VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

KOMUNIKAČNÍ HARDWARE PRO I4.0 BARMAN
COMMUNICATION HARDWARE FOR I4.0 BARMAN

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Richard Kubíček

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Václav Kaczmarczyk, Ph.D.

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Kybernetika, automatizace a měření**

Ústav automatizace a měřicí techniky

Student: Bc. Richard Kubíček ***ID:*** 164753

Ročník: 2 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Komunikační hardware pro I4.0 Barman

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte hardware komunikačního rozhraní, které bude použitelné jako vstupní brána pro poskytování informací a řízení autonomních výrobních buněk testbedu Industry 4.0.

1. Seznamte se s testbedem průmyslu 4.0 a s jeho funkcí
2. Definujte požadavky na komunikační rozhraní umožňující distribuované řízení a zvolte vhodný komunikační protokol.
3. Navrhněte strukturu univerzálního hardwarového modulu pro implementaci komunikačního rozhraní procesních buněk tesbedu.
4. Zkonstruujte zařízení, ověřte jeho funkčnost
5. Navrhněte a realizujte programové moduly sloužící jako interface k hardwarovým perifériím
6. Řešení otestujte, dokumentujte a vyhodnoťte dosažené výsledky.

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 1.6.2020

Vedoucí práce: Ing. Václav Kaczmarczyk, Ph.D.

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

ABSTRAKT

Tato práce se zabývá popisem a realizací AAS modulu. V práci je popsán výběr jednotlivých komponent zařízení. Modul bude součásti projektu testbedu Barman. Tento projekt je realizován v souladu s trendy Průmyslu 4.0. Zařízení bude sloužit jako čtečka informací, ingrediencí na NFC tagu umístěném na skleničce. Tyto informace budou sloužit nadřízené buňce k rozhodnutí jaké operace budou provedeny s položenou skleničkou. V rámci této práce bude popsán a vytvořen program pro obsluhu hardware. Tento program bude dále komunikovat s nadřazeným programem řídicím samotnou funkcionalitu buňky.

KLÍČOVÁ SLOVA

AAS modul, NFC, Průmysl 4.0, testbed Barman, PoE, Modus TCP, Python, OLED

ABSTRACT

This thesis deals with description and implementation of AAS module. This thesis describes the selection of individual components of device. This module will be a part of the testbed Barman project. This project is implemented in accordance with Industry 4.0 trends. Device will be work as reader of information and ingredients from NFC tags placed on a glass. With this information, parent cell can make decision which action will be done with glass. In this thesis, software for hardware operation with AAS module will be created and described. This program will further communicate with the parent program controlling the functionality of the cell itself.

KEYWORDS

AAS module, NFC, Industry 4.0, testbed Barman, PoE, Modbus TCP, Python, OLED

KUBÍČEK, Richard. Komunikační hardware pro I4.0 Barman [online]. Brno, 2020 [cit. 2020-05-31]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/126920. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce Václav Kaczmarczyk.

PROHLÁŠENÍ

Prohlašuji, že svoji diplomovou práci na téma Komunikační hardware pro I4.0 Barman jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.
Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne .....31.5.2020..... …….....
(podpis autora)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce Ing. Václavovi Kaczmarczykovi, Ph.D. a konzultantům Ing. Jakubovi Streitovi a Ing. Ondřeji Baštánovi za odborné vedení, konzultace, cenné rady a návrhy k práci.

V Brně dne .....31.5.2020..... …….....
(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Cílem této práce je vytvoření a realizace elektrického zařízení, umožňující čtení RFID tagů ze skleniček na nápoje. Realizované zařízení bude ve spojení s řídícím programem z tagů číst receptury na přípravu nápojů. Tyto receptury budou předávány nadřízeným PLC buňkám pomocí OPC UA, STEP 7 nebo Modbus TCP protokolu.

V rámci této práce bude popsán návrh, výběr komponent a realizace tohoto zařízení.

Dále bude rozebrán obslužný software umožňující nízkoúrovňové ovládání tohoto zařízení tak, aby další obsluhu a komunikaci s dalšími buňkami mohl zajišťovat jiný program. Bude rozebrán i nadstavbový software nad nízkoúrovňové ovládání, který bude propagovat toto ovládání do Modbus TCP datového prostoru.

Poté bude vytvořen postup pro instalaci a používání napsaných programů.

1 PRŮMYSL 4.0 – TESTBED BARMAN

Koncepce testbedu Barmana pro Průmysl 4.0 byla vytvořena za účelem zkvalitnění výuky automatizace na Fakultě elektrotechniky a komunikačních technologií, VUT v Brně, Ústavu automatizace. Cílem projektu je umožnit studentům určitý vhled do života absolventů průmyslové automatizace v praxi, kde se nelze spoléhat jen naznalosti v oblasti návrhu algoritmů, regulátorů, ale jsou nutné i znalosti interoperability IT systémů, základy designů mechanických systémů apod.

1.1 Průmysl 4.0

Průmysl 4.0 je zkráceně řečeno pojem, který byl do češtiny přeložen z původního Industry 4.0. Pojem Industry 4.0 byl prezentován 8. dubna 2013 na veletrhu v německém Hannoveru členy skupiny „Working Group Industry 4.0“. Zde došlo ke spojení sil společnosti Siemens a německé vlády s cílem intenzivní propagace vývoje a použití nových technologií jak pro průmysl, tak i do domácností. Tuto iniciativu následně převzaly další země a evropské společnosti.

Výsledkem čtvrté průmyslové revoluce, jak bývá někdy Průmysl 4.0 nazýván, by mělo být propojení již existujících a používajících se moderních zařízení pro výrobu, jako jsou například manipulační roboti, CNC stroje apod. s výrobky, dohledovým centrem, případně i se zákazníky. Toto propojení může být realizováno například pomocí sítě IoT.

Nástupem Průmyslu 4.0 by mělo dojít i k nárůstu objemu sbíraných dat. Může se jednat o data získaná z komunikace stroj – stroj, stroj – člověk nebo i člověk – člověk. Analýzou nasbíraných dat bude možné optimalizovat chod zařízení ve firmě.

Koncept Průmyslu 4.0 v oblasti průmyslové výroby je přechod z jednotlivých automatizovaných výrobních jednotek na plně integrované, automatizované a průběžně optimalizované prostředí. Základními prvky, které jsou aplikovatelné v produkci jsou: virtualizace, decentralizace řízení, modularita, schopnost měnit vlastnosti při změnách nastavení. Dále také integrace systému z hlediska propojení objednávkového systému, systému pro podporu výroby až po systém zajišťující zákaznickou podporu. Důležitou vlastností Průmyslu 4.0 je také schopnost komunikovat s ostatními jednotkami v továrně po dedikované síti (čerpáno z [1]).

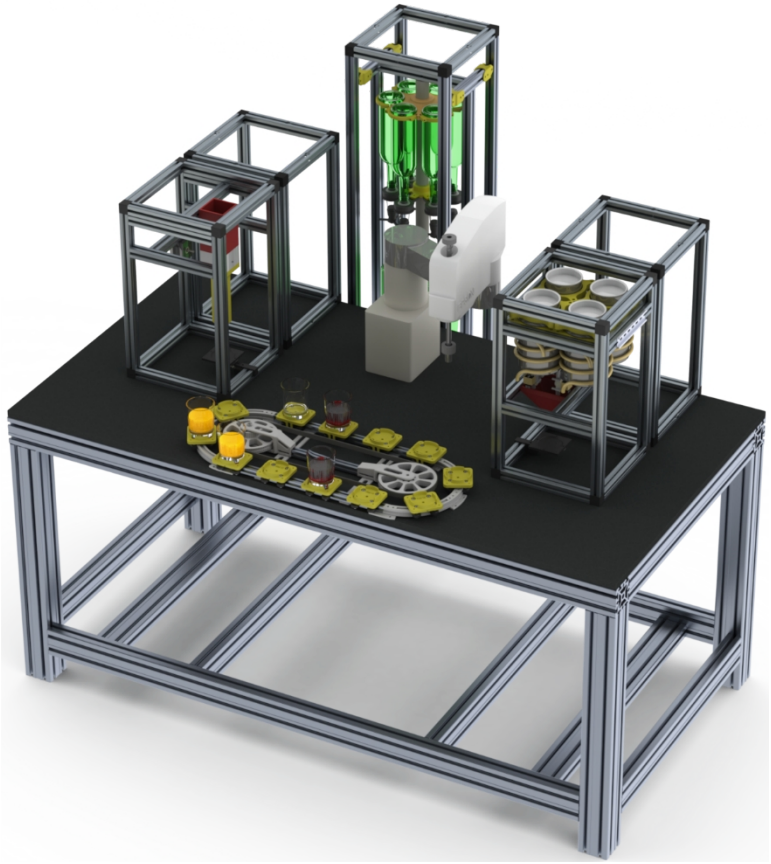
1.2 Testbed Barman

Testbed je tvořen několika funkčními jednotkami, kde se každá z nich stará o část výroby zamýšleného nápoje. Celá funkcionalita Barmana je vytvořena na ploše o rozměrech 2 x 1 m. Jednotky jsou tvořeny z hlinkových profilů. Testbed obsahuje buňky pro dávkování nealkoholických nápojů, například šťáv, pro dávkování

alkoholických nápojů, jednotku s drtičem ledu, shaker, sklad skleniček, manipulátor a pás. Vizualizaci testbedu zobrazuje Obrázek 1.

Jednotlivé buňky budou využívat recepturu umístěnou v NFC tagu, který bude nalepen na dno skleničky. Tuto recepturu bude číst právě zařízení popisované v této práci. V paměti NFC tagu budou umístěny nejen receptury ale i stavy, ve kterých se aktuálně objednávka, respektive daná sklenička, nachází.

Výroba nápoje započne vytvořením objednávky uživatelem přes webové rozhraní. Po potvrzení objednávky je objednávka předána do ERP systému. Následně je objednávka předána do MES systému. Výrobní systém objednávku přijme a zařadí ji do výrobní fronty.



Obrázek 1 - náhled na namodelovaný testbed

2 NÁVRH ZAŘÍZENÍ

2.1 Požadavky na zařízení

Koncepce Průmyslu 4.0 a tedy i koncepce testbedu Barmana vyžaduje propojení jednotlivých buněk, respektive částí digitalizované továrny pomocí moderních komunikačních prostředků, jako je průmyslový ethernet. Z tohoto důvodu byla jedním z hlavních požadavků na realizaci AAS modulu možnost jeho připojení do ethernetu.

Za účelem čtení NFC tagů ze dna skleniček a získání tak receptury na výrobu nápoje je nutná přítomnost čtečky NFC tagů.

Pro konfiguraci zařízení je nevhodnější vzdálený přístup. Ten je podmíněn znalostí konkrétní IP adresy zařízení. Nejjednodušším způsobem, jak zjistit IP adresu modulu je její zobrazení na displeji. Displej je také vhodný pro zobrazení stavů, v nichž se zařízení nachází. S displejem je také vhodné umístit tlačítka sloužící pro základní navigaci na displeji, případně na jednoduchá nastavení.

Z hlediska napájení je standardem v oblasti průmyslové automatizace napájení pomocí stejnosměrného napětí 24 V. Jelikož je ale přítomno ethernetové rozhraní, které se bude vždy osazovat, je efektivnější opatřit zařízení možností napájení přes ethernetový kabel. Jedná se tedy o napájení pomocí Power over Ethernet, tzv. PoE.

NFC čtečka může například stále detekovat, zda byl přiložen tag. Pokud se tomuto nemůže řídicí procesor věnovat je vhodné použít i jiný systém na detekci položené skleničky s tagem na podložku.

K vizuální stránce zařízení patří signalizace například položené skleničky pomocí RGB LED.

Řídicí procesor musí být schopen komunikovat po síti ethernet s různými komunikačními parametry. Procesor také musí mít dostatek periférií, které budou použity ke komunikaci s dalšími součástmi zařízení, jako je NFC čtečka, displej, RGB LED a jiné.

2.2 Výběr komponent

2.2.1 NFC čtečka

NFC je technologie založená na principu pole působícího v blízkém okolí čipu. V zásadě se tak jedná o možnost rychlé a bezpečné obousměrné komunikace mezi dvěma prvky na krátké vzdálenosti. Vzdálenost, na kterou jsou schopny dva prvky spolu komunikovat, je menší než 10 cm.

NFC se nejčastěji používá ve třech různých režimech. Prvním režimem je propojení například mobilního telefonu s prvky v reálném světě. Telefon disponující technologií

NFC si může přečíst informace uložené na tagu a navštívit tak například internetové stránky, nebo změnit nastavení svého telefonu.

Dalším režimem je propojení dvou telefonů v režimu přímé komunikace. V tomto režimu dochází k výměně dat, například kontaktů, hesel k bezdrátovým sítím, mezi dvěma telefonními přístroji.

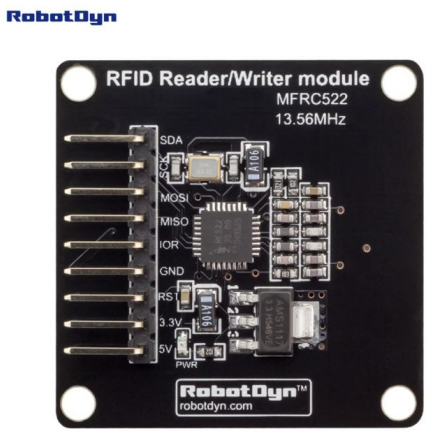
Posledním nejčastěji zmiňovaným režimem je emulace karty. V tomto režimu dochází k propojení platební karty s telefonem. Pro bezkontaktní platby se tak místo karty využívá přiložení telefonu.

NFC se řídí dvěma mezinárodními standardy. Prvním z nich je ISO/IEC 1800-3, ve kterém je definován kmitočet, na kterém probíhá komunikace, 13,56 MHz. Stejný standard se používá i pro čipové karty a obecně čipy. Tyto tagy jsou vždy pasivní a jejich napájení je z RFID čtečky. Komunikace mezi čtečkou a tagem je vždy zahájena čtečkou. Čtečka umí komunikovat v režimu plného duplexu, zatímco tag pouze v poloviční duplexu. Čtečka předává příkazy tagu pomocí malých změn fáze nosné napájecího pole. Tag odpovídá induktivní vazbou, kdy je napětí modulováno na subnosnou. Přenosová rychlost příkazů, tedy komunikace od čtečky k tagu, je 423,75 kbit/s. Přenosová rychlost odpovědí tagu je 105,9375 kbit/s. Obě tyto komunikace využívají modifikovanou frekvenční modulaci. Aby bylo možné v jednom okamžiku komunikovat s více tagy najednou, používá se ke komunikaci přístup MFTDMA, tedy multiplex s časovým dělením přístupu na více kmitočtech.

Druhým standardem je ISO/IEC 14443. Tato norma popisuje bezkontaktní karty, které jsou používány pro identifikaci. V normě jsou popsány komunikační protokoly pro komunikaci s těmito tagy. Tagy definované touto normou používají předchozí normu (ISO/IEC 1800-3) a říkají tedy, že komunikace probíhá právě na kmitočtu 13,56 MHz.

Pravděpodobně nejznámějším typem bezkontaktních karet, čipů a tagů v oblasti vysokých kmitočtů, tedy operujících na kmitočtu 13,56 MHz, je třída MIFARE® od firmy NXP Semiconductors. Značka MIFARE® obsahuje několik podtříd: MIFARE® DESFire®, MIFARE® Ultralight®, MIFARE® Plus®, MIFARE® Classic® a SmartMX®. Každá z těchto podtříd je určena pro jinou aplikaci. Například MIFARE® DESFire® je určeno pro použití v přístupových systémech, studentských kartách nebo třeba v zákaznických programech obchodů. Tyto zmíněné systémy často využívají pouze identifikátory karty nebo čipu. Ty si ukládají do svých databázových systémů a dále je využívají.

V oblasti NFC tagů je asi nejrozšířenější třídou NTAG®. I tato třída respektuje dříve zmíněné normy. Zatímco MIFARE® se častěji využívá pro identifikace, NTAG® bývá spojován s aplikacemi, ve kterých dochází ke čtení tagu pomocí mobilních telefonů. NTAG® tagy mívají také větší celkovou i uživatelskou paměť, do které lze zapisovat vlastní data. Výhodou například tagů NTAG21x je přítomnost interního počítače čtecích příkazů.



Obrázek 2 - RFID čtečka, převzato z [4]

Realizovat NFC na vlastním hardware by znamenalo navrhnout správně meandr s anténou na plošném spoji. Na internetu lze nalézt již hotové layouty, které lze importovat do návrhového systému, ale i tak je nutná znalost postupu návrhu takovéhoto plošného spoje. Ne vždy se však musí podařit navrhnout 100 % funkční čtečku v první iteraci návrhu designu. V zájmu větší jistoty při návrhu zařízení a s přihlédnutím k velikosti výrobní dávky bylo pro zvoleno využití již hotového zakoupeného modulu RFID čtečky na Obrázek 2. Zvolený modul využívá integrovaný obvod od NXP, MFRC522. Tento obvod disponuje třemi druhy komunikačních rozhraní a to SPI, UART a I^2C. Modul umožňuje čtení a zapisování dat na tagy z rodiny MIFARE® i NTAG®. V AAS modulu bude k řídicímu procesoru čtečka připojena přes rozhraní SPI.

2.2.2 Displej

Jako displej, který bude sloužit pro zobrazení IP adresy, provozních informací, případně pro zobrazení menu bylo zvoleno použití OLED displeje. Schopnost vyzařování světla u OLED displejů umožňuje zkonstruování velmi tenkých displejů nevyžadujících podsvícení. Zvolený displej disponuje na úhlopříčce 0,91 palců rozlišením 128 x 32 pixelů. Řízení tohoto displeje zajišťuje obvod SSD1306 po sběrnici I^2C z nadřazeného procesoru. Použití tohoto displeje se jevilo jako nejúspornější, z hlediska zabraného místa na plošném spoji a v samotné krabičce. Obrázek 3 ukazuje příklad takového displeje.

Obrázek 3 - OLED displej, převzato z [3]

2.2.3 Napájení

Jak již bylo zmíněno, v případě napájení pomocí 24 V, by stačilo použít relativně levné spínané zdroje, které často mívají maximální vstupní napětí do asi 30 V. Protože je ale zamýšleno mít možnost použít i napájení pomocí PoE, musel být zvolen výrazně dražší spínaný zdroj.

Napájení přivedené ethernetovým kabelem se řídí standardem IEEE 802.3. Existují dva typy napájení. Typ B využívá nevyužité páry drátů v ethernetovém kabelu. Používá páry 4-5 a 7-8. Typ A využívá takzvané fantómové napájení, kdy jsou data i napájení přenášena stejnými páry, tedy 1-2 a 3-6. PoE nejčastěji najde svoje uplatnění v zařízeních, které nejsou výkonově náročné a potřebují připojený ethernetový kabel. Typicky se jedná o telefony, bezpečnostní kamery, přístupové body. Zapojení typu A ukazuje Obrázek 4 a zapojení typu B zobrazuje Obrázek 5.

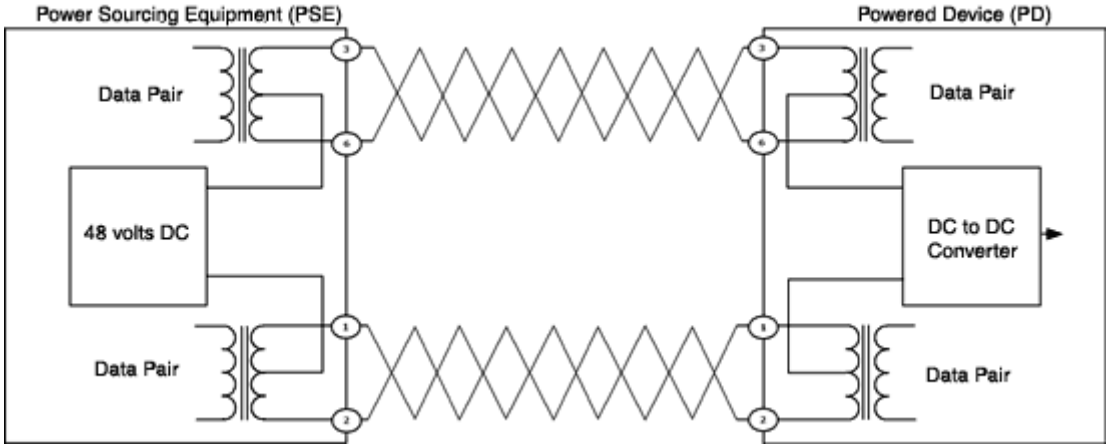
Tabulka 1 - parametry vybraných PoE standardů

vlastnost/standard	802.3af (802.3at typ 1)	802.3at typ 2 (PoE+)
výkon dodaný napájecím zařízením	15,4 W	30 W
výkon dodaný do napájeného zařízení	12,95 W	25,5 W
napěťový rozsah napájecího zařízení	44 - 57 V	50 - 57 V
napěťový rozsah napájeného zařízení	37 - 57 V	42,5 - 57 V

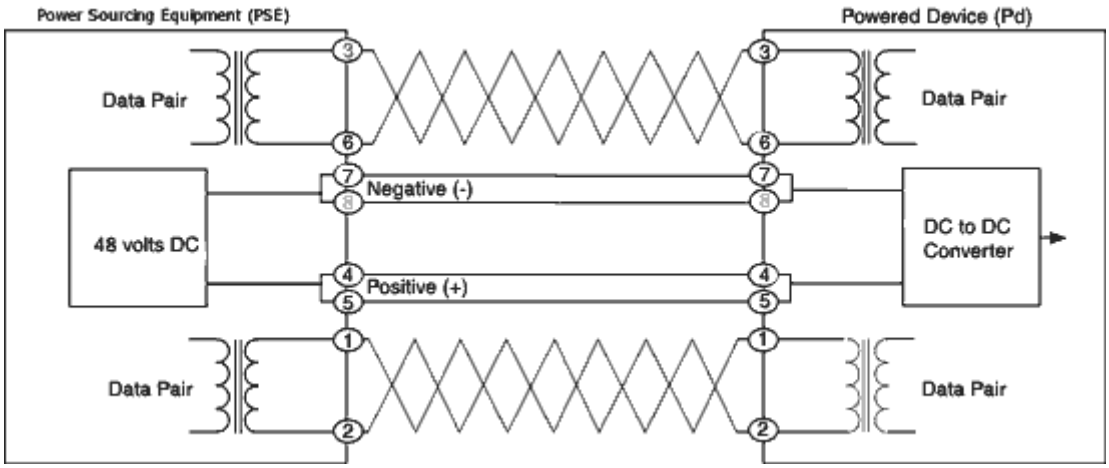
Zařízení napájená pomocí PoE využívají některý ze standardů normy IEEE 802.3. Některé ze standardů a jejich hlavní vlastnosti jsou zmíněny v Tabulka 1.

Typ B se také často využívá pro pasivně napájená zařízení. Používají se pasivní injektory, které jsou vloženy do cesty kabelu mezi napájené zařízení a síťový prvek (například router), který neumožňuje PoE napájení na svých portech. Tyto pasivní injektory jsou v zásadě jen 2 ethernetové konektory a stejnosměrný napájecí zdroj, jehož výstup je připojen na páry 4-5 a 7-8. Tento způsob napájení musí být podporován i napájeným zařízením, protože se jedná o nestandardní typ PoE. Obvykle napájecí

napětí v tomto případě nedosahuje úrovně 37 V, což je minimální úroveň napětí ve verzi 802.3af.



Obrázek 4 - PoE typ A, převzato z [2]



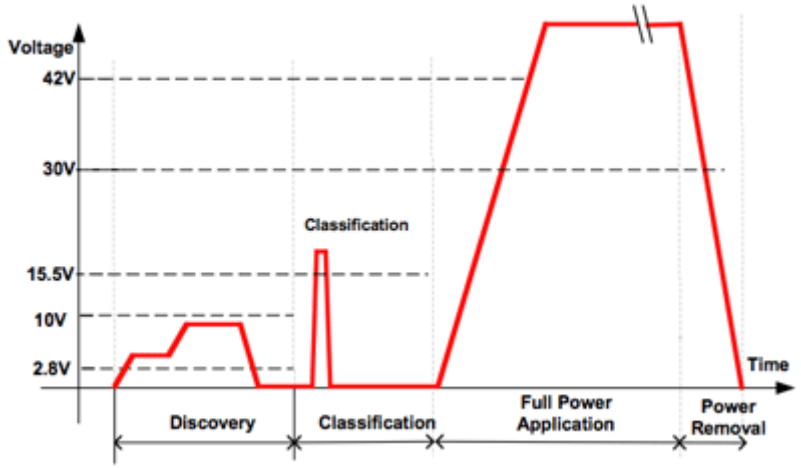
Obrázek 5 - PoE typ B, převzato z [2]

Tabulka 2 - klasifikační třídy PoE

třída	klasifikační proud [mA]	maximální příkon napájeného zařízení [W]	poznámka
0	0 - 4	12,95	klasifikace neimplementována
1	9 - 12	3,84	-
2	17 - 20	6,49	-
3	26 - 30	12,95	-
4	36 - 44	25,5	PoE+

Pokud by byl na napájeném zařízení pouze obvod se snižujícím měničem, mohlo by u některých napájecích zařízeních dojít k tomu, že nedetekují správně připojené zařízení, a to nebude napájeno. Z tohoto důvodu je nutné, aby byl mezi snižující měnič a ethernetový transformátor umístěn obvod, který zajistí, že bude zařízení správně detekováno. Musí dojít k detekci připojeného zařízení i ke správné klasifikaci zařízení. Napájecí zařízení po klasifikaci bude vědět jaký výkon bude napájené zařízení chtít

odebírat. Obrázek 6 ukazuje průběh klasifikace zařízení připojeného na PoE. V časovém úseku „Discovery“ dochází k detekci přítomnosti rezistoru 25 kΩ. Časový úsek „Classification“ má za úkol změřit proud, který identifikuje napájecí třídu. Pomocí tříd je možno napájecímu zařízení říct, že na daném portu může snížit výdej energie a přesměrovat ji například na jiný ethernetový port. Předposlední úsek „Full Power Application“ je úsek, v němž je napájené zařízení plně v provozu. Během posledního úseku „Power Removal“ dojde k odpojení napájeného zařízení. Třídy, do kterých mohou napájená zařízení patřit a proudy, při nichž jsou klasifikovány jsou v Tabulka 2.



Obrázek 6 - průběh klasifikace PoE, převzato z [5]

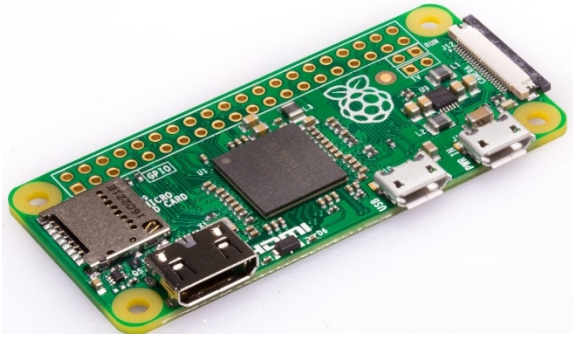
Třídy 1 – 3 jsou podporovány normami 802.3af a 802.3at typ 1. Pokud není implementována klasifikace, automaticky je zařízení zařazeno do třídy 0. Třída 4 je aplikovatelná pouze ve standardu 802.3at typ 2. Pokud napájecí zařízení podporuje pouze normu 802.3af a je k němu připojeno zařízení třídy 4, napájecí zařízení bude tuto třídu ignorovat a bude se chovat stejně, jako kdyby bylo připojeno zařízení třídy 0. Na trhu jsou ke koupi i taková napájecí zařízení, která pokud se připojené zařízení správně neklasifikuje, nedojde k zapnutí napájení na tomto portu.

Zmíněný obvod, který zajísti detekci připojeného zařízení a správnou klasifikaci do třídy 3 je použit i v návrhu popisovaném v této práci. Díky tomu, že ethernetový kabel k zařízení nepovede na velkou vzdálenost, není nutné, aby bylo napájení galvanicky odděleno. Proto byl použit spínaný regulátor BD9G341AFJ v topologii buck s maximálním vstupním napětím 76 V. Tento spínaný regulátor je schopen do zátěže dodat až 3 A. Obvod je nastaven tak, aby spínání probíhalo na kmitočtu 200 kHz. Zapojení tohoto regulátoru bylo převzato z doporučeného zapojení výrobce ROHM semiconductor. Minimální vstupní napětí tohoto obvodu je 12 V. Toto je i minimální úroveň napětí, kterým lze zařízení napájet přes konektor pro připojení externího stejnosměrného napětí. V případě nutnosti lze použít i střídavé napětí. V zapojení je obsažen i diodový usměrňovací můstek pro tyto případy. Je pak ovšem vhodné navýšit filtrační kapacitu z navržených 220 µF.

2.2.4 Řídicí procesor

Požadavky na řídicí procesor jsou dány všemi potřebnými komponentami, které zařízení bude obsahovat. Pokud by počet vyrobených zařízení byl větší, než je aktuálně plánovaných 30 ks, cenově výhodnější by bylo použít například mikrokontrolér od firmy STMicroelectronics a k němu připojit pomocí rozhraní RMII nebo MII ethernetovou PHY. Toto řešení má na druhou stranu nevýhodu v tom, že je nutné implementovat vlastní nebo některý z již existujících IP stacků. Nad stack by bylo nutné dále implementovat podporu komunikačních protokolů. Také by bylo zapotřebí navrhnout a realizovat způsob, jež by umožnil nastavení zařízení bez nutnosti přehrávání programu v procesoru.

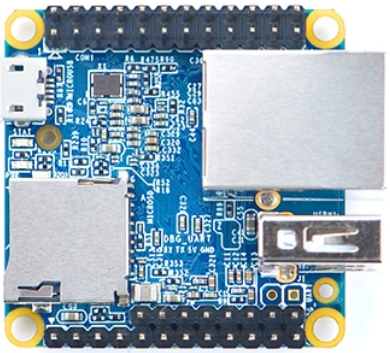
Další možností je použití procesoru s linuxem, respektive linuxového modulu. Prvním zvažovaným modulem bylo Raspberry Pi Zero (na Obrázek 7) za asi 320 Kč.



Obrázek 7 - Raspberry Pi Zero, převzato z [6]

Tento modul disponuje periferiemi, které jsou pro AAS modul zapotřebí. Jsou to SPI, I^2C a další GPIO piny. Bohužel tento modul neobsahuje ethernetové rozhraní, ale jen pouze WiFi. Ethernetové rozhraní by bylo zapotřebí připojit jako USB dongle do přítomného µUSB slotu. Modul je tvořen jedním čipem v konfiguraci SoC, kdy je v jednom pouzdře jak samotný 1 GHz procesor, 512 MB RAM paměti a grafický procesor. Protože se na modulu nenachází eMMC paměť, nebo jiná FLASH paměť, je nutné, aby v modulu byla vždy osazena paměťová karta.

Dalším zvažovaným modulem je NanoPi NEO (na Obrázek 8) v ceně asi 300 Kč od firmy FriendlyElec. Modul je založen na procesoru Allwiner H3. K němu je připojena DD3 RAM paměť standardně o velikost 256 MB.



Obrázek 8 - NanoPi NEO, převzato z [7]

Modul také obsahuje potřebné komunikační rozhraní. Kromě nich, díky zabudované ethernetové PHY je osazen i konektor pro připojení ethernetového kabelu. Bohužel vlastní eMMC paměť se na modulu nenachází. Musí být tedy vždy přítomna paměťová karta. Použitím tohoto modulu by se zkomplikovalo použití PoE napájení. RJ45 konektor na tomto modulu má integrovaný ethernetový transformátor a není tedy možné PoE napájení vyvézt do snižujících měničů. Na plošném spoji by tedy musely vzniknout další dva RJ45 konektory. První by byl vstupní, přivedl by se do něj ethernet a druhý by byl propojovací, ze kterého by vedl propojovací patch kabel do modulu. Tyto dva přidané konektory by byly na plošném spoji propojeny a mohlo by dojít k vyvedení napájení.

Volba nakonec padla na modul od stejné firmy NanoPi NEO Core (na Obrázek 9) za asi 400 Kč.

Obrázek 9 - NanoPi NEO Core, převzato z [8]

Zmíněný modul obsahuje navíc kromě dříve zmíněných periférií také eMMC paměť (v základní verzi o velikosti 4 GB). Kromě paměti má tento modul vyvedeno ethernetové rozhraní, které lze připojit přímo do ethernetového transformátoru. Díky přítomnosti eMMC paměti je možné nahrát linuxový obraz z paměťové karty na tuto paměť. V takovém případě není nutná přítomnost paměťové karty, protože se systém bude moci načíst z eMMC paměti. Z tohoto důvodu se tento modul cenově vyrovná i ostatním uvažovaným, protože by k nim bylo zapotřebí započítat i cenu paměťové karty.

2.2.5 Detekce položené skleničky

Pro detekování položené skleničky na krabičku AAS modulu bylo navrhnuto použití kapacitních plošek. Za tímto účelem byl použit integrovaný obvod od firmy Atmel (nyní již Microchip), AT42QT1070. Jedná se o 7kanálový kapacitní senzor. Má možnost komunikovat s nadřazeným procesorem po rozhraní I^2C, nebo přepnutím do jiného módu pomocí pěti výstupních pinů. Pokud je obvod použit v komunikačním módu, je možné nastavovat citlivosti, detekční úrovně a kalibrovat jednotlivé kanály. Jako ochranný kanál lze v tomto módu nastavit jakýkoliv kanál. V režimu s pěti výstupními piny je ochranný kanál pevně na kanálu 0. Tento kanál mívá podobu

polygonu rozlitého mezi snímacími ploškami a slouží tak k ochraně proti detekci falešného stisku dvou tlačítek.

2.2.6 RGB LED

Klasické RGB LED jsou konstruovány tak, že mají jednu elektrodu společnou a poté tři vývody pro jednotlivé barvy. Změna úrovně jasu jednotlivých barevných segmentů je závislá na úrovni napětí, které je na elektrodu přiloženo. Pokud by bylo zapotřebí řídit nezávisle několik LED představuje to jisté obtíže v jejich řízení.

Místo těchto klasických LED lze použít čipy, které někdy bývají označovány jako chytré LED. Jedním z těchto čipů je WS2812. Jedná se o čip se sériovým asynchronním řízením, který dokáže svítit v 256 odstínech pro každou barvu (červená, zelená, modrá), celkem tedy 16777216 barev. Pro každou diodu v sérii se posílá 24 bitů, tedy 8 bitů pro každou barvu. Na výstup čipu se již neposílají použitá data. Odeslání jednoho bitu trvá asi 1,25 µs. Data pro jeden čip se tedy odesílají asi 30µs. Zrealizovat tento signál na výstupu procesoru s Linuxem, který nepracuje v reálném čase bez použití periferií, jako například SPI, je nemožné.

Čip, který lze použít pro řízení z linuxem řízeného procesoru, je například APA102. Tento čip má synchronní řízení a taktéž se zapojuje do série. Umí zobrazit stejný počet barev, ale má navíc i možnost tyto barvy škálovat v 32 jasových úrovních. Díky synchronnímu řízení je možné tento čip připojit například na periférii procesoru SPI na signály MOSI a CLK. Obrázek 10 definuje datový rámec pro odeslání zobrazovaných dat do více LED v kaskádě. Rámec začíná čtyřmi bajty logických 0 a končí čtyřmi bajty logických jedniček. Uprostřed tohoto rámce jsou takzvané Rámce LED definované dle Obrázek 11. Tento rámec obsahuje kromě 24 bitů pro tři barevné složky, také jeden bajt, jehož spodních 5 bitů udává úroveň jasu konkrétní LED.

<table>
<tr><td>0*32</td><td>LED 1</td><td>LED 2</td><td>---</td><td>LED N</td><td>1*32</td></tr>
<tr><td>Start rámec</td><td colspan="4" rowspan="2">Rámce LED</td><td>Ukončovací rámec</td></tr>
<tr><td>32 nulových bitů</td><td>32 jedničkových bitů</td></tr>
</table>

Obrázek 10 - datový rámec pro odeslání do více LED

<table>
<tr><td colspan="2">MSB LSB</td><td>MSB LSB</td><td>MSB LSB</td><td>MSB LSB</td></tr>
<tr><td>111</td><td>jas</td><td>modrá</td><td>zelená</td><td>červená</td></tr>
<tr><td>3 bity</td><td>5 bitů</td><td>8 bitů</td><td>8 bitů</td><td>8 bitů</td></tr>
</table>

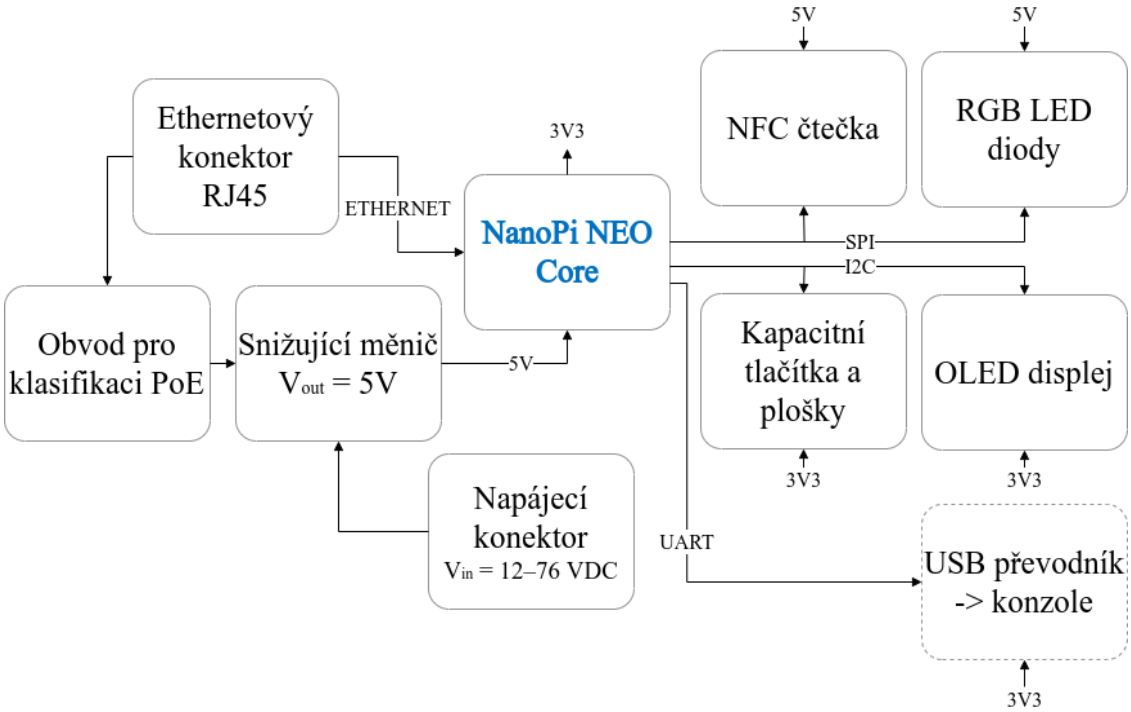
Obrázek 11 - datový rámec pro jednu LED

3 REALIZACE

3.1 Návrh schématu

Vzhledem k faktu, že se někdy stává, že schémata jsou velice rozsáhlá a orientace v nich může být matoucí, je vhodné doplnit schéma blokovým diagramem. Z blokového diagramu by měl být schopen člověk, který o zařízení téměř nic neví, pochopit, jak jsou jednotlivé části zařízení propojeny a jak spolu komunikují. Zároveň může blokový diagram usnadnit konzultaci s ostatními kolegy.

Pro návrh AAS modulu vznikl blokový diagram, který je vyobrazen na Obrázek 12.

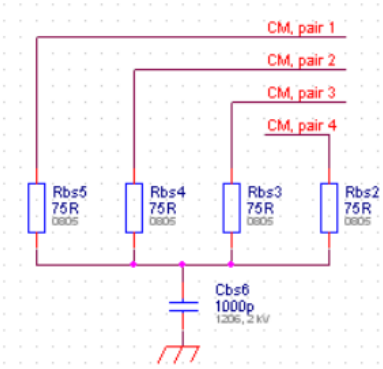


Obrázek 12 - blokový diagram AAS modulu

Schéma zapojení bylo navrženo podle této verze blokového diagramu, i když se v průběhu návrhu některé bloky přidávaly a jiné odebíraly. Požadavky na zařízení se měnily vždy vzhledem k ploše, která mohla být na plošném spoji obsazena komponenty. Mimo jiné i z tohoto důvodu návrh plošného spoje vznikal postupnými iteracemi, vždy po přidání nebo odebrání nějaké části.

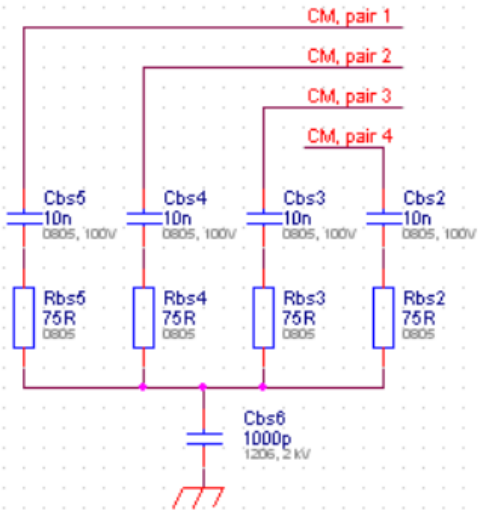
Jednotlivé části schématu vycházejí z doporučených zapojení výrobce. Dále budou popsány některé části zapojení.

Impedanční zakončení ethernetového portu se často provádí na straně konektoru pomocí zapojení „zakončení Bob Smith“. Toto zapojení je znázorněno na Obrázek 13.



Obrázek 13 - zakončení Bob Smith, převzato z [9]

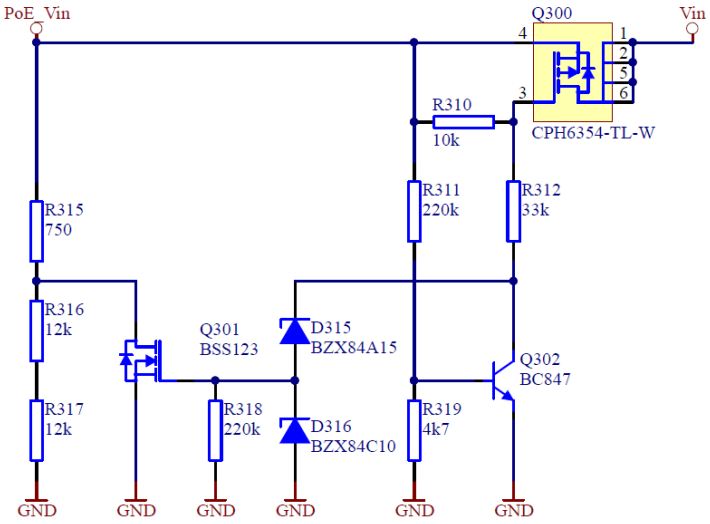
Toto zapojení není použitelné pro PoE. Musí být použito modifikované zapojení z Obrázek 14, které blokuje velké stejnosměrné proudy mezi ukončovacími odpory.



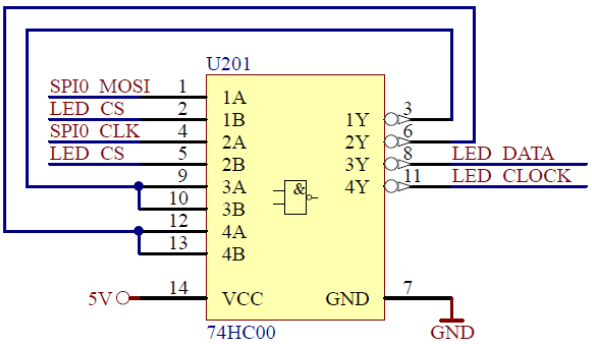
Obrázek 14 - modifikované zakončení Bob Smith, převzato z [9]

Za účelem ochrany zařízení před účinky elektrostatických výbojů nejen při připojování ethernetového kabelu, tedy doteku dvou částí s různým potenciálem, bylo zařazeno diodové pole SLVU2.8-4 jako ochranný prvek.

Na Obrázek 15 je obvod, který zajišťuje úspěšnou detekci 25 kΩ na napájeném zařízení přes PoE. Dále umožní v režimu klasifikace zkratovat rezistory R316 a R317, a umožnit tak zařazení AAS modulu do třídy 3. Po klasifikaci dojde k zapnutí plného napájení na ethernetovém PoE portu. Následně se sepne tranzistor Q300 a zapne se i AAS modul.



Obrázek 15 - obvod pro identifikaci PoE class

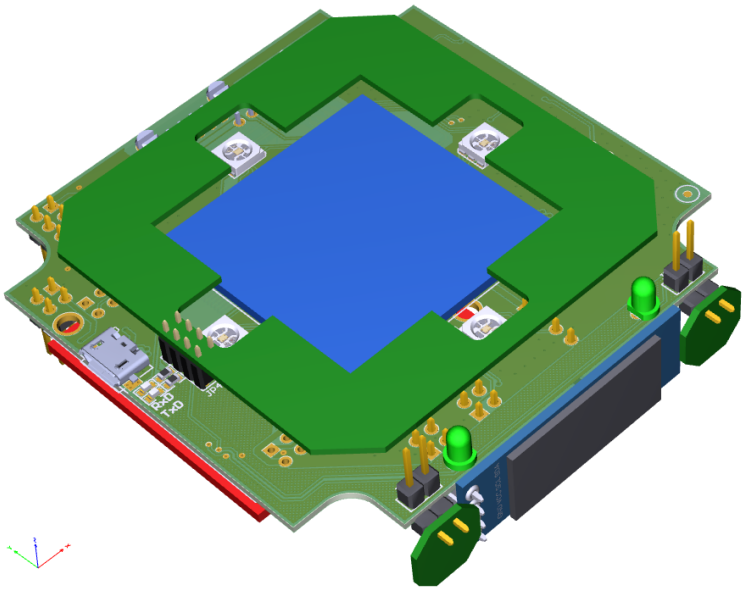


Obrázek 16 - převaděč napěťových úrovní pomocí hradel NAND

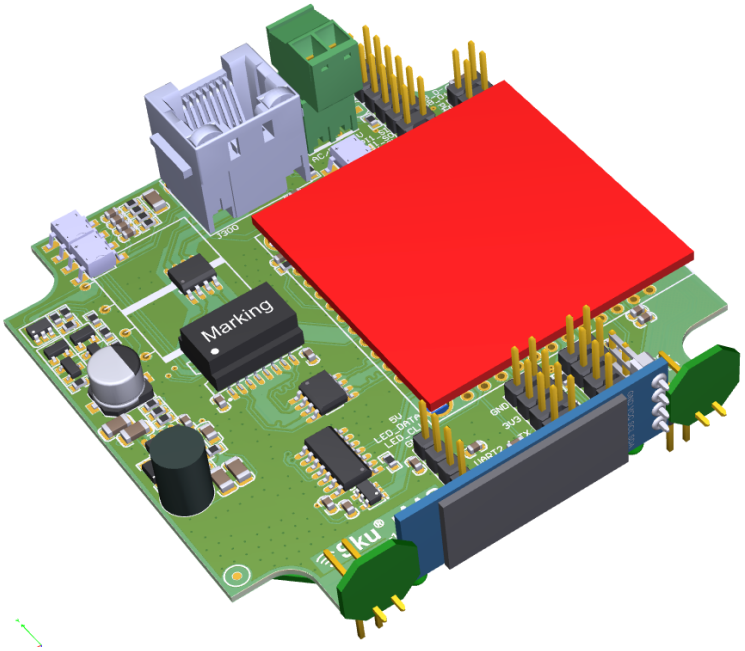
Protože minimální úroveň logické 1 u diody APA102 je 3,5V a NanoPi NEO Core má piny s výstupním napětím 3,3V, bylo jako převaděč napěťových úrovní zařazeno dvou vstupové hradlo NAND. Jeden vstup je použit jako povolovací a druhý vstup je datový. Výstup hradla je přiveden zpět na další hradlo, které je použito jako invertor. Toto zapojení, které je na Obrázek 16, je použito na signály SPI0 MOSI a SPI0 CLK. Pokud by nebyl použit převaděč úrovní, muselo by být sníženo napájecí napětí diody APA102 například pomocí diody.

3.2 Návrh plošného spoje revize 1.1

Návrh probíhal v návrhovém programu Altium Designer 18. Na Obrázek 17 a Obrázek 18 jsou zobrazeny 3D modely realizovaného plošného spoje. Modrý blok na modelu představuje NFC čtečku a červený představuje NanoPi NEO Core.

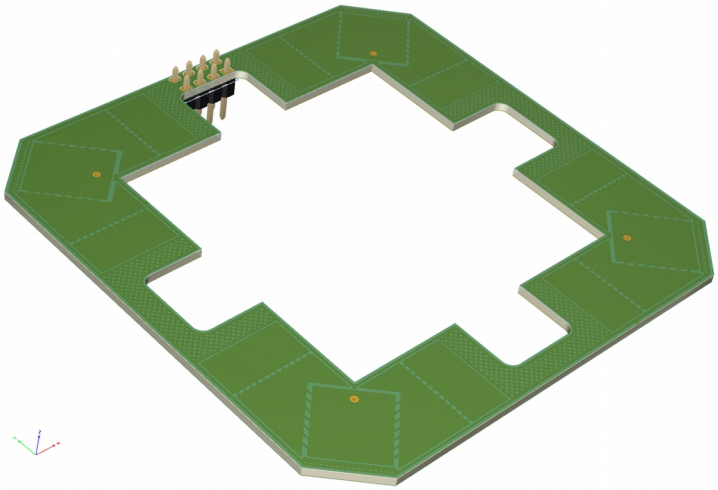


Obrázek 17 - model AAS modulu z horní strany



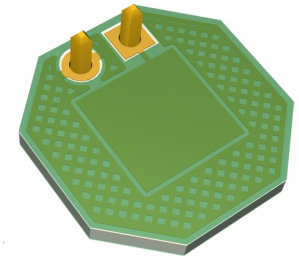
Obrázek 18 - model AAS modulu ze spodní strany

Dotykové plošky, které budou sloužit pro detekci položené skleničky, jsou umístěny na zvláštním plošném spoji, který bude umístěn ve stejné výšce nad hlavní deskou jako čtečka NFC. V tomto plošném spoji jsou kromě výřezu na NFC čtečku také výřezy na RGB LED. Tyto dva plošné spoje jsou propojeny pomocí dvouřadé 8pinové lišty s roztečí 2 mm. Plošný spoj s dotykovými ploškami je na Obrázek 19. DPS obsahuje v každém rohu jednu plošku, která je zvětšena ploškami, které lze odřezat, a tak je odpojit, v případě nespolehlivého čtení.



Obrázek 19 - model plošného spoje s dotykovými ploškami

Dvě menší dotykové plošky (na Obrázek 20) jsou umístěny vedle displeje a budou připojeny pomocí 2pinové jednořadé úhlové lišty. Tyto plošky mohou složit k navigaci na displeji. LED, které jsou na modelu vedle displeje budou ohnuty o 90° a budou směřovat stejným směrem jako displej.



Obrázek 20 - model plošného spoje dotykové plošky

4 VÝROBA, OŽIVENÍ A OPRAVA CHYB

4.1 Výroba zařízení

Výroba plošného spoje byla svěřena firmě JLCPCB, která se zabývá prototypovou výrobou plošných spojů.

Došlo k osazení plošných spojů na osazovacím automatu MYDATA MY9.

Osazení THT součástek bylo realizováno ručně po optické kontrole osazení SMD součástkami. Již během osazování SMD součástek bylo zjištěno několik nedostatků a chyb při návrhu rev1.1 plošného spoje. Mezi chybami můžeme nalézt například chybné pouzdro v návrhové knihovně pro tranzistor Q300 sloužící jako spínací prvek obvodu pro detekci PoE class. Díky tomuto chybnému pouzdru bylo zapotřebí uvedený tranzistor osadit ručně doprostřed plošek.

Další zásadní problém nastal při pájení RGB LED diod APA102. Zde bylo navrženo pouzdro které by se pravděpodobně správně zapájelo v pájecí peci přetavením. Bohužel však LED dorazily od dodavatele až po zapájení některých THT součástek. Diody tak byly pájeny ručně. Vzhledem k malým ploškám docházelo k velkému tepelnému namáhání plošek diod. Prvních několik kusů, kde byly použity diody z čerstvě otevřeného sáčku přežily toto pájení bez problému. Zbývající kusy AAS modulu byly osazovány LED až po nějaké době. Diody v tomto mezidobí nebyly uskladněny ve vzduchotěsném sáčku. Z tohoto důvodu se k nim dostala vlhkost. Před pájením nebyly diody vysušeny například v elektrické troubě. Díky tomu došlo ke zničení buďto celé LED, výstupu diody nebo jednoho či více barevných segmentů. Z tohoto důvodu bylo nutné objednat nové LED a nefunkční kusy vyměnit. Nové pájení probíhalo pomocí přetavení nanesené pájecí pasty hrotem pájky.

Největší chyba která byla během návrhu rev1.1 zanesena byla chybná orientace NanoPi NEO Core modulu. Tato chyba způsobila že bylo zapotřebí z dodaných modulů s již připájenými pinovými lištami tyto lišty vypájet. Pro odsání a vytažení takového množství pinů je zapotřebí mít k dispozici kvalitní odsávací stanici. Zvláště v případě, kdy se jedná o vícevrstvou DPS. Toto odsávání způsobilo u několika modulů jeho poškození. Po odsání pinových lišt mohlo dojít k jejich zapájení z druhé strany modulu a osazení modulu do plošného spoje zrcadlově oproti původní orientaci lišt.

4.2 Testování

Za účelem ověření správné funkce jednotlivých periférií byly napsány jednoduché testovací programy ve vysokoúrovňovém skriptovacím jazyce Python. Otestování elektrických vlastností proběhlo připojením zařízení na zdroj napětí 24V. Ověření správné funkce identifikace PoE proběhlo připojením zařízení PoE switchům

podporujícím standardy 802.3at a 802.3af. AAS modul byl na manažovatelném switchi úspěšně detekován jako zařízení s PoE class 3.

Obrázek 21 - zapnutý AAS modul s textem na displeji a rozsvícenými LED

Pro ověření funkce LED byl napsán testovací skript [10]. Skript pošle zobrazovací data – barvy a úroveň jasu do všech 4 ks LED APA102.

Pro ověření funkčnosti OLED displeje byl rovněž napsán testovací skript [11]. Tento skript inicializuje displej a vypíše na něj testovací text, aktuální datum a čas a IP adresu zařízení. S funkcí OLED displeje nebyl na žádném z realizovaných kusů hardware žádný problém.

Posledním napsaným testovacím skriptem byla obsluha čtečky RFID tagů [12]. Skript umožňuje čtení i zápis do tagu.

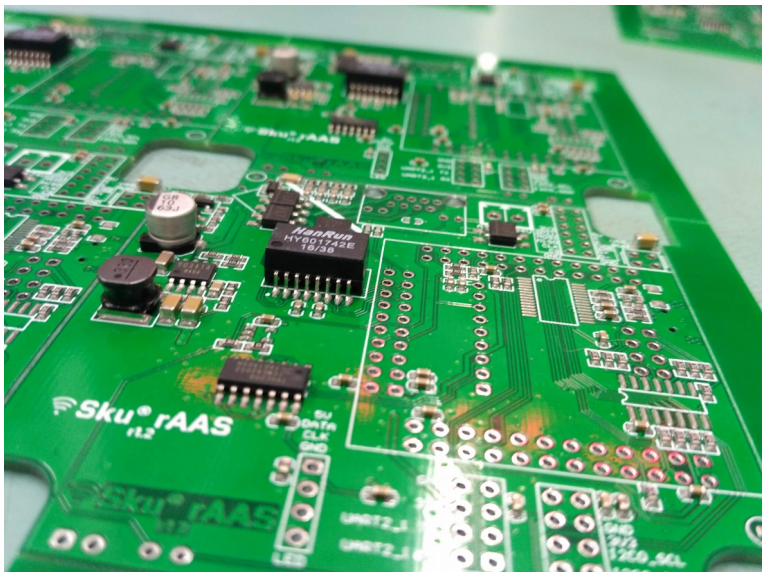
Zapnutý AAS modul s rozsvíceným textem na displeji a rozsvícenými LED je vyobrazen na Obrázek 10.

4.3 Návrh revize 1.2 plošného spoje

Z důvodu potřeby dalších modulů jak pro vývoj tak i pro zakomponování do projektu Barmana byla navržena revize 1.2 plošného spoje, která opravuje výše zmíněné nedostatky.

Bylo opraveno pouzdro a typ spínacího tranzistoru v obvodu pro identifikaci třídy PoE. Také byla provedena záměna tlumivky spínaného zdroje z THT na SMD. V neposlední řadě byla provedena změna počtu a pouzdra vyhlazovacích kondenzátorů na napájecí větvi před spínaným zdrojem.

Pro LED APA102 došlo k úpravě pouzdra a zvětšení jejich plošek. Diody tak lze lépe pájet ručně. Díky zkušenostem z revize 1.1 a následného osazování, byly diody osazeny do pájecí pasty a zapájeny v přetavovací peci.



Obrázek 22 - plošný spoj osazen SMD součástkami do pasty

Asi ta největší změna, která zapříčinila potřebu opětovného rozmístění prvků a následného opětovného routování spojů na celé DPS je zrcadlové otočení všech prvků tak, aby bylo možné využít NanoPi NEO Core moduly s již zapájenými pinovými lištami.

Pro lepší možnosti uchycení DPS ve vozíku osazovacího automatu byla DPS panelizována. To znamená, že reálný návrh byl umístěn do jednoho jiného většího návrhu, kde byly plošné spoje zkopírovány celkem čtyřikrát do pole 2x2. K takovýmto panelům se navíc přidávají technologické okraje. Tento okraj se obvykle vyrábí o tloušťce 5 – 10 mm. Okraj se používá pro uchycení plošných spojů, případně pro popisky určené pro výrobu. Technologický okraj je od DPS oddělen takzvanou V drážkou. Jedná se o drážkování pomocí frézy kónického tvaru s vrcholovým úhlem obvykle 30 - 45°. Fréza zajede do materiálu plošného spoje, z obou stran, do hloubky přibližně 1/3 celkové tloušťky plošného spoje. Při nejčastější tloušťce plošného spoje 1,6 mm činí hloubka penetrace asi 0,53 mm. Některé společnosti, zabývající se výrobou plošných spojů, využívají pro tvorbu V drážek místo frézy speciální pilové kotouče. Pro oddělování takto vyrobených plošných spojů je vhodné použít specializované přístroje. Pokud tyto specializované přístroje nejsou k dispozici, lze oddělení provézt pomocí plochých, případně kombinovaných kleští.

K panelizaci došlo i u desky s dotykovými ploškami. Důvod panelizace u této části je čistě jen ekonomický. Při výrobě bylo levnější objednat 5 ks panelů než požadovaných 20 ks jednotlivých DPS.

Podobně jako revize 1.1 byl i tento návrh proveden v programu Altium Designer 18. Z hlediska konstrukce a celkové sestavy nedošlo k výrazným změnám, které by zásadně ovlivnily podobu zařízení. Kromě zmíněných úprav došlo i k úpravě na desce s dotykovými ploškami. Propojovací pinová lišta desky s dotykovými ploškami se změnila z rozteče 2 mm na lištu s roztečí 2,54 mm. Dále dotykové plošky

umisťované kolmo k hlavní desce vedle displeje byly vyrobeny zvlášť a nebylo tedy nutné je vyřezávat z vnitřní strany dotykových ploch.

5 SOFTWARE

Obslužný software AAS modulu byl rozdělen na několik částí – obsluha HW periférií a komunikace v rámci testbedu.

Obsluha HW periférií byla psána ve skriptovacím jazyce Python. Komunikace v rámci testbedu a logika buněk bude psána v jazyce C++.

5.1 Skriptovací jazyk a vývojové prostředí

Jazyk Python, ve kterém byla psána obsluha HW periférií a adaptér pro komunikaci přes Modbus TCP, je otevřený programovací intepretovaný jazyk navržený v roce 1991 Nizozemským programátorem Guido van Rossum. Jeho instalační balíky nabízí podporu pro většinu standardních platforem, jako jsou Linux, Windows, macOS případně Android. Během let vyšlo několik verzí tohoto programovacího jazyka. Jednotlivé verze mají definován konec podpory. S koncem podpory souvisí začátkem roku 2020 často skloňovaný pojem v Python komunitě, kdy skončila podpora pro verzi 2.7 a množství aplikací nebylo na tento konec připraveno. Vývojáři tak byli v případě, že chtěli zajistit, že aplikace bude spouštěna v aktuální verzi interpretru, nuceni aktualizovat verzi na některou z verzí Python 3.x. Přestože velké množství vlastností z verze 3 byla zanesena zpětně do verze 2, stále zde existuje nesourodost v práci s textovými řetězci, kdy jedna verze pracuje s 8bitovým kódováním a druhá verze používá pro kódování řetězců Unicode. To, že je jazyk interpretovaný znamená, že není třeba spouštět kompilátor tak jako v případě jazyka C, ale v momentě spuštění dojde k interpretaci příkazů zapsaných v souborech *.py. Standardní interpretr je napsán v jazyce C. Nabízí řadu knihoven umožňujících komunikaci s hardwarovými periférierniemi zařízení případně knihovny pro komunikaci po internetu. Velké množství knihoven je nainstalováno přímo s interpretrem.

Vývoj probíhal ve vývojovém prostředí od firmy JetBrains, PyCharm v jeho profesionální edici. Tato edice mimo jiné umožňuje ladění na vzdáleném zařízení. Je takto možné vytvářet záchytné body v rámci ladění přímo na zařízení a použít tedy vzdálený interpretr.

5.2 Komunikace mezi procesy

V případech, kdy má na jednom zařízení být spuštěno několik nezávislých aplikací bývá velmi často zapotřebí zajistit vzájemnou komunikaci.

Pro zajištění vzájemné komunikace mezi procesy existuje velké množství možností realizací. Mezi tyto možnosti patří například soubor, databáze, socket, MQTT, D-Bus a jiné.

5.2.1 Komunikace pomocí souboru

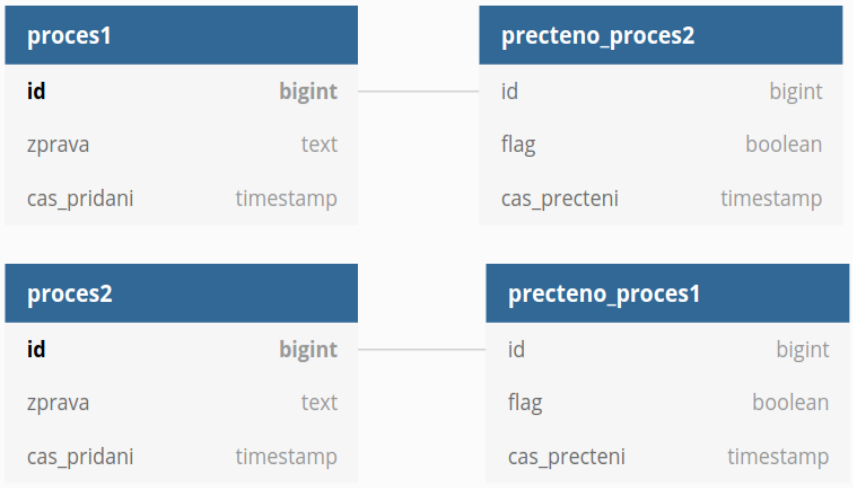
Pravděpodobně nejjednodušší způsob jak komunikovat mezi procesy je sdílený soubor do kterého se budou předávaná data zapisovat. Tento způsob komunikace je jednoduchý na implementaci ať už na UNIX systémech tak i na systémech typu Windows.

Zapisování do souboru sebou nese i nějaké potíže. K potížím může dojít v momentě, kdy se dva procesy budou snažit do souboru v jeden okamžik. Podobná situace může nastat i v následujícím příkladu. Proces 1 otevře soubor pro zápis, poté jej pro zápis otevře proces 2. Proces 2 dokončí zápis. Proces 1 se snaží dokončit zápis do již změněného souboru. Tato situace může vyústit v několik v několik výsledků. Aplikace může havarovat, zápis se nemusí povést nebo se může poškodit soubor. Jednou z nevýhod této metody je přepisování jednoho a téhož souboru. Tato vlastnost může v závislosti na souborovém systému zapříčinit rychlejší poškození paměťového média.

5.2.2 Komunikace pomocí databáze

V systému může být nainstalována databáze, například PosgreSQL, nebo lze použít jedno souborovou databázi, například SQLite. Výhodou systémové databáze je, že obvykle poskytuje přístupový bod který lze použít pro databázové dotazy. Častým typem databáze je relační databáze, která používá pro práci SQL jazyk.

Relačních databáze, založená na relačním modelu, se vyznačuje tím, že používá tabulky, jejíž řádky mohou, případně nemusí mít souvislost s řádky jiné tabulky. Relace, vztahy, nemusí být jen mezi dvěma tabulkami. Lze je vytvořit i mezi několika různými tabulkami. Vždy záleží na struktuře ukládaných dat.



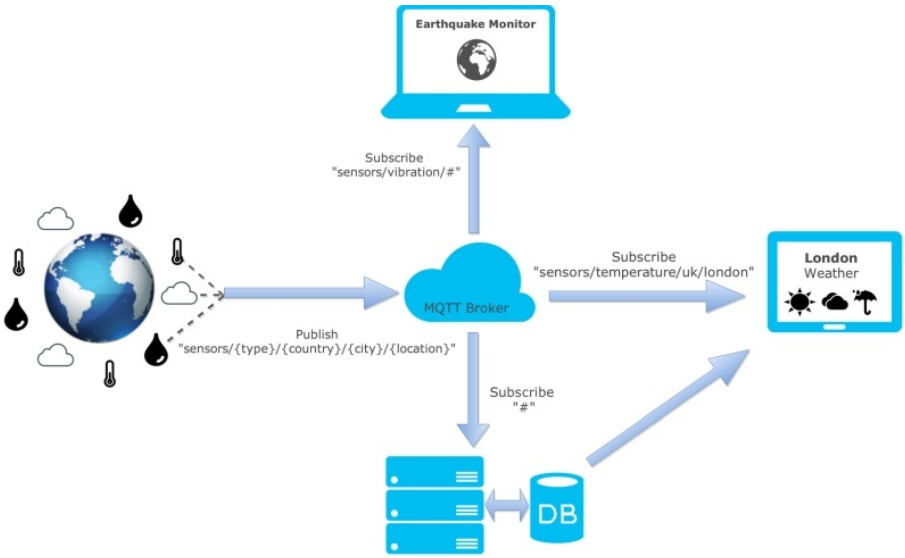
Obrázek 23 - příklad struktury databáze a její relace

Pokud je databáze použita na komunikaci mezi přenosy lze zvolit několik možností jak tyto data ukládat. Je možné vytvořit zvláštní tabulku pro každý proces a jen přidávat do konkrétní tabulky nově odesílaná data. Data lze například opatřit časovou značkou nebo flagem, vlajkou, označující přečtení zprávy jiným procesem. Příklad takové databáze je na Obrázek 23. V této databázi každý z procesů zapisuje zprávy do tabulky

se stejným názvem, tedy proces 1 do tabulky `proces1`. Pomocí relace je s každým záznamem v tabulce `proces1` svázán záznam o přečtení procesem 2 v tabulce `precteno_proces2`. V případě, že by byla databáze navržena tímto způsobem nedojde ke kolizi dat, respektive přístupů a oba procesy tak mohou mít přehled o tom kterou zprávu již jiný proces přečetl.

5.2.3 Komunikace pomocí MQTT

MQTT protokol je postavený nad TCP/IP. V internetu věcí se tento protokol, případně jeho modifikace, často používají pro komunikaci se senzory. S tímto protokolem se lze setkat i u některých služeb Amazon Web Services (AWS), Facebook.com, a jiných.



Obrázek 24 - funkční schéma MQTT protokolu, převzato z [13]

Protokol využívá systém typu zveřejnit/odebírat. Senzor, řídicí aplikace apod. publikují (proces zvaný publish) do centralizovaného prvku (MQTT Broker) zprávu do tématu. Témata lze volit libovolně, avšak pro využití všech vlastností MQTT je vhodné skládat témata do hierarchie, například `floor/1/bathroom/lamp`. Tuto zprávu obdrží všechna zařízení, aplikace pro obsluhu HW, apod., které jsou přihlášeny k odběru daného tématu (proces zvaný subscribe). Centralizovaným prvkem se rozumí aplikace, která běží na nějakém serveru, nebo na zařízení.

Protokol umožňuje například nastavit zprávy typu Last Will and Testament, což je druh zprávy, kterou klient nařídí centralizovanému prvku aby odeslal, pokud dojde k odpojení klienta. V IoT bývá tato vlastnost využívána pro notifikaci, že některý ze senzorů vypadl ze sítě. MQTT umožňuje nastavení QoS pro jednotlivé zprávy a tak se ujistit, že zpráva dorazí od klienta k centralizovanému prvku. Jedná se o systém potvrzování.

V režimu subscribe je možné využít zástupných symbolů, tzv. wildcards, umožňující nahrazení části názvu tématu jakýmkoliv slovem. Pro nahrazení části názvu uprostřed tématu se využívá symbol +.

```
floor/1/bathroom        // zprávy z koupelny na 1. podlaží
floor/2/bathroom        // zprávy z koupelny na 2. podlaží
floor/3/bathroom        // zprávy z koupelny na 3. podlaží

floor/+/bathroom        // zprávy z koupelen na všech podlažích
```

Pokud chceme odebírat zprávy, které jsou v kaskádě úrovní níže, lze použít symbol #.

```
floor/1/bathroom        // zprávy z koupelny na 1. podlaží
floor/1/bedroom         // zprávy z ložnice na 1. podlaží
floor/1/kitchen         // zprávy z kuchyně na 1. podlaží

floor/1/#               // zprávy z místností na 1. podlaží
```

Nejběžnější způsob použití tohoto protokolu je umístění MQTT brokeru na centrální server, ke kterému se mohou připojit klienti. Centrální server může být jak v lokální síti, tak i v internetu. Vždy záleží na použití. Protokol lze provozovat pouze pokud je zajištěna komunikace s MQTT brokerem. Pokud není jistota, že zařízení má přístup k vhodnému MQTT brokeru, je zapotřebí jej nainstalovat přímo na zařízení. Jedním z možných brokerů je volně šiřitelný produkt Mosquitto [15] z dílny firmy Eclipse. Kromě samotného brokeru nabízí i klienty použitelné z příkazové řádky.

V případě, že je broker nainstalován na zařízení, které je přístupné do LAN sítě a jsou-li povoleny příslušné komunikační porty, je možné přistupovat k brokeru i z jiných zařízení. Tuto vlastnost lze využít pro snadné sledování a případné ovlivňování probíhající meziprocesové komunikace.

5.2.4 Komunikace pomocí D-Bus

D-Bus je meziprocesová komunikační sběrnice. D-Bus byl vyvíjen pro Linux a Unix distribuce. Sběrnici využívají GNOME a KDE Linuxová prostředí pro komunikaci například mezi systémovými aplikacemi a desktopovým prostředím. Nicméně existuje i portace pro systém Windows. Podle oficiálních stránek [16], lze port pro Windows provozovat na systémech Windows XP až Windows 7.

Objekt sběrnice je popsána pomocí textového řetězce odděleného tečkami, například `org.freedesktop.NetworkManager`. Tento řetězec se skládá ze jména objektu na poslední pozici a z cesty k objektu. Objekt může obsahovat několik funkcí (metod), nebo callbacky (signály). Funkce mohou přejímat argumenty nebo mohou vracet návratovou hodnotu. V případě signálů se jedná o asynchronní možnost vyvolat akci u klientů, kteří si zaregistrovali odběr daného signálu.

Popis vlastností metod, tedy typ a počet parametrů, popisuje D-Bus pomocí Signatures, podpisů. Každá metoda musí obsahovat jednoznačný popis argumentů, který se skládá z jednotlivých znaků. Každý znak v popisu odpovídá nějakému typu

argumentu. Má li metoda popis `iis`, tak to znamená, že funkce očekává dvě 32bitová znaménková celá čísla následovaná řetězcem.

5.3 Formát dat vhodných pro komunikaci mezi procesy

Přenášená data musí být ve vhodném formátu, který je srozumitelný všem stranám. Následující formáty dat lze použít pro téměř jakýkoliv způsob ze zmíněných komunikačních protokolů.

5.3.1 Čistý text

V případě, že je formát dat přenosu zpráv mezi procesy čistě textový, je na vybranou mezi několika různými formáty zpráv. Pokud má programátor tyto zprávy číst, případně pomocí nich ladit svoji aplikaci je nejvhodnějším formátem čistý text. Jedná se o zprávy, nejčastěji větu nebo uspořádaný seznam hodnot. Výhodou těchto zpráv je rychlá a jednoduchá interpretace programátorem. Na druhou stranu, strojové zpracování těchto zpráv může být relativně komplikovaná záležitost.

5.3.2 XML

Jedním z formátů dat je XML. Lze jej použít nejen pro komunikaci, ale například i pro popis konfigurace aplikace. Díky svojí syntaxi, umožňuje popisovat téměř jakoukoliv situaci a předávat jakékoliv informace. Pouze kompletní XML datový soubor, nebo zprávu, lze zpracovat pomocí mnoha různých parserů. Parser kontroluje správnost syntaxe dokumentu, případně jej umí i rozklíčovat a předat data ze souboru aplikaci. Tagy XML souboru mohou obsahovat kromě konkrétní hodnoty také atributy.

```
<?xml version="1.0"?>                  // informace o použité verzi xml
<house>                                // začátek kořenového elementu
      <floor level="1">                // začátek elementu s atributem
            <room>Kitchen</room>       // element s hodnotou
            <room>Bathroom</room>
            <room>Dining room</room>
      </floor>                         // konec elementu s atributem
      <floor level="2">
            <room>Bathroom</room>
            <room>Bedroom</room>
      </floor>
</house>                               // konec kořenového elementu
```

Obrázek 25 - struktura zprávy z ukázky XML dokumentu

Uvedený příklad XML zprávy lze pomocí parseru zpracovat do objektu s polem, obsahujícím dva prvky (každý prvek odpovídá jednomu patru). Každý z těchto prvků obsahuje číselný parametr definující podlaží a pole s názvy místností. Ukázka takovéto struktury odpovídající ukázce je na Obrázek 23. Jedno z mála pravidel pro vytváření XML popisů je dodržování integrity. Je tedy nutné aby element začal i skončil. Koncový element je význačný tím, že na jeho začátku se nachází znak lomeno. Každý z popisů může mít jen jeden kořenový element.

5.3.3 JSON

Další možností formátování zprávy je JSON. Tento formát je relativně jednoduchý pro zpracování lidským okem a tedy je jednoduše zpracovatelný. Podobně jako pro XML, existuje i pro JSON velké množství parserů pro různé programovací jazyky. Do JSON zprávy lze uložit pole, řetězce, čísla, logické hodnoty `true` (ano), `false` (ne) a `null` (žádná hodnota). Stejnou situaci, kterou jsme ukázali u XML lze popsat v JSON formátu takto.

```
{
  "house": {                          // kořenový element
    "floors": [                       // pole prvků
      {                               // první prvek pole
        "level": 1,                   // číselný argument
        "rooms": [
          "Kitchen",
          "Bathroom",
          "Dining room"
        ]
      },                              // konec prvního prvku pole
      {                               // druhý prvek pole
        "level": 2,
        "rooms": [
          "Bathroom",
          "Bedroom"
```

```
        ]
      }
    ]
  }
}
```

Obrázek 26 - struktura zprávy z ukázky JSON dokumentu

Zpracovaná struktura zprávy z ukázky je na Obrázek 23.

Podobně jako XML, musí být i JSON řádně zakončen. Jinými slovy, musí mít stejný počet pravých složených závorek, definujících objekt, jako levých. Toto pravidlo se týká i hranatých závorek definujících pole. Jedinou výjimkou jsou závorky uzavřené v textovém řetězci.

5.4 Obsluha hardwarových periférií

Obsluhu lze rozdělit na dvě základní části. Na část komunikující přes sběrnici SPI a na část komunikující přes sběrnici I^2C. Vývoj těchto částí probíhal nejprve separátně tak aby byl vývoj ulehčen. Na konci vývojového procesu byly obsluhy těchto dvou částí spojeny do jednoho obslužného programu dostupného z [14].

Pro komunikaci mezi procesy, tedy mezi skriptem obsluhy HW periférií a programem zajišťujícím vlastní funkcionalitu buňky byl zvolen protokol MQTT. Tento protokol byl zvolen mimo jiné i pro možnost snadného ladění obou aplikací. Dalším důvodem proč byl zvolen tento protokol je možnost pracovat na vývoji obou aplikací nezávisle na sobě. Jako formát datové zprávy byl zvolen JSON. Tento formát je dostatečně univerzální a variabilní pro použití v dané aplikaci a obsahuje všechny potřebné datové typy.

5.4.1 Společný základ

Aby bylo možné oddělit implementaci ovládání SPI od kontroléru čtečky, případně LED byla vytvořena třída `NanoPiSpi`. Tato třída obsahuje přímý přístup k implementaci spi pomocí doinstalované knihovny pro Python, `spidev`. Základem SPI jsou signály

MOSI (Master Out, Slave Input – data odesílaná hlavním prvkem), MISO (Master Input, Slave Output – data odesílaná podřízeným prvkem) a CLK (Clock – hodinový signál generovaný hlavním prvkem). Každý z prvků dále musí mít svůj vlastní pin CS (Chip Select – slouží pro výběr podřízeného prvku se kterým bude ten hlavní komunikovat). Čtečka i LED mají svoje CS signály. Tyto CS signály jsou ovládány pomocí kontroléru `gpio`. Tento kontrolér, převzatý z [17], nastavuje stavy pinů přes virtuální systém souborů `sysfs`. Kontrolér umožňuje stav pinů nejen nastavovat ale i číst. Pro případ, že je pin nastaven jako vstupní, je možné nastavit callback reagující na například na hranu pinu.

5.4.2 Část ovládaná po sběrnici SPI

Mezi periferiemi připojenými na sběrnici SPI se nachází čtečka NFC tagů a LED.

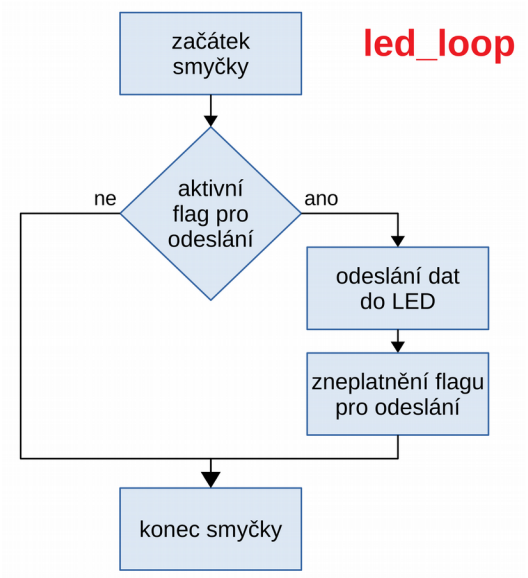
Komunikace se čtečkou NFC, modulem s obvodem MRC522, je obsažena ve stejnojmenném modulu s odpovídající třídou `MFRC522`. Tato třída je založena na knihovně [18]. Třída obsahuje metody umožňující komunikaci po SPI, kde využívá třídu `NanoPiSpi`. Pravděpodobně nejdůležitější funkcí této třídy je metoda `to_card`. Tato funkce zajišťuje odeslání dat do čtečky, přijetí a zpracování přijatých dat. Využívají ji ostatní funkce, jež se využívají například pro vyčtení obsahu přiloženého tagu, pro zápis dat do tagu, apod. Vlastní funkcionalita, tedy algoritmus pomocí kterého dochází k vyčtení dat tagu, případně algoritmus zápisu dat do tagu není v této třídě obsažen. Tato třída jen poskytuje potřebné metody.

Algoritmus, zajišťující komunikaci s tagem, tedy vyčtení uložených hodnot a zápis nových dat zajišťuje metoda `reader_loop` ze třídy `AasSpi`. Vývojový diagram této smyčky je na Obrázek 10. V této třídě jsou obsaženy další funkce umožňující zápis, nebo čtení z tagu. Požadavek pro zápis jednoho nebo více sektorů je indikován nastaveným příznakem. Nastavení příznaku probíhá mimo tuto třídu uvnitř callbacku pro odběr zpráv z MQTT. Po dokončení zápisu je tento příznak vymazán. Taktéž se v této třídě nachází několik callbacků, které jsou implementovány v jiných třídách. Jedná se o callback `on_event`, který přebírá parametry nutné pro publikování dat po MQTT, a `log_debug`, zapisující zprávy do logovacího souboru s úrovní debug. Tyto záznamy v logovacím souboru slouží pro ladění, případně pro diagnostiku programu. Zmíněné callbacky jsou v této třídě implementovány tak, že mají definovány argumenty stejně jako originální metoda která se bude používat k jejich volání a v těle funkce je pouze výraz `pass`. Ten nedělá nic. V tomto případě umožní vytvoření funkce, která neobsahuje tělo.

Ve zmíněné třídě se také nachází funkce `led_loop`, která se stará o zápis nového stavu LED. Vývojový diagram této smyčky je na Obrázek 10. K tomu dojde v případě, že přišel přes MQTT požadavek k tomuto úkonu. Ten je signalizován nastaveným příznakem, který je po odeslání dat do LED vymazán. Tato funkce využívá metody třídy `APA102`. Tato třída umožňuje vytvořit základní datový rámec se všemi diodami dle

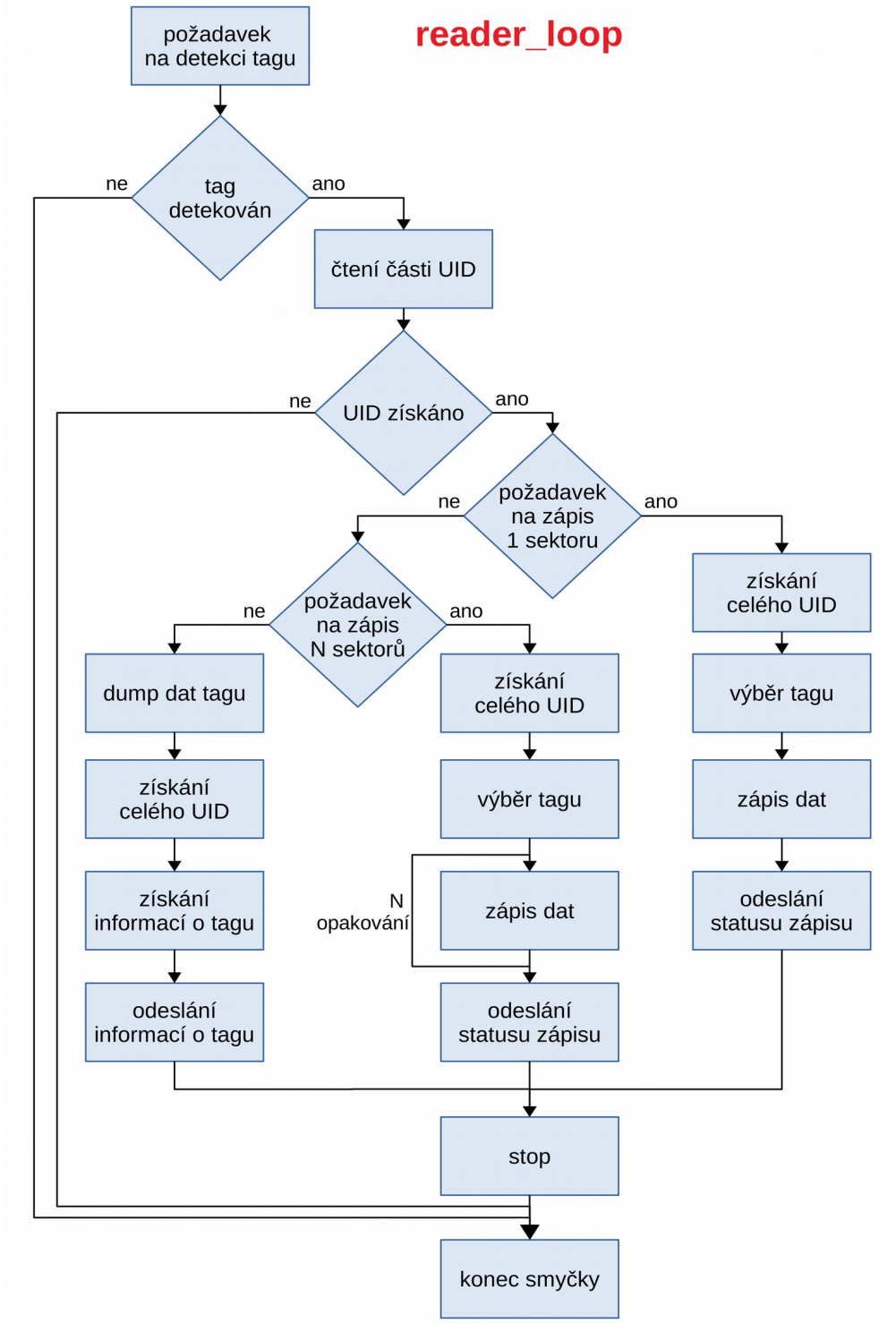
Obrázek 10. Dále metoda `prepare_data` slouží pro nastavení úrovně jasu v rozsahu 0 až 100 % a úrovně barevných složek v rozsazích 0 – 255. Tato třída slouží jen pro vytvoření odesílaných dat. Funkce `get_data` této třídy vrací list obsahující zobrazovaná data určená pro odeslání do LED.

Pro programy, které jsou spuštěny a pracují kontinuálně existují v zásadě dva přístupy programování. Jedním z přístupů jsou thready, tedy vlákna, kdy každá funkcionalita má svoje vlákno ve kterém pracuje. Další možností jak psát takovýto program je vytvořit sadu stavových automatů, které se periodicky volají. Místo stavových automatů lze použít jen obyčejné podmíněné volání funkcí v nekonečné smyčce, takzvané supersmyčce. Tento přístup je zvolen i v tomto případě. Proto hlavní aplikace volá periodicky funkce končící výrazem `_loop`. Tyto funkce jsou na takovéto volání přizpůsobeny.



Obrázek 27 - vývojový diagram smyčky LED diod

Třída `AasSpi` při vytvoření svojí instance, tedy zavolání funkce `__init__` mimo jiné otevře a inicializuje SPI sběrnici a nastaví komunikační rychlost na 2 MHz. Také vytvoří instanci třídy `APA102` a nastaví počet LED na čtyři. V této funkci dojde i k inicializaci instance třídy `MFRC522`, tedy NFC čtečky.



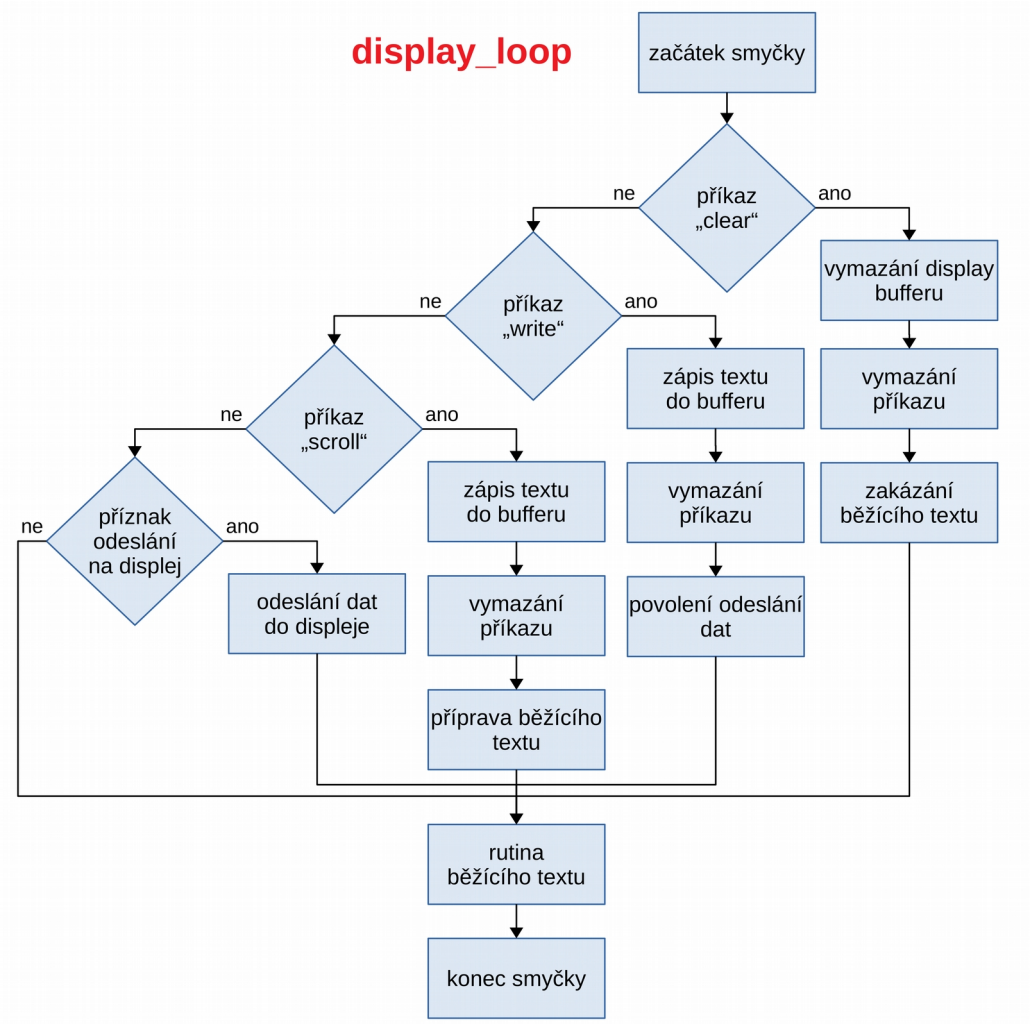
Obrázek 28 - vývojový diagram smyčky NFC čtečky

5.4.3 Část ovládaná po sběrnici I²C

Ke sběrnici I²C je připojen OLED displej a kontrolér dotykových plošek.

Ovládání OLED displeje s rozlišením 128x32 pixelů a kontrolérem SSD1306 je zajištěno pomocí knihovny Adafruit SS1306 [19]. Knihovna umožňuje komunikovat

s různými velikostmi displeje a to jak po sběrnici I^2C, tak i po SPI. Tato knihovna kromě inicializace poskytuje potřebné funkcionality pro odeslání zobrazovaných dat do displeje. Tyto data musí být předány ve formě instance PIL (Python Image Library) Image. Tento objekt lze vytvořit pomocí knihovny [20]. Objekt displeje je v kontroléru třídy `AasI2C` tvořen jako `display`. Objekt PIL Image je nazvána `image`. Tento objekt vytváří virtuální obrázek s potřebnými rozměry. Do tohoto obrázku je dále zapisováno objektem `draw`, což je instance třídy `ImageDraw` knihovny PIL. Objekt umožňuje zapisovat do zmíněného obrázku text, čáry a jiné obrazce. Nicméně v programu je využita jen vlastnost zapisovat text a obdélníky. Text lze zapisovat na zvolené souřadnice zvoleným fontem.



Obrázek 29 - vývojový diagram smyčky obsluhy displeje

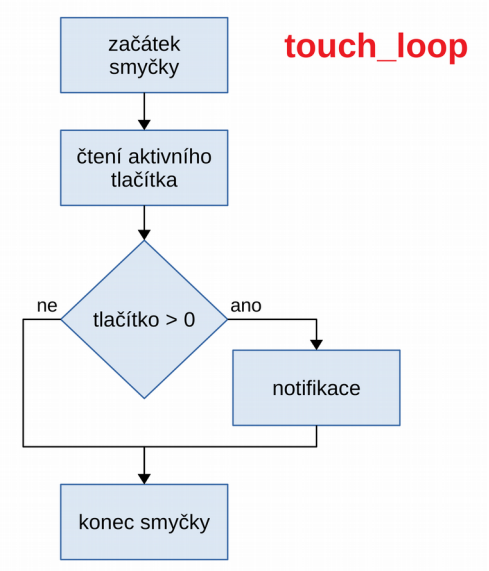
Objekt `fonts` obsahuje dict s nainstalovanými písmy o zvolených velikostech. Instalace fontů probíhá při startu programu pomocí funkce `load_fonts`. Tato funkce vyhledá soubory s příponou `ttf` umístěné ve složce `static` celého programu. Při volání této funkce se volí velikost písem v pixelech. Funkce je v programu volána tak, aby vytvořila písma o velikostech 10, 12 a 15 pixelů. Výsledné pojmenování písem v

daném dict je tvořeno názvem písma a jeho velikostí. Pro 12px font Vafle VUT Regular je použit název `Vafle_VUT_Regular-12`. Ve výchozím stavu je kromě zmíněného písma k dispozici také písmo Arial ve všech třech velikostech.

Kromě metod pro odeslání dat do displeje, jeho vymazání apod., obsahuje třída `AasI2C` také objekty třídy `TouchControl`, nazvaný `touch`, a `ScrollText`, nazvaný `scroll`.

Třída `ScrollText` obsahuje algoritmus umožňující vykreslení běžícího textu na displeji. Objektu této třídy lze nastavit zobrazovaný text, písmo a souřadnici v ose y. Objekt poskytuje metodu `run`, která se pomocí periodického volání stará o posun textu po displeji. Text se vykresluje do obdélníku, pomocí kterého tato metoda zajistí vymazání daného řádku při každém volání. Kromě vykreslení obdélníku přes již existující znaky, metoda vypíše i zobrazovaný text posunutý o potřebný počet pixelů. Přepisování displeje a jeho reakci na povely řídí smyčka `display_loop`, jejíž vývojový diagram je znázorněn na Obrázek 30.

Třída `TouchControl` obsahuje metody určené pro komunikaci s AT42QT1070 a detekci stisknutého tlačítka. Také se zde nachází callback `_read_state` pro `gpio` kontrolér, který se vyvolá při sestupné hraně pinu `CHANGE` zmíněného obvodu. V daném callbacku proběhne nastavení příznaku pro vyčtení obvodu. Nastavenost zmíněného příznaku se testuje v periodicky volané funkci `read_active_key`, která vrací číslo stisknutého tlačítka. Po úspěšném přečtení stisknutého tlačítka dojde k odeslání této informace přes mqtt. Volání této funkce je uzavřeno v další smyčkové funkci, `touch_loop`. Vývojový diagram této smyčky je na Obrázek 10.



Obrázek 30 - vývojový diagram smyčky dotykových plošek

Metoda `init_screen` třídy `AasI2C` se volá při startu programu během inicializace. Tato metoda zobrazí na displeji Obrázek 10. Jak lze vidět, dojde k vypsání i aktuální IP adresy zařízení. Toto zobrazení na displeji zůstane do doby, než program přijme příkaz pro přepsání nebo vymazání displeje.

Obrázek 31 - úvodní obrázek zobrazený na OLED displeji

5.4.4 Ovládání programu

Program lze spustit buďto ručně přes konzoli, nebo je možné jej nainstalovat jako službu a nechat běžet na pozadí. Program zapisuje logovací hlášky do souboru `aas-low-level.txt` ve složce `var/log`. Tyto hlášky lze použít pro diagnostiku, případně ladění programu.

Kromě zmíněných log hlášek je dalším komunikačním bodem programu mqtt protokol. Pomocí tohoto protokolu se na lokálně instalovaný broker posílají notifikace jak v textové podobě určené pro ladění, tak především strojově zpracovatelné zprávy ve formátu json.

Notifikace o stisknutém tlačítku je posílána do topicu `i2c/touch` ve formátu.

```
{
  "button": btn
}
```

Kde `btn` reprezentuje číslo stisknutého tlačítka v rozsahu 1 – 6.

Další notifikací je informace o přiloženém tagu. Tato informace je posílána do topicu `spi/reader/data/read`. Příklad těchto informací:

```
{
  "data": [
    [4, 131, 105, 102],
    [66, 236, 76, 129],
    [99, 72, 0, 0],
            ⋮
            ⋮
    [0, 0, 0, 0],
    [0, 0, 0, 0]
  ],
  "tag": {
    "tag_type": "NTAG213",    // typ tagu
    "tag_size": 144,          // velikost paměti tagu
    "tag_vendor": "NXP",      // výrobce tagu
    "user_memory_offset": 4,  // první sektor do kterého lze zapisovat
    "tag_protocol": 3         // číslo protokolu
  },
  "read_state": "OK",
  "uid": [4, 131, 105, 66, 236, 76, 129],
  "timestamp": 1588328123.5859733
```

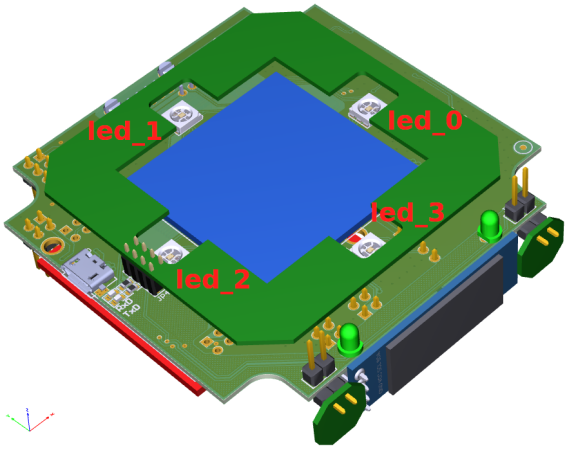
```
}
```

Tato zpráva obsahuje pole `data`, které obsahuje pro NTAG 213 45 polí dat uložených v tagu. Z prvních třech polí je vyextrahováno UID – jedinečná identifikace tagu a uložena do pole `uid`. Položka `timestamp` obsahuje takzvaný Unix Time Stamp v zóně UTC, tedy počet vteřin, které uplynuly od 1.1.1970. Položka `tag` obsahuje různé informace o přiloženém tagu. Tato zpráva je odeslána při přiložení tagu ke čtečce pouze jednou. Opětovné odeslání se provede v případě, že dojde k chybě při čtení, nebo k oddálení tagu a k jeho opětovnému přiložení.

Další zprávy, které program podporuje jsou příkazové. Jedním z příkazů je nastavení stavu LED. Tento příkaz nadřazený systém posílá do topicu `spi/led`. Příkaz vypadá následovně:

```
{
  "led_0": {
    "brightness": 50,
    "red": 50,
    "green": 0,
    "blue": 0
  },
  "led_1": {
    "brightness": 50,
    "red": 0,
    "green": 50,
    "blue": 0
  },
  "led_2": {
    "brightness": 50,
    "red": 0,
    "green": 0,
    "blue": 50
  },
  "led_3": {
    "brightness": 50,
    "red": 20,
    "green": 50,
    "blue": 100
  }
}
```

Každá z LED zde obsahuje nastavení jasu v rozsahu 0 – 100 % (položka `brightness`) a nastavení úrovně barevných složek v úrovni 0 – 255 (položky `red`, `green` a `blue`). LED jsou číslovány dle Obrázek 10.



Obrázek 32 - číslování LED diod

Pro nastavení zobrazovaného textu, případně vymazání dat na displeji slouží další topic, `i2c/display`. Příkaz pro zobrazení statického textu na daných souřadnicích s fontem Arial o velikosti 10 pixelů následuje.

```
{
  "cmd": "write",
  "text": "Hello world",
  "font": "Arial-10",
  "pos_x": 10,
  "pos_y": 2
}
```

Pro zobrazení běžícího textu se zpráva změní tak, že se vynechá klíč `pos_x` a položka `cmd` se změní na `scroll`.

```
{
  "cmd": "scroll",
  "text": "Hello world",
  "font": "Arial-10",
  "pos_y": 2
}
```

Pro vymazání obsahu displeje je zpráva redukována jen na položku `cmd`.

```
{
  "cmd": "clear"
}
```

Pokud nejsou nainstalována žádná jiná písma, lze vybírat z možností: `Arial-10`, `Arial-12`, `Arial-15`, `Vafle_VUT_Regular-12`, `Vafle_VUT_Regular-12` a `Vafle_VUT_Regular-15`. Některý z těchto názvů lze tedy zapsat do položky `font`.

Poslední sada příkazů slouží pro zápis uživatelských dat do vnitřní paměti tagu. Za tímto účelem se využívá topic `spi/reader/data/write`. Zde existují dva různé příkazy. Ten první zapisuje jeden sektor dat do určeného sektoru.

```
{
  "write": [
    {
      "data": [5, 3, 8, 10],
      "sector": 15
    }
  ]
}
```

Další příkaz se stará o to, aby bylo možné zapsat více sektorů naráz.

```
{
  "write_multi": [
    {
      "data": [1, 3, 5, 7],
      "sector": 8
    },
    {
      "data": [2, 4, 6, 8],
      "sector": 9
    }
  ]
}
```

Položka `data` obsahuje zapisovaná data a položka `sector` značí číslo sektoru do kterého se bude zapisovat s indexací od 0. Program nekontroluje, zda lze do zadaného sektoru zapisovat. Signalizace úspěšného zápisu je provedena pomocí zprávy v topicu `spi/reader/state`.

```
{
  "write": {
    "sector": 8,
    "status": "OK"
  }
}
```

5.5 Adaptér hardwarových periférií pro komunikaci přes Modbus TCP

Pro případ, že neexistuje nadřazená aplikace, která by komunikovala s aplikací `aas-low-level` přímo na AAS modulu, lze nainstalovat a spustit aplikaci `aas-modbus` [22]. Tato aplikace se stará o přemostění komunikace z MQTT na Modbus TCP. Díky tomuto lze komunikovat s AAS modulem například z PLC. Instalace a provozování tohoto programu není v rozporu s případným provozem jiné instance Modbus TCP například nadřazenou aplikací. Jedinou podmínkou, aby tyto aplikace mohly vedle sebe koexistovat, je potřeba vyhnout se konfliktu komunikačního portu.

Protokol Modbus TCP byl vyvinut na základech Modbus RTU a byl přizpůsoben pro použití v počítačových sítí. Protokol je založen na principu master-slave. V režimu master se obvykle nachází centrální prvek, například PLC, které se dotazuje na zařízení typu slave, v tomto případě na AAS modul.

Aplikace se zaregistruje na odběr notifikačních zpráv z aplikace `aas-low-level` přes MQTT. Tyto zprávy jsou v aplikaci zpracovány a uloženy do datové struktury vhodné pro vyčítání přes Modbus TCP. Datová struktura může mít podobu zprávy ve formátu MessagePack [21], ve formátu JSON, případně jsou data namapována přímo do registrové struktury. Formát ukládaných dat se nastavuje v souboru `config.json`, nacházejícím se v adresáři programu.

Tabulka 3 - přehled slave id a jejich popis ve výchozím stavu

slave id	popis
0	data přečtená z tagu
1	příkaz pro zápis dat do tagu
2	stav zápisu dat do tagu
3	číslo stisknutého tlačítka
4	příkazy pro displej
5	příkazy pro LED diody

Pro použití formátu MessagePack je zapotřebí nastavit volbu `use_msgpack` na `true` a volbu `use_registers` na `false`. Pro ukládání užitných dat přímo do registrů se tato volba obrátí, nastaví se tedy `use_msgpack` na `false` a volbu `use_registers` na `true`. Nastavením obou voleb na `false` se budou data ukládat přímo ve formátu JSON, respektive v jeho ASCII vyjádření, kdy jeden registr odpovídá jednomu znaku v ASCII. V konfiguračním souboru lze také nastavit komunikační síťový port aplikace a slave id jednotlivých datových prostorů. Pokud aplikace tento soubor nenalezne, použijí se výchozí hodnoty. Pro komunikační port to je 5020. Jako formát zpráv se zvolí MessagePack. Výchozí identifikátory slave id jsou popsány v Tabulka 3. Pod pojmem slave id zde rozumíme identifikátor virtuální jednotky. Do každé z těchto jednotek je namapován datový prostor. V každém z těchto prostorů se ukládá jiný typ dat.

Stejný formát, který je využit pro komunikaci směrem z AAS modulu, je použit i pro komunikaci směrem do AAS modulu.

Formát MessagePack slouží k redukování počtu znaků JSON zprávy. Tato redukce je provedena nahrazením určitých znaků, případně datových typů vyhrazenými hexadecimálními hodnotami. Následující zprávu z identifikátoru jednotky 3, lze redukovat pomocí MessagePack do zprávy `0x81 0xA6 button 0x05`. Zde hodnota 0x81 znamená, že zpráva je jednoprvková. Hodnota 0xA6 znamená, že řetězec, jež po této hodnotě následuje má délku 6 znaků a 0x05 je hodnota zmíněného klíče, tedy `button`. V registrovém prostoru by takto redukovaná zpráva byla vyjádřena pomocí

sekvence dvanácti registrů počínaje registrem 0: `0x81 0xa6 0x62 0x75 0x74 0x74 0x6f 0x6e 0x05`.

```
{
  "button": 5
}
```

Pomocí MessagePack byla tedy zmenšena velikost zprávy ze 12 znaků na 9 znaků. Následující registr po konci takto zpracované zprávy obsahuje výchozí hodnotu 0xFFFF.

Předchozí zprávu lze také zapsat do registrového prostoru jako JSON v ASCII formátu. Takto zapsaná zpráva je složena ze sekvence třinácti registrů počínaje registrem 0: `0x7b 0x22 0x62 0x75 0x74 0x74 0x6f 0x6e 0x22 0x3a 0x20 0x35 0x7d`. Následující registr po konci takto zpracované zprávy obsahuje výchozí hodnotu 0xFFFF.

Poslední možností jak tuto zprávu zapsat do datového prostoru je přímé mapovaní do registrů. Uvedená zpráva je namapována tak, že požadovaná hodnota je uložena do registru 0.

Protože ukládání zpráv ve formátech JSON a MessagePack se vždy řídí stejnými pravidly, nevyžaduje nadále další popis ke konkrétním zprávám. Proto bude uveden popis vždy jen k ukládání pomocí registrů.

Tabulka 4 - přehled registrů uložených ve slave id 0 pro data přečtená z tagu

offset registru	popis dat v registrech
0 – 6	`uid` – každý bajt pole je uložen do jednoho registru; uloženo ve stejné posloupnosti v jaké je odesláno z `aas-low-level`
7	0xFFFE – oddělovač
8	`tag_protocol` – číslo protokolu
9	`tag_size` – velikost paměti tagu
10	`user_memory_offset` – první sektor, do kterého lze zapisovat
11	0xFFFE – oddělovač
$12+N*5$	první položka ze sektoru N ($N \in \langle 0; po\check{c}etsektor\mathring{u}-1 \rangle$)
$13+N*5$	druhá položka ze sektoru N ($N \in \langle 0; po\check{c}etsektor\mathring{u}-1 \rangle$)
$14+N*5$	třetí položka ze sektoru N ($N \in \langle 0; po\check{c}etsektor\mathring{u}-1 \rangle$)
$15+N*5$	čtvrtá položka ze sektoru N ($N \in \langle 0; po\check{c}etsektor\mathring{u}-1 \rangle$)
$16+N*5$	0xFFFE – oddělovač sektoru N ($N \in \langle 0; po\check{c}etsektor\mathring{u}-1 \rangle$)

Data, která jsou přečtena z tagu jsou ve slave id 0 popsána v Tabulka 4.

Tabulka 5 - přehled registrů uložených ve slave id 1 pro zápis jednoho sektoru dat do tagu

offset registru	popis dat v registrech
0 – 3	každý z registrů obsahuje jeden z bajtů ukládaných seřazených dat
4	číslo sektoru
5	0xFFFE – oddělovač

Tabulka 6 - přehled registrů uložených ve slave id 1 pro zápis více sektorů dat do tagu

offset registru	popis dat v registrech
$0+N*6$	první položka ze sektoru N ($N \in \langle 0; po\check{c}etsektorů - 1 \rangle$)
$1+N*6$	druhá položka ze sektoru N ($N \in \langle 0; po\check{c}et sektorů - 1 \rangle$)
$2+N*6$	třetí položka ze sektoru N ($N \in \langle 0; po\check{c}etsektorů - 1 \rangle$)
$3+N*6$	čtvrtá položka ze sektoru N ($N \in \langle 0; po\check{c}etsektorů - 1 \rangle$)
$4+N*6$	číslo sektoru N ($N \in \langle 0; po\check{c}etsektorů - 1 \rangle$)
$5+N*6$	0xFFFE – oddělovač sektoru N ($N \in \langle 0; po\check{c}etsektorů - 1 \rangle$)

Příkaz pro zápis jednoho sektoru dat do tagu ve slave id 1 je popsán v Tabulka 5. Tabulka 6 popisuje registry pro zápis více sektorů.

Tabulka 7 - přehled registrů uložených ve slave id 2

offset registru	popis dat v registrech
0	číslo sektoru
1	stav zápisu (1 pro OK, 0 pro NOK)

Stav zápisu dat do tagu je ve slave id 2 popsán v Tabulka 7.

Tabulka 8 - přehled registrů uložených ve slave id 4 pro zápis na displej

offset registru	popis dat v registrech
0	příkaz pro displej (0 – vymazání displeje; 1 – statický text; 2 – běžící text)
1	`pos_x` – souřadnice x textu
2	`pos_y` – souřadnice y textu
$\langle 3; N \rangle$	`font` – název použitého písma, np.: `Arial-10`
$N+1$	0xFFFE – oddělovač
$\langle N+2; M \rangle$	`text` – zobrazovaný text
Pozn.: pokud je obsah registru 0 roven 0 (vymazání displeje), dále se žádný registr nevyplňuje	

Příkazy pro zápis na displej jsou mapovány ve slave id 4. Jejich popis je uveden v Tabulka 8.

Tabulka 9 - přehled registrů uložených ve slave id 5 pro rozsvícení LED diod

offset registru	popis dat v registrech
$0+N*4$	úroveň červené složky LED diody číslo N ($N \in \langle 0;3 \rangle$) v rozsahu 0 – 255
$1+N*4$	úroveň zelené složky LED diody číslo N ($N \in \langle 0;3 \rangle$) v rozsahu 0 – 255
$2+N*4$	úroveň modré složky LED diody číslo N ($N \in \langle 0;3 \rangle$) v rozsahu 0 – 255
$3+N*4$	úroveň jasu LED diody číslo N ($N \in \langle 0;3 \rangle$) v rozsahu 0 – 100

Tabulka 9 popisuje mapování příkazu pro rozsvícení LED diod zvolenou barvou.

5.6 Instalace operačního systému

NanoPi NEO Core obsahuje v závislosti na zakoupené verzi interní eMMC paměť. V této paměti je již předinstalován linuxový operační systém. Pro účely snazšího ladění, případně pro hromadnou výrobu je vhodnější využít µSD paměťovou kartu pro průmyslové použití. Takovouto paměťovou kartu je zapotřebí správně naformátovat a nahrát na ni požadovaný operační systém.

Prvním krokem pro instalaci systému je získání vhodného obrazu systému. Ten lze stáhnout z internetových stránek nebo vytvořit na počítači. Případně je možné duplikovat již existující obsah paměťové karty s nainstalovaným systémem.

Pro vývoj aplikací a následnou distribucí byl používán operační systém FriendlyCore ve verzi 4.14 založený na jádře systému Ubuntu 16.04. Tento systém byl stažen z oficiálního úložiště [23]. Ke stažení jsou zkomprimované soubory typu zip. Velikost takového archivu je asi 400 MB. Uvnitř archivu je obraz systému s vysokou úrovní komprese o velikosti asi 2,4 GB. Archiv není zapotřebí rozbalovat. Softwarové vybavení zajišťující nahrání obrazu na paměťovou kartu si archiv zvládne interně rozbalit. Pro nahrávání obrazu lze použít například multiplatformní program balenaEtcher [24]. Tento program po spuštění provede uživatele vybráním a nahráním obrazu na vybrané paměťové médium. Kromě samotného nahrání, zajistí i rozdělení paměťové karty na logické disky a jejich naformátování. Oddíl pro boot sekci se naformátuje na fat16 systém souborů. Zbývající oddíly pro data a systém se naformátují na systém souborů ext4. Takto připravenou paměťovou kartu lze zasunout do připraveného slotu na modulu NanoPi NEO Core. Připojením napájecího napětí (alespoň 13,5V) ke konektoru J301 dojde k zapnutí AAS modulu. Alternativní způsob napájení je použití PoE. Pokud je modul připojen pomocí ethernetového kabelu k počítačové síti, je možné se k modulu vzdáleně připojit. K připojení lze použít ssh protokol. Vhodným programem pro ssh je například program PuTTY [25]. Po vytvoření spojení je uživatel vyzván k zadání uživatelského jména. V systému jsou vytvořeni ve výchozím stavu dva uživatelé:

1. běžný uživatel:

- uživatelské jméno: `pi`
- heslo: `pi`

2. uživatel s právy super uživatele:
   - uživatelské jméno: `root`
   - heslo: `fa`

Po zadání uživatelského jména je uživatel vyzván k zadání hesla. Zadávání hesla není indikováno zapsanými znaky ani hvězdičkami – je skryto.

Protože je používání již prefabrikovaného obrazu operačního systému značně omezující a z hlediska sériové produkce velmi obtížné, je vhodné obraz sestavit svépomocí. Často se za tímto účelem využívají takzvané docker kontejnery. Tyto kontejnery umožňují vytvořit v operačním systému oddělené místo pro aplikace. Pro NanoPi byl nalezen návod [27] a zdrojové soubory umožňující přípravu obrazu sytému. Dle uvedeného návodu by mělo být možné vytvořit obraz systému s upravenými soubory. Tento repozitář obsahuje sadu shell skriptů umožňujících například z několika částí obrazu vytvořit jeden kompletní.

S ohledem na potřeby projektu by bylo vhodné mít možnost nainstalovat přímo do obrazu systému různé potřebné programy. Mezi těmito programy je například i MQTT broker mosquitto. Pro zmíněný způsob tvorby obrazu bohužel však nebyl nalezen způsob jak nainstalovat program. Touto procedurou by tak mohly být pouze uloženy například knihovny pro Python, případně samotné programy `aas-low-level` a `aas-modbus`.

5.7 Instalace programů

Obě aplikace obsahují kromě zdrojových kódů, které jsou interpretovány interpretem, také shell skripty `install.sh`. Tento skript po spuštění pomocí příkazu `sh install.sh` zavolá několik příkazů. Pro aktualizaci seznamu balíků a jejich zdrojů slouží první z příkazů v souboru. Další příkaz zajistí nainstalování mosquitto brokeru, mosquitto klientů, balíkovacího systému pro python a dalších programů, které jsou nutné pro správnou funkci Pillow knihovny. Poslední příkaz zajistí nainstalování potřebných Python knihoven pro samotný skript pomocí seznamu požadavků, souboru `requirements.txt`.

Po úspěšné instalaci je možné programy spustit. V případě programu `aas-low-level` je pro spuštění určen příkaz `sudo python3 aas-low-level.py`. Pro `aas-modbus` to je příkaz `sudo python3 aas-modbus.py`. Takto spuštěný program bude spuštěn do momentu zavření konzole, případně ukončení spojení.

Pro spuštění na pozadí je vhodné program spustit jako službu. K tomu slouží skript `start_service.sh`. Následující příklad ukazuje chování v případě programu pro obsluhu hardware. Tento skript, po spuštění pomocí `sh start_service.sh`, zajistí zkopírování souboru popisující službu z cesty `service/aas-low-level.service`

do systémové složky `/lib/systemd/system`. Poté se pomocí příkazu `sudo systemctl enable aas-low-level.service` vytvoří symlink na tento soubor v cestě `etc/systemd/system/multi-user.target.wants/aas-low-level.service`. Příkaz `sudo systemctl start aas-low-level.service` službu spustí. Nadále je pak možnost používat příkaz `sudo systemctl command aas-low-level.service`, kde slovo `command` lze nahradit výrazem `status` pro zjištění stavu služby, `stop` pro její zastavení, `restart` pro restartování, apod. Další popis je uveden v manuálových stránkách [26].

Aby bylo možné spustit zmíněné shell skripty, případně ručně spustit Python programy je zapotřebí, aby se prompt příkazové řádky nacházel v kořenové složce programu.

Pokud se program nachází v jiné cestě než `/home/pi/aas-low-level`, případně `/home/pi/aas-modbus`, je zapotřebí upravit tuto cestu i v příslušných `*.service` souborech a to v sekcích `WorkingDirectory` a `ExecStart`. V tomto souboru se také nachází podmínka, kterou je třeba splnit než se program spustí. Tato podmínka se nachází v sekci `ExecStartPre`. Ve výchozím stavu je to podmínka na úspěšný výsledek ping příkazu na server google.com. Tato podmínka je zde zvolena tak, aby došlo ke spuštění programu až po obdržení IP adresy zařízení. Podmínku je možné libovolně upravit, změnit podmínku na vykonání nějakého příkazu nebo je možné tuto sekci odstranit.

6 ZÁVĚR

Byly zformulovány požadavky na AAS modul. Podle požadavků došlo k výběru komponent, které byly pro realizaci použity.

Tyto požadavky byly zformulovány na základě konceptu projektu testbed Barman, což je platforma pro Průmysl 4.0.

Dále byly popsány principy použité při návrhu první revize hardware pro AAS modul. Tato revize byla otestována a předána k vývoji obslužného software. Dále byly z chyb v této revizi vyvozeny důsledky a poučení z nich bylo použito při návrhu revize 1.2 plošného spoje.

Pátá kapitola se zabývá popisem obslužného software. V této kapitole byly představeny a rozebrány různé způsoby komunikace mezi procesy. Dále byly popsány formáty datových zpráv, které jsou vhodné pro předávání hodnot nejen mezi procesy. Mimo jiné zde byly popsány i postupy pro instalaci programů, jež byly v rámci této práce vytvořeny. V této kapitole byly zmíněny i možnosti tvorby vlastního obrazu s operačním systémem. Došlo i k několika pokusům o vytvoření vlastního obrazu. Bohužel však se nepodařilo obraz úspěšně vytvořit tak, aby bylo možné jej načíst v NanoPi NEO Core modulu.

Celkově tedy byly navrhnuty a vyrobeny dvě revize plošných spojů pro AAS modul. Pro tyto moduly byly napsány dva obslužné programy v jazyce Python. První program obsluhuje hardwarové periférie. Druhý program umožňuje přemostění MQTT protokolu, použitého pro komunikaci mezi procesy, a JSON zpráv do Modbus TCP datového prostoru. Tyto obslužné programy byly na oživeném hardware otestovány a zdokumentovány.

LITERATURA

[1] KACZMARCZYK, Václav, Ondřej BAŠTÁN, Zdeněk BRADÁČ a Jakub ARM. An Industry 4.0 Testbed (Self-Acting Barman): Principles and Design. *IFAC-PapersOnLine* [online]. 2018, 51(6), 263-270 [cit. 2019-01-02]. DOI: 10.1016/j.ifacol.2018.07.164. ISSN 24058963. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S2405896318309108

[2] POWER OVER ETHERNET (POE). In: *B+B SmartWorx* [online]. [cit. 2019-01-02]. Dostupné z: http://www.bb-elec.com/Learning-Center/All-White-Papers/Ethernet/Power-over-Ethernet-PoE.aspx

[3] OLED I2C Displej 128x32 0,91". In: *Návody Arduino* [online]. [cit. 2019-01-02]. Dostupné z: https://navody.arduino-shop.cz/navody-k-produktum/oled-i2c-displej-128x32.html

[4] RFID Reader/Writer, NFC module, MFRC522. In: *RobotDyn* [online]. [cit. 2019-01-02]. Dostupné z: https://robotdyn.com/rfid-reader-writer-nfc-module-mfrc522.html

[5] PSE Operating Modes and Sequence. In: *Embedded* [online]. [cit. 2019-01-02]. Dostupné z: https://www.embedded.com/print/4014361

[6] Raspberry Pi Zero. In: *Raspberry Pi* [online]. [cit. 2019-01-02]. Dostupné z: https://www.raspberrypi.org/products/raspberry-pi-zero/

[7] NanoPi NEO. In: *FriendlyElec* [online]. [cit. 2019-01-02]. Dostupné z: https://www.friendlyarm.com/index.php?route=product/product&path=69&product_id=132

[8] NanoPi NEO Core. In: *FriendlyElec* [online]. [cit. 2019-01-02]. Dostupné z: https://www.friendlyarm.com/index.php?route=product/product&path=69&product_id=212

[9] *EVBUM2156 - Power-over-Ethernet PD Interface Evaluation Board User's Manual* [online]. [cit. 2019-01-02]. Dostupné z: https://www.onsemi.com/pub/Collateral/EVBUM2156-D.PDF

[10] led-control. In: *github.com/rici4kubicek* [online]. [cit. 2020-01-02]. Dostupné z: https://github.com/rici4kubicek/led-control

[11] oled-display. In: *github.com/rici4kubicek* [online]. [cit. 2020-01-02]. Dostupné z: https://github.com/rici4kubicek/oled-display

[12] rfid-reader. In: *github.com/rici4kubicek* [online]. [cit. 2020-01-02]. Dostupné z: https://github.com/rici4kubicek/rfid-reader

[13] A Brief, but Practical Introduction to the MQTT Protocol and its Application to IoT. In: *zoetrope* [online]. [cit. 2020-01-02]. Dostupné z: https://zoetrope.io/tech-blog/brief-practical-introduction-mqtt-protocol-and-its-application-iot/

[14] aas-low-level. In: *github.com/rici4kubicek* [online]. [cit. 2020-01-02]. Dostupné z: https://github.com/rici4kubicek/aas-low-level

[15] Eclipse Mosquitto [online]. [cit. 2020-05-13]. Dostupné z: https://mosquitto.org/

[16] dbus. In: *freedesktop.org* [online]. [cit. 2020-05-15]. Dostupné z: https://www.freedesktop.org/wiki/Software/dbus/

[17] Linux SysFS FPIO access via Python. In: *github.com/derekstavic* [online]. [cit. 2020-05-17]. Dostupné z: https://github.com/derekstavis/python-sysfs-gpio/
[18] MFRC522-python. In: *github.com/mxgxw* [online]. [cit. 2020-05-17]. Dostupné z: https://github.com/mxgxw/MFRC522-python/
[19] Adafruit-SSD1306. In: *PyPI* [online]. [cit. 2020-05-17]. Dostupné z: https://pypi.org/project/Adafruit-SSD1306/
[20] Pillow. In: *PyPI* [online]. [cit. 2020-05-17]. Dostupné z: https://pypi.org/project/Pillow/
[21] MessagePack: It‘s like JSON, but fast and small. [online]. [cit. 2020-05-17]. Dostupné z: https://msgpack.org/
[22] aas-modbus. In: *github.com/rici4kubicek* [online]. [cit. 2020-05-17]. Dostupné z: https://github.com/rici4kubicek/aas-modbus
[23] FriendlyElec H3. In: *Google Drive* [online]. [cit. 2020-05-17]. Dostupné z: https://drive.google.com/open?id=15yJCIVY2A3_lk4H0NEoN-EteLR_umKFD
[24] balenaEtcher – Flash OS images to SD cards & USB drives [online]. [cit. 2020-05-17]. Dostupné z: https://www.balena.io/etcher/
[25] PuTTY [online]. [cit. 2020-05-17]. Dostupné z: https://www.putty.org/
[26] systemctl(1). In: *Linux manual page*. [online]. [cit. 2020-05-17]. Dostupné z: https://www.man7.org/linux/man-pages/man1/systemctl.1.html
[27] sd-fuse_h3. In: *github.com/friendlyarm* [online]. [cit. 2020-05-17]. Dostupné z: https://github.com/friendlyarm/sd-fuse_h3

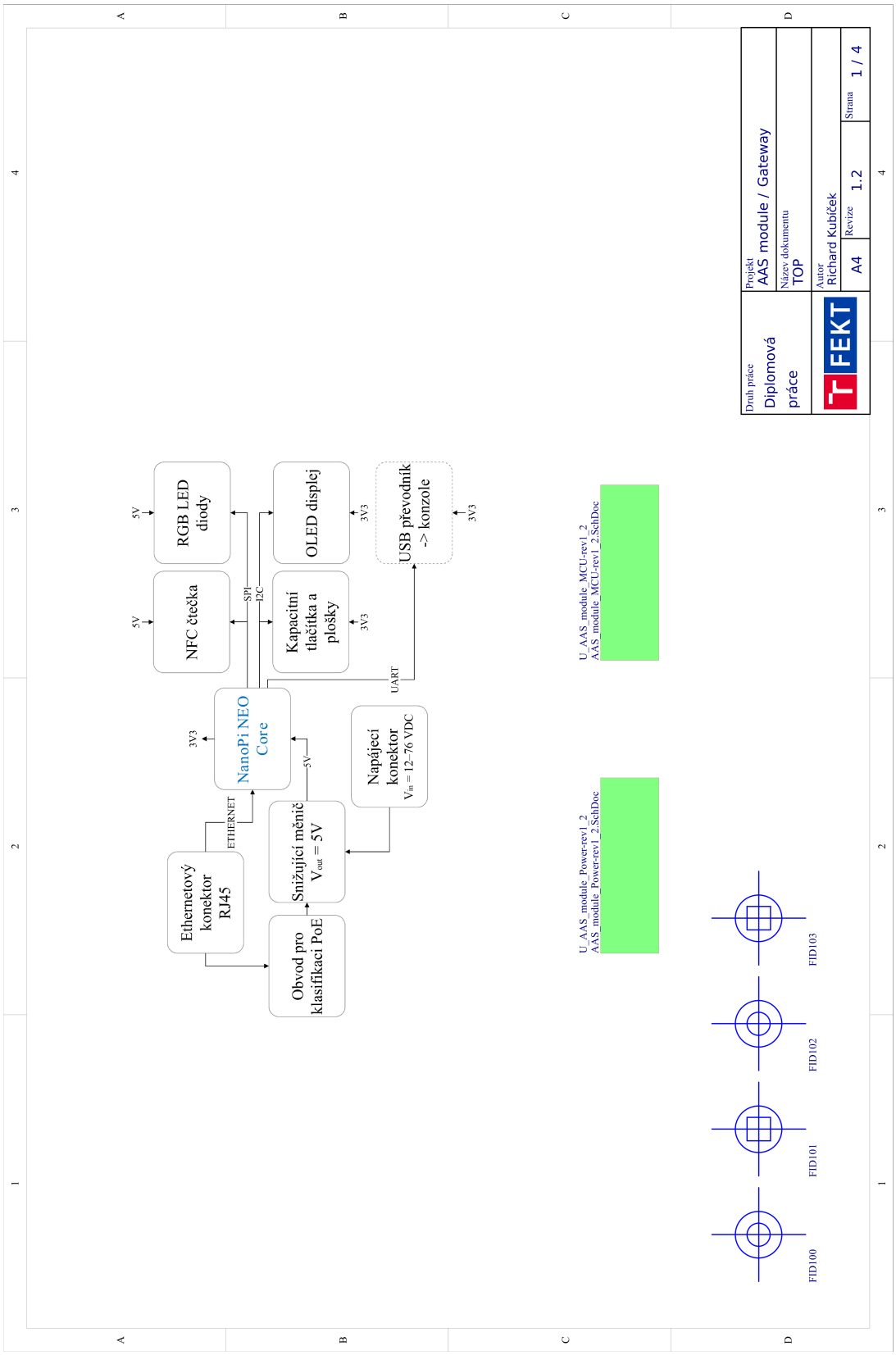
SEZNAM SYMBOLŮ, VELIČIN, ZKRATEK A CIZÍCH SLOV

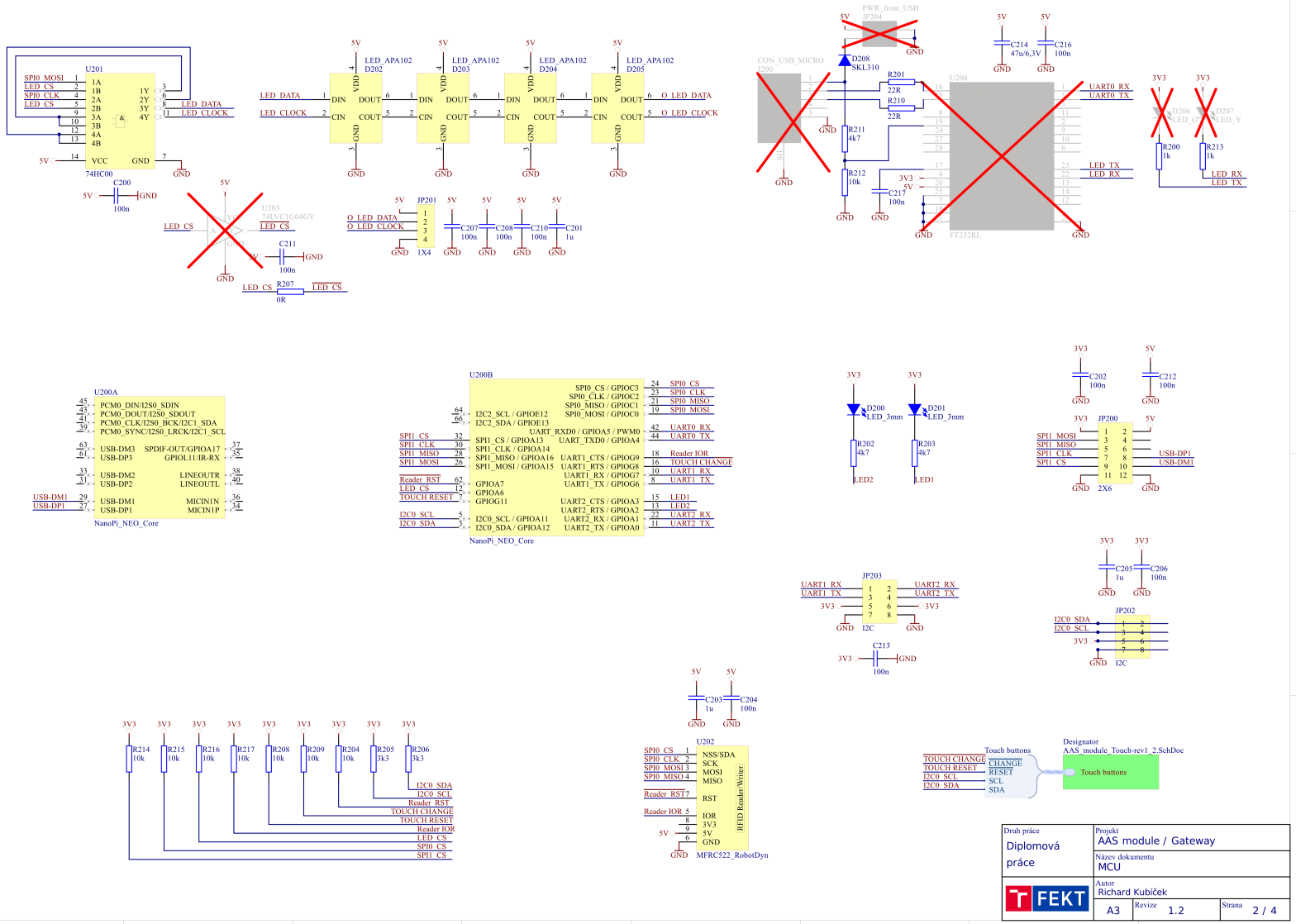
ASCII	American Standard Code for Information Interchange – kódová tabulka definující znaky v anglické abecedě
DICT	dictionary (slovník) – množina neuspořádaných dvojic klíč – hodnota v Pythonu
DPS	deska plošných spojů
eMMC	nevolatilní paměť jejíž princip je složen z principů paměťové karty a flash paměti
ERP	systém pro plánování podnikových zdrojů
FLASH	nevolatilní elektricky programovatelná paměť
I^2C	sériová sběrnice pro komunikaci dvěma vodiči
IoT	Internet of Things – internet věcí
IP adresa	číslo identifikující síťové rozhraní v síti
JSON	JavaScript Object Notation – způsob zápisu dat
LED	luminiscenční dioda
MASTER	hlavní
MES	výrobní informační systém tvořící vazbu mezi podnikovým informačním systémem a systémem pro automatizaci výroby
MQTT	MQ Telemetry Transport – jednoduchý a nenáročný protokol postavený nad TCP/IP
NFC	Near Field Communication – modulární technologie rádiové bezdrátové komunikace
OLED displej	zobrazovací displej využívající technologii organických elektroluminiscenčních diod
OPC UA	komunikační protokol průmyslové automatizace pro komunikaci stroj-stroj
PHY	obvod realizující fyzickou vrstvu OSI modelu
PLC	programovatelný logický automat
QoS	Quality of Services – kvalita služeb
RAM	volatilní polovodičová paměť s přímým přístupem
RFID	Radio Frequency Identification – identifikace zařízení pomocí rádiové frekvence
SHELL	intepretr příkazů vytvářející rozhraní pro uživatel
SLAVE	podružné
SMD	Surface Mount Device – součástka pro osazování na povrch plošného spoje

SPI	sériové periferní rozhraní používané pro komunikaci mezi řídicím procesorem a ostatními integrovanými obvody
SQL	Structured Query Language – standardizovaný strukturovaný dotazovací jazyk, užívá se pro přístup do relačních databází
SSH	Secure Shell – označení pro program a zabezpečený komunikační protokol
STEP 7	komunikační protokol firmy Siemens
SYMLINK	symbolický odkaz s absolutní či relativní cestou k souboru
THT	Through-Hole Technology – osazování desek plošných spojů součástkami s drátovými vývody
USB	univerzální sériová sběrnice
VUT	Vysoké učení technické
WiFi	označení pro standardy IEEE 802.11 popisující bezdrátovou síť
XML	eXtensible Markup Language – obecný značkovací jazyk vyvinutý a standardizovaný W3C

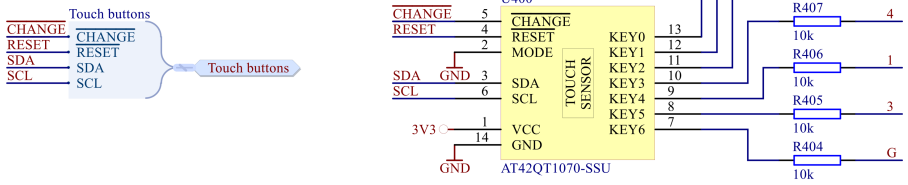
SEZNAM PŘÍLOH

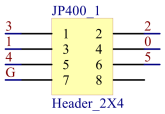
A. SCHÉMA ZAPOJENÍ AAS MODULU



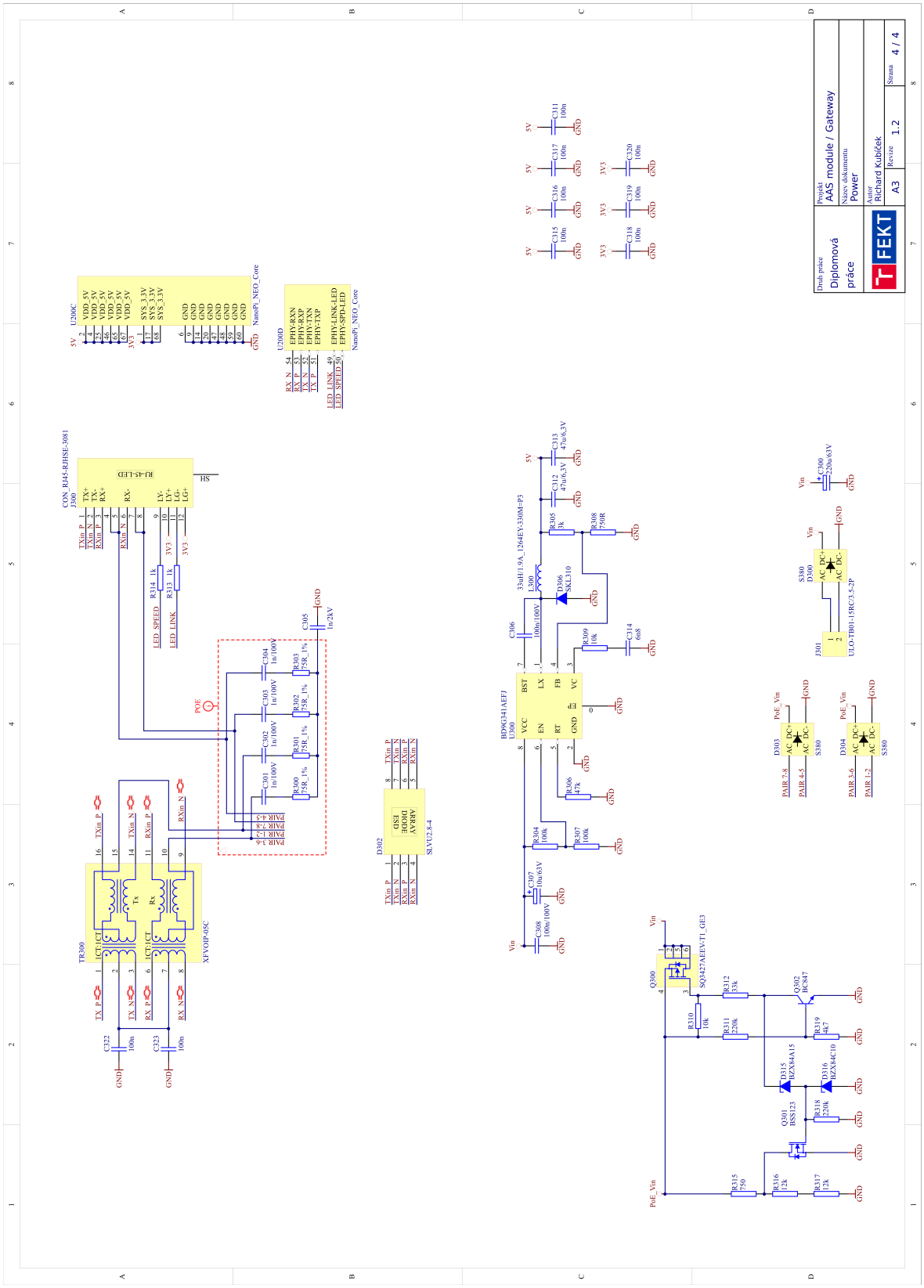


Druh práce: Diplomová práce	Projekt: AAS module / Gateway
	Název dokumentu: MCU
FEKT	Autor: Richard Kubíček
A3	Revize: 1.2
	Strana: 2 / 4

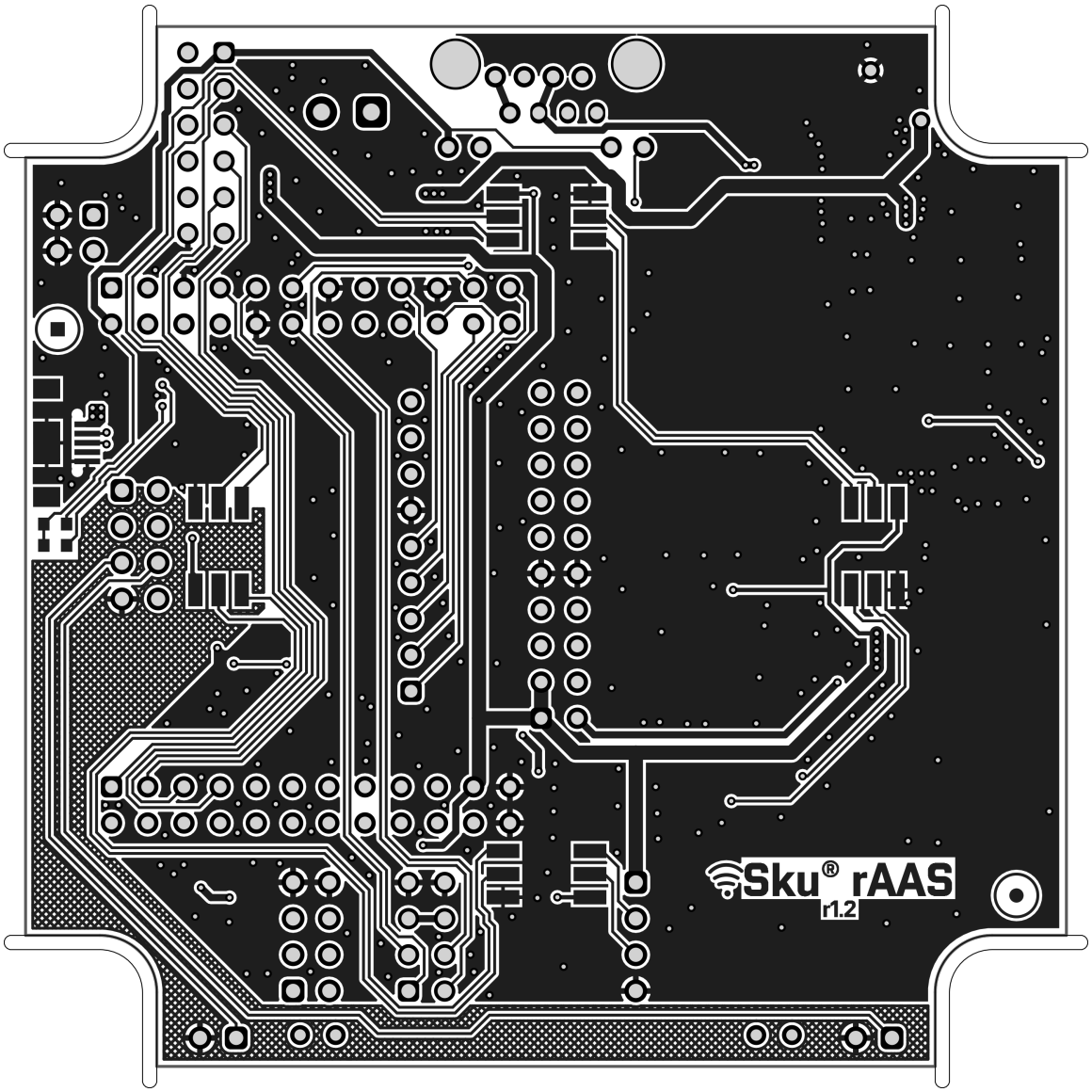




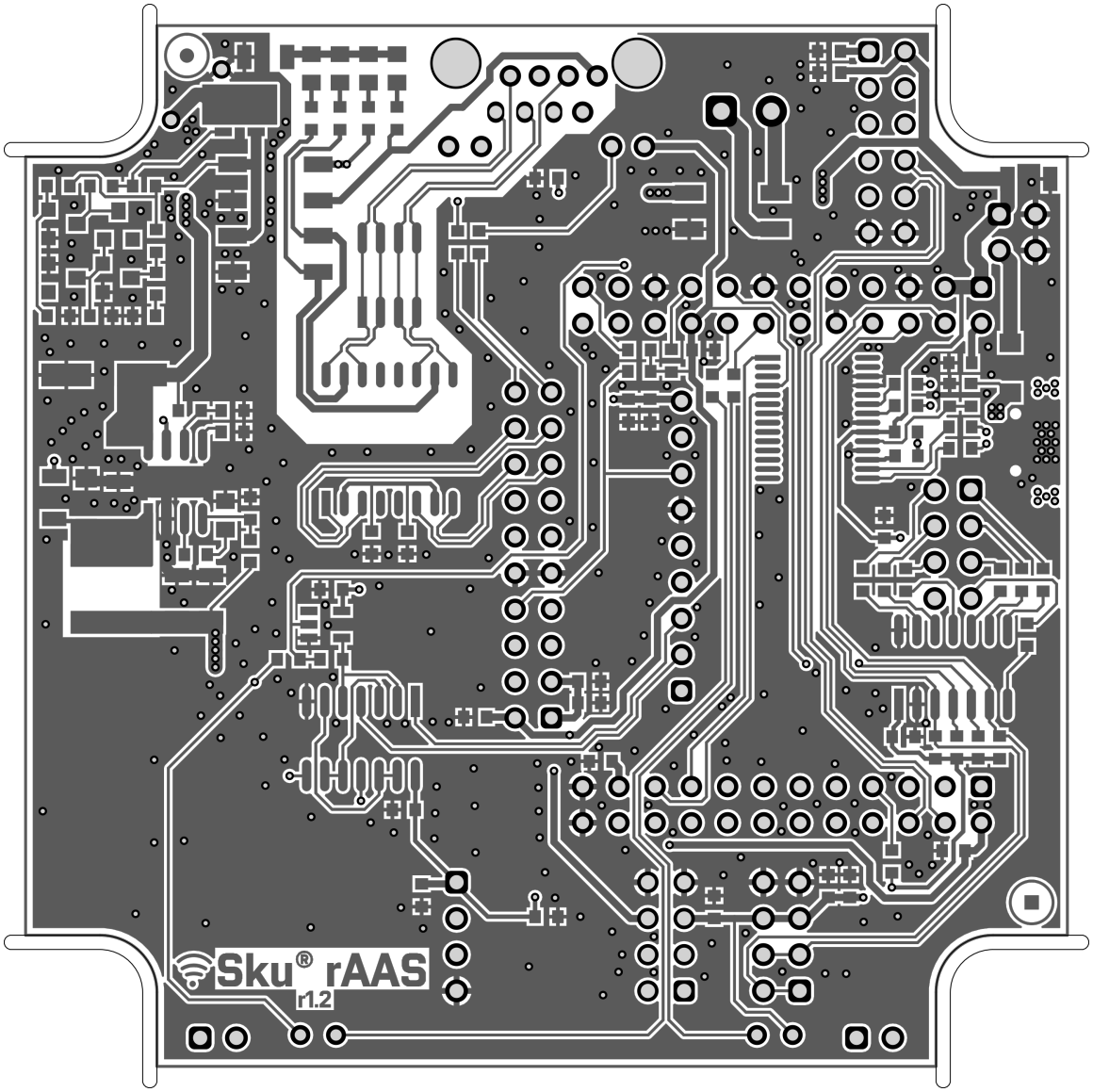
Druh práce: Diplomová práce	Projekt: AAS module / Gateway	
	Název dokumentu: Touch controller	
FEKT	Autor: Richard Kubíček	
	A4	Revize: 1.2 — Strana: 3 / 4


Druh práce
Diplomová práce
Projekt
AAS module / Gateway
Název dokumentu
Power
Autor
Richard Kubíček
A3
Revize
1.2
Strana
4 / 4
FEKT

B. MOTIV PLOŠNÉHO SPOJE AAS MODULU



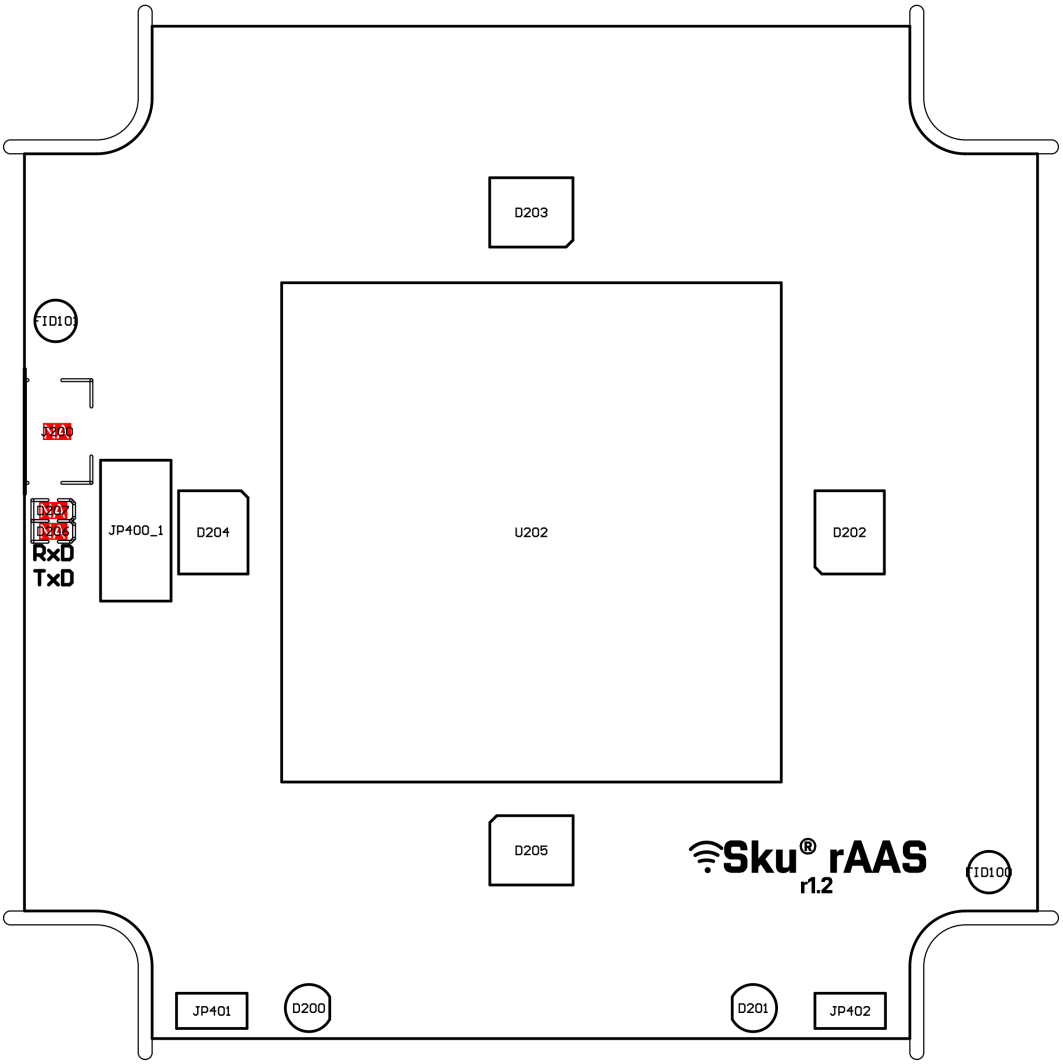
Motiv plošného spoje, strana TOP, měřítko 2:1, rozměry 73x73 mm



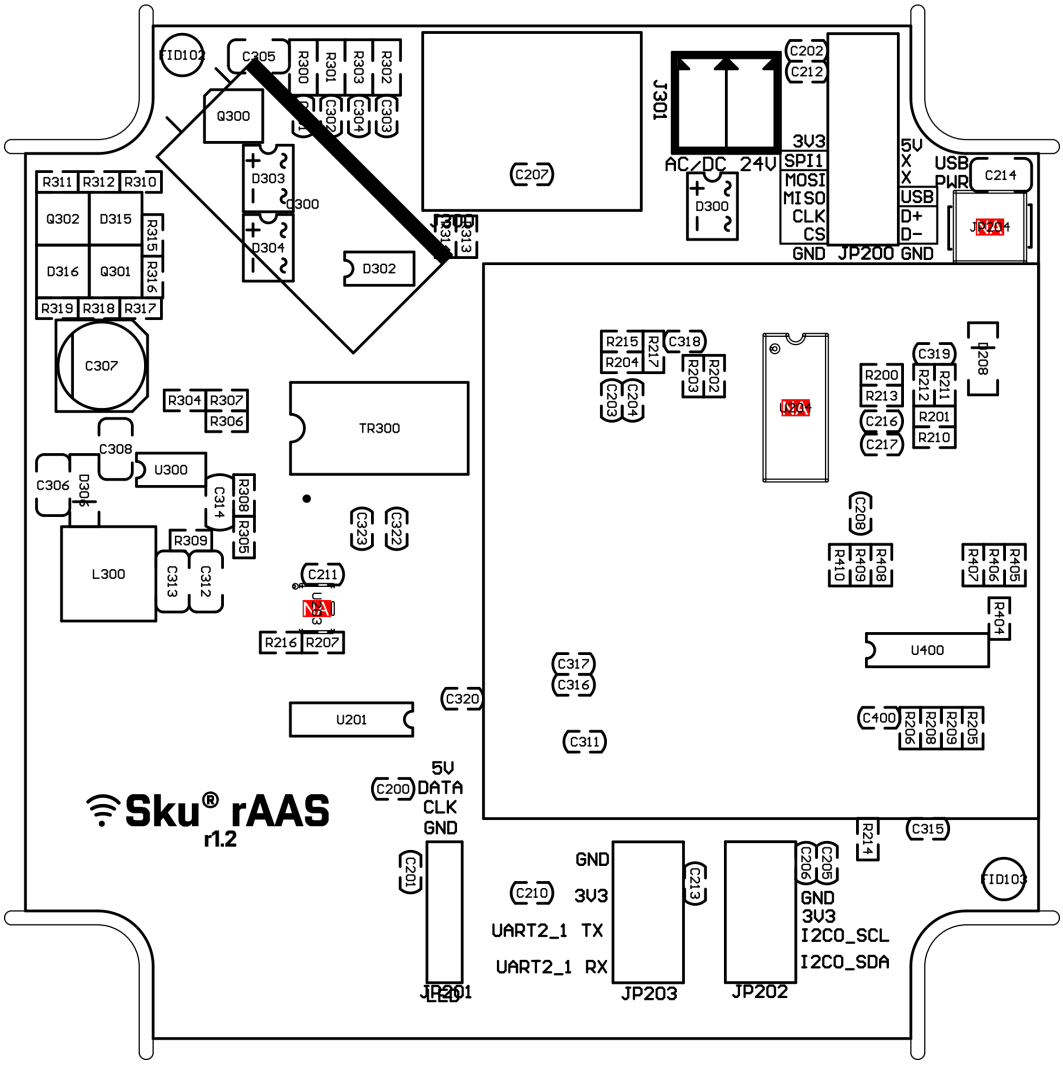
Motiv plošného spoje, strana BOTTOM, měřítko 2:1, rozměry 73x73 mm

C. OSAZOVACÍ VÝKRESY AAS MODULU

AAS_module_rev1_2.PcbDoc	
Variant STD	
TOP	31.05.2020



AAS_module_rev1_2.PcbDoc	
Variant STD	
BOTTOM	31.05.2020

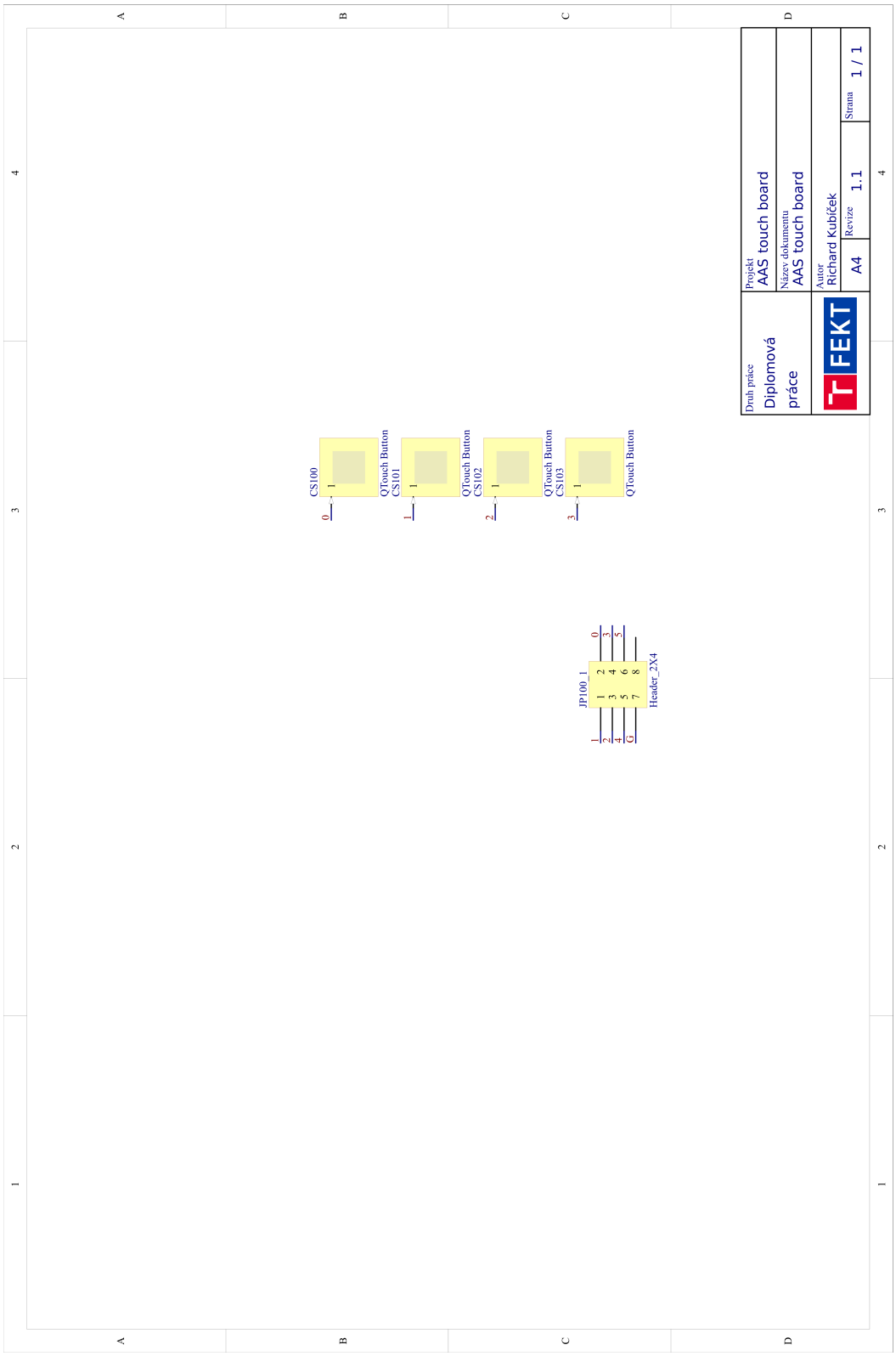


D. SEZNAM POUŽITÝCH SOUČÁSTEK AAS MODULU

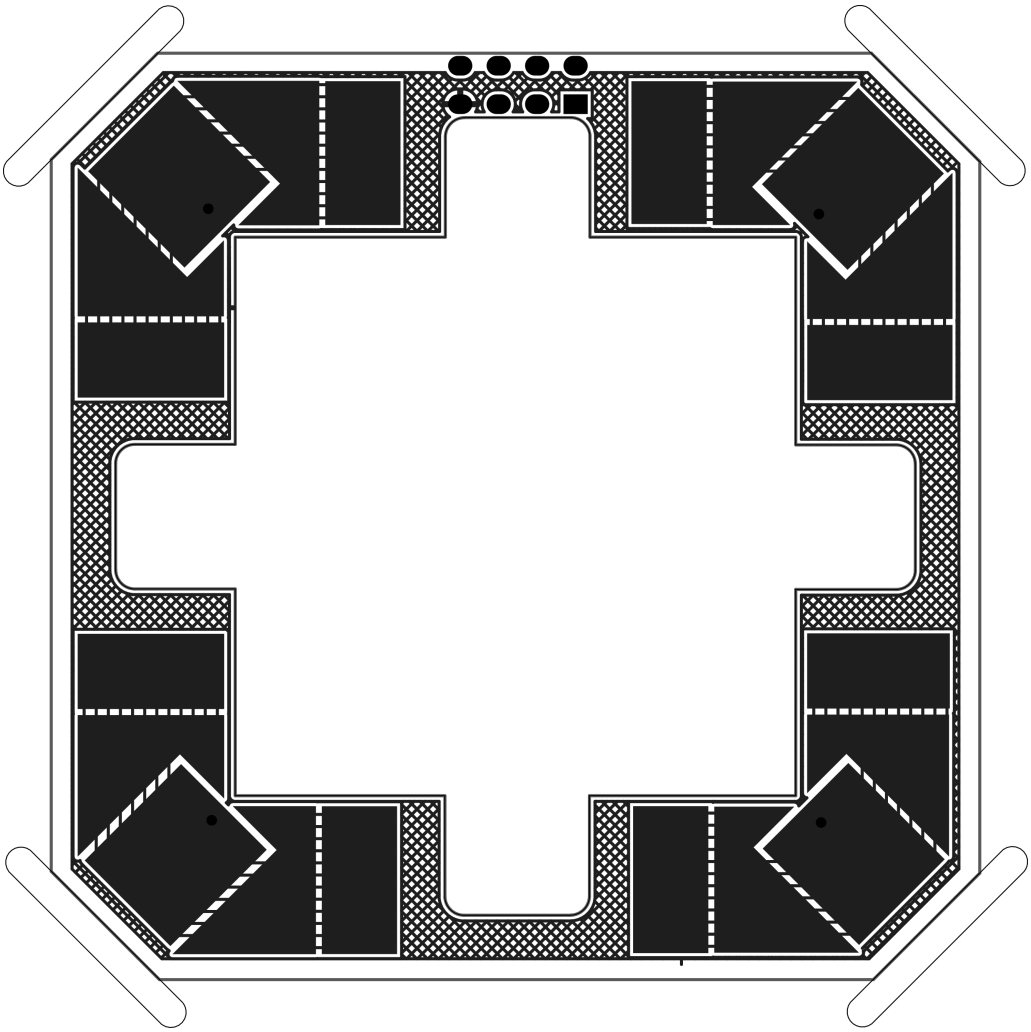
Hodnota	Komponent	Pouzdro	Počet
100n	C200, C202, C204, C206, C207, C208, C210, C211, C212, C213, C216, C217, C311, C315, C316, C317, C318, C319, C320, C322, C323, C400	C0603	22
1u	C201, C203, C205	C0603	3
47u/6,3V	C214, C312, C313	C1206	3
220u/63V	C300	CAP_D10xL20_P5_HR	1
1n/100V	C301, C302, C303, C304	C0603	4
1n/2kV	C305	C1206	1
100n/100V	C306, C308	C1206	2
10u/63V	C307	C_ALU_D6x5.8	1
6n8	C314	C0805	1
LED_3mm	D200, D201	LED_THT_3MM_P2.5_GREEN	2
LED_APA102	D202, D203, D204, D205	APA102	4
SKL310	D208, D306	SOD123	2
S380	D300, D303, D304	MBS	3
SLVU2.8-4	D302	SO-8	1
BZX84A15	D315	SOT-23	1
BZX84C10	D316	SOT-23	1
FID_CIRCLE	FID100, FID102	FID-CIRCLE1MM	2
FID_SQUARE	FID101, FID103	FID-SQUARE1MM	2
CON_RJ45-RJHSE-308	J300	RJHSE-3081	1
ULO-TB01-15RC/3.5-	J301	ULO-TB01-15VC-3.5-2P	1
2X6	JP200	HDR2x6	1
1X4	JP201	HDR1X4	1
I2C	JP202, JP203	HDR2X4	2
Header_2X4	JP400_1	HDR2X4	1
1X2	JP401, JP402	HDR1X2	2
33uH/1.9A_1264EY-	L300	L_MURATA-DG6050C	1
SQ3427AEEV-T1_GE3	Q300	SOT-23-6	1
BSS123	Q301	SOT-23	1
BC847	Q302	SOT-23	1
1k	R200, R213, R313, R314	R0603	4
22R	R201, R210	R0603	2
4k7	R202, R203, R211, R319	R0603	4
10k	R204, R208, R209, R212, R214, R215, R216, R217, R309, R310, R404, R405, R406, R407, R408, R409, R410	R0603	17
3k3	R205, R206	R0603	2
0R	R207	R0603	1
75R_1%	R300, R301, R302, R303	R_0805	4

Hodnota	Komponent	Pouzdro	Počet
100k	R304, R307	R0603	2
3k	R305	R0603	1
47k	R306	R0603	1
750R	R308	R0603	1
220k	R311, R318	R0603	2
33k	R312	R0603	1
750	R315	R0603	1
12k	R316, R317	R0603	2
XFVOIP-05C	TR300	XFVOIP-05	1
NanoPi_NEO_Core	U200	NanoPi_NEO_Core_right	1
74HC00	U201	SO-14	1
MFRC522_RobotDyn	U202	MFRC522_RobotDyn	1
BD9G341AEFJ	U300	SO-8_EP	1
AT42QT1070-SSU	U400	SO-14	1

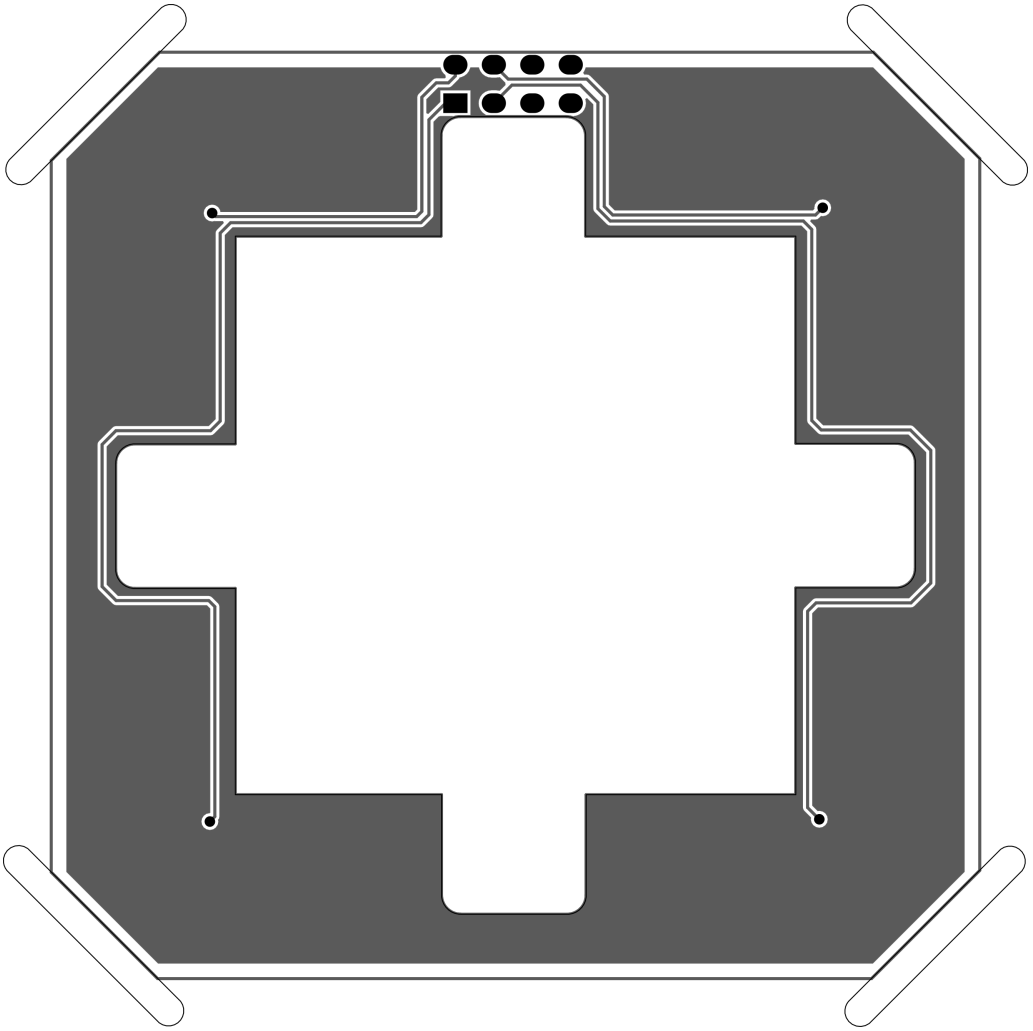
E. SCHÉMA DESKY DOTYKOVÝCH PLOŠEK



F. MOTIV PLOŠNÉHO SPOJE DOTYKOVÝCH PLOŠEK



Motiv plošného spoje, strana TOP, měřítko 2:1, rozměry 61,2x61,2 mm



Motiv plošného spoje, strana BOTTOM, měřítko 2:1, rozměry 61,2x61,2 mm