP adresy a v tretej je uložená maska podsiete. V prípade povolenia iba konkrétnej IP adresy je táto hodnota nastavená na 32, čo je aj najvyššia hodnota, ktorú môže táto hodnota nadobudnúť. V prípade vyššieho čísla naskytla chyba čítania dát a všetky dáta sú pokladané za neplatné.

5.6 FreeRTOS úlohy

Obsluha jednotlivých súčastí zariadenia je rozdelená do niekoľkých úloh, ktoré zaisťujú istú formu paralelného správania pre užívateľa. Pre efektívnejšie využitie pamäťového priestoru zariadenia bolo využitých čo najmenej RTOS úloh, no stále dostatok pre dosiahnutie paralelného správania. Na pozadí existujú aj ďalšie systémové úlohy mimo tých, ktoré sú spomenuté nižšie, tieto systémové úlohy majú na starosť obsluhu niektorých protokolov ako je HTTP, DNS a podobných. No tieto úlohy sú súčasťou zabudovaných knižníc výrobcu a ich popis sa dá nájsť v príslušnej dokumentácií.

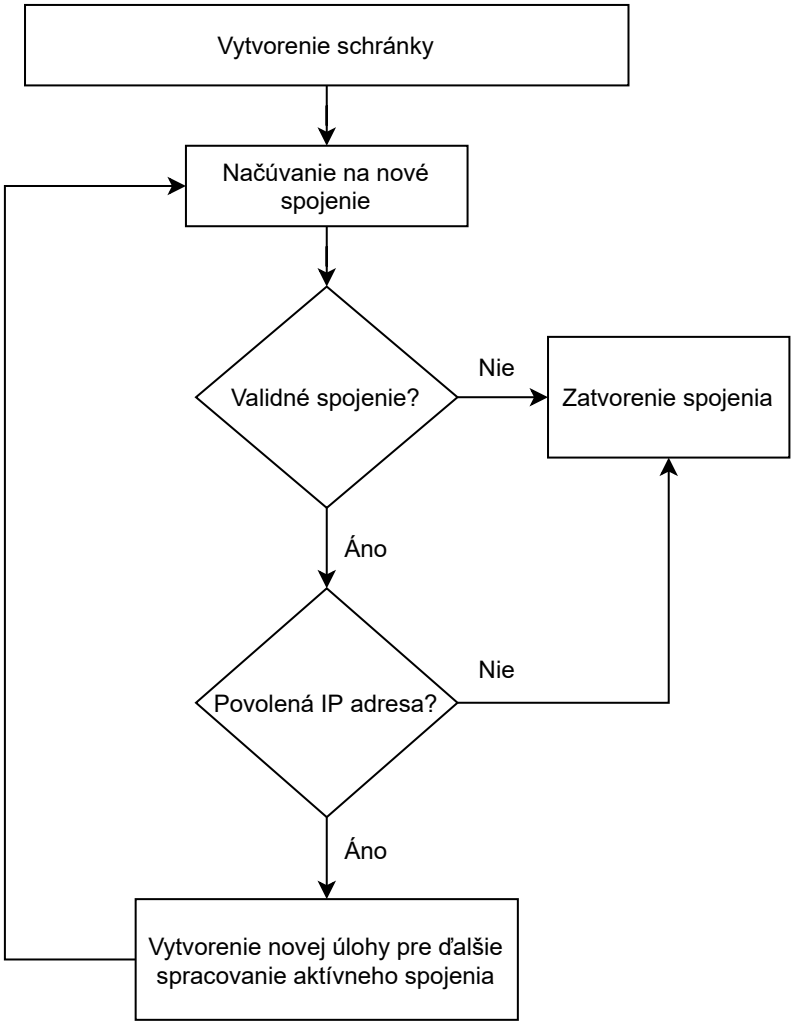
Úlohy, špecifické pre firmvér implementovaného zariadenia:

- **StartThread** (obr. 5.7) má za úlohu inicializovať niektoré prvky sieťového nastavenia a následne poskytuje užívateľovi spätnú odozvu vo forme tepu (ang. heartbeat). Jedná sa o informáciu o stave zariadenia prostredníctvom LED diódy. V prípade, ak dôjde v zariadení na chybu, z ktorej sa nevie zotaviť táto dióda prestane blikať.

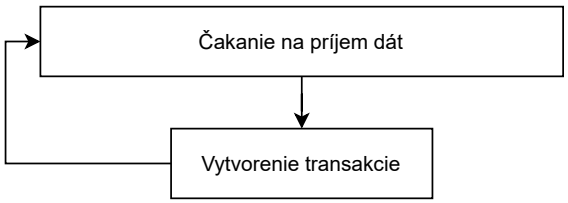


Obr. 5.7: Diagram popisujúci fungovanie StartThread úlohy.

- **TCPListenerThread** (obr. 5.8) spracováva všetky prichádzajúce spojenia na definovanom TCP Modbus porte. Najdôležitejšou úlohou tejto úlohy je identifikovať, či sa jedná o správny paket od povoleného zdroja a následne vytvoriť novú dynamickú úlohu, ktorej predá spojenie. Táto nová úloha (obr. 5.9) slúži iba na jednoduchú obsluhu prichádzajúcich paketov od verifikovaného zdroja. Tieto pakety iba zaobalí do štruktúry tzv. „transakcie“ a spolu s cieľom, na ktorý má zariadenie odoslať odpoveď, zaradí túto transakciu do fronty transakcií.

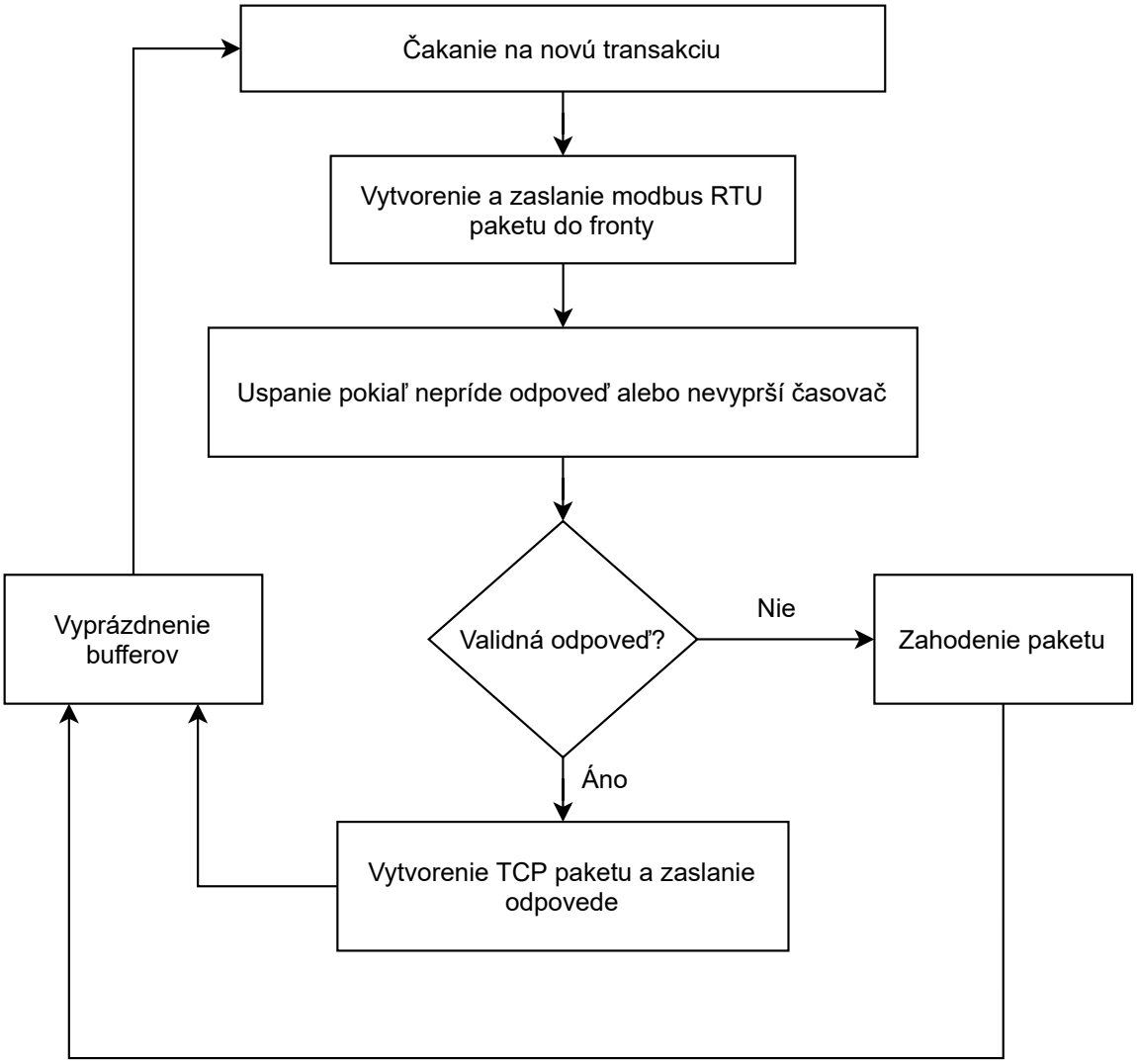


Obr. 5.8: Diagram popisujúci fungovanie TCPListenerThread úlohy.



Obr. 5.9: Diagram popisujúci fungovanie dynamickej ServeClient úlohy, ktorá je vytvorená v prípade pripojenia nového klienta.

- **TransactionHandlerTask** (obr. 5.10) spracováva dáta prichádzajúce do fronty transakcií, zaisťuje správne priradenie modbus RTU paketu k modbus TCP paketu. Keďže po modbus RTU zbernici môže komunikovať iba jedno zariadenie naraz a nie je predom špecifikované vybratie iba jedného konkrétneho zariadenia pre komunikáciu musí byť dodržané poradie odchádzajúcich paketov a odpovedí na dané pakety. Táto úloha zaisťuje, že v jeden moment sa čaká iba na jednu odpoveď z konkrétneho zariadenia naraz aj napriek tomu, že zariadenie dokáže prijímať niekoľko modbus TCP paketov súčasne. Po každom odoslanom RTU pakete je spustený časovač, ktorý označuje maximálny čas pre čakanie na odpoveď. Hodnota tohto časovača sa dá dynamicky nastaviť cez webové rozhranie. Ak nepríde žiadna odpoveď do konca tohto časovača je transakcia zahodená a neodosiela sa na zariadenie žiadna odpoveď. Naopak ak príde korektná odpoveď od zariadenia (alebo výnimka v prípade chyby) je táto odpoveď zaobalená do modbus TCP paketu a odoslaná na zariadenie od ktorého prišla požiadavka.



Obr. 5.10: Diagram popisujúci fungovanie TransactionHandlerTask úlohy.

- **ModbusMasterThread** úloha je implementovaná v rámci modbus knižnice, ktorá bola z veľkej časti prebratá z Arduino knižnice pre komunikáciu po modbus RTU zbernici. Táto úloha bola upravená pre efektívnejšiu komunikáciu na mikrokontroléri STM32 a to využitím UART periférie a jej prerušení. Prerušenie je vyvolané po prijatí jedného bajtu na zbernici, po ktorom je vždy resetovaný časovač. Tento časovač má za úlohu určiť koniec správy, keďže predom nie je známe aká bude odpoveď dlhá. Prednastavená hodnota časovača obmedzuje najmenšiu rýchlosť komunikáciu pre zariadenie. Vybraná hodnota postačuje pre komunikáciu so zariadeniami v rýchlosti aspoň 1800 bitov za sekundu, čo je dostačujúce pre väčšinu dostupných zariadení. Pri použití pomalšieho resp. rýchlejšieho zariadenia je možné túto hodnotu upraviť. V prípade využitia nižšej hodnoty časovača je možné komunikáciu značne zrýchliť. V knižnice je implementovaná aj kontrola CRC súčtu pre overenie validity prijatej správy.

Kapitola 6

Funkčnosť a testovanie

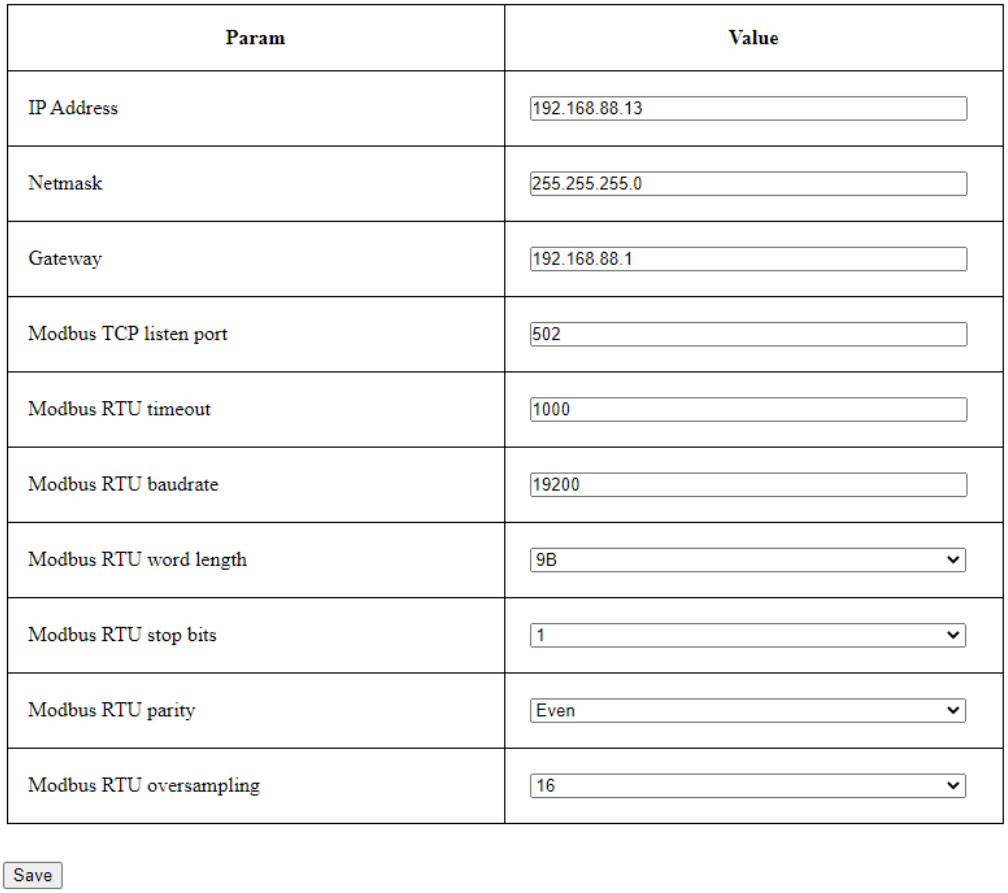
Funkčnosť zariadenia bola testovaná s Modbus zariadeniami PPL110 (obr. 6.1). Tieto zariadenia umožňujú čítanie teploty z RTD (ang. Resistance Temperature Detector) senzora. PPL110 umožňuje čítanie konkrétnej hodnoty rovnako ako zápis do registra pre úpravu krivky alebo odchýlky teploty senzora vďaka čomu môžeme testovať zápis aj čítanie registrov po Modbus RTU. Ako konkrétny senzor bol vybraný senzor Pt100, ktorý patrí medzi nepoužívanejšie senzory v rámci industriálneho využitia vďaka jeho vysokej presnosti a širokého rozsahu meranej teploty.

Obr. 6.1: Modbus RTU prevodník pre meranie teploty pomocou RTD senzorov (vľavo) a tepelný RTD senzor PT100 (vpravo).

Pre komunikáciu so zariadením bol využitý program Modpoll. Jedná sa o jednoduchú aplikáciu v príkazovom riadku, ktorá umožňuje generovanie Modbus TCP a RTU paketov. Zároveň číta odpovede zo zariadenia a implementuje aj chybné funkčné Modbus kódy, ktoré informujú o chybe pri požiadavke na zariadenie.

6.1 Konfigurácia prostredníctvom webového rozhrania

Konfigurácia zariadenia je možná pomocou jednoduchého webového rozhrania (obr. 6.2), ktoré umožňuje nastaviť základne parametre zariadenia popísané v sekcií 5.5. Pre využitie protokolu DHCP pre nastavenie parametrov je nutné nechať IP adresu zariadenia, masku siete a IP adresu brány prázdnu. V prípade tohto testu sú ponechané parametre rovnaké, aké sú zobrazené na obrázku webového rozhrania.



Obr. 6.2: Konfigurácia zariadenia prostredníctvom webového rozhrania.

V rámci konfigurácie je nutné nastaviť aj povolené IP adresy (podsiete), z ktorých je možne posielať Modbus TCP požiadavky na prevodník. Prevodník spojenia z nepovolených IP adries zahadzuje bez čítania obsahu prijatých paketov.

Konfigurácia povolených IP adries (podsieti) prebieha rovnako cez webové prostredie (obr. 6.3). Aj keď počet IP adries nie je softvérovo obmedzený je treba brať na vedomie, že kapacita Flash pamäte je obmedzená. Pre každú uloženú IP adresu (podsieť) sa ukladá 6 bajtov (4 bajty pre IP adresu a 2 bajty pre masku). Preto je možné do pamäte uložiť relatívne vysoký počet IP adries (podsieti) a nemalo by to užívateľa obmedzovať.

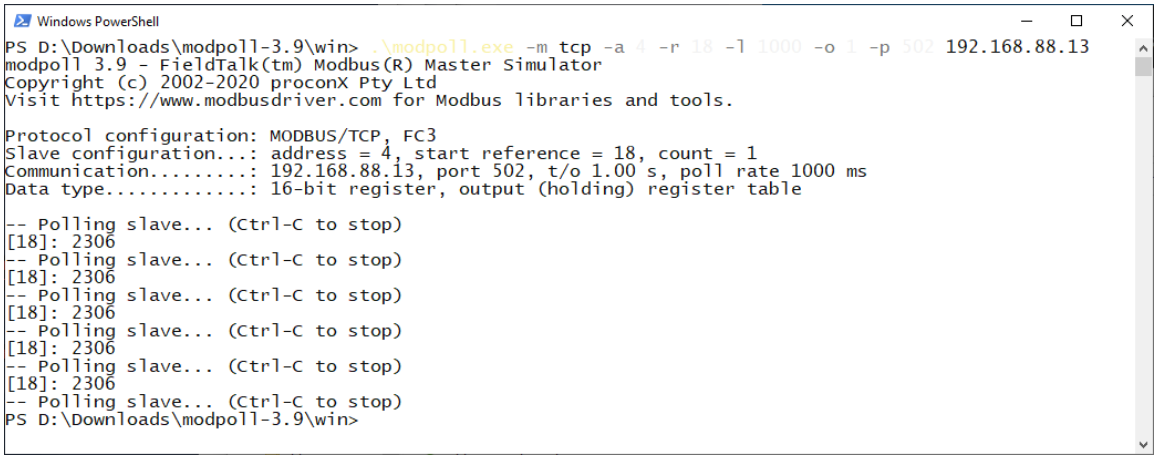
6.2 Meranie teploty prostredníctvom prevodníka PL110

K získaniu údaju o teplote nám pomôže už vyššie spomenutý program Modpoll. Pre jednoduchý test potrebujeme získať hodnotu z registra na adrese 18 (0x12). V tomto registri sa nachádza priamo hodnota nameranej teploty. Pri prvom zmeraní teploty je jasné, že sa nejedná o správnu hodnotu a treba ju upraviť (nameraná hodnota je približne o 3 stupne nižšia). Zápisom do registra 4140 (0x102C) je možné túto hodnotu kompenzovať. Po porovnaní s druhým (už nakalibrovaným) teplomerom vychádza táto nová zapísaná hodnota na 500. Po zapísaní tejto hodnoty vidíme v registri už správne nakalibrovanú hodnotu vnútornej teploty (obr. 6.4).

Delete	Remote IP		Mask
	192.168.1.0	/	24
	192.168.88.0	/	24
	192.168.88.53	/	32
☐		/	

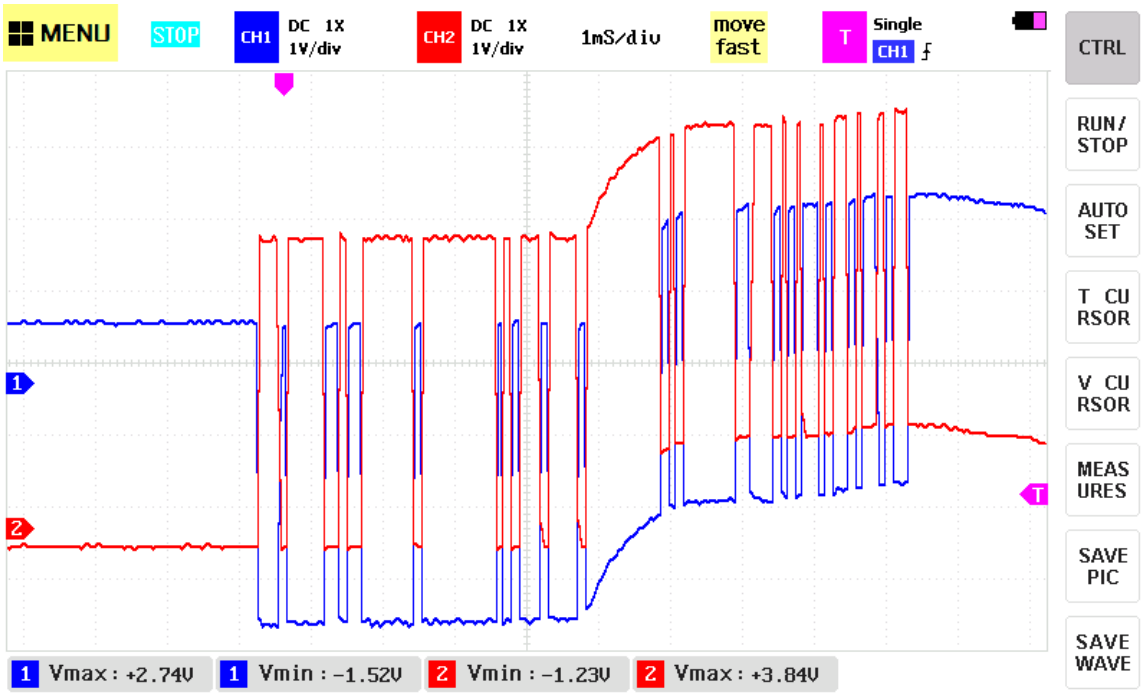
Save

Obr. 6.3: Pridávanie povolených IP adries (podsietí) prostredníctvom webového rozhrania.

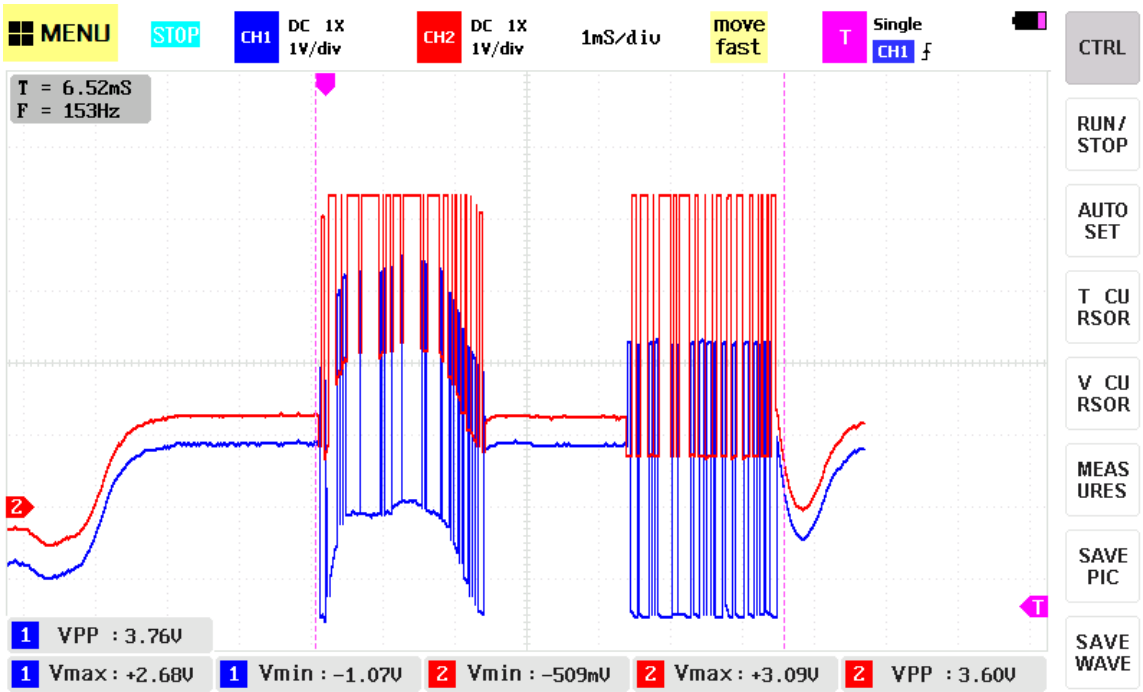


Obr. 6.4: Meranie teploty pomocou programu Modpoll, zariadenia PPL110 a senzora Pt100.

Pri testovaní funkčnosti RS485 zbernice bol využitý osciloskop, vďaka ktorému boli doladené posledné konfiguračné chyby Modbus RTU protokolu. Priebeh požiadavky pre prečítanie teploty zo zariadenia PL110 je možné sledovať na obrázku 6.5. Na obrázku 6.6 je možné sledovať ako požiadavku, tak aj korektnú odpoveď zariadenia.



Obr. 6.5: Priebeh požiadavky na MODBUS RTU zbernici pre slave zariadenie.



Obr. 6.6: Priebeh komunikácie dvoch zariadení na MODBUS RTU zbernici.

Kapitola 7

Záver

Výsledkom tejto diplomovej práce je navrhnutý a implementovaný kompletný vstavaný systém, schopný prevodu medzi priemyselnými protokolmi Modbus TCP a Modbus RTU. Pre prvotný vývoj softvéru bol využitý vývojový kit NUCLEO - F746ZG. Tento softvér bol následne upravený pre vybraný mikrokontrolér STM32F427VGT a navrhnutú dosku plošných spojov (DPS).

Výsledný systém je možné napájať 4 až 12 V s využitím lineárneho napäťového regulátora AMS1117, zároveň je chránený proti prepólovaniu vďaka ochrannej schottkyho dióde. Pre ochranu vnútornej konfigurácie zariadenia je využitá integrovaná Flash pamäť mikrokontroléra, ktorá plní funkciu externej EEPROM pamäte, vďaka čomu redukuje nutný počet komponentov. Pri prípadnom výpadku napájania a opätovnom spustení je teda zariadenie schopné načítať internú konfiguráciu a pokračovať bez zásahu užívateľa.

Pre komunikáciu po RS485 rozhraní je využitý integrovaný obvod MAX3485 a UART periféria mikrokontroléra, vďaka ktorej je dostupná dostatočná rýchlosť pre komunikáciu bez zaťaženia samotného jadra mikrokontroléra. Vytvorená DPS je osadená JTAG rozhraním, umožňujúcim programovanie externým JTAG/SWD programátorom.

Pripojené Ethernetové rozhranie využíva integrovanú MAC vrstvu mikrokontroléra, integrovaný obvod LAN8710A implementujúci fyzickú vrstvu Ethernetu a RJ45 konektor.

Softvérová časť je navrhnutá s využitím operačného systému reálneho času FreeRTOS, ktorý umožňuje prehľadnú definíciu úloh a súbežné spracovanie viacerých spojení. Pre konfiguráciu zariadenia bola vytvorená webová stránka, v ktorej je možné zmeniť základné parametre systému a povoliť prístup zariadeniam.

Softvér zariadenia obsahuje základnú ochranu pred nepovoleným prístupom povolením iba nakonfigurovaných IP adries (podsietí). V prípade zahltenia systému bude dochádzať k postupne vyššej pravdepodobnosti zahodenia prichádzajúceho paketu. Napriek týmto opatreniam sa predpokladá, že vstavaný systém bude umiestnený v rámci vnútornej, chránenej siete.

Pre zníženie výslednej ceny zariadenia bol pri návrhu DPS využitý dvojvrstvový návrh a súčiastky boli vybrané vo vhodnej veľkosti pre pohodlné ručné osadenie.

Systém bol testovaný s využitím voľne dostupného programu ModPoll a RTU zariadeniach PPL110. V kapitole 6 je dokázaná funkčnosť hotového vstavaného systému. V práci boli popísane aj základné princípy komunikácie, možnosti pripojenia Ethernetového rozhrania k vstavanému zariadeniu, ako aj samotné priemyselné protokoly Modbus RTU a Modbus TCP. Výsledná schéma zariadenia je uvedená v prílohe 1.

Literatúra

[1] IEEE Standard for Ethernet. *IEEE Std 802.3-2015 (Revision of IEEE Std 802.3-2012)*. 2016, s. 1–4017. DOI: 10.1109/IEEESTD.2016.7428776.

[2] *SN65HVD7x 3.3-V Supply RS-485 With IEC ESD protection* [online]. Marec 2019 [cit. 2020-10-03]. Dostupné z: https://www.ti.com/lit/ds/symlink/sn65hvd75.pdf.

[3] ARCELECT.COM. *RS232 Data Interface*. [cit. 2021-5-12]. Dostupné z: https://arcelect.com/rs232.htm.

[4] BARRY, R. *Mastering the FreeRTOS^{TM} Real Time Kernel*. Real Time Engineers Ltd., 2016. 44-97 s. Dostupné z: https://www.freertos.org/fr-content-src/uploads/2018/07/161204_Mastering_the_FreeRTOS_Real_Time_Kernel-A_Hands-On_Tutorial_Guide.pdf.

[5] BÉLAI, I. a DURFINA, M. *PRIEMYSELNÝ ETHERNET* [online]. [cit. 2021-1-10]. Dostupné z: https://senzor.robotika.sk/pkom/html/kapitola6.htm.

[6] CATSOULIS, J. *Designing Embedded Hardware*. 2. vyd. O'Reilly Media, Inc., 2005. ISBN 9780596007553.

[7] GUARESE, G. B. M., SIEBEN, F. G., WEBBER, T., DILLENBURG, M. R. a MARCON, C. *Exploiting Modbus Protocol in Wired and Wireless Multilevel Communication Architecture*. IEEE, 2012. ISBN 978-1-4673-5747-0.

[8] INTERFACEBUS.COM. *EIA-232 Bus*. [cit. 2021-5-12]. Dostupné z: http://www.interfacebus.com/Design_Connector_RS232.html.

[9] JOELIANTO, E. a HOSANA. *Performance of an industrial data communication protocol on ethernet network*. IEEE, 2008. ISBN 978-1-4244-1979-1.

[10] MAXIMINTEGRATED.COM. *MAX3485 Datasheet* [online]. [cit. 2021-4-18]. Dostupné z: https://datasheets.maximintegrated.com/en/ds/MAX3483-MAX3491.pdf.

[11] MELLON, L. *Data Transmission – Parallel vs Serial* [online]. [cit. 2021-1-10]. Dostupné z: https://www.quantil.com/content-delivery-insights/content-acceleration/data-transmission/#:~:text=There%20are%20two%20methods%20used,same%20time%20over%20multiple%20channels.

[12] MICROCHIP.COM. *LAN8710A/LAN8710Ai Datasheet* [online]. [cit. 2021-4-18]. Dostupné z: https://ww1.microchip.com/downloads/en/DeviceDoc/00002164B.pdf.

[13] MULTICOMP. *RJ45 Receptacle R/A Tabup W/F 100/1000M, Shielded with Led* [online]. [cit. 2021-4-26]. Dostupné z: http://www.farnell.com/datasheets/2189841.pdf.

[14] Noergaard, T. *2.2.4 OSI Model*. Elsevier, 2013. 54-67 s. ISBN 978-0-12-382196-6.

[15] omega.com. *RS485, RS422 and RS232*. 2019 [cit. 2021-5-12]. Dostupné z: https://www.omega.com/en-us/resources/rs422-rs485-rs232.

[16] Ozeki. *Modbus TCP* [online]. [cit. 2021-1-15]. Dostupné z: http://www.ozeki.hu/p_5858-ozeki-modbus-tcp.html.

[17] ozeki.hu. *Modbus RTU*. [cit. 2021-5-13]. Dostupné z: https://ozeki.hu/p_5854-ozeki-modbus-rtu.html.

[18] Reynders, D. a Wright, E. *Practical TCP/IP and Ethernet Networking*. Newnes, 2003. ISBN 07506 58061.

[19] Rojas, C. a Morell, P. Guidelines for Industrial Ethernet infrastructure implementation: A control engineer's guide. In: *2010 IEEE-IAS/PCA 52nd Cement Industry Technical Conference*. 2010. DOI: 10.1109/CITCON.2010.5469772.

[20] RTAutomation. *An Introduction to Modbus RTU Addressing, Function Codes, and Modbus RTU Networking Overview* [online]. [cit. 2020-10-17]. Dostupné z: https://www.rtautomation.com/technologies/modbus-rtu.

[21] STElectronics. *EEPROM emulation in STM32F40x/STM32F41x microcontrollers* [online]. [cit. 2021-5-7]. Dostupné z: https://www.st.com/resource/en/application_note/dm00036065-eeprom-emulation-in-stm32f40x-stm32f41x-microcontrollers-stmicroelectronics.pdf.

[22] Strauss, C. *Practical Electrical Network Automation and Communication Systems*. 1. vyd. Newnes, 2003. ISBN 0750658010.

[23] Tamboli, S., Rawale, M., Thoraiet, R. a Agashe, S. Implementation of Modbus RTU and Modbus TCP communication using Siemens S7-1200 PLC for batch process. In: *2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM)*. 2015, s. 258–263. DOI: 10.1109/ICSTM.2015.7225424.

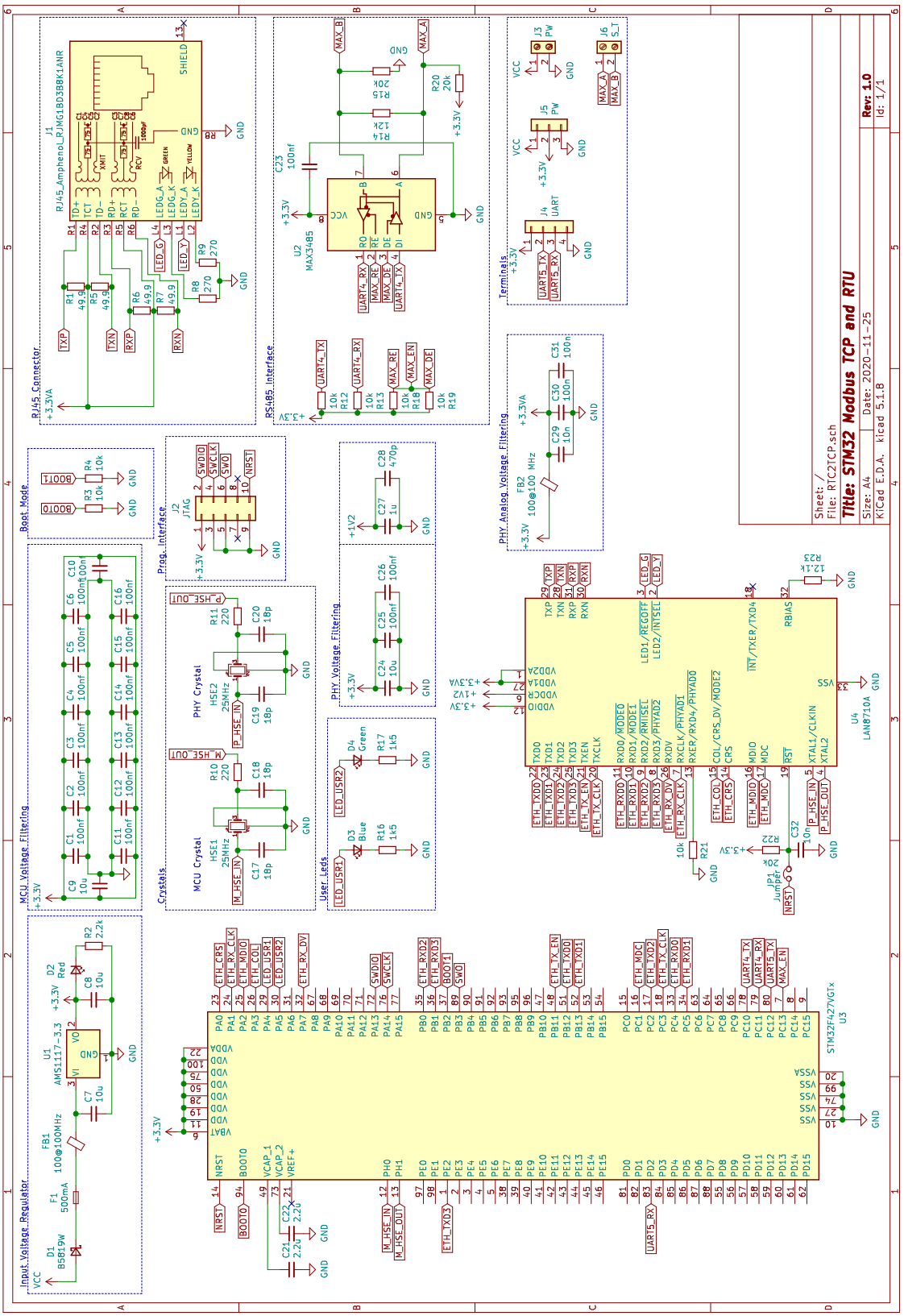
[24] TexasInstruments. *AN-1405 DP83848 Single 10/100 Mb/s Ethernet Transceiver Reduced Media Independent Interface^{TM} (RMII^{TM}) Mode* [online]. [cit. 2021-1-11]. Dostupné z: https://www.ti.com/lit/an/snla076a/snla076a.pdf.

[25] Urrea, C., Morales, C. a Muñoz, R. *Design and implementation of an error detection and correction method compatible with MODBUS-RTU by means of systematic codes* [online]. Máj 2016 [cit. 2020-10-04]. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0263224116302123.

[26] wikipedia.org. *Model OSI* [online]. 2019 [cit. 2021-04-20]. Dostupné z: https://sk.wikipedia.org/wiki/Model_OSI.

[27] Černohorský, J. a Srovnal, V. *Systémy reálneho času, AT&P journal*. [cit. 2021-4-28]. ISSN 1335-2237.

[28] Šimíček, V., Drápalová, J. a Ilčík, V. *Spojovací technika*. Code Creator, s.r.o., 2015. ISBN 978-80-88058-14-4.



Obr. 1: Schéma prototypu vstavaného zariadenia pre preklad medzi protokolmi Modbus RTU a Modbus TCP