České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra mikroelektroniky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **M A L Ý Jan**

Studijní program: Komunikace, multimédia a elektronika
Obor: Aplikovaná elektronika

Název tématu: **Univerzální komunikační jednotka pro embedded zařízení**

Pokyny pro vypracování:

Navrhněte a realizujte univerzální komunikační jednotku pro oblast embedded zařízení umožňující řídicímu systému (např. PC) ovládat jednu nebo několik samostatných periferií.

1) Proveďte rešerši z oblasti komunikace řídicího systému (např. PC) s embedded zařízeními.
2) Navrhněte vhodné řešení pro ovládání embedded periferií prostřednictvím řídicího systému (např. PC).
3) Realizujte prototyp komunikační jednotky, která bude přijímat data z PC (místo PC můžete použít tablet nebo chytrý telefon) a na základě těchto informací bude ovládat připojené periferie.
4) Pro komunikační jednotku vytvořte řídicí firmware a její funkci demonstrujte na modelu minisumorobota.
5) Změřte vlastnosti Vámi navrženého systému a zhodnoťte dosažené výsledky.

Seznam odborné literatury:

[1] Robert B.Reese: Microprocessors From Assembly Language to C Using The PIC18Fxx2, Da Vinci Engineering Press, Hingham Massachusetts 2005
[2] Brian W. Kernighan, Dennis M. Ritchie: Programovací jazyk C, Computer Press, a.s., Brno 2006

Vedoucí: **Ing. Tomáš Teplý**

Platnost zadání: 31. 8. 2015

Prof. Ing. Miroslav Husák, CSc.
vedoucí katedry

Prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 5. 2. 2014

České vysoké učení technické v Praze

Fakulta elektrotechnická

Katedra mikroelektroniky

Studijní program: Komunikace, multimédia a elektronika

Obor: Aplikovaná elektronika

Univerzální komunikační jednotka pro embedded zařízení

BAKALÁŘSKÁ PRÁCE

Student : Jan Malý

Vedoucí : Ing. Tomáš Teplý

2014/2015

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a že jsem uvedl všechny použité prameny a literaturu, ze které jsem čerpal.

V Praze dne 21. 5. 2015 . . . . . .

Jan Malý

Poděkování

Rád bych poděkoval svému vedoucímu práce za trpělivost, cenné rady a pomoc při realizaci hardwarové části práci.

Abstrakt

Bakalářská práce popisuje návrh a realizaci univerzální komunikační jednotky pro oblast embedded zařízení umožňující řídicímu systému ovládat jednu nebo několik samostatných periferií. Výsledné řešení umožňuje nejen ovládat periferie způsobem podobným řešení webových služeb – vzdálenému volání metod, ale dovoluje i vzájemnou komunikaci mezi periferiemi s důrazem na rychlost a spolehlivost. Výsledný produkt je vytvořený tak, aby co nejvíce ulehčil práci implementátorovi embedded zařízení, proto práce kromě samotného návrhu a popisu řešení obsahuje i postup implementace.

Klíčová slova

Komunikace, embedded zařízení, mikrokontrolér, řízení systému, protokol

Abstract

Thesis describes design and realization of universal communication unit for area of embedded devices. Using proposed solution control system can manage several discrete peripheries in meaner similar to remote method invocation from web services whit speed and reliability in mind. Embedded system can also use this solution to communicate with each other. Implementation of communication on end device should be easy thanks to description of usage.

Keywords

Communication, embedded device, microcontroller, system control, protocol

Obsah

1 Úvod

Hledání způsobů jak si ulehčit život je jednou z typických lidských vlastností. V dnešní době se proto stále častěji setkáváme s různými sofistikovanými zařízeními vytvořenými za tímto účelem. Jedná se například o roboty, kteří za člověka přebírají pro něj nebezpečnou práci, automatické linky v hromadné výrobě, které dokáží produkovat výrobky výrazně rychleji a podstatně kvalitněji než lidé, nebo inteligentními systémy v budovách, jenž zajišťují komfort jejich obyvatel.

Všechna výše jmenovaná zařízení mají jedno společné – jsou to velmi komplexní systémy tvořené mnoha subsystémy. Integrací několika subsystémů do jednoho systému, je zajištěna funkcionalitu, kterou by samotné části nenabídly. Výzvou pro tuto práci je zajistit efektivní komunikaci mezi jednotlivými částmi podobného systému a umožnit jeho efektivní řízení.

1.1. Cíle práce

Cílem práce je navrhnout a realizovat univerzální komunikační jednotku pro oblast embedded zařízení umožňující řídicímu systému (například PC) ovládat jednu nebo několik samostatných periferií. Tato univerzální komunikační jednotka bude primárně použita v projektu vytvářeném na katedře mikroelektroniky pro dálkové ovládání modelu automobilu a vzájemnou komunikaci embedded modulů různých funkcí, ze kterých bude projekt vytvořen. Jelikož jsou požadavky velmi obecné povahy, musí být i výsledná podoba produktu universální. Výstupem práce tedy bude nejen deska plošného spoje realizující bezdrátové propojení soustavy modulů, ale i program pro vytvářenou jednotku, připojené embedded systémy a řídící zařízení – PC. Vzhledem k aplikaci by výsledný program pro embedded zařízení měl být velmi jednoduchý, nenáročný na implementaci a snadno přenositelný v duchu „separation of concerns“[1]. To samé platí i pro řídící software, jenž by měl být ještě navíc rozšiřitelný. Navržená deska by měla podporovat standadrdní rozhraní jak pro komunikaci s embedded zařízeními, tak i s řídící jednotkou.

Za účelem naplnění cíle bylo nutno stanovit několik podcílů:

- Provést rešerši z oblasti komunikace řídicího systému (PC) s embedded zařízeními.

[1] „*V programování se proces oddělení zodpovědností (Separation of Concerns - SoC) rozumí rozdělení počítačového programu na různé části tak, aby se z hlediska funkcionality tyto části co možná nejméně překrývaly. To znamená, aby určitou funkcionalitu vykonávala pouze část programu k tomu určená. Další část programu by pak neměla kopírovat v sobě stejnou funkcionalitu* [14].“

- Navrhnout vhodné řešení pro ovládání embedded periferií prostřednictvím řídicího systému.
- Realizovat prototyp komunikační jednotky, která bude přijímat data z řídicího systému a na základě těchto informací bude ovládat připojené periferie (či informace dále distribuovat).
- Vytvořit firmware pro komunikační jednotku a řídicí systém a demonstrovat funkci řešení na modelu minisumorobota.
- Změřit vlastnosti vytvořeného systému a zhodnotit dosažené výsledky.

1.2. Struktura práce

Aby bylo dosaženo cílů popsaných v minulé podkapitole, a kvůli budoucímu dalšímu užití, jsem práci rozdělil do několika kapitol. Ty se v zásadě snaží adresovat jednotlivé dílčí cíle a společně tak přispívají k naplnění hlavního cíle. Pro lepší přehlednost jednotlivé kapitolu dodržují určitou strukturu – na začátku každé kapitoly je představen obsah, který je případně doplněný o cíl kapitoly. Poté následuje samotná hlavní část kapitoly a na závěr jsou zpravidla zhodnoceny dosažené výsledky nebo následuje krátký souhrn.

Kapitoly práce by šli víceméně rozdělit na teoretické, jakými jsou například kapitoly Vymezení oblasti embedded zařízení specifikující oblast aplikace, na kterou navazuje kapitola Model komunikace s představením konceptu komunikace podle modelu OSI. Tyto kapitoly jsou doplněny částečně teoretickými kapitolami Komunikační rozhraní a Výběr protokolu pro komunikaci, kde vedle analýzy dostupných komunikačních rozhraní, respektive protokolů, jsou vybrány ještě kandidáti pro realizaci na universální komunikační desce. Teorii doplňují praktické kapitoly s postupem a popisem samotného řešení, případně s návodem k používání. Jsou to části Základní koncepce, Definice vlastního protokolu, Implementace řídícího softwaru a Realizace prototypu komunikační jednotky, ve které je navržena deska plošného spoje, na níž je nahrán řídící software implementující vlastní protokol.

2 Rešerše použité odborné literatury

Jelikož jsou zdroje a toky myšlenek uvedeny vždy v jednotlivých kapitolách, omezím se v této části pouze na spíše telegrafických výčet použité odborné literatury.

Ve 3 kapitole práce jsem převážně vycházel ze závěrečných prací věnující se projektu Minidarpa, který je detailně představen v další části. Závěrečné práce a různé oficiální stránky výrobců se staly dobrým základem i pro definici komunikačních rozhraní. Při definici konceptu řešení jsem se inspiroval projektem Minidarpa a různými návrhy komunikace popsanými v závěrečných pracích. Při vytváření řídícího software pro PC v jazyce JAVA jsem problematiku s odbornou literaturou konzultovat nemusel. O to více jsem ale musel studovat [10], jelikož psaní efektivního kódu v C pro mikrokontroléry bylo pro mě velkou novinkou. Samotné hardwarové řešení je z velké části založeno na referenčním řešení vývojového kitu [2].

3 Vymezení oblasti embedded zařízení

Tato kapitola je klíčová z hlediska této práce. Účelem je vymezit těžce uchopitelné oblasti embedded zařízení. Zaměřím se převážně na embedded zařízení vyskytující se v robotice a domovní automatizační technice, která s projekty na katedře přímo souvisí.

V následujícím textu vycházím především ze závěrečných prací zabývajících se projektem vývoje robota Minidarpa na VUT v Brně [1], [10], [19], [23], [23] a [26]. Problematika tohoto projektu co se hardwarové a komunikační části týče, bohatě pokrývá projekt na katedře. Na základě těchto prací vypracuji rozbor, jak spolu subsystémy (embedded moduly) komunikují, jak jsou řízeny a z čeho se skládají. Díky tomu je navíc možné si udělat představu o typických periferiích, použitých technologiích, výzvách a řešeních v této oblasti, což mi následně umožní vypracovat konceptu a výběr technologií pro technickou realizaci univerzální komunikační jednotky.

3.1 Definice pojmu embedded zařízení

Podle [5] lze embedded zařízení definovat jako jednoduchý počítač vytvořený za účelem vykonání konkrétního úkolu nebo několika menších souvisejících úkolů, což je podstatný rozdíl oproti například klasickému PC, které má řadu účelů. Embedded zařízení je tedy „*kombinace hardwaru a softwaru, která plní určitý úkol* [14].“ Úkol může plnit nezávisle na člověku nebo s jeho intervencí. Zařízení interaguje s okolním prostředím a zadaný úkol splňuje co nejefektivněji.

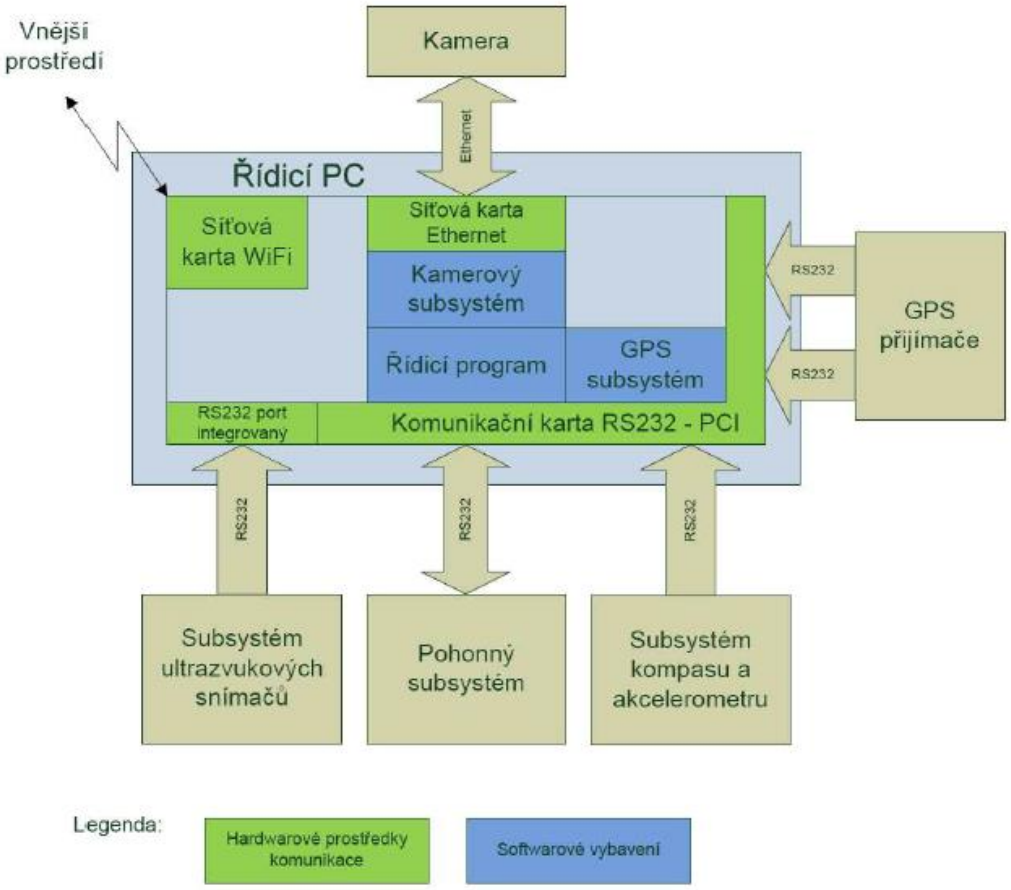
3.2 Projekt Minidarpa

Cílem projektu Minidarpa bylo vytvoření autonomního robota pro soutěž Robotour[2]. V této soutěži je hlavním úkolem účastníků vytvořit samostatného robota, který musí v zadaném časovém limitu projet určitou dráhu podle zadané mapy, přičemž nesmí sjíždět z cesty. Za účelem vytvoření takového robota vznikl na VUT v Brně v roce 2008 pětičlenný tým, jenž měl za úkol „*zpracovat robota po mechanické stránce, vytvořit systém pro ovládání podvozku, senzorický a navigační subsystém a hlavní řídicí systém* [1].“ V roce 2010 na tento projekt bylo navázáno staronovým týmem, který robota přepracoval.

[2] http://robotika.cz/competitions/robotour/cs

3.2.1 Systém robotu Minidarpa v roce 2008

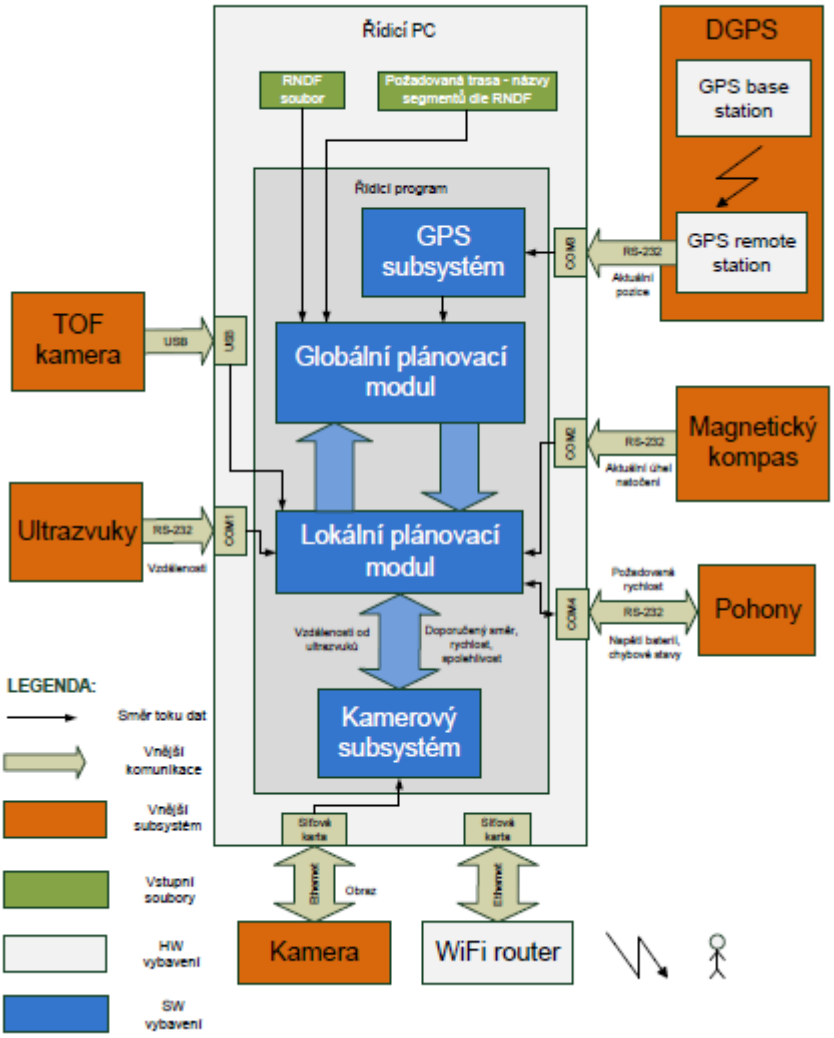
Blokové schéma systému Minidarpa z roku 2008 je na obrázku **1.** K řídící jednotce tvořené v tomto případě komponentami z běžného PC je připojeno celkem pět subsystémů: několik modulů senzorů pro vnímání okolí a polohy a subsystém pro řízení motorů. Řídící program implementuje dva režimy řízení – automatický režim, jenž na základě vyhodnocování údajů ze senzorů a zadané mapy ovládá pohyb robota, a manuální, kdy robot je řízen ze vzdáleného počítače. Vyjma kamery, která pro přenos informací používá Ethernet, jsou všechny moduly připojeny k řídící jednotce pomocí aspoň jedné sériové linky RS232. „*Navržený řídicí program se skládá z pěti současně zpracovávaných vláken, aby bylo zajištěno paralelní přijímání a vysílání na sériových portech a zároveň běh systémové části aplikace* [19].“ Informace si řídící jednotka s jednotlivými moduly předává prostřednictvím specifických zpráv lišícími se podle modulu. Sériová linka je nastavena na rychlost přenosu 9600 baudů, bez parity, s jedním stop-bitem a osmi datovými bity.



***Obrázek** 1:*
Blokové schéma systému Minidarpa z roku 2008 (Zdroj: [19])

3.2.2 Systém robotu Minidarpa v roce 2010

V navazujícím projektu z roku 2010 byl řídicí systém na obrázku **2** přepracován, aby reflektoval požadavky, které vyplynuly z účasti robota na několika soutěžích. Nové blokové schéma je zachyceno na obrázku . Jak je patrné z obrázku, hardwarová realizace zůstala téměř nedotčena, embedded zařízení zůstaly totožné a jsou stále většinou připojeny přes sériovou linku. Přesto bylo nutné, s ohledem na požadavky, kompletně přepracovat řídicí systém a implementovat rozhraní pro jednoduchou komunikaci s různými senzorickými subsystémy.



Obrázek 2:
Blokové schéma systému Minidarpa z roku 2010 (Zdroj: [23])

3.2.3 Nedostatky v komunikaci s jednotlivými embedded zařízeními

Po pečlivém prozkoumání popsaných systémů jsem našel hned několik úskalí těchto řešení. Kromě problémů, jako byla nutnost vytvořit jednotný interface pro komunikaci s různými senzorickými subsystémy a potřebě redefinovat komunikaci se zaměněnými moduly (GPS subsystém), které se snažila adresovat další iterace řídicího systému robota Minidarpa, jsem objevil několik dalších, které by mohla odstranit v této práci navržená univerzální komunikační jednotka. Ta by totiž zajistila nejenom jednotný interface pro komunikaci řídící jednotky s moduly senzorů, ale i se všemi embedded zařízeními jako jsou například pohony, zobrazovače. Jednotka by dále odstínila řídicí systém od realizace samotné komunikace, tak, že by vyčítání a zadávání údajů do modulů nemuselo běžet v několika samostatných vláknech. Řídicí systém by se tak soustředil pouze na zpracování a odeslání zprávy komunikační jednotce, která by už zajistila vše potřebné. V ideálním případě by nebylo ani třeba měnit kód řídícího programu při záměně modulu či rozšíření systému o další modul. Co je ovšem podstatné je fakt, že by bylo možné vytvářet řídicí systém a programovat jednotlivé moduly zcela samostatně a nezávisle na sobě, což podle prací kolem robota Minidarpa možné nebylo. Jako další možné problémy vidím absenci jakékoliv kontroly správnosti zpráv a počet připojitelných modulů omezený počtem portů na řídicím systému, což by rovněž do jisté míry mohla vyřešit mnou navržená jednotka.

3.3 Komplexita systémů embedded zařízení, topologie sítí jejich zapojení a způsob komunikace jednotlivých modulů

K obecnému popisu a kategorizaci takových systémů jako je například robot Minidarpa lze použít dělení systémů podle jejich komplexity popsané v online knize [4], která komplexitu systémů dělí na tři kategorie:

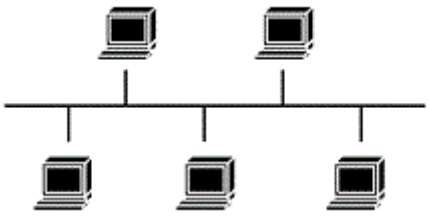
- Centralizovaný hardware a řízení, ve které jsou všechny informace soustředěně na jednom místě
- Distribuovaný hardware a centrální řízení, kde se systémy skládají z několika subsystémů, jenž má svojí vlastní řídící jednotku. Za každou řídící jednotku těchto subsystémů rozhodu jedna společná hlavní řídící jednotkou, se kterou komunikují
- Distribuovaný hardware a řízení, ve které je systém rovněž složený z několika subsystémů s vlastní řídící jednotkou. V tomto uspořádání neexistuje žádná centrální řídící jednotka, jelikož každý subsystém se rozhoduje sám

Typickým zástupcem první kategorie je například systém pro ovládání výtahu nebo automat na nápoje, do druhé kategorie spolehlivě zapadne robot Minidarpa a příkladem pro třetí kategorii by mohl být systém několika robotů na automobilové lince. Dělení do tří

kategorií ovšem není absolutní, jak ostatně upozorňuje i [4], lze nalézt i takové systémy, které mají znaky z více kategorií.

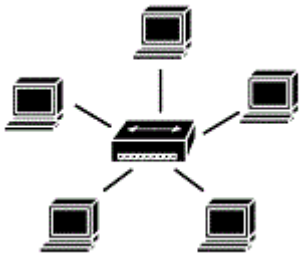
Důležité při zkoumání systémů embedded zařízení je způsob zapojení jednotlivých prvků, jímž se zabývá topologie sítí. „*Topologii lze zvažovat jako určitý tvar či strukturu dané sítě* [28].“ Při zkoumání topologie sítě je nutné rozlišovat fyzické a logické zapojení, které nemusí vždy být nutně stejné. „*Fyzická topologie popisuje reálnou konstrukci sítě, zapojená zařízení a jejich umístění včetně instalovaných kabelů. Logická topologie se vztahuje k tomu, jak jsou data v síti přenášena a kudy protékají z jednoho zařízení do druhého* [28].“ Podle [28] můžeme rozlišovat následující topologie sítí:

- Sběrnicová, která je znázorněna na obrázku **3**, kdy k jednomu společnému médiu (sběrnici) jsou připojena všechna koncová zařízení. Toto zapojení je velice jednoduché a levné, má však nevýhodu v podobě možných kolizí, kdy chce vysílat na jedné sdílené sběrnici více zařízení naráz.



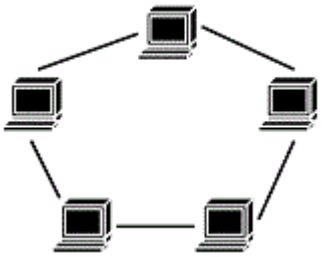
Obrázek** 3**:
Sběrnicová topologie (Zdroj: [28])

- Hvězdicová, ve kterém jsou zařízení propojeny do útvaru připomínající hvězdu. Každé zařízení je připojeno k centrálnímu prvku, které mu zprostředkovává komunikaci s dalším zařízením. Drobnou variací na hvězdu je stromové zapojení, které spojuje několik hvězd. Hvězdicová topologie je na obrázku **4**.



Obrázek** 4**:
Hvězdicová topologie (Zdroj: [28])

- V kruhové topologii vyobrazená na obrázku **5** jsou uzly vzájemně propojeny tak, že vytvoří kruh. Data z jednoho zařízení tak musí projít mezi několika zařízeními, aby se dostaly do cílové jednotky, což není tak efektní jako v případě hvězdy.



Obrázek** 5**:
Kruhová topologie (Zdroj: [28])

V případě robota Minidarpa se jak po logické, tak i fyzické stránce, pokud pod jednotlivými bloky chápeme moduly (senzory, kamera, pohon) jedná o hvězdicové zapojení, kde všechny prvky jsou zapojeny k řídící jednotce.

V souvislosti s předchozím textem je pro správné pochopení celého systému embedded zařízení a komunikace jednotlivých prvků nutné se podívat i na samotné role neboli vystupování jednotlivých prvků v komunikaci. Jedním z modelů komunikace je architektura master/slave, kde jedno zařízení – master má kontrolu nad jedním nebo několika zařízeními označenými jako slave. Zařízení typu master by se v počítačové terminologii dalo přirovnat ke klientovi, který ovládá server reprezentovaný zařízením typu slave. Dalším modelem komunikace je architektura multimaster (známé spíše pod pojmem peer-to-peer), ve které kterékoliv zařízení může začít komunikaci, a zařízení jsou postavena na stejné úrovni. Robot Minidarpa je zástupce první zmíněné architektury, řídící PC je v roli Mastera, který inicializuje komunikaci s jednotlivými moduly, které jsou na pozici slave, jimž zadává příkazy či z nich vyčítá informace.

3.4 Způsoby řízení embedded systémů typu Minidarpa

Způsoby řízení systémů obecně jsou velice rozsáhlou oblastí a hlavní náplní oboru kybernetika, proto tuto oblast nebudu podrobněji rozebírat a spokojím se spíše s konkrétními příklady. Řízení jsem částečně načal v minulé podklapitole. Nyní na něj navážu. Jedním příkladem způsobů řízení je například samotný průběh správy komunikace popsaný v závěru předchozí podkapitoly.

Minidarpa je typickým příkladem centrálně řízeného systému, kdy celý systém ovládá řídicí systémem. Tento model řízení by také šel označit za Master/Slave, kde prvku Master odpovídá řídicí systém a jednotlivé moduly tvořící robota jsou v režimu Slave. Dalším typem řízení je distribuované, kdy systém není řízen centrálně. Tento model řízení má oproti centralizovanému stylu řadu výhod jako rychlejší reakční dobu, ale i jednu velkou nevýhodu – jeho implementace je většinou složitější.

Minidarpa je řízeného pomocí událostí [6] (zpětná vazba v kybernetice). Velmi zajímavé je, že robot si události z periferií ve většině případů vyčítá, než aby byl přímo notifikován periferiemi v případě události, což jistě prodlužuje reakční dobu a vede k nízké utilizaci zdrojů v podobě několika běžících vláken programu, které by šly například využít k lepšímu výpočtu další trasy.

3.5 Typické periferie embedded zařízení

Jak se vidět na projektu Minidarpa periférií je v oblasti embedded skutečně mnoho, zařízení mohou zahrnovat různé senzory, zobrazovače a prvky, které přímo ovlivňují okolí zařízení. Nejsnadnější asi bude periferii chápat jako přídavné zařízení, které rozšiřuje funkčnost celku. V případě robota je to například GPS senzor, který pomáhá robotu lokalizovat jeho polohu, motory, které umožnují se robotu pohybovat v terénu, ale periferií je třeba i indikátor stavu baterie. V širším slova smyslu může být periferií i samotný modul robota – například deska na ovládání motorů. Jak je patrné z prací je komunikace s periferiemi věc velmi náročná a na hardwarové úrovni asi ani nikdy nebude jednotná, proto je vždy zásadní v ovládání softwarová vrstva, která na vyšší úrovni ucelené ovládání zaručit může.

3.6 Shrnutí

Účelem této kapitoly bylo poskytnout úvod do řešené problematiky, která je velmi těžko uchopitelná, a poskytnout tak náhled na možná řešení projektu na katedře na příkladu robota Minidarpa. V rámci této kapitoly byly v rámci možností definovány pojmy embedded zařízení a periferií z větší částí za pomoci konkrétního příkladu. Dále byly definovány oblasti řízení systémů embedded zařízení a jejich zapojení. V průběhu textu jsem identifikoval hned několik problémů při komunikaci a řízení embedded zařízení, které lze zobecnit nejen na roboty, ale i na další komplexní systémy. Část těchto problémů, tedy aspoň pro vymezenou oblast, adresuje moje bakalářská práce.

4 Model komunikace

Pokud je řeč o komunikaci, je velice užitečné se podívat na model ISO/OSI, který poskytuje aspoň základní teoretický rámec pro její realizaci podle principu „separation of concerns“. I z pohledu této práce je nutné se zmínit o tomto modelu, jelikož se během řešení problematiky pohybuji na různých vrstvách. V kapitole tedy stručně popíši jednotlivé vrstvy toho modelu, abych se na ně mohl později odkazovat.

4.1 ISO/OSI model

Open Systém Interconnection model (OSI model) je v [15] charakterizován jako koncepce charakterizující vnitřní funkce komunikačního systému, tak aby byla možná komunikace mezi rozdílnými systémy bez nutnosti měnit logiky použitého hardwaru nebo softwaru. Nejedná se tedy o konkrétní implementaci nějakého komunikačního protokolu, ale o model návrhu síťové architektury, která bude flexibilní, robustní a přenositelná.

Layer		OSI protocols	TCP/IP protocols
No.	Name		
7	Application	FTAM · X.400 · X.500 · DAP · ROSE · RTSE · ACSE[9] · CMIP[10]	NNTP · SIP · SSI · DNS · FTP · Gopher · HTTP · NFS · NTP · DHCP · SMPP · SMTP · SNMP · Telnet · BGP · FCIP
6	Presentation	ISO/IEC 8823 · X.226 · ISO/IEC 9576-1 · X.236	MIME · SSL · TLS · XDR
5	Session	ISO/IEC 8327 · X.225 · ISO/IEC 9548-1 · X.235	Sockets (session establishment in TCP / RTP / PPTP)
4	Transport	ISO/IEC 8073 · TP0 · TP1 · TP2 · TP3 · TP4 (X.224) · ISO/IEC 8602 · X.234	TCP · UDP · SCTP · DCCP
3	Network	ISO/IEC 8208 · X.25 (PLP) · ISO/IEC 8878 · X.223 · ISO/IEC 8473-1 · CLNP X.233	IP · IPsec · ICMP · IGMP · OSPF · RIP
2	Data link	ISO/IEC 7666 · X.25 (LAPB) · Token Bus · X.222 · ISO/IEC 8802-2 · LLC (type 1 / 2)[11]	PPP · SBTV · SLIP
1	Physical	X.25 (X.21bis · EIA/TIA-232 · EIA/TIA-449 · EIA-530 · G.703)[11]	

***Obrázek** 6:*
Sedm vrstev modelu OSI s příklady protokolů v kontrastu s TCP/IP (Zdroj: [15])

OSI model je rozdělen do 7 hierarchicky uspořádaných logických vrstev, kde každá vrstva obsluhuje vyšší vrstvu a je zároveň obsluhována vrstvou o úroveň níže. V případě

vzájemně komunikujících systémů navržených podle tohoto modelu spolu komunikují vždy vrstvy na stejné hierarchii. 7 vrstev konceptu je znázorněno na **6**. Jednotlivé vrstvy zajišťují následující funkce:

1. **Fyzická vrstva** je nejnižší vrstvou, která fyzicky propojuje dvě zařízení a přenáší jednotlivé bity mezi komunikujícími zařízeními. Příkladem realizace jsou např. technologie Ethernet nebo WiFi. Technologie realizují různé topologie popsané v předchozí části.
2. **Linková vrstva** seskupuje bity přenášené předchozí vrstvou do větších bloků tzv. rámců a vytváří tak spojení mezi dvěma přímo propojenými body. Detekuje a následně opravuje chyby způsobené fyzickým přenosem na předchozí vrstvě. Tato vrstva kontroluje přístup zařízení k datům a řídí přenos – tok dat. Příkladem realizace je například Point-to-Point protokol u TCP/IP protokolu.
3. **Síťová vrstva** má za úkol zajistit přenos datových sekvencí (datagramů nebo také paketů) mezi dvěma komunikačními uzly, mezi kterými neexistuje přímá komunikační cesta. Pomocí různých routovacích protokolů a znalosti topologie sítě se snaží najít sub optimální cestu, která vede přes ostatní uzly.
4. **Transportní vrstva** ve své podstatě staví na funkcích předchozí vrstvy. Tato vrstva zabezpečuje skutečné doručení paketů, které mohou putovat různými cestami, a dále opravuje chyby způsobené takovýmto přenosem. Zjednodušeně řečeno zajišťuje kvalitu spojení další kontrolou toku dat – udržuje si přehled o přijatých paketech, posílá potvrzení přijatých paketů či chybovou zprávu. Za příklad implementace je označován Transmission Control Protocol.
5. **Relační vrstva** vytváří a ukončuje relace (nebo také sezení). Relace je podobná telefonnímu hovoru, dialogu mezi dvěma stranami, který je zahájen vytočením telefonního čísla, příjmem hovoru, vlastním telefonátem a ukončením hovoru. Na stejném principu funguje právě tato vrstva.
6. **Prezentační vrstva** zajišťuje transformaci přijatých dat ze síťového podoby, do které je předtím převedla u odesílatele, do původní podoby interpretovatelné příjemcem.
7. **Aplikační vrstva** je vrstvou nejblíže uživateli. Tato vrstva interaguje se softwarovými aplikacemi, které implementují komunikační komponenty. Příkladem jsou například HTTP, FTP a SMTP, jejichž služby pro vzájemnou komunikaci následně používají softwarové aplikace.

4.2 Shrnutí

OSI model se stal předlohou pro různé komunikační protokoly. Asi nejznámější implementací, která vychází z tohoto modelu, je rodina protokolů TCP/IP, jenž je hlavní prostředkem pro komunikaci v síti Internet. Při plné implementaci modelu OSI a dodržení

doporučení by řízení embedded zařízení mělo odpovídat softwarové aplikaci na všech jednotkách využívající služeb aplikační vrstvy OSI. O univerzální aplikaci koncepce nebo TCP/IP na řešenou problematiku mám vážné pochybnosti vzhledem k poměrně velké náročnosti na koncová zařízení. Přesto základní myšlenku „separation of concerns“ modelu OSI považuji za velmi nadčasovou a koncept komunikace a definované funkce jsou velmi relevantní i pro tuto práci, proto z modelu vycházím.

5 Základní koncepce

Cílem této části je definovat obecnou koncepci pro následnou fyzickou realizaci univerzální komunikační jednotky pro oblast embedded zařízení, která bude umožňovat řídicímu systému řídit připojené periferie. Výsledná koncepce by měla být snadno přenositelná, což platí do jisté míry o samotné softwarové a hardwarové realizaci, jelikož je zadání dost obecné a po rozboru oblasti embedded zařízení v předminulé kapitole si pod pojmem řízení lze představit i upozornění řídicího systému na různé události a různé možnosti způsobů samotného řízení jako je například distribuované.

V této kapitole nejdříve popíši cílový stav. Na jeho základě znázorním jednotlivé role v systému a jejich funkce pomocí USE-CASE diagramu společně se zjednodušeným modelem systému a definicí jeho jednotlivých částí. Následně je vypracován model procesu samotné komunikace. Nakonec se zaměřím na samotnou komunikační jednotku, kdy stanovím kritéria na komunikační rozhraní pro projekt na katedře.

5.1 Cílový stav

Jak už bylo řečeno, navržené řešení bude použito k řízení bezdrátového modelu autíčka složeného z několika modulů. Bude se tedy jednat o podobné zařízení jako Minidarpa s tím rozdílem, že mezi řídicí systém a embedded zařízení bude vložen ještě prostředník - navrhovaná univerzální komunikační jednotka. K naplnění cíle v obecné rovině by výsledné řešení mělo mít následující parametry:

- Řešení by mělo umožnit vzájemnou komunikaci několika zařízení takovým způsobem, že jakékoliv zařízení bude moci komunikovat s jiným zařízením, což umožní řídit celou soustavu pomocí událostí a přímo využívat ostatních subsystémů bez nutnosti souhlasu řídícího centra (pokud bude tak systém zamýšlen).
- Do vzniklého systému by měla být možnost připojit subsystém, který bude vystupovat jako nadřazené zařízení (v tomto případě master), které bude v systému pouze jedno a bude jednoznačně identifikovatelné. Čímž se usnadní celkové řízení systému, jelikož bude jasně definovaný jeden řídící bod.
- Výsledné řešení by mělo být velmi jednoduché na implementaci ve stylu „separation of concerns“, kdy implementátor nové periferie nebo řídící jednotky bude co nejvíce odstíněn od implementace samotné komunikace. Tento parametr zajistí, že i někteří implementátoři, kteří se zaměřují především na hardware, a software není jejich silnou stránkou, budou moci používat sofistikované řešení pro komunikaci s minimálním úsilím.

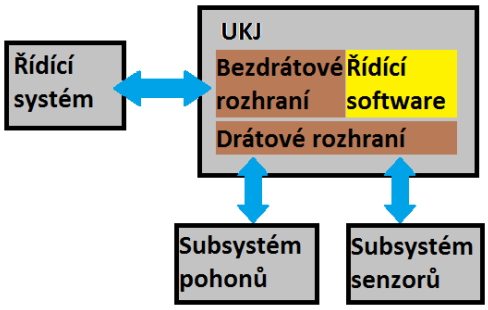
- Řešení by mělo být univerzální, aplikovatelné nejen na určitý druh projektů, a přenositelné.
- Řídící software stojí až na poslední vrstvě. Tak bude zabezpečeno, že programátor bude co nejvíce oddělen od implementace samotné hardwarové části.

S ohledem na aplikaci, oblast a konkrétní zadání by dále řešení mělo splňovat následující požadavky:

- Komunikace mezi zařízeními by měla probíhat pomocí volání funkcí se zadaným parametry, stejně jako je tomu například v programovacím jazyce C. Toto řešení umožní přesunout program periferie do ohraničeného bloku programu (například funkce v programu C) a tento blok jednoduše vykonat zavoláním jeho identifikátoru na zařízení z jiného zařízení. Tento požadavek koresponduje s běžným způsobem návrhu embedded zařízení – vytvoření za účelem realizace nějaké funkce.
- Řídicí systém by měl se zbytkem systému (s periferiemi) komunikovat nějakým běžně podporovaným bezdrátovým rozhraním.
- Subsystémy – jednotlivé periferie, budou mezi sebou komunikovat přes drátové rozhraní.
- Univerzální komunikační jednotka bude prostředníkem takovéto komunikace. Bude routerovat požadavky mezi zařízeními a vytvářet fyzické spojení mezi zařízeními na různých komunikačních rozhraních.
- Výsledné řešení bude nenáročné na výpočetní zdroje, jelikož výkon u mikroprocesorů v oblasti embedded není hlavní prioritou, proto jsou tyto procesory výpočetně velmi slabé.

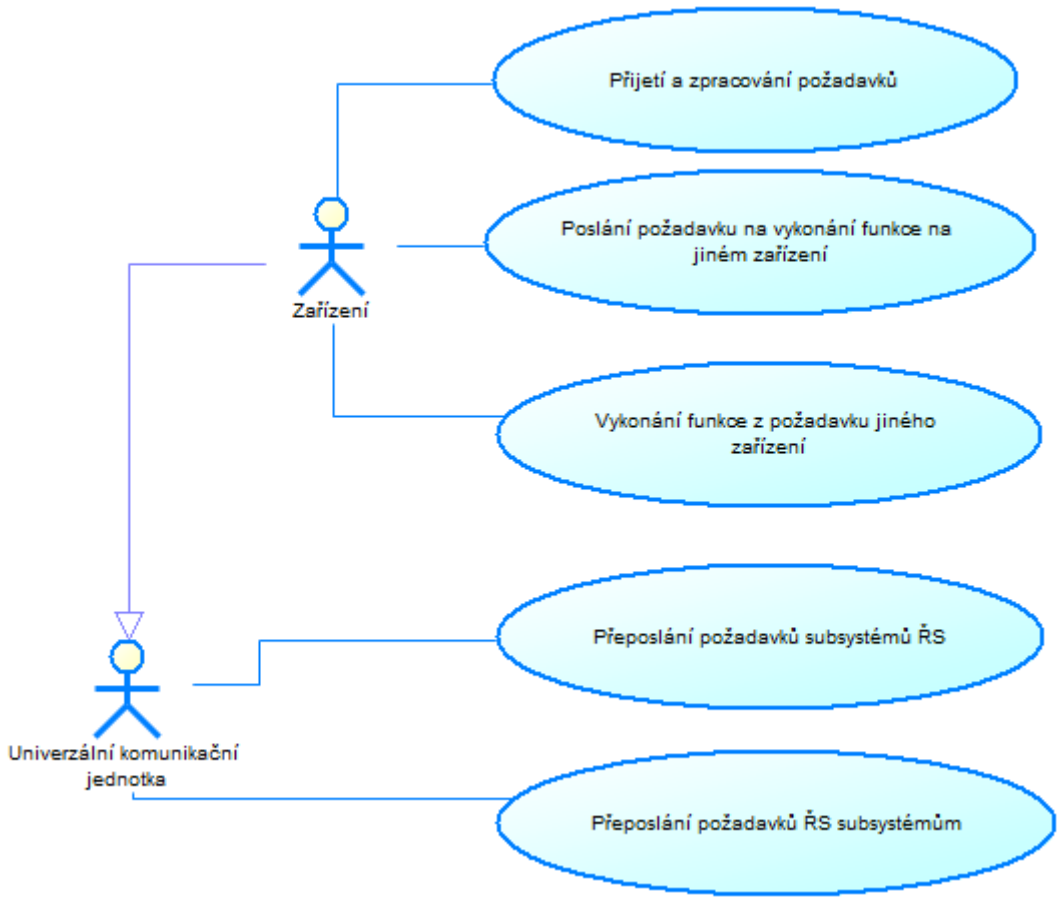
5.2 Navržené řešení

Na základě těchto požadavků jsem vypracoval schéma zapojení systému zobrazené na obrázku **7**. Tento obrázek představuje zjednodušené zapojení podobné tomu, které je na obrázku **2** u robota Minidarpa, s tím rozdílem, že se na schématu nachází ještě prostředník – v podobě univerzální komunikační jednotky. Ta zajišťuje spojení mezi různými komunikačními rozhraními periferií a řídicího systému. Funkce jednotlivých zařízení podporující tento způsob komunikace jsou znázorněny USE-CASE diagramem na obrázku **8** pod rolí „zařízení“. Univerzální komunikační jednotka rozšiřuje tuto roli o další dvě funkce. Samotná realizace komunikace je namodelována na obrázku **9.**



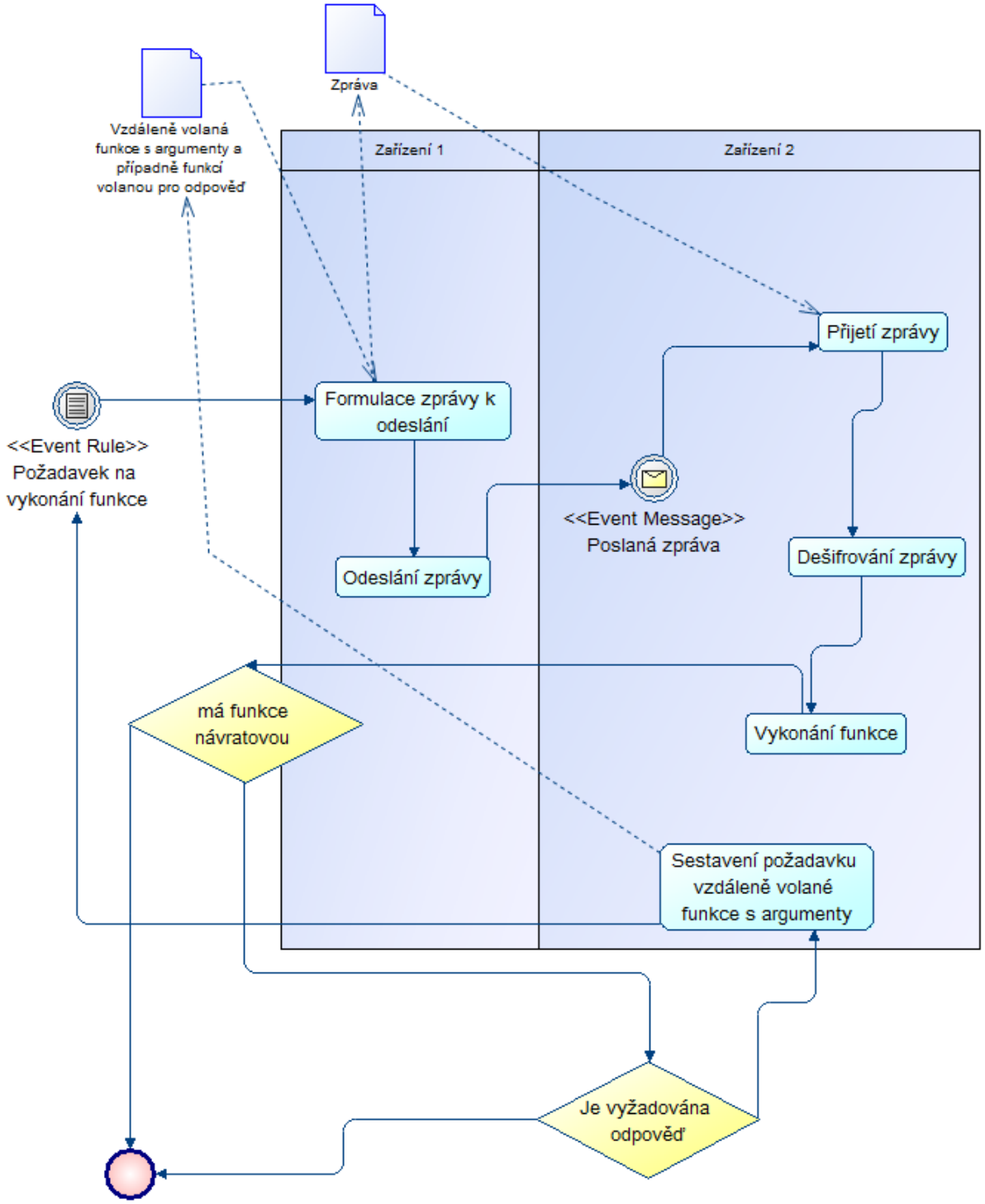
Obrázek 7:
Blokové schéma zapojení (Zdroj: autor)

V rámci softwarové části budu implementovat nějaký jednoduchý protokol. V ideálním případě bude stačit jeden na všechny subsystémy. Na protokolu pak následně vystavím vlastní aplikaci, která bude vykonávat program zobrazený na obrázku **9**. Pokud zvolím vhodný protokol vycházející z konceptu OSI, bude stačit jen aplikační vrstvě protokolu předat zprávu (funkci s argumenty, která se má vykonat) a zbylé vrstvy se postarají o samotnou komunikaci a doručení zprávy na druhé zařízení, kde následně bude vykonána. To znamená naimplementovat pouze 3 aktivity z obrázku **9** – „formulace zprávy k odeslání“ a „vykonání funkce“ a „sestavení požadavku vzdáleně volané funkce s argumenty“. Zbytek by měl řešit protokol.



***Obrázek** 8:*
Diagram rolí s funkcemi znázornění pomocí USE-CASE (Zdroj: autor)

Co se týče hardwarové realizace prostředníka, budu ho řešit podobným způsobem, jako jsou realizovány periferie – pomocí jednoduchého mikrokontroléru z rodiny PIC18, který je založený na 8-bitové architektuře. Jednotlivé periferie budou muset implementovat program s podobnými funkcemi jako navrhovaný prostředník, proto budu moct pro všechna zařízení použít podobný základ programu. Usnadní se mi tak samotná implementace vybraného protokolu a aplikační části, kterou bude možné napsat v jazyce C. Zápis v jazyce C splní požadavky na přenositelnost, jelikož takto napsaný program lze většinou zkompilovat do strojových instrukcí pro většinu zařízení. Navíc jsem se těmito procesory setkával v některých předmětech bakalářského studia.



***Obrázek** 9:*

Proces vzdáleného volání funkce vytvořen podle notace BPMN (Zdroj: autor)

5.2.1 Kritéria výběru protokolu

Jak už jsem v minulé podkapitole naznačil, preferuji jeden protokol v rámci všech zařízení v systému, ale není to nutnost. Vybraný protokol by mně měl práci ušetřit a odstínit mě v nejlepším případě od samotné komunikace přičemž by vlastní aplikace interagovala s ekvivalentem k aplikační vrstvě u vybraného protokolu. Co se týče realizace hardwarové části, měl by mě vybraný protokol maximálně ulehčit implementaci vybraných rozhraní. Softwarová část implementace protokolu by měla být velice nenáročná tak, aby embedded zařízení mohlo kromě komunikace vykonávat ještě další činnost. Implementace by tedy embedded zařízení neměla nějak omezovat a přehnaně zatěžovat. Přidávání nějakých externích modulů k periferiím k realizaci komunikace beru jako poslední a nejextrémnější možnost, kterou bych nerad praktikoval. Znamenalo by to předělání stávajících subsystémů po hardwarové stránce, prodražení těch nových a další nároky na energii kvůli přídavným obvodům.

5.2.2 Kritéria na komunikační rozhraní

Stejně jako u protokolu bych se rád při výběru komunikačních rozhraní co nejvíce omezil na rozhraní, které už zařízení sami podporují. Mikroprocesory již řadu komunikačních rozhraní podporují, proto se pro vzájemnou komunikaci zaměřím na ně a pokusím se co nejvíce vyhnout přídavným obvodům. Jediným požadavkem je opět schopnost zpracovávat data z rozhraní takovým způsobem, které příliš nezatěžuje embedded zařízení. U bezdrátového rozhraní se přídavnému modulu podporující nějakou technologii bezdrátového přenosu nevyhnu. Volit budu technologii, kterou podporuje i řídící jednotka (v tomto případě PC).

5.3 Shrnutí

V této kapitole byl definován koncept celého řešení, ze kterého následně čerpám dále při samotné realizaci řešení. Na tuto kapitolu přímo navazují zbylé kapitoly. Na základě kritérií na komunikační rozhraní jsou po analýze dostupných technologií v další kapitole vybrány dvě konkrétní technologie. Ve dvou následovných kapitolách jsou zmapovány dostupné protokoly a je realizována implementace. Poté je v rámci jedné kapitoly vytvořen software pro všechna zařízení a dále je realizovaná hardwarová část s podporou komunikačních rozhraní a dostatečným výkonem pro běh softwarové části.

6 Komunikační rozhraní

Cílem této kapitoly je vybrat vhodné komunikační rozhraní, která budou zajišťovat jak propojení komunikační jednotky s několika embedded zařízeními, tak komunikaci této jednotky s řídicím systémem s ohledem na zamýšlenou aplikaci. Za tímto účelem je v kapitole popsáno hned několik průmyslových standardů komunikačních drátových a bezdrátových rozhraní. Výběr řešení je velmi důležitý, volba totiž ovlivní samotnou fyzickou realizaci zařízení, zapojení ostatních systémů a implementaci a výběr protokolu. S ohledem na předchozí kapitolu, kde byla popsáno koncepce, budou vybrána dvě rozhraní – drátové - pro propojení ostatních zařízení s komunikační jednotkou, a bezdrátové pro komunikaci jednotky s řídicím systémem. Obsah této kapitoly vychází hlavně ze závěrečných prací [9] a [18].

6.1 Bezdrátová komunikační rozhraní

6.1.1 WiFi

Wifi je obchodní značka pro standard IEEE 802.11x, což je „*populární technologie bezdrátové síťové, která používá rádiové vlny k zajištění rychlého bezdrátového sítového připojení* [29].“ Tuto technologii netřeba dále představovat proto jen uvedu dělení standardů podle vysílacích frekvencí, které se od sebe mohou odlišovat například rychlostí, dosahem a dalšími vlastnostmi:

- 2.5 GHz: 802.11b, 802.11g a 802.11n
- 5 GHz: 802.11a

6.1.2 Bluetooth

Bluetooth je bezdrátové komunikační rozhraní umožňující připojení dvou a více zařízení. Technologie umožňuje vytvořit síť až 8 zařízení tzv. piconet typu Master-Slave. Nejčastěji se však technologie používá k vytvoření přímého dvoubodového spoje (ad-hoc síť). Bluetooth se v posledních letech stalo prakticky standardem v oblasti připojování bezdrátového příslušenství k mobilním telefonům, kde typickým zástupcem příslušenství jsou náhlavní soupravy. Tato technologie se vyskytuje prakticky u všech novějších telefonů a také stále častěji u přenosných PC. Bluetooth je poměrně úsporná a rychlá varianta komunikace, viz obrázky a , proto je možné ji uplatnit ke komunikaci i v oblasti embedded zařízení. Při nasazení v této oblasti je nutné použít specializované obvody (moduly), u kterých je většinou již integrovaná anténa.

Bluetooth je definováno standardem IEEE 802.15.1., a patří mezi osobní sítě typu PAN. K identifikaci zařízení slouží adresa *BT_ADDR*, kterou má každé zařízení. Pracuje stejně jako Wi-Fi v pásmu 2,4 GHz. Přenos dat probíhá pomocí metody FHSS (Frequency-hopping spread spectrum), kdy je provedeno 1600 skoků mezi 79 frekvencemi za sekundu, čímž se zlepšuje odolnost proti rušení. I pro tuto technologii existuje několik verzí lišící se převážně v přenosové rychlosti, dosahu a podporovaném šifrování. Zařízení se dělí podle výkonu a dosahu následovně (platí pro volný prostor):

- Class 1 – až 100 metrů (100 mW)
- Class 2 – až 10 metrů (2,5 mW)
- Class 3 – až 1 metr (1 mW)

Rozdělení podle jednotlivých verzí:

- Verze 1.2 1 Mbit/s
- Verze 2.0 + EDR 3 Mbit/s
- Verze 3.0 + HS 24 Mbit/s
- Verze 4.0 24 Mbit/s

V současnosti za nejpoužívanější verzi lze považovat verzi 2.0.

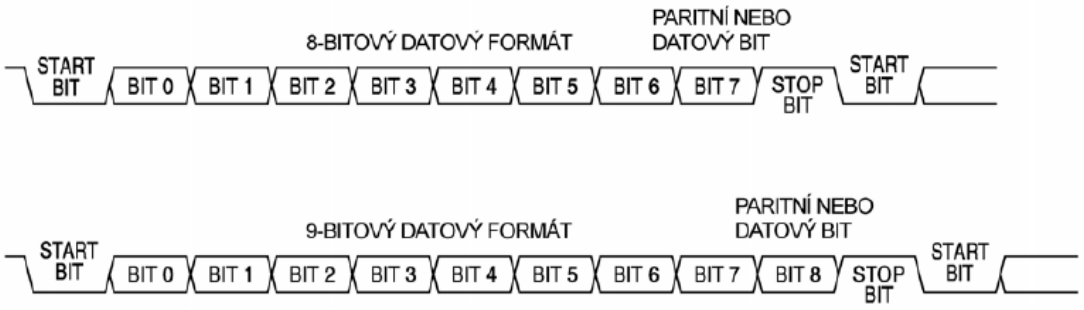
6.2 Drátová komunikační rozhraní

6.2.1 RS232

RS232 je zástupcem sériového komunikačního rozhraní umožňujícího komunikaci dvou zařízení s podporou plně duplexního módu. Přestože toto rozhraní bylo již téměř vytlačeno rozhraním RS485 či USB, stále zůstává průmyslovým standardem, proto je i modul realizující toto rozhraní běžnou součástí mikroprocesorů.

Standard tohoto rozhraní pro přenosovou rychlost 19200 Bd[3] definuje maximální délku propojení až 15 metrů nebo délku vodiče o kapacitě 2500 pF. Spoj může být v plném zapojení realizován 25 vodiči. Běžnější varianta zapojení je s 9 vodiči, přesto se většinou realizuje zapojení pouze s 3 vodiči: RxD (Receive Data), TxD(Transmit Data) a GND (System Ground).

[3] Rychlost v Bd můžeme podle [27] v tomto případě považovat za rychlost přibližně bit/s.



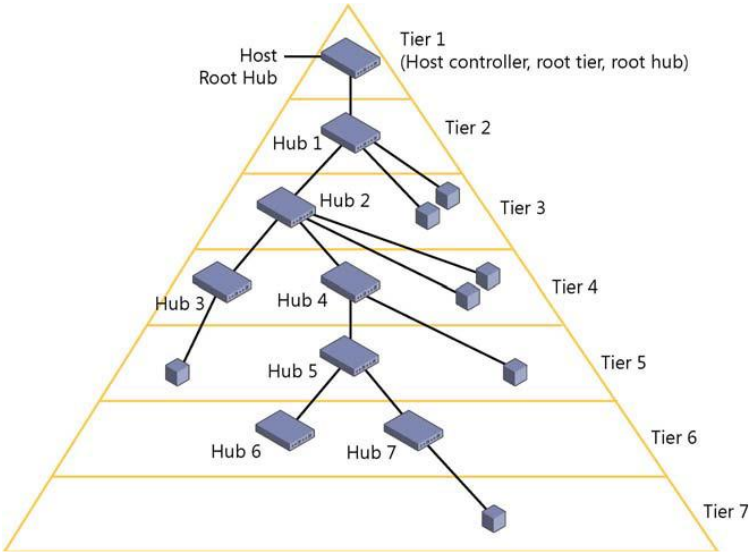
Obrázek** 10**:
Ukázka přenosového rámce (Zdroj: [18])

Datový přenos je realizován pomocí tzv. datových rámců, viz obr. **10**, v rámci něho je přenášeno většinou 8 bitů. Přenos dat je realizován buď synchronně, kdy mimo datového vodiče je na sběrnici přítomný ještě jeden vodič přenášející signál, nebo asynchronně, kdy jsou data přenášena v určitých sekvencích přesně danou rychlostí, na kterou se synchronizují všechna přijímací zařízení.

Správnost přenášených dat je zabezpečena paritou tak, že „ve vysílacím zařízení se sečte počet jedničkových bitů a doplní se paritním bitem tak, aby byla zachována předem dohodnutá podmínka sudého nebo lichého počtu jedničkových bitů [8].“ Datový rámec je ukončen Stop bitem, který zajišťuje časovou proměnu mezi jednotlivými rámci.

6.2.2 USB

USB (Universal Serial Bus) je jeden z nejrozšířenějších standardů pro sériový přenos dat z periferií typu Slave do nadřazeného systému typu Master. Sběrnice je typu singlemaster, na sběrnici je tedy zapojeno jedno zařízení typu Master, které řídí komunikaci s ostatními zařízeními typu Slave. „*USB sběrnice používá víceúrovňovou hvězdicovou topologii. Hub je nejvyšším bodem hvězdicového propojení. Každé zařízení má svoji adresu a může komunikovat s jedním nebo více koncovými uzly (nodes) v několika úrovních (tier)* [18].“ Hvězdicová topologie sběrnice USB je znázorněna na obr. **11.**



***Obrázek** 11:*
Ukázka topologie USB (Zdroj: [18])

Data jsou mezi zařízeními přenášena ve formě paketů o děle 8 až 64 Bytů. Pakety jsou přenášeny v rámcích, přičemž v jednom rámci může být obsaženy pakety pro více zařízení. Přenosová rychlost USB verze 1.1 je u rychlých zařízení (Full-Speed) až 12 Mb/s u pomalejších (Low-Speed) až 1,5 MB/s. Verze USB 2.0 může dosahovat přenosové rychlosti až 480 Mb/s a u nejnovější verze USB 3.0 je maximální přenosová rychlost až 5 Gb/s. Standard podle [18] definuje maximální délku kabelu mezi dvěma zařízeními až 5 metrů, přičemž kabel u prvních dvou verzí je realizován 4 vodiči. Přestože USB 3.0 používá vodičů 8, je zpětně kompatibilní s verzí 2.0.

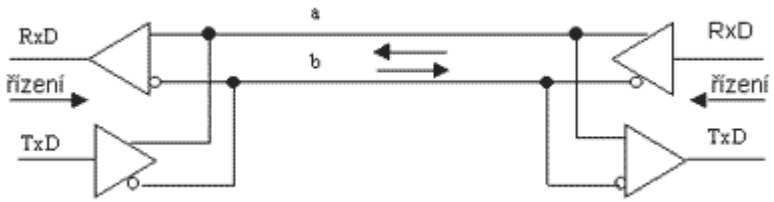
Vzhledem k velké rozšířenosti standardu, můžeme se s podporou USB setkat i u nejjednodušších mikroprocesorů, případnou podporu lze navíc zajistit i formou specializovaného obvodu – převodníku UART – USB, který umožňuje mikroprocesoru vystupovat minimálně v režimu Slave. Navíc je nutno podotknout, že sběrnice může přímo napájet připojená zařízení napětím 5 V a dodávat zařízení proud 100 mA (krátkodobě až 500 mA).

6.2.3 RS485

Sériová sběrnice RS485 patří v současnosti mezi nejpoužívanější komunikační rozhraní pro propojení několika zařízení. Pomocí této sběrnice může komunikovat 32 vysílačů a 32 přijímačů, což je velký posun oproti sběrnici RS232, která umožňuje připojení pouze 2 zařízení. Systémy RS485 používají architekru Master/Slave s adresací Slave zařízení, které nezačínají komunikaci, nebo architekturu Multimaster, kde každé zařízení může odesílat data. Sběrnice nepoužívá žádné řídící vodiče.

Funkčnost sběrnice je podle [25] následující: „*Všechny přijímače i neaktivní vysílače se v klidu musí nacházet ve stavu vysoké impedance, tj. nijak neovlivňují komunikující*

zařízení. Pouze jedno zařízení na sběrnici může v daném čase pracovat jako vysílač.“ Specifika sběrnice RS485 už samotnou arbitraci – domluvu zařízení, už nedefinuje, ta je ponechána na protokolu vyšší vrstvy. Standard RS485 operující v poloduplexním módu definuje pouze zapojení a technické požadavky ale ne způsob, jakým způsobem budou jednotlivá zařízení komunikovat. Běžně se pro komunikaci používá stejný asynchronní sériový protokol známý z rozhraní RS-232C.



Obrázek** 12**:
Schéma zapojení linky RS485 (Zdroj: [8])

Velká přenosová rychlost až 10 MBit/s na poměrně velkou vzdálenost 1200 metrů jsou zaručeny díky použití kroucené dvojlinky a diferenciálního kódování. Diferenciální kódování je zde realizováno stejně jako u sběrnice RS232, kdy jedna polaritu představuje jednu logickou úroveň (logickou 1) a k ní opačná polarita druhou logickou úroveň (logická 0). S použitím kroucené dvoulinky už není potřeba žádná napěťová reference a společná zem, jelikož rozdílná napětí přijímače a vysílače nezpůsobují problémy. Přijímač tak pro zjištění hodnoty bitu rozlišuje pouze rozdíl napěťových potenciálů mezi vodiči místo porovnávání absolutní hodnoty napětí. Velmi důležité je proto správné zapojení přenosové linky (terminace a přizpůsobení). Pokud by se linka správně nezapojila, „*mohl by se šum na „plovoucích“ linkách při odpojeném vysílači (tento stav není definován) projevit rozdílem větším než 0,2 Voltů, což je, jak již víme, požadovaná citlivost přijímače (přijímač či přijímače by tedy přečetly datový šum)* [25].“ Z pohledu fyzické realizace je ještě nutno podotknout, že zapojení je možno realizovat jedním (Single TwistedPair RS485) či dvěma kroucenými vodiči (Double TwistedPair RS485). V prvním případě po jednom páru probíhá komunikace v obou směrech, v druhém případě je na druhém vodiči realizován přenos dat Slave > Master. Realizace prvního zapojení je zachycena na obr. **12.** Podpora sběrnice RS485 ze strany zařízení je obdobná podpoře RS232.

6.3 Výběr vhodného komunikačního rozhraní

Na základě analýzy dostupných rozhraní jsem zvolil WiFi technologii pro realizaci bezdrátové komunikace, a sériovou sběrnici RS485, RS232 i USB. WiFi jsem zvolil proto, že jsem byl o to požádán, abych tím zároveň vyzkoušel WiFi moduly, kterými disponuje škola. Drátové sběrnice jsem zvolil 3, jelikož u prvních dvou je jejich implementace velice

podobná a celkem jednoduchá. Zařízení navíc doplním i o USB, abych mohl produkt snadněji testovat a ladit.

6.4 Shrnutí

Cílem této kapitoly bylo zmapovat možnosti realizace komunikačního rozhraní jak v oblasti bezdrátové, tak i drátové komunikace. Řadu technologií jsem vynechal, ale na výsledném výběru se to podepsalo jen minimálně. S vybranými sběrnicemi jako WiFi, RS485, RS232 a USB jsem pracoval již dříve, proto jejich volbu preferuji.

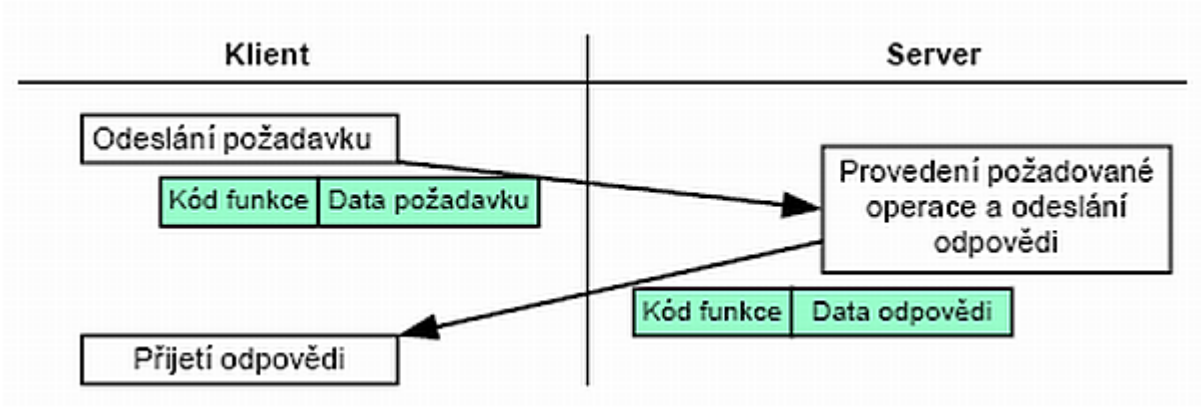
7 Výběr protokolu pro komunikaci

Cílem této kapitoly mělo být původně výběr nějakého na trhu dostupného protokolu splňujícího kritéria zadaná v kapitole s konceptem. Bohužel žádné nalezené dostupné řešení stanovené požadavky nesplňovalo, proto jsem se nakonec uchýlil k vytvoření vlastního protokolu popsaného v navazující kapitole.

7.1 Dostupná řešení

7.1.1 Modbus

Komunikační protokol MODBUS se pohybuje na úrovni aplikační vrstvy OSI modelu, který umožňuje komunikaci klient server na různých typech sítí a sběrnic. Podporovány jsou například sériové linky typu RS-232, RS-422 a RS-485 a síť Ethernet s využitím protokolu TCP/IP. Komunikace probíhá metodou požadavek-odpověď zobrazená na obrázku **13.** V rámci požadavku je volána funkce specifikována pomocí kódu funkce. Funkce většinou představují zápis/čtení nějakého paměťového místa na cílovém zařízení, kde jsou data uložena jedením z řady datových typů, které protokol definuje. Možné je například definovat si i vlastní funkce.



Obrázek** 13**:
MODBUS transakce s bezchybným provedením požadavku (Zdroj: [17])

Zařízení jsou v protokolu identifikovány pomocí adres (od 1 do 247), jelikož požadavek může většinou číst více zařízení, ale vykoná ho pouze adresát. V závislosti na implementaci nejen klient může inicializovat komunikaci. Jak už bylo řečeno Modbus je až na aplikační vrstvě OSI modelu, proto je nutné implementovat další protokol jednající na nižších vrstvách jako je například TCP/IP. Implementací lze na internetu nalézt několik,

dokonce i pro procesory z rodiny PIC. Ve většině případů je určená podstatně výkonnějším zástupcům z rodiny procesorů PIC, než PIC18.

7.1.2 Middleware pro robotiku

Middlewary pro robotiku, například popsané v [21], splňují principy navrženého konceptu. Problém těchto řešení je, že mají o několik řádů vyšší požadavky na výkon, než nabízí procesory PIC18. Tato řešení se pohybuje nad aplikační vrstvou OSI.

7.1.3 Definice vlastního protokolu

Dalším možným řešení problému výběru protokolu je návrh zcela vlastního, ušitého na míru aplikaci, který by přesně splnil všechna kritéria.

7.2 Důvody vytvoření vlastního protokolu

Po zralé úvaze a zhodnocení všech alternativ jsem se nakonec rozhodl pro řešení vytvoření vlastního protokolu. Za druhou nejlepší alternativu jsem považoval implementaci Modbus, ale jak se po prozkoumání hotových řešení ukázalo, znamenalo by to psát skoro celý vlastní software pro podporu tohoto protokolu, který nepovažuji zrovna za přímočarý. Jistě požadovaná funkcionalita řešení by použitím Modbus mohla být dosažena – na vzdáleném zařízení by se zavolalo několik funkcí na zápis dat do proměnných a následně by se zavolala uživatelská funkce, která by hodnoty z proměnných vyčetla a uživatelskou funkci s nimi vykonala. Tato řešení mi přijde jako velmi špatně čitelné pro koncového uživatele a znamenalo by použití ne moc dobrých praktik návrhu programu.

7.3 Shrnutí

Tato kapitola se zabývala analýzou a následným výběrem protokolu. Moc dostupných variant řešení jsem neobjevil. Jelikož jsem zanedbával různá jednorázová řešení pro konkrétní aplikaci, zůstali mi nakonec 3 možné varianty. Přičemž jsem se nakonec rozhodl dát přednost možnosti vlastní definice protokolu, které popíši v další kapitole.

8 Definice vlastního protokolu

Jak bylo napsáno v předchozí kapitole – účelem této části je definovat vlastní protokol, který by mi umožnil naplnit cíl. Díky analýze několika protokolů, znalosti modelu OSI a definovaných kritérií, definuji manifest protokolu, na jehož základě protokol definuji ve zbytku této kapitoly. Pro protokol v následující kapitole představím program, který ho implementuje.

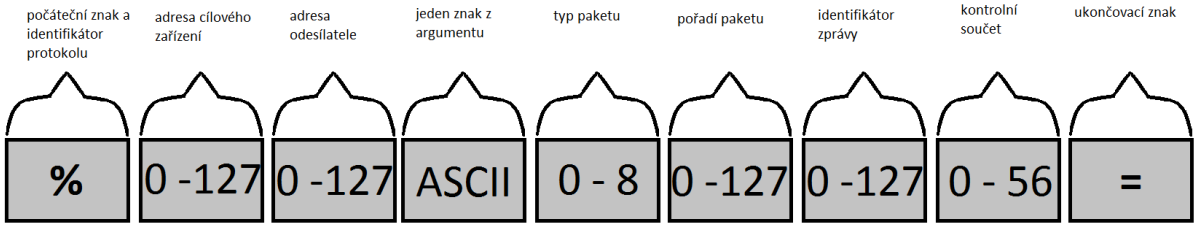
8.1 Manifest protokolu

Cílem protokolu je být co nejjednodušší, přesto by měl protokol naplnit všechny požadavky z konceptu. Navržené řešení se bude pohybovat na aplikační vrstvě modelu OSI, čímž se zajistí požadovaná přenositelnost a kompatibilita s různými rozhraními. Navržený protokol umožnuje:

- V přijatých datech detekovat požadavek ze znalosti formátu požadavku, začátku a konce
- Definovat sémantiku přijatých dat požadavku pro člověka čitelným způsobem
- Ověřovat správnost přijatých dat v požadavku
- Popisovat chování běžných pro zajištění plynulosti komunikace – příjem a potvrzení požadavku, nakládání s požadavky v případě chyby, způsob sestavení požadavku z dat, odesílání požadavku
- Navržený protokol může přijímat více požadavků od jednoho zařízení

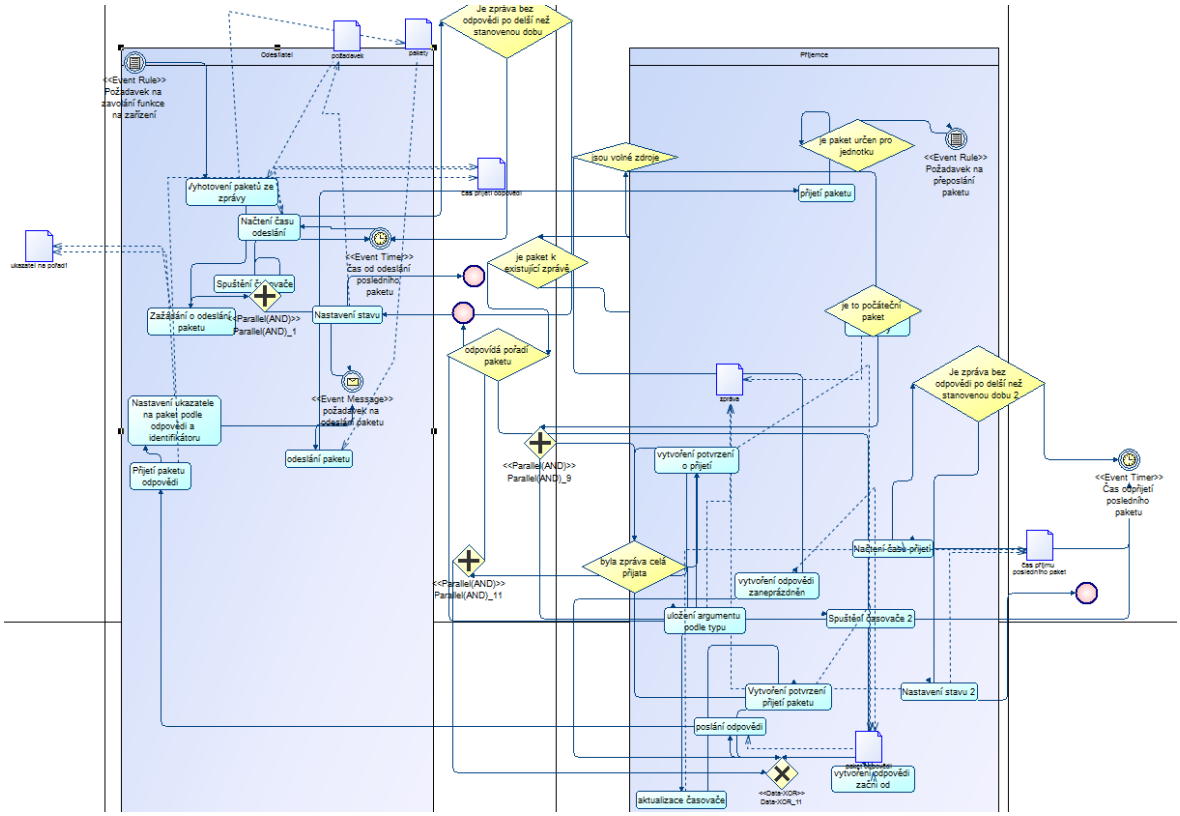
8.2 Obecný popis

Výsledný protokol jsem se rozhodl koncipovat tak, že jednotlivé každý požadavek je reprezentován zprávou, která obsahuje identifikátor vzdáleně volané metody, adresu zařízení na kterém je funkce volána, argumenty se kterými má být funkce volána, adresu žadatele o vykonání požadavku a případně zda je požadována nějaká návratová hodnota z vykonané funkce, která se má vložit jako argument do zadané funkce na zařízení, které zprávu odesílalo.



Obrázek** 14**:
Znázornění jednoho paketu (Zdroj: autor)

Jelikož argumenty ve zprávě mohou mít rozdílnou délku, některé parametry jako návratová funkce jsou nepovinné a nerad bych zbytečně zatěžoval na příliš dlouhé časové úseky komunikační linku, rozhodl jsem se zprávu rozdělit na menší datové části – pakety. Model paketu je znázorněn na obrázku **14**. Paket je složen z 9 bajtů, kde první a poslední bajt se interpretují jako 2 konkrétní znaky, které uvozují tělo paketu. První znak těla paketu obsahuje adresu cílového zařízení v rozsahu 0 -127, kde 0 je rezervovaná pro řídící jednotku. Pomocí protokolu je tedy možné identifikovat až 127 zařízení. Druhý znak identifikuje odesílatele. Třetí bajt obsahuje data interpretovaná jako znak podle kódování ASCII a odpovídá tělu paketu s daty je naloženo podle typu paketu, který se nachází na další pozici a odpovídá číslu od 0 do 8. Interpretace bude vysvětlena v navazující podkapitole. Pátý znak obsahuje číslo s pořadím paketu, aby bylo následně možné sestavit zpětně požadavek seřazením dat postupně podle pořadí. Předposlední bajt těla paketu obsahuje identifikátor zprávy, aby bylo možné jednoznačně odlišit různé požadavky od stejného odesílatele. Poslední bajt představuje kontrolní součet – v tomto případě počet jedniček v bajtech těla zprávy bez kontrolního součtu.



Obrázek** 15**:
Proces komunikace (Zdroj: autor)

Definice chování protokolu je znázorněna na obrázku **15**. Vyobrazený proces rozšiřuje obrázek **9** o konkrétní implementaci komunikace, znázorňuje tedy celý proces od vytvoření požadavku k zavolání na vzdáleném zařízení až po jeho faktické zavolání s tím že bere v úvahu i funkci prostředníka, kdy paket jednoduše přepošle (tuto činnost periferie neimplementují, ty paket jednoduše ignorují, když není určen pro ně). Proces definuje i chování v případě ukončení spojení či nějaké závažné chyby. Požadavek se bere za nedoručený.

8.3 Popis jednotlivých typů paketů

Argument těla paketu se vždy interpretuje podle čísla na pozici typ paketu podle obrázku **14**. Pro číslo 0 představuje argument číslo s celkovým počet paketů, ze kterých se zpráva skládá. Tento paket zprávy je nutné odeslat jako první, aby příjemce věděl, kdy přijal celou zprávu. V případě čísla 1 představuje tělo identifikátor volané funkce na vzdáleném zařízení, 8 zase představuje číslo funkce na vzdáleném zařízení. Samotná část argumentu pro volanou funkci je v těle obsažena pokud se typ paketu rovná číslu 3. Jednotlivé data argumentů jsou oddělena pomocí paketu s typem 2. Tělo paketu je v tomto případě nedefinováno stejně jako v případě čísla 4 a 5, které představují potvrzení přijetí paketu respektive celé zprávy. Číslo 7 je nositelem informace, že zařízení je zaneprázdněno.

Pakety s typem čísla 4, 5 a 7 představují samostatné pakety odpovědi řídící pouze tok komunikace.

8.4 Shrnutí

Tato část práce byla klíčová pro další kapitolu a naplnění cíle, jelikož v rámci ní byl definován aplikační protokol, na kterém budu stavět při řešení softwarové části. Návrh protokolu hodnotím jako povedený oproti Modbus mi přijde daleko jednoduší a navíc s možnostmi aplikace, které vyžaduje naplnění cíle.

9 Implementace řídícího softwaru

Obsahem této kapitoly je postup implementace protokolu navrženého a popsaného v předchozí kapitole jak pro řídící zařízení, tak pro periferie a samotnou komunikační jednotku. Přičemž pro periferie a prostředníka je výsledný kód vytvořen v jiném programovacím jazyku a pomocí jiného paradigmatu než program pro PC. Důvody tohoto řešení zdůvodním hned v první části.

V této části práce je rovněž nastíněno používání jednotlivých programů. Jednotlivé ukázky implementace a používání jsou demonstrovány na příkladu sumo robota. Tyto příklady lze velice snadno zobecnitelné i pro jiné použití. Řízení systému je řešeno formou grafického rozhraní, které implementuje program pro řídící zařízení (PC). Program je dělán tak, aby šlo grafické rozhraní nahradit, či doplnit, i dalšími formami řízení jakými jsou například různé formy umělé inteligence, kterou používal i projekt Minidarpa. Všechny okomentované zdrojové kódy jsou nahrány na přiloženém CD.

9.1 Implementace software pomocí Java a C

Program pro univerzální komunikační jednotku a periferie jsem se rozhodl napsat v jazyce C. Tento programovací jazyk je velice mocný nástroj, kterým lze zapsat jakýkoliv program, a navíc dokáže programátora dostatečně dobře abstrahovat od hardwarové realizace. Program napsaný v jazyce C lze v ideálním případě dobře porotovat na jiná zařízení. Tento jazyk má však jednu slabinu, pro některé možná i výhodu. Programátor je nucen v některých případech spravovat paměť sám. Může přímo zapisovat data na konkrétní místa v paměti a může z konkrétních adres číst, což při neopatrnosti může vést k pádům aplikace.

Po zralé úvaze jsem se rozhodl nepoužít zdrojový kód v C i jako základ pro řídící program na samotném PC, ale raději jsem zvolil jazyk JAVA, který bohužel procesory PIC nepodporují. K použití jazyku JAVA mě vedlo hned několik důvodů. Ukáži tím, že navržený protokol lze implementovat i v dalším jazyce a je skutečně nezávislí na platformě, a pak si tím také ulehčím výrazně práci. Dobře napsaný program napsaný pomocí objektového paradigmatu půjde velmi dobře rozšířit, s programováním v objektových jazycích jako JAVA mám velmi dobré zkušenosti a již jsem pracoval s celou řadou grafických knihoven, které tento jazyk podporuje. Při programování budu navíc moci použít celou řadu návrhových vzorů, které mi některá řešení zápisu kódu výrazně ulehčí.

9.2 Software napsaný v C pro embedded zařízení

Software v C jsem se snažil napsat co nejjednodušší a zároveň co nejuniverzálnější, aby šel použít jak na samotných periferiích, tak i na prostředníkovi. Navíc bylo nutné umožnit implementátorovi výsledný program nastavit tak, aby dokázal běžet i na slabších zařízení, proto jsem některé části programu definoval pomocí maker. Pomocí maker se definují některé konstanty, které mají přímí vliv na paměťovou náročnost a velikost programu. Všechna tato nastavení včetně nastavení adresy zařízení se provádí pomocí editace souboru „konfigurace.h“. Pro bližší informace odkážu na komentáře, které jsou součástí zdrojového kódu. Zmiňovaný soubor včetně souboru „konfiguraceProtokolu.h“, kde jsou definovány konstanty protokolu, „datoveTypy.h“, který definuje datový typ boolean, souboru „zarizeni.h“, ve kterém jsou definovány šablony metod, struktury pro program a globální proměnné, a „zarizeni.c“, jenž obsahuje těla některých těchto metod, jsou jediné soubory, které se musí nahrát do adresáře projektu a zahrnout do souboru, kde budou funkce volány. To se udělá pomocí následující instrukce:

```
#include "zarizeni.h"
```

Ve výsledném programu se dále naimplementuje například následující metoda:

```
void nahradSlova(Zprava *zprava){
     int i, j;
     for (i = 0; i < (*zprava).indexAktualnihoTela[0]; i++) {
          boolean obsahujeRetezecKNahrazeni = TRUE;
          for (j = 0; j < (*zprava).indexAktualnihoTela[1]; j++) {
               if ((*zprava).argumenty[0][i+j]!=(*zprava).argumenty[1][j]){
                   obsahujeRetezecKNahrazeni=FALSE;
                   break;
               }
          }
          if (obsahujeRetezecKNahrazeni==TRUE){
              for (j = 0; j < (*zprava).indexAktualnihoTela[2]; j++) {
                    (*zprava).argumenty[0][i+j]=(*zprava).argumenty[2][j];
              }
          }
     }
     if ((*zprava).vyzadujeOdpoved==TRUE){
        odesliOdpoved(zprava, (*zprava).argumenty[0]);
     }
}
```

Což je ukázka metody, která nahradí řetězce ve slově jiným řetězcem. Tato metoda je velice jednoduchá a rozhodně v některých případech nemusí být bezpečná – například nahrazovaný řetězec je kratší než řetězec, který ho nahrazuje. Důležité je, aby metoda měla návratový typ VOID a jediný argument v podobě ukazatele na strukturu Zprava. V této struktuře jsou uloženy všechny data k příkazu, který tuto funkci vzdáleně zavolal. Lze

vyčíst argumenty číslo n pomocí ukazatele na pole znaků zavoláním (*zprava).argumenty[n]. Konec tohoto řetězce je ukončen ukončovacím znakem. Samotná aplikace si hlídá správný počet argumentů, pokud počet nesedí, funkce není zavolána. Z funkce lze odeslat i odpověď na příkaz. To se provede pomocí metody odesliOdpoved.

Metodu je ještě nutné zaregistrovat tak, aby mohla být zavolána a to například následujícím kódem, který definuje tělo inicializujUzivatelskeMetody, a k výše definované funkci registruje ještě další dvě na práci s čísly.

```
void inicializujUzivatelskeMetody(){
    zarizeni.metodyZarizeni[0].metodaKVykonani=&sectiCisla;
    zarizeni.metodyZarizeni[0].pocetArgumentu=2;
    zarizeni.metodyZarizeni[1].metodaKVykonani=&porovnejCisla;
    zarizeni.metodyZarizeni[1].pocetArgumentu=2;
    zarizeni.metodyZarizeni[2].metodaKVykonani=&nahradSlova;
    zarizeni.metodyZarizeni[2].pocetArgumentu=3;
}
```

Samotné zařízení se inicializuje pomocí metody inicializaceZarizeni("jmenoMehoZarizeni") volané jednou v hlavní smyčce programu. Následně může být v nekonečném cyklu volána metoda vykonaniProgramu(), která se postará o samotnou komunikaci a vykonání zavolaných funkcí. Nyní už stačí pouze implementovat funkci odeslání paketu pro použití s WiFi modelem například následovně:

```
void odesliPaket(char paket[POCETBYTUPAKETU+1]){
     unsigned int numOfSentBytes;
     Net_Wireless_MCW1001_TCP_SendBytes(socketChild, paket, POCETBYTUPAKETU,
&numOfSentBytes);
}
```

A volat funkci příjem znaků (nejlépe ve druhém vlákně pomocí přerušení) ze sériové linky následovně:

```
void interrupt() {
  if (RC2IF_bit==1){
         prectiZnakPaketu(&UART2_Read);
         RC2IF_bit=0;
  }
}
```

Důležitá je metoda prectiZnakPaketu. Do té se vloží přijatý bajt. Při načítání znaků došlých přes WiFi je při použití knihovny z MikroC o něco složitější. Načítání znaků je nutné vykonávat v hlavním vlákně před voláním hlavního programu. Metoda může být definována například následovně:

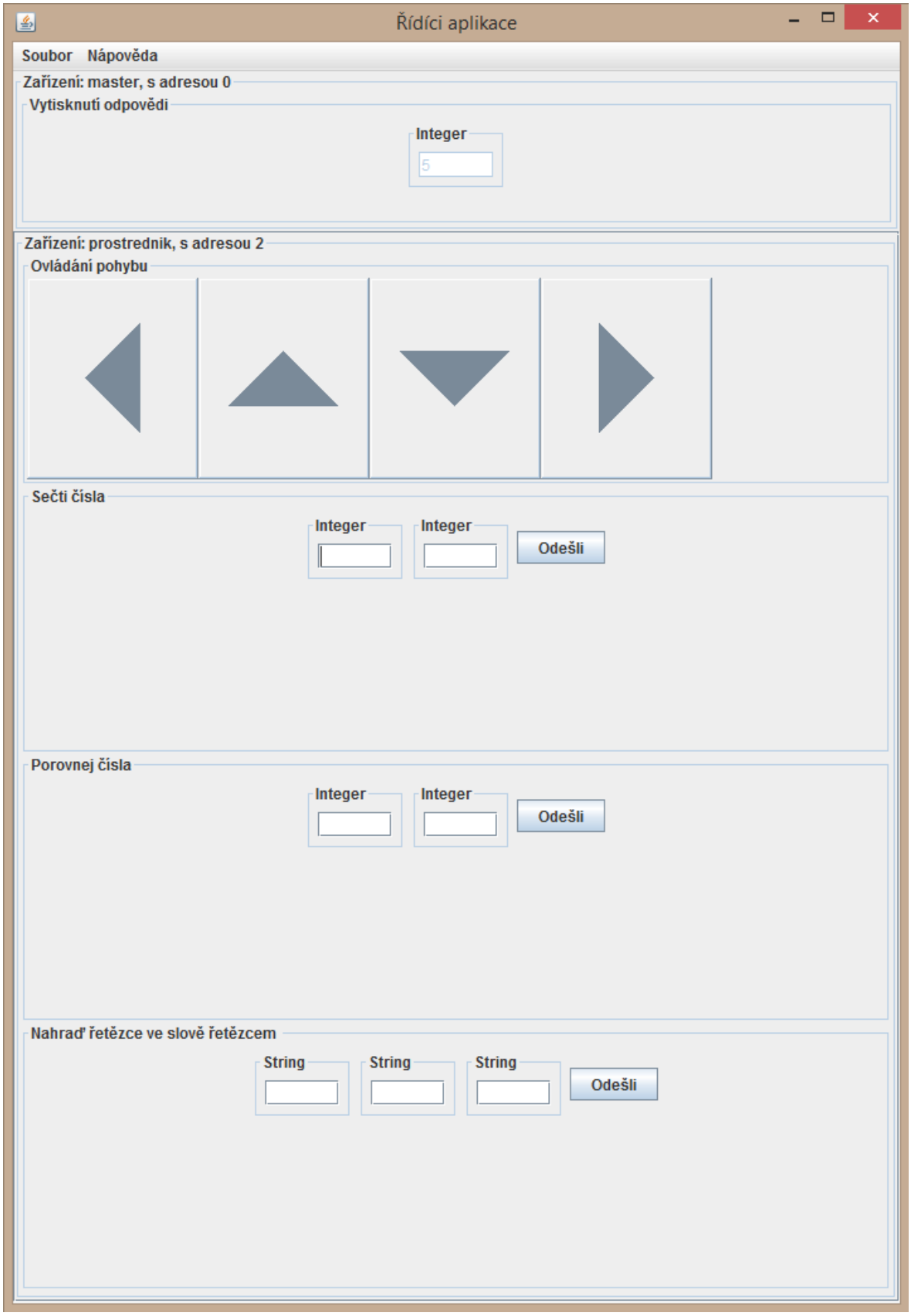
```
void odesliPaket(char paket[POCETBYTUPAKETU+1]){
     unsigned int numOfSentBytes;
```

```
    Net_Wireless_MCW1001_TCP_SendBytes(socketChild, paket, POCETBYTUPAKETU,
&numOfSentBytes);
}
```

Na závěr ještě podotknu, že všechny výše vypsané soubory jsou psané podle dlouholetého standardu C89, takže jsou bez problému kompilovatelné jakýmkoliv novějším kompilátorem C.

9.3 Řídící software napsaný v JAVA

Program napsaný v jazyce JAVA implementuje stejný protokol jako program v C, rozdíl je ovšem ve způsobu, jakým je protokol implementovaný. V programu popsaném výše jsem používal hlavičkový soubor „zarizeni.h“ pro definici vlastních datových struktur jako je třeba Metoda či Zpráva. V jazyce JAVA všechny tyto objekty zapisuji pomocí samostatných tříd. Program napsaný tímto způsobem je tedy daleko větší, ale na druhou stranu pro mě daleko přehlednější, jelikož každý objekt řeší určitou problematiku a logika aplikace je daleko více členěná do samostatných nezávislých celků, které mezi sebou interagují, než aby bylo všechno nahuštěné většinou pohromadě, jako je tomu v předchozím případě. Hlavním důvodem pro volbu JAVA byla ovšem myšlenka ulehčení si práce a zajištění další rozšiřitelnosti výsledného programu. Proto jsem volil architektonický návrhový vzor Model-View-Controll, který odděluje datový model od logiku ovládání a samotného zobrazení výsledků. Na úrovni datového modelu je implementován protokol, tato komponenta programu je univerzální pro všechny aplikace. K datům modelu se přistupuje přes různé pohledy (view), které jsou již závislé na aplikaci. Jelikož systém bude řídit člověk, jako pohled naprogramuji grafické uživatelské rozhraní. Výsledek je na obrázku **16**. To si bude načítat data přímo z modelu a případně zasílat modelu události pomocí kontroleru.



***Obrázek** 16:*
Uživatelské rozhraní řídícího programu (Zdroj: autor)

V programu dále hojně využívám dědičnosti a rozhraní a dalších návrhových vzorů jako je Singleton, Adaptér a třeba Observer. Návrhový vzor Strategy je zde používán pro definování funkcí, které se mají vykonat na řídícím zařízení. Účel je podobný jako u Funkcí v předchozím programu – tedy vzdáleně zavolat funkci s odeslanými argumenty. Příkladem implementace je například tato jednoduchá třída implementující rozhraní MetodaInterface, která vrátí první přijatý argument jako řetězec. Tato metoda je v řídícím programu používaná pro zobrazení výsledku například funkce „nahradSlova“ definované v předchozí části.

```
public class MetodaVytisknutiZpravy implements MetodaInterface{

    @Override
    public String vykonejMetodu(ArrayList<String> teloArgumenty, int odesilatel)
throws IllegalArgumentException {
        return teloArgumenty.get(0);
    }

}
```

V současné verzi programu je nutné nejdříve jednotlivé periferie zaregistrovat. Nejdříve je vytvořena nová instance Zarizeni a následně se toto zařízení přidá pod správu řídicího systému:

```
Zarizeni zarizeni2 = new Zarizeni(2, "prostrednik");
Master.getInstance().pridejZarizeni(zarizeni2);]
Metoda nahradSlova se pro zařízení registruje následovně:
[ArrayList<Class> argumentyTypy4 = new ArrayList<>();
argumentyTypy4.add(String.class);
argumentyTypy4.add(String.class);
argumentyTypy4.add(String.class);
zarizeni2.pridejMetodu(new Metoda(argumentyTypy4, false,
Master.getInstance().vratMetoduPodleIndexu(0), "Nahraď řetězce ve slově řetězcem
"));
```

Nejdříve je nutné vytvořit seznam argumentů, se kterými se metoda má volat, a poté je nutné definovat novou metodu. Ta v případě zavolání bude na zařízení s adresou 2 posílat požadavek na nahrazení zadaných znaků ve slově jinými znaky, navíc vzdálená metoda bude volána tak, aby následně vrátila výsledek zavoláním metody ze třídy MetodaVytisknutiZpravy. Na základě takto registrovaných dat se pak vytvoří pohled na obrázku **16** s boxy na tři argumenty v panelu zařízení 2 s tlačítkem, které odešle požadavek.

Program je psán tak, aby bylo možné používat více rozhraní pro komunikaci. Na ukázku jsem implementoval dvě – síťové pro komunikaci přes WiFi pomocí TCP/IP a sériovou linku. Ještě podotknu, že program běží v několika výpočetních vláknech – pro grafické zobrazení, pro odesílání paketů a pro čtení paketů. Příkazy na periferie je tedy možné

přidávat do fronty, ze které jsou postupně vykonávány, zatímco je najednou přijímáno třeba několik zpráv. Pro demonstraci robota jsem vytvořil speciální panel se směrovými šipkami.

9.4 Shrnutí

Pevně věřím, že se mi touto kapitolou pomohlo objasnit principy fungování programů a jejich implementace. Implementovat řídící software v objektovém programovacím jazyce se ukázalo jako dobrý nápad, jelikož mi to umožnilo vytvořit aplikaci, kterou bych pomocí C vytvářel velice těžce. Nasazením programů na PC a vývojový kit jsme si ještě navíc vše otestoval a potvrdil si funkčnost. Nyní už mi zbývá realizovat vlastní desku univerzální komunikační jednotky, na kterou hotový program nasadím.

10 Realizace prototypu komunikační jednotky a demonstrace na Minisumo robotovi

Cílem této kapitoly je nejdříve navrhnout a následně realizovat desku plošného spoje schopnou vykonávat software navržený v předchozí kapitole na vývojovém kitu [2]. Tato deska představuje hardwarovou realizaci univerzální komunikační jednotky popsané v kapitole s konceptem a umožňuje tedy řídící jednotce bezdrátově komunikovat s několika periferiemi připojenými k desce. V poslední části této kapitoly navrženou desku oživím nahráním firmware a zapojím do systému s minisumo robotem, čímž demonstruji její funkčnost.

10.1 Výběr součástek

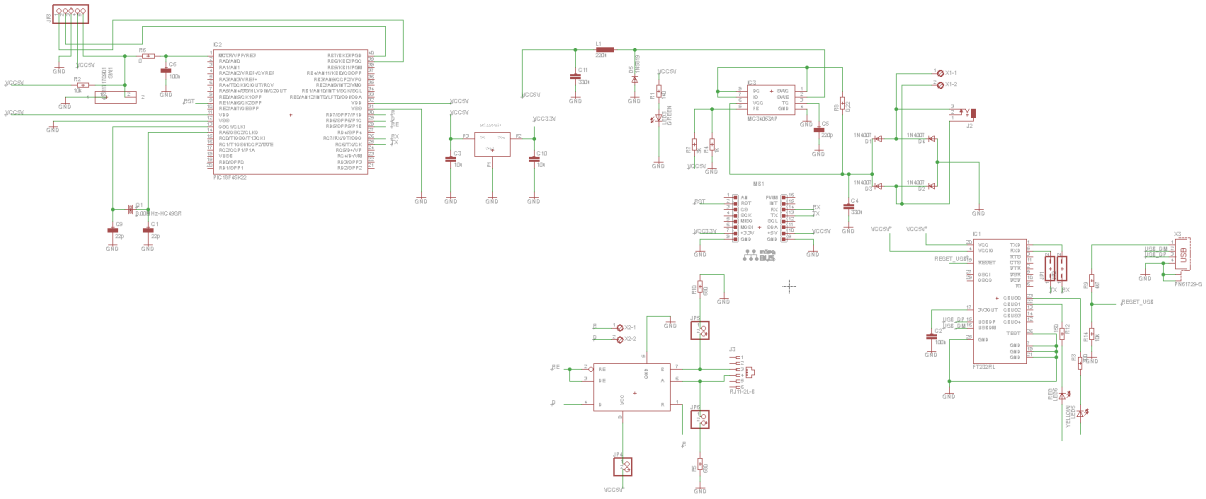
Nejdůležitější součástkou na vytvářené desce je bezesporu samotný procesor. Oproti referenčnímu vývojovému kitu jsme zvolil model PIC18F46K22 [15] s více jak dvojnásobnou pamětí RAM a o něco větší pamětí na samotný program. Tento model jsem volil z několika důvodů. Prvním z nich je fakt, že mikrokontrolér je vystavění na stejné architektuře jako PIC18F45K22 a kromě paměti se od svého slabšího bratříčka ničím zásadním neliší. Dalším důvodem je získání jisté rezervy, která se vždycky hodí. Tato rezerva navíc není vykoupena přílišným rozdílem v cenně obou procesorů.

Komunikace s PC bude realizovaná pomocí zapůjčeného WiFi modulu [30] od stejné firmy, která vyrobila i vývojový kit. Výhodou zapůjčeného modulu je přítomnost obvodu a softwaru implementující TCP/IP stack, který by jinak bylo nutné řešit programem nahraným na vytvářené desce, který by se velmi těžce implementoval a znamenal by velké vytížení mikroprocesoru při obsluze vysílače. Tuto část nemusím tak vůbec řešit a stačí se pouze prostřednictvím sériové linky připojovat k modulu a vyzvednout si sadu přijatých bajtů. Komunikace přes linku RS485 tak náročná na výpočetní výkon není, proto ji řeším přímím připojením běžně dostupného transiveru k mikrokontroleru.

10.2 Schéma zapojení

Schéma vlastního zařízení je zobrazeno na obrázku **17**. Zapojení vychází převážně z návrhu používaného kitu [2]. Používá „mikro BUS“ sběrnici, což je sběrnice výrobce kitu - společností Mikroelektronika. Díky této sběrnici bude v budoucnu možné připojit nejenom samotný WiFi modul, ale i jiné kompatibilní rozšíření pro komunikaci od tohoto výrobce jako například modul pro komunikaci prostřednictvím Bluetooth. Kromě rozhraní RS485, jehož zapojení je rovněž realizováno na základě jednoho z modulů, umožňuje

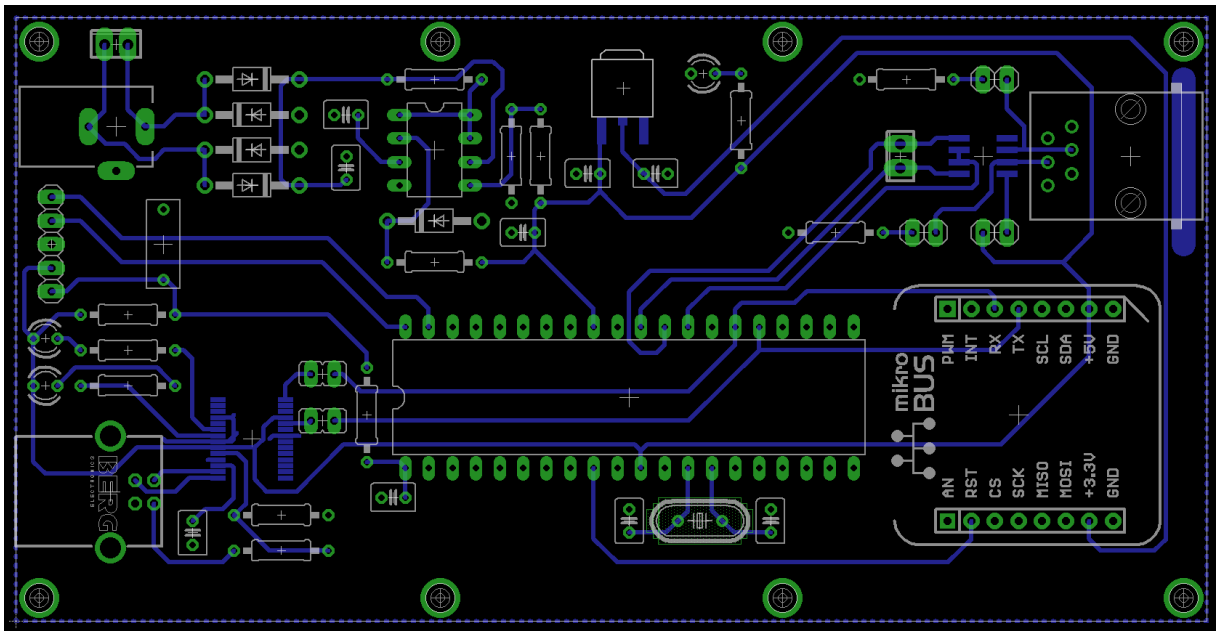
deska komunikovat pomocí USB (je instalován převodník sériové linky na USB) a na krátkou vzdálenost i prostřednictvím RS232. Schéma zapojení je dostupné i na přiloženém CD.



***Obrázek** 17:*
Schéma zapojení (Zdroj: autor)

10.3 Návrh desky plošného spoje

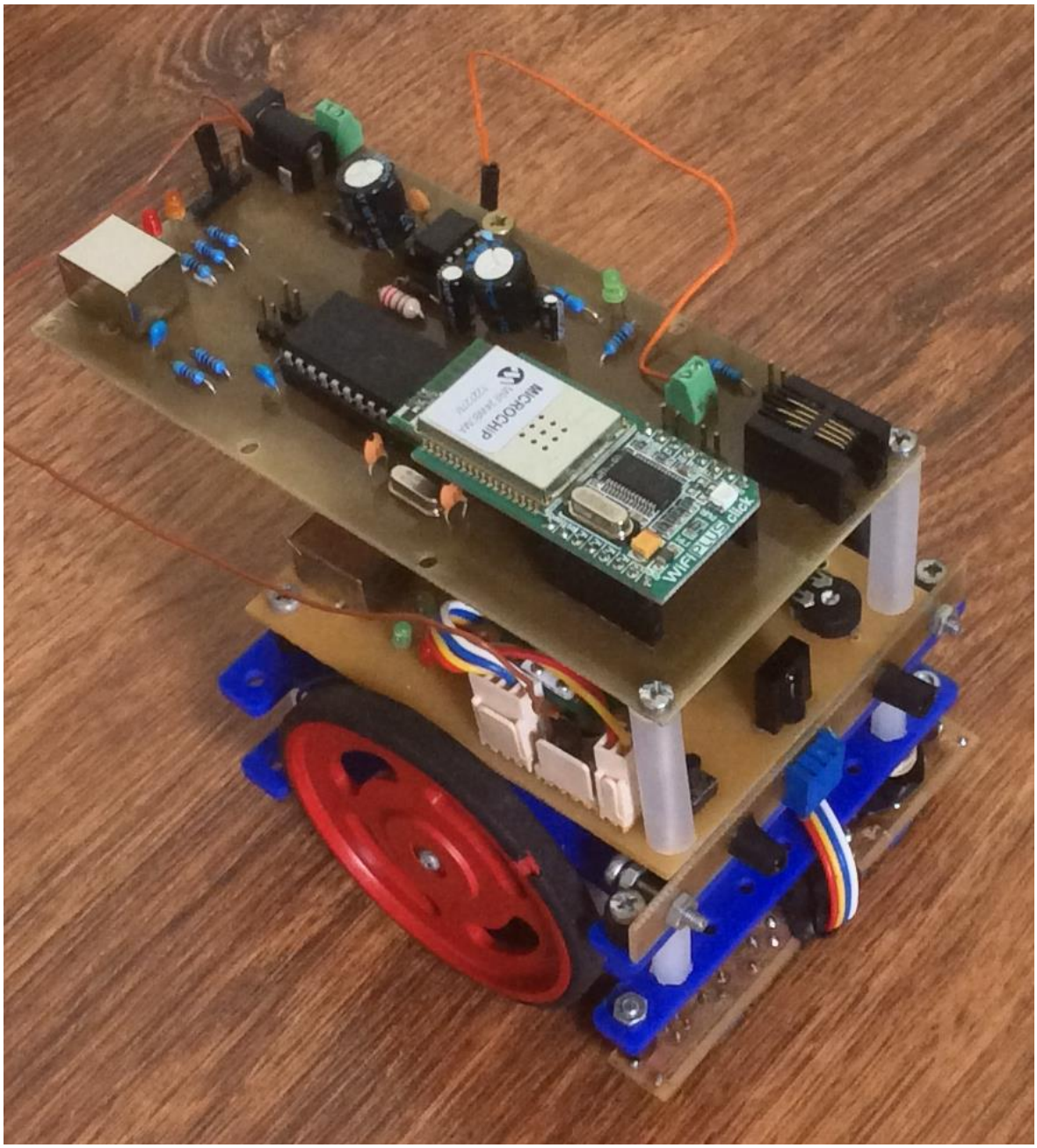
Samotnou desku plošného spoje vyobrazeno na obrázku **18**, jsem se rozhodl realizovat většinou pomocí součástek v diskrétních pouzdrech, jelikož s vytvářením podobných produktů mám nulové zkušenosti. Za pomoc při návrhu a samotné realizaci desky velmi vděčím svému konzultantovi. Bez něho bych výrobek v podobné kvalitě realizoval jen těžko. Detailní seznam součástek lze nalézt v příloze číslo 1 či v podobě tabulky na přiloženém CD. Na přiloženém CD naleznete i samotný návrh z obrázku **17** a **18**. Celková výroba desky i s cenou součástek a následným osazením mě vyšla na cca 600,-.



Obrázek** 18**:
Návrh desky plošného spoje (Zdroj: autor)

10.4 Ukázka aplikace a zhodnocení vlastností

Na obrázku číslo **19** je zachycen finální produkt již osazený na Minisumo robota, na kterém své řešení demonstruji. Robot je k desce připojen pomocí sériové linky a univerzální komunikační jednotka komunikuje s PC prostřednictvím WiFi. Robot implementuje funkce jízdy dopředu, couvání a otočky doleva nebo dopravu. V reálu to znamená zavolání správné funkce, kde uvnitř těla funkce se nastaví hodnota registru, kterou jsou řízeny serva na minisumo robotovi.



Obrázek** 19**:
Fyzická realizace desky instalovaná na sumo robota (Zdroj: autor)

Samotné měření vlastností systému případně nějaké porovnání s jiným obdobným systémem lze velmi těžko realizovat, navíc navržené řešení lze stěží považovat za finální, spíše se jedná o první verzi, proto vlastnosti nemá smysl nějak extenzivně měřit. Skvělou zprávou je, že systém reaguje na příkazy z řídícího PC velice svižně. Komunikace je velice rychlá a to hlavně díky jednoduchému protokolu a implementací samotného programu, který se zbytečně nevětví a většinou vykonává pouze elementární výpočty, jenž jsou volány podle aktuálního stavu. Pokud je tedy za úkol vykonat jen nějakou jednoduchou funkci s žádnými nebo velice krátkými argumenty na vzdáleném zařízení jako je třeba již zmiňované nastavení registru k servu, je odezva okamžitá. Pokud je za úkol například

provést složitější funkci, příkladem může být již popisované nahrazení určitých znaků v řetězci jinými znaky, výsledky jsou také velmi dobré a rychlost je víceméně omezena pouze výkonem embedded zařízení a nutností přenášet o něco větší zprávu. Změření rychlosti komunikace by si jistě zasloužilo vlastní kapitolu s návrhem vlastní metodiky měření. Toto by bylo velmi pracné, proto se uchýlím k pouhému konstatování, že navržené řešení je rychlejší než třeba Modbus již z prosté podstaty věci, že k vykonání zprávy je potřeba přenést podstatně méně bajtů. Pokud bych měl srovnat rychlost komunikace použité bezdrátové technologie s kabelem, vše hovoří ve prospěch kabelu. To je způsobené hlavně samotnou implementací knihovny k WiFi modulu. Při komunikaci pomocí kabelu jsou přijaté bajty postupně ukládány hned po jejich přijetí pomocí přerušení. U WiFi jsou přijatá data načítána vždy po vykonání jednoho cyklu hlavního programu. Pro finální řešení projektu na katedře by to chtělo vypracovat vlastní knihovnu, která z modulu bude znaky vyčítat rovnou pomocí přerušení.

10.5 Shrnutí

V rámci této kapitoly byly realizovány poslední střípky do skládačky k vytvoření univerzální komunikační jednotky pro embedded zařízení. Byla navržena a vyrobena deska, která byla následně oživena již hotovým firmwarem a otestována na minisumo robotovi, který tak mohl být ovládán bezdrátově prostřednictvím PC. Součástí této kapitoly bylo i shrnutí pozorování vlastností navrženého systému. Shrnutí nahradilo exaktní měření, které by si zasloužilo nějakou vlastní metodiku pro jeho realizaci s ohledem na povahu řešení, což by vyžadovalo udělat spoustu další práce nad povolený rozsah bakalářské práce.

11 Závěr

Cílem této závěrečné práce bylo navrhnout a realizovat univerzální komunikační jednotku pro oblast embedded zařízení umožňující řídicímu systému (například. PC) ovládat jednu nebo několik samostatných periferií. Cíl považuji za splněný. Jeho naplnění jsem demonstroval v závěru předchozí kapitoly na minisunorobotovi, který představuje jeden z možných příkladů aplikace výsledného produktu. Řešení bude jistě dobře použitelné i na projektech na katedře.

Výsledné řešení hodnotím jako unikátní a velice přínosné, jelikož přesně splňuje na něj kladené požadavky definované a postupně upřesňované v průběhu práce. Ke komunikaci je využíván vlastní protokol, který i přes svoji jednoduchost umožňuje vykonávat složité příkazy. Mezi hlavní přednosti řadím vzdáleného volání funkcí na jednotlivých periferiích způsobem jako je tomu u middlewaru pro roboty. Implementace komunikace prostřednictvím mého řešení na embedded zařízení je navíc velice jednoduchá a po přečtení kapitoly 9 by měla být i snadno pochopitelná. Stávající programy pro embedded systémy bude velice snadné přenést a využívat výhod komunikace. Subsystémy mohou v rámci systému komunikovat takřka neomezeně. Návrh desky v poslední kapitole také považuji za zdařilí. Může posloužit řešitelům jako reference.

Další možnou práci spatřuji v dalším rozvoji protokolu a zejména vylepšení jeho implementace. Programu v C by jistě slušela možnost dynamicky alokovat paměť pro některá data, jako jsou Metody a Zprávy, čímž by se pravděpodobně ušetřilo místo. Dalším výrazným zlepšením by bylo definovat metody protokolu – například pro nastavení adres a vyčítání konfigurace periferie pro dynamické registrování zařízení a jejich metod se způsobem volání na řídící jednotce. Také by bylo velice vhodné kód, komentáře a návod k jeho používání přeložit do angličtiny, aby z něho mohlo profitovat více lidí.

Seznam literatury

[1] ROUSE, Margaret. Definition master/slave. *SearchNetworking.com: Networking information, news and tips* [online]. 2008 [cit. 2014-03-30]. Dostupné z: http://searchnetworking.techtarget.com/definition/master-slave

[2] EasyPIC v7. *MikroElektronika* [online]. 2015 [cit. 2015-05-12]. Dostupné z: http://www.mikroe.com/easypic/ - odkaz

[3] GROULÍK, Tomáš. *Elektronika pro mobilní robot Minidarpa: Minidarpa robot - electronic layout design* [online]. Brno: Vysoké učení technické, Fakulta elektrotechniky a komunikačních technologií, 2008 [cit. 2014-03-23]. 1 elektronický optický disk [CD-ROM / DVD]. Dostupné z: https://dspace.vutbr.cz/handle/11012/3511. Bakalářská práce. VUT Brno, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky.

[4] Embedded Control Systems Design/Categories of system complexity. *Wikibooks* [online]. 2011, 10.3.2011 [cit. 2014-03-30]. Dostupné z: http://en.wikibooks.org/wiki/Embedded_Control_Systems_Design/Categories_of_system_complexity

[5] Embedded Systems/Embedded Systems Introduction. *Wikibooks* [online]. 2015 [cit. 2015-05-18]. Dostupné z: http://en.wikibooks.org/wiki/Embedded_Systems/Embedded_Systems_Introduction

[6] MICHELSON, Brenda. Event-Driven Architecture Overview. In: *Event-Driven Architecture Overview* [online]. 2011 [cit. 2015-04-16]. Dostupné také z: https://dl.dropboxusercontent.com/u/20315902/EventDrivenArchitectureOverview_ElementalLinks_Feb2011.pdf

[7] Freely Available Standards. *The International Organization for Standardization* [online]. 1993 [cit. 2015-05-18]. Dostupné z: http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html

[8] OLMR, Vít. HW server představuje - Sériová linka RS-232. *Hw.cz* [online]. 2005 [cit. 2014-02-12]. Dostupné z: http://www.hw.cz/rozhrani/hw-server-predstavuje-seriova-linka-rs-232.html

[9] DLABAČ, Petr. *Metody a způsoby bezdrátového přenosu dat ze snímačů na centrální jednotky*. Zlín, 2013. Dostupné z: http://dspace.k.utb.cz/handle/10563/25126. Bakalářská práce. UTB ve Zlíně, Fakulta aplikované informatiky, Ústav bezpečnostního inženýrství.

[10] REESE, Robert Bryan. *Microprocessors: from assembly language to C using the PIC18Fxx2*. Hingham, Mass.: Da Vinci Engineering Press, 2005, p. cm. ISBN 15-845-0378-5.

[11] MikroC PRO for PIC - C compiler for Microchip PIC microcontrollers. *MikroElektronika* [online]. 2015 [cit. 2015-05-16]. Dostupné z: http://www.mikroe.com/mikroc/pic/

[12] KUNA, Zdeněk. *Navigační subsystém robotu Minidarpa: Minidarpa robot - navigation system design* [online]. Brno: Vysoké učení technické, Fakulta elektrotechniky a komunikačních technologií, 2008 [cit. 2014-03-23]. 1 elektronický optický disk [CD-ROM / DVD]. Dostupné z: https://dspace.vutbr.cz/handle/11012/3513. Bakalářská práce. VUT Brno, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky.

[13] Network WiFi Library. *MikroElektronika* [online]. 2014 [cit. 2014-03-16]. Dostupné z: http://www.libstock.com/projects/view/356/network-wifi-library

[14] Oddělení zodpovědností. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2015-05-18]. Dostupné z: http://cs.wikipedia.org/wiki/Odd%C4%9Blen%C3%AD_zodpov%C4%9Bdnost%C3%AD

[15] OSI model. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2015-01-22]. Dostupné z: http://en.wikipedia.org/wiki/OSI_model

[16] PIC18F46K22. *MICROCHIP* [online]. 2014 [cit. 2015-01-18]. Dostupné z: http://www.microchip.com/wwwproducts/Devices.aspx?product=PIC18F46K22

[17] RONEŠOVÁ, Andrea. Popis protokolu Modbus I. *MCU.cz* [online]. 2008 [cit. 2015-02-23]. Dostupné z: http://mcu.cz/news.php?extend.1104

[18] HLADIŠ, Ondřej. *Programová knihovna pro podporu RS485 komunikace na vývojovém kitu M68EVB908GB60*. Zlín, 2011. Dostupné z: http://dspace.k.utb.cz/handle/10563/17562?show=full. Diplomová práce. UTB ve Zlíně, Fakulta aplikované informatiky Ústav informatiky a umělé inteligence.

[19] KERNIGHAN, Brian a Dennis RITCHIE. *Programovací jazyk C*. Vyd. 1. Brno: Computer Press, 2006, 286 s. ISBN 80-251-0897-X.

[20] KOPECKÝ, Martin. *Programové vybavení mobilního robotu Minidarpa: Minidarpa robot - software design* [online]. Brno: Vysoké učení technické, Fakulta elektrotechniky a komunikačních technologií, 2008 [cit. 2014-03-23]. 1 elektronický optický disk [CD-ROM / DVD]. Dostupné z: https://dspace.vutbr.cz/handle/11012/2529. Bakalářská práce. VUT Brno, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky.

[21] Robotics middleware. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2015-03-02]. Dostupné z: http://en.wikipedia.org/wiki/Robotics_middleware

[22] RS485 click 3.3V. *MikroElektronika* [online]. 2014 [cit. 2014-12-14]. Dostupné z: http://www.mikroe.com/click/rs485-3.3v/

[23] KOPECKÝ, Martin. *Řídicí systém mobilního robotu Minidarpa: Minidarpa robot - software design* [online]. Brno: Vysoké učení technické, Fakulta elektrotechniky a komunikačních technologií, 2010 [cit. 2014-03-24]. 1 elektronický optický disk [CD-ROM / DVD]. Dostupné z: https://dspace.vutbr.cz/bitstream/handle/11012/1702/Kopeck%C3%BD_Martin_DP_2010.pdf?sequence=2&isAllowed=y. Diplomová práce. VUT Brno, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky.

[24] LIBRA, Jaroslav. *Řízení pohonů mobilního robotu Minidarpa* [online]. Brno, 2010 [cit. 2014-03-23]. Dostupné z: https://dspace.vutbr.cz/handle/11012/1698. Diplomová práce. VUT Brno, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky.

[25] TIŠNOVSKÝ, Pavel. Sběrnice RS-422, RS-423 a RS-485. *Root.cz* [online]. 2008 [cit. 2014-02-12]. Dostupné z: http://www.root.cz/clanky/sbernice-rs-422-rs-423-a-rs-485/

[26] SEDLÁK, Luboš. *Senzoricky subsystém mobilního robotu Minidarpa: Minidarpa robot - sensors* [online]. Brno: Vysoké učení technické, Fakulta elektrotechniky a komunikačních technologií, 2008 [cit. 2014-03-23]. 1 elektronický optický disk [CD-ROM / DVD]. Dostupné z: https://dspace.vutbr.cz/handle/11012/3509. Bakalářská práce. VUT Brno, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky.

[27] TIŠNOVSKÝ, Pavel. *Sériový port RS-232C* [online]. 2008 [cit. 2014-02-12]. Dostupné z: http://www.root.cz/clanky/seriovy-port-rs-232c/

[28] MAŘÍKOVÁ, Denisa. *Topologie počítačových sítí* [online]. 2013 [cit. 2014-03-30]. Dostupné z: http://home.zcu.cz/~topinkov/index.html

[29] Wi-Fi. *Webopedia* [online]. 2014 [cit. 2014-9-12]. Dostupné z: http://www.webopedia.com/TERM/W/Wi_Fi.html

[30] WiFi Plus Click. *MikroElektronika* [online]. 2014 [cit. 2014-03-5]. Dostupné z: http://www.mikroe.com/click/wifi-plus/

Seznam obrázků a tabulek

Seznam obrázků

Přílohy

Seznam tabulek

Příloha A: Seznam součástek

Tabulka 1: *Přehled použitých součástek*

Součástka	Popis	Po-čet	Cena za kus	Cena celkem
MC33269DT-3.3G	Stabilizátor napětí, převod 5V na 3,3	1	15	15
PIC18F45K22/PIC18F46K22	mikrokontrolér	1	75	75
SOKL 40	pouzdro na PIC	1	5	5
FT232RL	převodník USB na UART	1	90	90
SN75176BD	budič RS485	3	5	15
MPT0.5/2-2.54	VCC ext. konektor přívod dráty	1	21	21
ST MICROELECTRONICS MC34063ACN	měnič DC-DC pro převod napětí na vstupu na 5V	1	10	10
Konektor RJ	zásuvka RS485	5	10	50
Konektor RJ	konektor, plochý kabel	10	2,3	23
USBB-G-SMD	USB B konektor	1	15	15
ZL262-8SG	Konektor PIN 8, 2,54mm	10	3,5	35
JUMPER-H/B	Jumper	20	1,5	30
DS1021-1*2SF1-1	Konektor PIN 2, 2,54mm	50	0,5	25
PC-GK2.1	napájecí konektor	1	5	5
LF XTAL003156	oscilátor 8MHz	1	20	20
Rezistory, pouzdro 207				
680		2	2	4
150		4	2	8
140		1	2	2
300		1	2	2
10k		2	2	4
1k		4	2	8
3k		1	2	2
0,22		1	2	2
4k7		1	2	2
Diody				
Green		1	2	2
Red		4	4	16
Yellow		1	4,5	4,5
1N4007		4	1,5	6
1N5819		1	3	3
Cívky				
200u		1	8	8
Kondenzátory				
100n		4		0,5
22p		2		0,4
10u		2		0,5

Součástka	Popis	Po-čet	Cena za kus	Cena celkem
330u/35V		2		0,4
220p		1		0,5

509,8