ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ,
FAKULTA ELEKTROTECHNICKÁ,
KATEDRA ŘÍDICÍ TECHNIKY

Diplomová práce

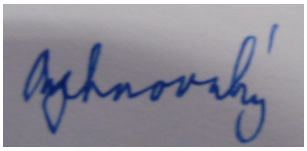
Řídicí sytém s využitím bezdrátové komunikace ZigBee

2011

Bc. Václav Rychnovský

Prohlášení

Prohlašuji, že jsem svou bakalářskou / diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.



V Praze dne 13.5.2011 …….

podpis

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra řídicí techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Václav Rychnovský**

Studijní program: Kybernetika a robotika
Obor: Systémy a řízení

Název tématu: **Řídicí sytém s využitím bezdrátové komunikace ZigBee**

Pokyny pro vypracování:

Práce má za cíl poskytnout sadu hardwarových modulů vhodných pro bezdrátové řízení modelové železnice, s tím že je vyžadovaný zpětný kanál pro odometrické snímání polohy jednotlivých vozidel a možnost uzavření zpětné vazby přes nadřazený řídicí počítač.
1. Na základě předchozích výsledků bakalářské práce navrhněte hardware, který bude vhodný jak pro vlastní řízení modelové železnice včetně prvků kolejiště. Přitom se zaměřte na to, aby bylo možné procesorový systém kombinovat s různými variantami návrhu RF části.
2. Upravte a navrhněte softwarové vybavení pro výše navržený HW. Ověřte možnosti použití otevřených komponent TinyOS a otevřených ovladačů pro použitý RF obvod
3. Pečlivě zdokumentujte navržené řešení a práci s vývojovými nástroji tak, aby bylo možné projekt dále rozvíjet a případně i použít v rámci výuky.

Seznam odborné literatury:

TinyOS: open-source operating system designed for wireless embedded sensor networks, http://tinyos.net/ ZigBee Alliance home page, http://www.zigbee.org/

Vedoucí: Ing. Pavel Píša, Ph.D.

Platnost zadání: do konce letního semestru 2011/2012

L.S.

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

prof. Ing. Boris Šimák, CSc.
děkan

V Praze dne 10. 2. 2011

Abstrakt

Práce předkládá rozbor a návrh kompletního řešení bezdrátového řízení modelové železnice s využitím přenosové vrstvy IEEE 802.15.4 a zpětným kanálem z řízených jednotek do řídicího počítače. Vlastní podrobně rozebírané řešení je postaveno na součástkách firmy Texas Instruments. Text práce se pak především zaměřuje na dokumentaci nového vývoje HW a SW, dalších kroků a testování alternativ, které následovaly po ukončení první etapy obhajobou bakalářské práce. Výstupem práce je hardware s ovládacím softwarem vhodným jak k trvalému nasazení v Ústavu teoretické informatiky a automatizace v Praze, tak i k další možné výuce na katedře Řídicí techniky. Jedná se sice o zakázkovou aplikaci, nicméně je možno ji použít v jakékoliv aplikaci pro sběr dat v jednoduché bezdrátové senzorové síti se zpětným předáváním informace senzorům, kde se hlavní důraz klade na dobu odezvy.

Abstract

This work presents the analysis and design of complete wireless solutions to control model railroad through using transport layer of the IEEE 802.15.4 and the return channel from the controlled unit to the control computer. Custom in detail analyzed solution is based on devices from Texas Instruments. The text of the work is primarily focused on documentation of new hardware and software developments, next steps and testing alternatives that followed the end of the first stage of the bachelor thesis defence. The output of this work is hardware and driver software suitable for permanent deployment at The Institute of Information Theory and Automation in Prague, as well as to other possible teaching at the Department of Control Engineering. Although this is a custom application, it can be used in any application for collecting data in a simple wireless sensor network and the return channel back to sensors, where the main emphasis is on response time.

Poděkování

Děkuji za spolupráci:
vedoucímu práce Ing. Pavlu Píšovi Ph. D.
a jeho spolupracovníkům,
firmě Texas Instrument za poskytnutý hardware,
zadavateli aplikace Ing. Květoslavu Beldovi Ph. D.

Obsah

1 Úvod 6
1.1 Cíle diplomové práce 6
1.2 Úvodní specifikace řešené aplikace 6
1.3 Požadavky na bezdrátovou komunikaci 6
1.4 Požadavky na rozměry 7
1.5 Požadavky na senzory a akční členy 7
1.6 Požadavky na napájení 8

2 Řešení bezdrátové komunikace 9
2.1 Popis řešeného problému 9
2.2 Navržené schéma komunikace 10
2.3 Protokol ZigBee, vlastnosti, vybrané řešení 12
2.4 SimpliciTI, vlastnosti, vybrané řešení 17
2.5 Analýza zpoždění přenosového řetězce 18

3 Hardwarové řešení 20
3.1 Přehled hardwarového řešení 20
3.2 Mikrokontrolér MSP430F2618 23
3.3 Transceiver CC2520 24
3.4 Ovladač motorku DRV8800 26
3.5 Optická myš PAN3401 (PAW3402) 27
3.6 Laserová myš PAW3601DH-NF 27
3.7 Návrh zálohovaní napájení 29
3.8 Návrh vf části 32

4 Softwarové řešení 33
4.1 Softwarové volby řešení bezdrátové komunikace 33
4.2 Vývojové nástroje 34
4.3 Kompilační parametry 35
4.4 Použití hodinového signálu 36
4.5 Ovladač transceiveru CC2520 38
4.6 Aplikační vrstva v Z-Stack 39
4.7 Aplikační vrstva v SimpliciTI 42
4.8 Soubor MeasExtension 43

5 Závěr 45

6 Literatura 46

Přílohy

Příloha A, tabulky propojení6 48

Příloha B, schémata zapojení6 50

Příloha C, fotogalerie 52

1 Úvod

1.1 Cíle diplomové práce

Tato diplomová práce popisuje výchozí specifikaci pro realizaci bezdrátového řídicího systému. Podrobnější specifikace obsahuje popis dalších částí, tak jak jsem je řešil v průběhu svého magisterského studia, kdy jsem postupně na základě pokynů zadavatele rozvíjel a odlaďoval předchozí návrh v reálném nasazení a v závěru provedl na základě získaných znalostí návrh nový, který staví na získaných zkušenostech. První část řešení si klade za cíl představit problém bezdrátového přenosu informace v malé a strukturou jednoduché bezdrátové senzorové síti. Srovnává dostupná řešení komunikace založená na bezdrátové síti ZigBee a její alternativy z hlediska vhodnosti použití pro řešenou aplikaci. Další část se soustředí na návrh vhodného hardwaru s obvody firmy Texas Instruments, který by obsahoval potřebné komunikační obvody spolu s měřicími a akčními členy. Třetí část podává přehled softwarových řešení zadaného problému, přičemž je hlavní důraz kladen na popis funkce jednotlivých částí kódu. Tyto dvě částí mají hlavně sloužit k předání poznatků z vývoje s hardwarem a softwarem k dalšímu rozvoji na katedře Řídicí techniky. Zaznamenal jsem zde vše, co považuji za důležité k archivaci použitých řešení. Podrobným popisem komunikačních rozhraní vrstev softwaru se zde nezabývám. Nakonec následuje zhodnocení dosažených výsledků a přílohy. Ty obsahují schémata zapojení s popisem uživatelského rozhraní desek a několik obrázků ilustrujících fyzické provedení aplikace.

1.2 Úvodní specifikace řešené aplikace

Výchozím předmětem k řešení je návrh a realizace bezdrátového systému pro přenos informace. V prvé řadě se jedná o specifickou aplikaci pro testování řídicích algoritmů v Ústavu teoretické informatiky a automatizace v Praze (UTIA). Celý systém má zabezpečit zejména přímé řízení rychlosti a měření polohy několika standardních modelových motorových jednotek v kolejové síti měřítka H0 (1:87). Rovněž jako další úkol má bezdrátový systém ovládat přestavovací prvky vlakové cesty a doplňkové měřicí prvky v kolejišti. Klíčové parametry jsou jak vhodné rozměry jednotlivých bezdrátových modulů, tak dostatečný dosah v přenosovém prostoru složeném z jedné místnosti. Cílem a zdrojem informací pro tento přenosový systém je osobní počítač s prostředím MATLAB Simulink. Hlavní důraz je kladen na dosažení dostatečné vzorkovací frekvence a spolehlivosti přenosu dat. Posledním požadavkem je možnost činnosti nezávisle na systému kolejového napájení.

Hotová aplikace (HW, SW) má být řádně vysvětlena pro případné další použití na katedře Řídicí techniky. K tomu náleží i určité zohlednění tohoto požadavku již při samotném návrhu. Navržené hardwarové řešení musí být dostatečně robustní a univerzální pro použití i v jiných aplikacích sběru dat. Softwarové řešení by pak mělo být navrženo z volně dostupných zdrojů softwaru kvůli možnosti libovolných úprav pro jiné aplikace.

1.3 Požadavky na bezdrátovou komunikaci

Bezdrátová senzorová síť se skládá z prostorově distribuovaných autonomních zařízení, která zpravidla předávají údaje ze svých senzorů do jednoho centrálního zařízení , které tato data následně vyhodnocuje. V mém případě je situace poněkud komplikovanější, neboť centrální zařízení zde musí ve stejném intervalu také předávat informaci všem jednotlivým uzlům. To vše při zajištění garantované doby odezvy, tj. doby trvání vyslání dat ke vzdáleným jednotkám, jejich zpracování na vzdálené jednotce a následné odeslání zpět do centrální jednotky a její zpracování. Při přímém řízení jednotek z PC pro předpokládané použití modelové železnice k testování řídicích algoritmů se uvažuje maximálním doba odezvy bezdrátové senzorové sítě 50 ms, tj. vzorkovací frekvence 20 Hz. V celé bezdrátové senzorové síti v uvažované aplikaci se počítá s maximálně 6

zařízeními pro měření/řízení (koncové uzly) a jedním centrálním zařízením (přístupovým uzlem, koordinátorem vysílání v síti) pro předávání nebo přijímání dat z PC. Dalším požadavkem na bezdrátovou síť je zajištění spolehlivosti přenosu dat. Toho lze dosáhnout využitím některého z komerčně používaných bezdrátových protokolů obsahujících mechanismus CSMA-CA (Carrier sense multiple access with collision avoidance), který zajišťuje bezkonfliktnost komunikace v senzorové síti. Pro tento účel jsou v této práci použit protokol IEEE 802.15.4. S fyzickou vrstvou a vrstvou přístupu k médiu protokolu IEEE 802.15.4 lze kombinovat více variant řešení vyšších komunikačních vrstev. Jedna z variant je použití plného protokolového zásobníku ZigBee. Alternativou je proprietární protokol SimpliciTI, který je jednodušší a používá vlastní zjednodušenou část kontroly kolizí na přenosovém médiu.

Posledním požadavkem je přenosový dosah bezdrátových zařízení. Pro použití na malém kolejišti v jedné místnosti postačí řádově jednotky metrů. Z hlediska univerzálnosti použití navrženého hardwaru by tento údaj měl být co největší. Maximální dosah je především závislý na použitém obvodu transceiveru a vlastnostech připojené anténky. Pro přenos je používán obvod CC2520, který dosahuje větší citlivosti a vysílacího výkonu oproti předchozímu obvodu CC2420. Předchozí obvod je na katedře Řídicí techniky i jinde [2],[3],[4] hojně používán v hotových modulech, které disponují zaručeným dosahem, což znamená, že autoři výše uvedených prací nemuseli tuto skutečnost řešit. Tyto moduly však nebylo možno použít z důvodu omezení týkajících se velikosti a malé přizpůsobitelnosti k vestavbě do mé aplikace. Navíc by jejich použití nepřineslo pro katedru poznatky z tvorby vlastního hardwaru s novějším obvodem CC2520, který je klíčovou částí navrženého hardwaru. Řešení anténky a navazující vf části na desce plošných spojů (samotná anténka je preferována také jako tištěný spoj) nelze rozhodně podcenit. Samotný předpoklad vyzkoušení několika typů antének se ukázal jako oprávněný. Vyzkoušel jsem několik typů antének navržených přímo výrobcem pro pásmo 2.4 GHz.

1.4 Požadavky na rozměry

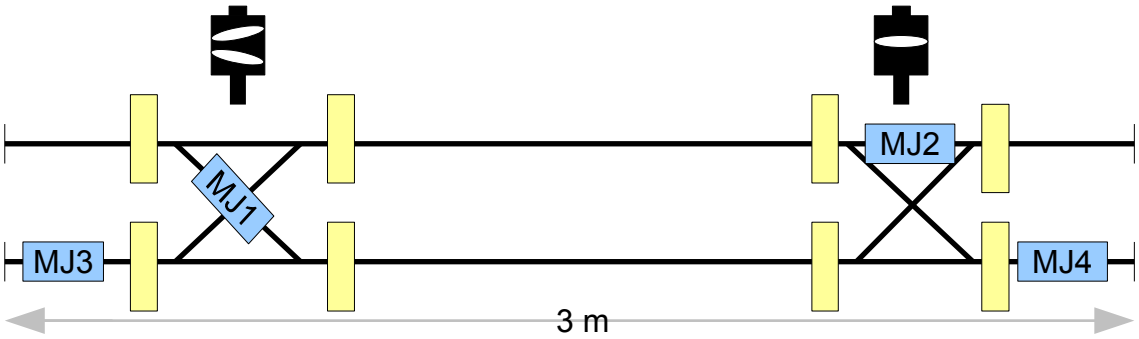
Vzhledem k použití hardwaru pro vestavbu do modelové železnice velikosti H0, musí být zvolena správná velikost fyzické realizace prvků bezdrátové senzorové sítě. Prvotním cílem návrhu hardwaru byla úplná miniaturizace tj. použití zejména nejmenší možné anténky pro VF obvod CC2520 a miniaturní spojovací konektory k senzorům, kvůli umístění do velmi malé modelové jednotky s vnitřní šířkou pouhých 21 mm a délkou zhruba 120 mm, která byla v Ústavu předem k dispozici. Tato malá modelová jednotka se však neosvědčila už ze samotného mechanického hlediska použití pro výzkumné účely v Ústavu. Její chodové vlastnosti v modelovém kolejišti jsou pro zpětnovazební řízení špatné (neustálý prokluz příliš malých hnacích kol). Proto jako konečné řešení byla zvolena možnost rozměrů větších a umístění na plošinový vůz typické šířky 35 mm a délky kolem 200 mm. Rozměry bezdrátových uzlů pro umístění do kolejiště nejsou těmito rozměry limitovány, nicméně v zájmu unifikace je výhodnější použít stejného hardwaru jako u jednotek pohyblivých.

1.5 Požadavky na senzory a akční členy

Neoddělitelnou součástí bezdrátové senzorové sítě jsou vlastní senzory. V tomto případě se jako nejvhodnější jeví použít optických senzorů myši. Pro přisvětlení běžně dostupné optické myši využívají buď klasickou LED nebo infračervenou laserovou LED. Obecně se dá říci, že laserové myši jsou vhodnější pro snímání na hladkém povrchu. Jako první jsem navrhl snímač s použitím senzoru PAN3401 (jiné označení téhož obvodu PAW3402). Tento obvod však vzhledem k použití ryze počítačového rozhraní PS/2, není pro odečítání polohy ve vestavných aplikacích vhodný. Nicméně byl již k dispozici, tedy vyzkoušel jsem nejdříve práci s tímto obvodem. Navíc je jisté, že jím disponuje každá počítačová myš, z které je nejsnadnější optické snímače vytěžit. Dále jsem proto ověřil obvod PAW3601DH-NF s přisvětlením infračervenou laserovou diodou. Tento obvod již disponuje inkrementálními výstupy a funguje i s napájecím napětím 3.3 V, které je kompatibilní s ostatními integrovanými obvody.

Jako základní akční člen pro řízení modelové motorové jednotky je nejvhodnější integrovaný můstek. Z důvodu kompatibility s ostatním hardwarem je použit DMOS můstek DRV8800, který i bez chlazení s velkou rezervou vyhoví pro malý stejnosměrný motorek s permanentními magnety s typickým napájecím napětím 9 V a proudem 100 *mA*. Pro řízení můstku se z důvodu minimalizace ztrát používá pulsně šířková modulace.

Pro obsluhu senzorů a akčních členů dvou bezdrátových jednotek kolejiště se již žádné další požadavky výkonového spínání nekladou, neboť výkonová část je umístěna na samostatné desce - pevné součásti kolejiště. Je zde pouze požadavek na generování signálu pro budič optických závor - společný pro 4 senzory. Tyto senzory registrují průjezd soupravy (příp. jednotlivých jednotek soupravy vozů) a určují tak pevně pozici pohybující se jednotky. Tato kalibrace je nutná vzhledem k tomu, že při přejezdu nerovného povrchu dochází ke ztrátě polohy z optické myši. Na každou jednotku dále připadají 4 výhybky tj. celkem 8 cívek. Pro přepínání směru však stačí pouze 2 signály - funkce cívek jde sdružit tak, že jsou vždy všechny výhybky přepnuty buď do rovna nebo na změnu trati.



Obr 1: Schéma kolejiště se dvěma bezdrátovými jednotkami, výhybky jsou navzájem propojené pro každé místo změny trati, které ovládá jedna bezdrátová jednotka, celkem je zde 8 optických závor sloužících ke kalibraci polohy

1.6 Požadavky na napájení

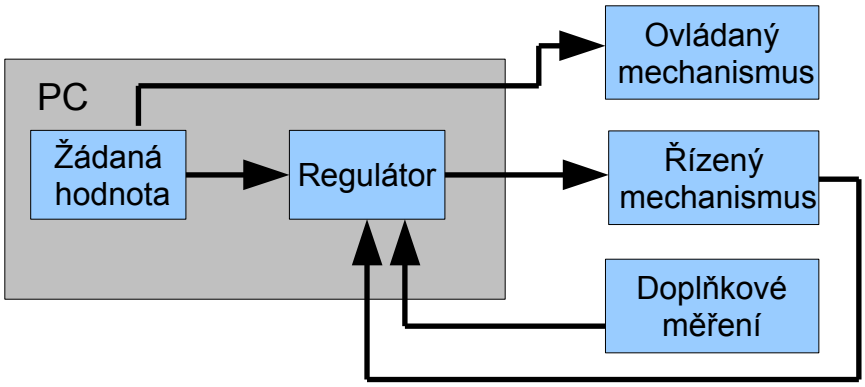
Pro napájení bezdrátových senzorových uzlů se typicky používá napětí které je násobkem některého z jmenovitých napětí elektrochemických zdrojů. Zde ani specializovaný obvod CC2520 není výjimkou neboť pracuje v rozmezí 2,4 - 3,3 V . Dá se tedy napájet pomocí dvou standardních tužkových baterií/akumulátorů nebo jedné lithiové. V rámci unifikace je žádoucí mít toto napětí shodné pro všechny logické obvody. Pro výkonovou část je potřeba napětí mnohem vyšší. Pro motorek trakční jednotky to je 9 V . Pro spolehlivé přepnutí výhybky je to však až 15 V se špičkovým odběrem proudu až 1 A . Uvažujeme-li ztráty na usměrňovači (ty zde musí být, pokud nechceme aby záleželo na polaritě kolejí), je nutné napájecí napětí až 19 V .

Pokud chceme zajistit zálohování funkce motorové jednotky pro funkci i na kolejišti s nekompatibilním napájením a použít např. NiMh (Nikl-Metal-hydrid) akumulátory, je potřeba pro DC motorek a konečné vybíjecí napětí jednoho článku akumulátoru 1 V celkem 10 článků do série. Kapacita pro alespoň 1 hodinu činnosti je pak s dostatečnou rezervou kolem 300 mAh .

2 Řešení bezdrátové komunikace

2.1 Popis řešeného problému

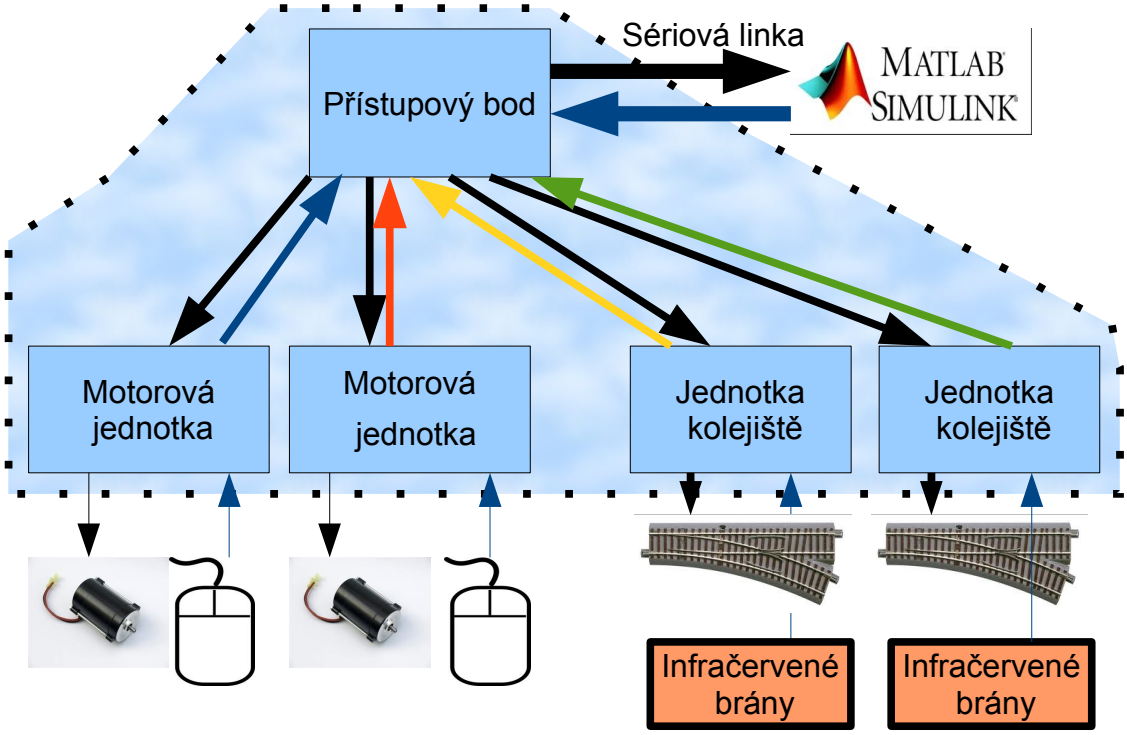
Jak již bylo řečeno, jedním z cílů práce je řešit komunikační schéma, kdy několik koncových zařízení je přímo řízeno prostřednictvím algoritmu prováděného v PC. Jedná se tedy o přenos dat s poměrně omezenou variabilitou toku informace mezi zařízeními. Samozřejmě, že běžné typy PC přímo nedisponují vhodným bezdrátovým rozhraním. Používá se proto jedno sběrné zařízení, které bezdrátově komunikuje s koncovými zařízeními. Sběrné zařízení zároveň obousměrně komunikuje s počítačem pomocí standardního rozhraní USB. Jelikož se jedná o řídicí systém, vystupují zde do popředí zejména požadavky na spolehlivost a periodu vzorkování. Jako samostatná část se ve schématu vyskytuje informace, která do regulační smyčky nevstupuje. Na tuto komunikaci jsou kladeny mírnější požadavky. V dalších úvahách však možnost rozdělení tříd přenosů uvažovaná není. V rámci minimalizace počtu přenášených zpráv, kterou zdůvodním v další části této kapitoly, používám pro přenos informace k ovládanému mechanismu kvalitativně stejný způsob přenosu jako k ostatním zařízením.



Obr 2: Blokové schéma informačních toků v aplikaci

Z pohledu fyzických zařízení se komunikační schéma skládá z bezdrátové senzorové sítě s hvězdicovou architekturou, neboť celkový počet zařízení v síti není vysoký (max 7). V bezdrátové síti je jedno sběrné zařízení (v bezdrátových sítích se užívá pojem uzel - node, zde navíc i pro nadřazené PC přístupový bod - access point), které slouží zároveň jako koordinátor sítě. Koordinátor bezdrátové senzorové sítě vlastní síť zakládá a udržuje v ní pravidla komunikace. Ostatní zařízení (používá se označení end node nebo end device), která obstarávají vlastní měření a akční zásah, jsou sběrnému zařízení podřízena. V řešeném schématu jsou podřízená zařízení 2 typů:

- pohyblivé koncové zařízení - generuje pulsně šířkový signál, odečítá inkrement ze senzoru polohy (optické myši)
- statické koncové zařízení - přepíná výhybky pro změnu vlakové cesty, odečítá stav optických závor



Obr 3: Schéma prvků bezdrátové sítě s vyznačením toku dat

Vzhledem k výjimečným vlastnostem přístupového uzlu je jeho identičnost s koordinátorem bezdrátové komunikace v síti logická. Případnou změnou parametrů přístupového uzlu lze snadno měnit parametry komunikace v síti. Navíc je v síti přítomen vždy, když jsou požadována data. Zároveň není nutné obsluhovat nadbytečného člena sítě, který by měla senzorová síť sama kvůli sobě. Pohyblivým nebo stacionárním jednotkám také není rozumné přidělit funkci koordinátoru sítě, neboť není jisté, že je jednotka s implementací funkce koordinátora sítě vždy v činnosti.

2.2 Navržené schéma komunikace

Na začátek shrňme předpoklady pro řešení přenosu užitečné informace v mnou realizované bezdrátové senzorové síti:

- izolovaná bezdrátová síť, použití pro výzkumné účely v jedné místnosti
- zajištění komunikace mezi stanoveným maximálním počtem uzlů s garantovanou dobou odezvy
- všechny datové přenosy začínají nebo končí ve sběrném uzlu, tento uzel je vždy dostupný pro podřízené uzly
- zabránění rušení mezi uzly tj. v komunikačním médiu (tj. v místnosti vysílá v každý okamžik nejvýše 1 uzel)
- z důvodu předchozího bodu využití již vytvořeného komunikačního protokolu ZigBee a alternativy v podobě SimpliciTI
- pevný formát přenášených zpráv

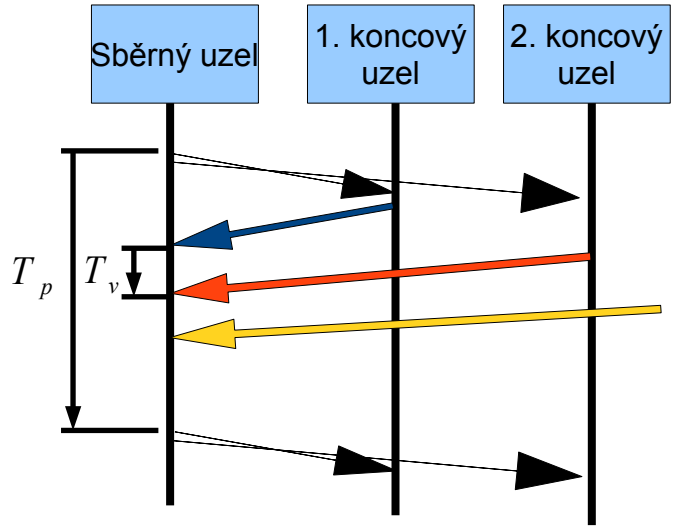
První předpoklad znamená, že není potřeba ošetřit konflikt s ostatními bezdrátovými sítěmi. Jde zejména o zabezpečení bezdrátové senzorové sítě. Nepoužívá se šifrování dat, ani není mechanismus, jak zabránit připojení cizího zařízení do vlastní sítě. Bezdrátová síť SimpliciTI ani

ZigBee tak, jak je zde použita nevyužívá své vlastní šifrování. Dokonce je zde možné, po vytvoření uzlu se stejnou adresou v síti, posílat sběrnému uzlu falešná data.

Dalším požadavkem je garantovaná doba odezvy v bezdrátové síti. Znamená to, že po odeslání informace koncovým uzlům, musí sběrný uzel dostat zpět informaci s odpovědí v časovém intervalu před dalším přenosovým cyklem. Záměrně zde píši koncovým uzlům, protože používáme-li mechanismus na zabránění kolizí při přenosu příliš často (jeho provedení trvá nezanedbatelnou dobu může se však upravit), dochází ke zbytečnému protahování přenosu dat. Pokud je v síti 6 koncových zařízení, musel by koordinátor posílat 6 zpráv každou periodu vzorkování, což by zvýšilo i obsazení přenosového média zprávami koordinátoru. Při vysílání pro všechny uzly stačí 1 jedno ověření volnosti přenosového média a odesláním 1 datového paketu. Samozřejmě obráceně je situace s požadavkem na dobu odezvy obdobná. Jestliže koncový uzel nedostane po předem stanovenou dobu informaci o nastavení svých výstupů, nastaví své akční členy do bezpečného stavu. Pevný maximální počet uzlů pak vyjadřuje zachování této vlastnosti pro všechny koncové uzly v síti, které postupně předávají změřené údaje sběrnému uzlu. Zde následuje volba pořadí komunikace se sběrným uzlem tak, aby implicitně nedocházelo ke kolizím:

- časové dělení komunikace
- návazné dělení komunikace

Při časovém dělení má každý koncový uzel předem stanovený pevný začátek svého vysílání po obdržení konfigurační nebo ještě lépe přímo zprávy ze sběrného uzlu. Časový interval mezi přijmutím zprávy a odesláním odpovědi je pro každou periodu komunikace stejný bez ohledu na to , jestli předchozí uzel s kratším intervalem skutečně vysílal. Může zde tak nastat období bez komunikace. Druhou možností je dělení podle návaznosti. Zde v ideálním případě nedochází k prodlevám mezi vysíláním uzlů. Každý koncový uzel si totiž pamatuje adresu koncového uzlu, který vysílá před ním a jakmile zjistí, že předchozí uzel vysílal, odešle svou zprávu. Tady je však situace ještě horší pokud předchozí uzel selže. Následující uzel nebude vysílat vůbec, pokud nepoužije ošetření této eventuality. Dále je zde nutnost rozumět zprávám ostatních uzlů. To znamená, že prakticky všechny uzly musejí být v každý okamžik mezi sebou dostupné. Jako lepší řešení se mi jeví časové dělení komunikace, které jsem i použil. Časové dělení se může vylepšit i předáváním aktivních uzlů. Sběrný uzel tak může ve své zprávě spolu s daty periodicky předávat informace o koncových zařízeních, která se mu v posledním cyklu ozvala nebo měla ozvat.



Obr 4: Posloupnost datové komunikace, opakuje se s periodou T_p. Koncové jednotky odpovídají v rozestupech T_v, první koncový uzel odpoví okamžitě

Následuje oddíl specifikování formátu dat. Jak již bylo řečeno, používám pouze jednu zprávu společnou pro všechny koncové jednotky. Technicky vzato je vlastní zpráva posloupností

bajtů, která dostává význam po převedení na příslušný datový typ. Zpráva sběrného uzlu, vysílaná v intervalu T_p, obsahuje v prvních čtyřech bajtech hodnotu pulsně šířkové modulace pro pohyblivé jednotky, neboli hodnotu, kterou koncová jednotka převede na interval sepnutí ovladače motorku. Další 2 bajty pak obsahují číslo kódující sepnutí cívek v přestavovacím mechanismu výhybek na jednom z míst výběru vlakové cesty. Prakticky se jedná o dvě hodnoty, které přestavují všechny výhybky do rovna nebo do zatáčky. Poslední bajt pak obsahuje bitovou mapu aktivních koncových jednotek, které by podle nadřazeného PC měly být aktivní.

1. byte	2.byte	3. byte	4. byte	5. byte	6. byte	7. byte
PWM_1	PWM_2	PWM_3	PWM_4	VYH_1	VYH_2	AKT_UZLY

Tabulka 1: Data vysílaná sběrným uzlem

U koncových jednotek obsahuje zpráva naměřená data. Vzhledem k rozdílu mezi pohyblivou a stacionární koncovou jednotkou, je význam datové části vysílaného paketu dvojznačný. Pohyblivá jednotka vysílá 16 bitovou hodnotu naměřeného inkrementu získanou ze senzoru myši za dobu T_p. Přenáší se nejprve inkrement ve směru X tj. směru pohybu do strany a poté ve směru Y tj. ve směru podélné osy optického senzoru. Poslední byte pak obsahuje identifikační číslo jednotky. Stacionární jednotka funguje obdobně s tím rozdílem, že pro přenos informace o stavu infračervených bran ji stačí pouhé 4 bity z druhého bajtu zprávy. Posledním údajem je identifikátor SENS_ID jednotky, která zprávu vyslala. Prakticky je jednotka identifikována svou adresou v síti. Nicméně pro abstrahování způsobu přidělování adres v síti jsem zvolil tuto možnost identifikace původu zprávy, ačkoliv je u každého bezdrátového zařízení v použité softwarové implementaci obou komunikačních protokolů předem jasné, jakou adresu bude jednotka v síti mít.

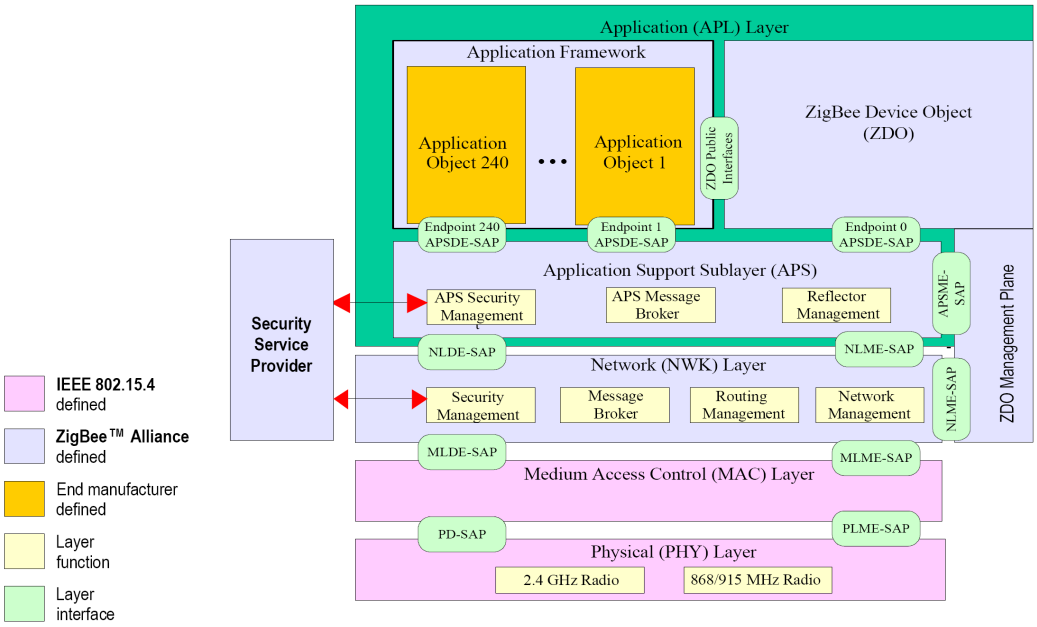
typ koncové jednotky	1. byte	2. byte	3. byte	4. byte	5. byte
Pohyblivá	INKREMENT_X		INKREMENT_Y		SENS_ID
Stacionární	0	INFRA_BITS	0		SENS_ID

Tabulka 2: Data vysílaná koncovým zařízením

2.3 Protokol ZigBee, vlastnosti, vybrané řešení

Komunikační protokol ZigBee je bezdrátový komunikační protokol určený zejména k propojení velkého množství (až 65536 členů v jedné síti) zařízení. Vhodný je pro aplikace, kde se požaduje nízká spotřeba a garantovaná doba odezvy. Jeho hlavní předností oproti obdobným standardům (zejména Bluetooth) je jednodušší implementace (zhruba 64 KB kódu oproti 720 KB u Bluetooth), dostatečný dosah (stovky metrů, oproti desítkám u BlueTooth), odolnost proti rušení a rychlé obnovení sítě po jejím restartu (30 ms). Základní nevýhodou je pak omezená datová propustnost (teoreticky 250 kb/s pro 2.4 GHz a DSSS modulaci) oproti jiným komplexnějším řešením z rodiny IEEE802.1x.

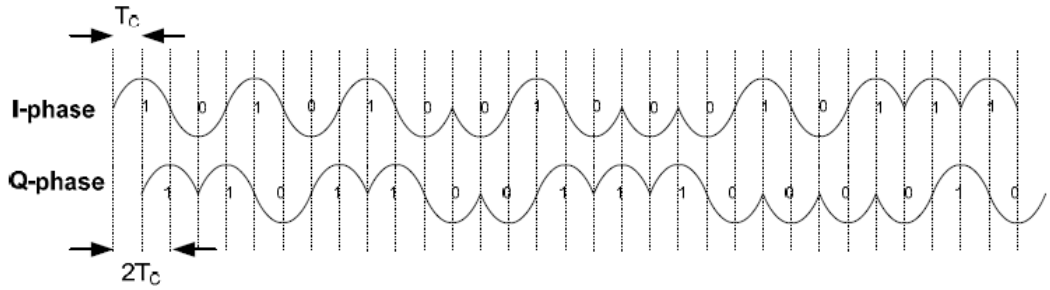
Na následujícím obrázku je ISO/OSI model protokolu z jeho podrobné specifikace [5]. Jako vlastní ZigBee by správně měla být označována část definovaná ZigBee Alliance, ale běžně se používá pro označení celého protokolu včetně části určené IEEE 802.15.4 [6], která obsahuje fyzickou a MAC vrstvu. Vyšší vrstvy jsou již definovány ZigBee Alliance. Vrstvami protokolu, jejichž funkčnost ve svém řešení nevyužívám, ale jsou neoddělitelnou součástí komunikačního protokolu, se nebudu podrobně zabývat.



Obr 5: Přehled vrstev ISO/OSI modelu ZigBee protokolu

Základní vrstvou protokolu IEEE 802.15.4 je vrstva fyzická. Tato vrstva definuje parametry elektromagnetických vln, které se používají pro komunikaci. Jsou to frekvenční pásma, způsob modulace a kódovaní přenášených dat. K dalším úkolům fyzické vrstvy patří komunikace s vrstvou řízení přístupu k médiu (medium access control - dále jen MAC) vrstvou, pro kterou vykonává následující úlohy:

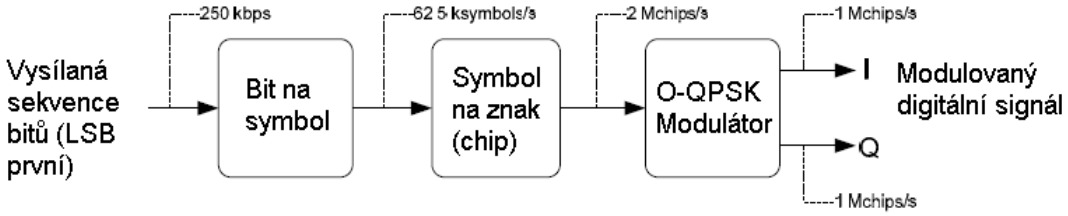
- aktivaci a deaktivaci vlastního transceiveru
- detekci síly signálu v požadovaném kanálu
- sledování kvality zpráv (link quality indication LQI)
- funkce Clear channel assessment (CCA) pro sledování nosné a zabránění kolizím (CSMA-CA)
- výběr vysílacího kanálu
- vysílání a příjem dat



Obr 6: Fáze signálu před namodulováním na nosnou frekvenci

V pásmu 2,4 GHz (fyzická vrstva definuje i jiná pásma) je definováno použití modulace O-QPSK (offset-quadrature phase shift keying) s půl-sinovým tvarováním. Jedná se o fázovou modulaci, kde je vstupní digitální signál rozdělen na sudé a liché bity. Každý bit se vytvaruje na kladnou nebo zápornou část sinusovky a s posunutím o polovinu bitu se střídavě moduluje na nosnou frekvenci. Perioda vysílání jednoho bitu T_C = 500 ns, což je frekvence 2 MHz. V pásmu 2,4 GHz je definováno celkem 15 přenosových kanálu, každý o šířce pásma 5 MHz označených čísly 11 až 26.

Frekvence nosné se vypočte podle vztahu: $f_c = 2405 + 5(k-11),\ k = 11,12,\ldots,26$. Zdrojem bitů pro vysílání je pak kódování pomocí DSSS (direct sequence spread spectrum). Každý bajt informace tj. 8 bitů se rozdělí na horní a spodní 4 bity. Tyto 4 bity tvoří jeden symbol, což je základní časová jednotka o délce trvání 16 µs. Symbol slouží dále jako klíč podle, kterého se vyhledá v tabulce jedna ze 16 sekvencí o délce 32 bitů. Tato sekvence bitů se poté zavádí do výše zmíněného modulátoru. Z obrázku je patrná maximální přenosová rychlost IEEE802.15.4 250 kbps, která je dosažitelná při trvalém vysílání. Kromě způsobu modulace je důležitou veličinou vysílání frekvence nosné vlny, na kterou se provádí modulace.



Obr 7: Způsob kódování zprávy

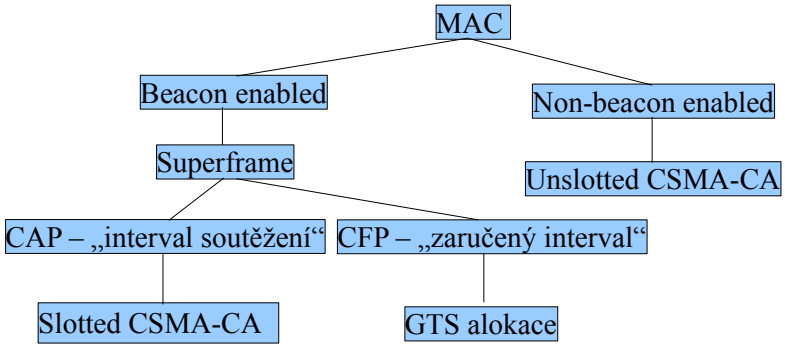
Další funkcí fyzické vrstvy, která si zaslouží vysvětlení je LQI. Je to číslo v rozsahu 0 až 255 udávající pro každý přijatý paket kvalitu signálu. Maximální kvalita odpovídá 255. V standardu IEEE802.15.4 není specifikováno, jakým způsobem se jeho hodnota získá. Je zde pouze uvedeno, že musí být zaručeno alespoň 8 hodnot. Je tak na výrobci transceiveru a programátorovi, jak provede rozmístění hodnot. Jako příklad uvádím konkrétní implementaci ZigBee v Z-Stacku s použitím transceiveru CC2520. Maximální citlivost transceiveru je -95 dBm . Praktická citlivost se však uvažuje na -85 dBm . Saturace pak obvod dosahuje pro +10 dBm . Vlastní transceiver provádí měření síly signálu RSSI přes 8 symbolů následujících po SFD, neboli startovací sekvenci rámce získaném demodulací signálu transceiverem. Samozřejmě, že se nejedná o skutečné hodnoty síly signálu v místě příjmu. Do hry vstupuje ještě vf část (referenční anténka představuje -76 dBm), která se přičte ke změřené úrovni signálu. Výsledná hodnota tedy odpovídá nule pro skutečnou úroveň signálu -85 dBm a 255 pro hodnotu +10 dBm .

Hodnota RSSI se také používá ve funkci fyzické vrstvy CCA (clear channel assesment), která zjišťuje obsazenost přenosového média. Typicky se zde opět uvažuje práh citlivosti +10 dBm nad citlivostí transceiveru. Doba trvání detekce je opět 8 symbolů tj. 128 µs. Mechanismus CCA tvoří základ pro CSMA-CA, který však již provádí MAC vrstva. K úkolům MAC vrstvy patří:

- definování dvou základních typů zařízení v síti - koordinátor sítě a koncové zařízení
- synchronizování komunikace v síti pomocí speciálních zpráv koordinátoru sítě (beacons)
- podpora přihlášení a odhlášení ze sítě
- podpora zabezpečení
- implementace CSMA-CA pro přístup ke kanálu
- Vytváření komunikační linky mezi MAC vrstvami zařízení v síti (přidělování 16 bitových adres)

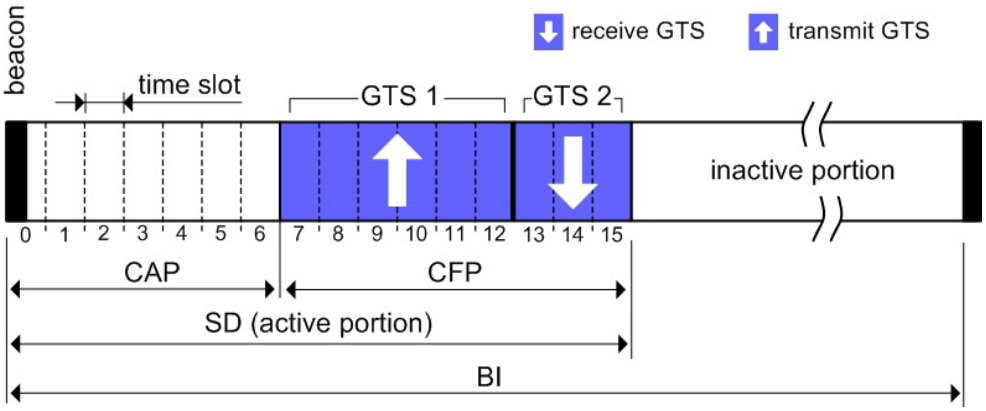
MAC vrstva slouží jako rozhraní mezi fyzickou vrstvou a vrstvou síťovou (např. definovanou ZigBee Alliance). Jedná se o klíčovou vrstvu celého standardu IEEE 802.15.4, která se v jednodušších aplikacích používá jako vrstva přímo napojená na vrstvu aplikační. I v aplikaci bezdrátového řízený motorových jednotek v podstatě tato vrstva obstarává veškerou komunikační logiku. Tato vrstva zabezpečuje fyzické spojení mezi komunikačními zařízeními v síti, neboť vyšší vrstva jí předává adresu cílového uzlu. K vytvoření spojení MAC vrstva specifikuje role

jednotlivých uzlů v síti na koordinátor sítě a koncový uzel (end device). Koordinátor je vlastním tvůrcem sítě, do které se připojují koncové uzly. Koordinátor sítě provádí fyzické řízení komunikace v síti buď jako náhodné (non-beacon enabled) nebo pravidelné (beacon enabled) pomocí vysílání svých zpráv - beacons (jednotné číslo beacon) koncovým zařízením.



Obr 8: Schéma volby druhu CSMA-CA mechanismu MAC vrstvy

Koncové zařízení rozlišuje zda pracuje v beacon enabled nebo non-beacon enabled síti podle obsahu beacon zprávy koordinátoru při žádosti o zařazení do sítě. Pokud beacon zpráva obsahuje pole s pořadím vysílání (beacon order - *BO*) rovným 15, je koncové zařízení seznámeno s tím, že mu odpověděl koordinátor non-beacon enabled sítě. Pokud má přijatá beacon *BO* menší než 15, jedná se o beacon enabled síť, kde jsou beacon rámce vysílány pravidelně. Beacon lze chápat jako startovací značku (synchronizaci) pro takzvaný super-rámec superframe, ve kterém



Obr 9:Struktura superframe v beacon-enabled síti IEEE802.15.4, obrázek převzat z [2]

jsou vymezeny časové sloty pro vysílání a příjem jednotlivých uzlů. Je to vhodné zejména při synchronním vzorkování z velkého počtu bezdrátových senzorů. Testováním a vylepšováním složitých senzorových sítí s IEEE802.15.4 se zabývají práce [2], [3]. S beacon resp. non-beacon síti je svázán přístupový mechanismus CSMA-CA, který je synchronní s beacon zprávou (slotted) resp. zcela nezávislý mezi zařízeními (unslotted). Každý superframe se skládá z periody volného soutěžení bezdrátových uzlů o přenosové médium (Contention Access Period - CAP), garantovaných časových slotů (Guaranteed Time Slots - GTS) ve vyhrazené části superframe (Contention free period - CFP) a neaktivní části. Platí, že vždy po beacon rámci následuje CAP minimální délky 440 symbolů (beacon je součástí CAP, navíc musí probíhat i jiná komunikace např. připojení nového uzlu do sítě). Máme zde tak 7,04 ms doby pro neperiodickou komunikaci,

kdy se provádí slotted CSMA-CA varianta kontroly média. Délka trvání části CFP je dána částí beacon zprávy vyhrazené pro trvání superframe (superframe duration - SD), perioda pak nastavenou hodnotou trvání beacon (beacon interval -BI):

BI = *aBaseSuperframeDuration* 2^{BO} *BO* - parametr beacon order
SD = *aBaseSuperframeDuration* 2^{SO} *SO* - parametr superframe order

kde $0 < SO < BO < 14$, *aBaseSuperframeDuration* = 15.36 ms. V síti tedy nelze vysílat superframe častěji. Do CFP lze umístit vysílání až 7 zařízení. Pro $SO = 0$ má každé zařízení pro sebe k dispozici interval 960 µs. Za tuto dobu (60 symbolů) lze uskutečnit pouze přenos několika bajtů (60 symbolů je fyzicky 30 bajtů, nicméně uživatelských dat bude kolem 10 bajtů).

U non-beacon sítě působí po celou dobu komunikace mechanismus CSMA-CA v unslotted podobě. Ta spočívá v tom, že na rozdíl od slotted podoby není mechanismus CSMA-CA synchronizován s beacon rámcem. MAC vrstva tedy předchází kolizím použitím CCA v náhodných intervalech nesynchronních s ostatními uzly v síti. Pokud je CCA neúspěšné zkusí ho znovu po náhodném intervalu (násobky backoff periody - 20 symbolů), který je dvojnásobkem předešlého. Pokud není CCA ani po 4 pokusech úspěšné, prohlásí se kanál za obsazený.

Pro moji bezdrátovou senzorovou síť nicméně používám non-beacon síť. Vlastní přenos totiž nelze použitím beacon sítě urychlit. Slouží k zavedení mechanismu pravidelného přenosu dat ve variabilní síti. Moje senzorová síť však má stále stejný maximální počet členů a jiná než datová komunikace se sběrným uzlem zde po založení sítě neprobíhá. Proto je kontraproduktivní využívat rozdělení komunikačního času na CAP a CFP. Řízení komunikace provádím ve vlastní aplikaci a MAC vrstva tak dostává již v přesném intervalu data, která má vysílat. Z vyšších vrstev definovaných ZigBee Alliance používám pouze výhody zprávy typu broadcast, která je definována v síťové vrstvě. Nejedná se však o nic jiného než, že sběrný uzel vysílá s cílovou adresou 0xFFFF, která způsobí, že zprávu přijmou všechny uzly v dosahu. Uspořádání sítě je pouze jednoúrovňové (koordinátor a koncové uzly - hvězdicová topologie sítě), takže není třeba zprávu typu broadcast přeposílat přes routery, kteří zde nejsou.

Princip použití ZigBee v mé aplikaci je tedy zřejmý. Koordinátor vysílá zprávy typu broadcast a všechny koncové uzly vysílají s cílovou adresou 0 (pevná adresa koordinátoru v síti). Koncové uzly by se vlastně ani do sítě nemusely přihlašovat, neboť cílovou adresa zařízení je neměnná. Připojení do sítě však jednak přináší výhodu kratší adresy zařízení (64 bitová IEEE adresa je nahrazena 16 bitovou krátkou adresou), a jednak by bylo zbytečně složité předělávat implementaci mezivrstev protokolu ZigBee, z kterého by se stal proprietární protokol. Příklad datového rámce vysílaného koordinátorem ukazuje následující obrázek. Modře je znázorněna část přidaná síťovou vrstvou. Žádné nové informace, které by ovlivnily příjem koncových jednotek samotné vrstvě MAC (šedě), neobsahuje. Červeně je znázorněna část přidaná aplikační vrstvou. Jsou zde informace o cílové aplikaci na koncové jednotce - číslo cílové aplikace, číslo clusteru, číslo profilu a číslo zdrojové aplikace. Opět platí, že pokud existuje pouze 1 aplikace, 1 profil a 1cluster, nejsou tyto informace k vlastnímu přenosu potřeba.

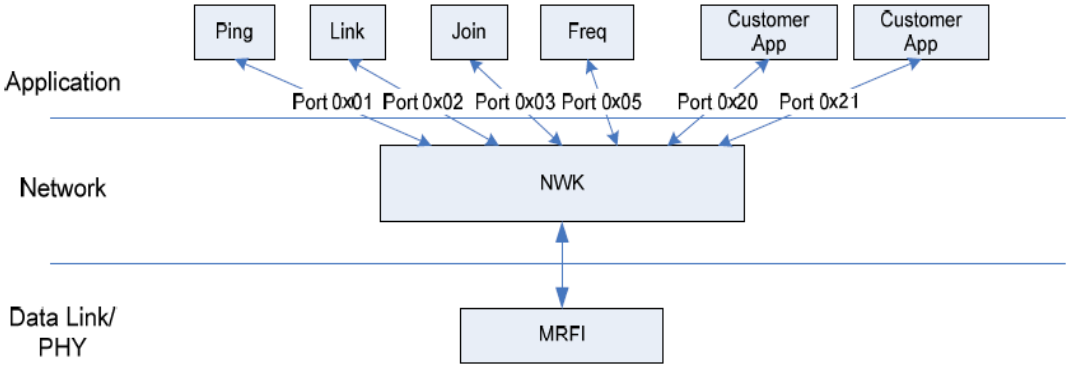
P.nbr.	Time (us)	Length	Frame control field	Sequence number	Dest. PAN	Dest. Address	Source Address	NWK Frame control field	NWK Dest. Address	NWK Src. Address	Broadcast Radius	Broadcast Seq.num
			Type Sec Pnd Ack.req PAN_compr					Type Version DR MF Sec SR DIEEE SIEEE				
RX 5	+48296 =198258	39	DATA 0 0 0 1	0xE3	0x0555	0xFFFF	0x0000	DATA 0x2 0 0 0 0 0 0	0xFFFF	0x0000	0x1E	0xE3
RX 6	+49835 =248093	39	DATA 0 0 0 1	0xE4	0x0555	0xFFFF	0x0000	DATA 0x2 0 0 0 0 0 0	0xFFFF	0x0000	0x1E	0xE4

APS Frame control field	APS Dest Endpoint	APS Cluster Id	APS Profile Id	APS Src Endpoint	APS Counter	APS Payload	LQI	FCS
Type Del.mode Ind.am Sec Ack								
Data Broadcast 0 0 0	0x0A	0x0002	0x0F04	0x0A	11	00 00 00 00 00 00 00 00 00 00 00 00	255	OK
Type Del.mode Ind.am Sec Ack								
Data Broadcast 0 0 0	0x0A	0x0002	0x0F04	0x0A	12	00 00 00 00 00 00 00 00 00 00 00 00	255	OK

Obr 10: Zachycené dvě zprávy koordinátoru, broadcast pro všechny uzly v síti, APS payload jsou vlastní užitečná data

2.4 SimpliciTI, vlastnosti, vybrané řešení

Komunikační protokol SimpliciTI je jednoduchý protokol bezdrátové komunikace pro baterií napájená zařízení, jako jsou různé snímače v budovách (detektory kouře, pohybu, teploty). Protokol byl vytvořen jako proprietární řešení komunikace pro obvody firmy Texas Istruments. Jedním z obvodů, se kterým je možné protokol provozovat, je transceiver CC2520. Při využití transceiveru CC2520 lze prohlásit, že fyzická vrstva protokolu je totožná s fyzickou vrstvou IEEE802.15.4. Nabízí tak i využití její funkce testováni obsazenosti přenosového kanálu (Clear Channel Assessment - CCA)



Obr 11: Struktura protokolu SimpliciTI

Základní částí protokolu SimpliciTI je vrstva minimálního rozhraní s transceiverem (Minimal RF interface - MRFI). Tato vrstva je ovladačem, který poskytuje jednotné rozhraní pro všechny podporované obvody. Mezi vrstvou MRFI a aplikací je síťová vrstva (NWK). Ta implementuje sadu základních příkazů pro práci aplikační vrstvy, jako je vytvoření komunikační linky mezi zařízeními a vytvoření sítě. Založení sítě však neznamená žádnou výměnu dat, jedná se čistě o konfiguraci uzlu po restartu. Síť SimpliciTI nepodporuje stromové topologie sítě. Jsou zde možné komunikace ve hvězdicové topologii, každý s každým (peer2peer) a případně komunikace přes opakovače (repeater). Aplikační vrstva SimpliciTI je vlastní aplikace napsaná přímo uživatelem. Aplikace v SimpliciTI nepracuje přímo s adresami zařízení, které jsou skryté v síťové vrstvě. Pro komunikaci využívá číslo linky, které bylo s koordinátorem sítě vytvořeno. K tomu využívá přenos zpráv s číslem linky. Na obrázku struktury SimpliciTI není zdokumentována možnost využít komunikaci přes Port 0x00. Tento port využívá přímo síťová vrstva pro všechna zařízení v dosahu (broadcastová zpráva). Vlastní rozhodnutí o příjmu uzlu do sítě pak probíhá v aplikační vrstvě zařízení typu Access point (AP, koordinátor sítě). Přihlášení koncového zařízení do sítě poskytuje výhodu v adresném určení cíle zprávy.

Formát přenosového rámce je v SimpliciTI podstatně úspornější. Užitečná informace má v přenosu velký podíl, proto se dá očekávat lepší datová prostupnost sítě oproti ZigBee.

P.nbr.	Time (us)	Length	Misc						Dest. Address	Source Address	Port	
			Type	Sec	Pnd	Ack.req	PAN_compr	Seq.num			Encryption	Number
3	+51891 =103778	23	DATA	0	0	0	0	0x4D	FF FF FF FF	78 56 34 12	NO	0x3F
4	+51894 =155672	23	DATA	0	0	0	0	0x4E	FF FF FF FF	78 56 34 12	NO	0x3F

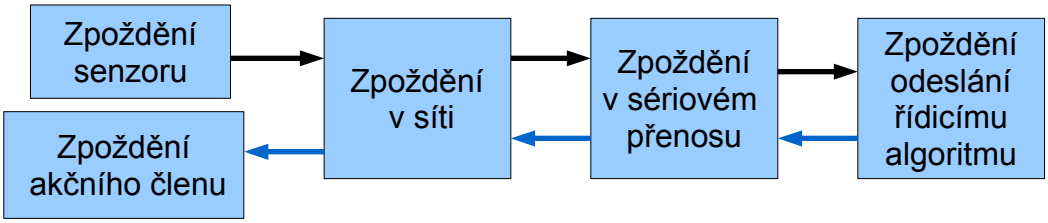
Device Info			Transaction ID	Applicaton payload	User Port	RSSI (dBm)
Rec.Type	Send.Type	HCount				
CONT_LISTEN	ACCESS_POINT	03	0xD3	01 02 03 04 05 06 00	0x3F	-25
CONT_LISTEN	ACCESS_POINT	03	0xD4	01 02 03 04 05 06 00	0x3F	-25

Obr 12: Zachycené dva pakety SimpliciTI vysílané AP, cílová adresa je typu broadcast

Shrneme-li poznatky o SimpliciTI, dojdeme k závěru, že obdobně jako protokol ZigBee nabízí broadcastové zprávy. Sběrný uzel vysílá v pravidelných intervalech zprávy všem koncovým zařízením. Ta pak odpovídají v přesně stanoveném pořadí pomocí definovaného zpoždění mezi vysíláními. Realizace komunikace tak odpovídá výstupu kapitoly 2.2.

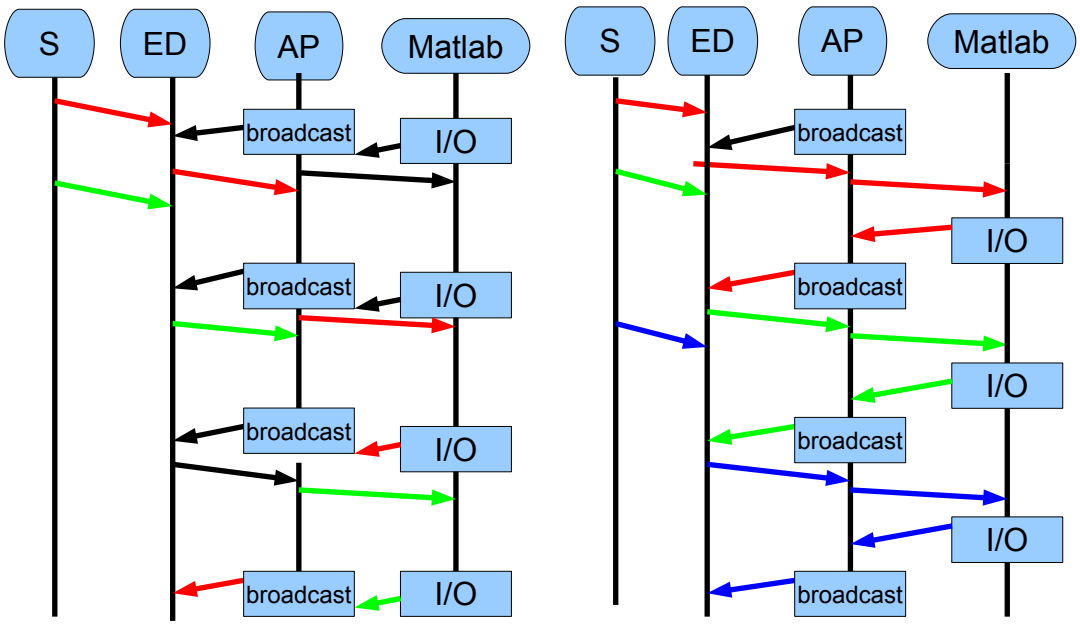
2.5 Analýza zpoždění přenosového řetězce

Důležitou vlastností přenosového řetězce je doba, za kterou se informace dostane z počátku na konec. V bezdrátové senzorové síti je to doba od zaregistrování změny měřené veličiny k předání informace nadřazenému řídicímu algoritmu. Obdobně lze definovat zpoždění z výstupu řídicího algoritmu k fyzickému vykonání akčního zásahu. Celkové zpoždění se skládá z několika částí. Zpoždění senzoru (a vyčtení informace z něj), zpoždění v bezdrátové síti, zpoždění odeslání hodnoty z přístupového bodu (sériová komunikace přes USB) a zpoždění při meziprocesní komunikaci v PC. Každé dílčí zpoždění ovšem není synchronní s ostatními. Přenos informace v bezdrátové senzorové síti probíhá v periodách, které jsou asynchronní vůči sériové komunikaci s PC a vyčítání dat ze senzoru v koncovém uzlu.



Obr 13: Zpoždění v přenosovém řetězci

Vlastní zpoždění v řídicí aplikaci je dané návrhem řídicích algoritmů řešených v rámci jiných projektů a jejich optimalizace a návrh je mimo rámec mojí práce zaměřené na zajištění komunikace a návrh hardware jednotek. Samotnou rychlost sériového přenosu lze ovlivnit pomocí návrhu sběrného zařízení. Je dána jednak vlastní přenosovou rychlostí (např 57600 Baud) a komunikačním formátem. Formát dat je vhodné zvolit jako ASCII řetězec s ukončovacím znakem. Nejdelší možný řetězec je o celkové délce 42 bajtů (6 hodnot po 6 bajtech, 5 oddělovacích znaků a ukončovací znak). Tento řetězec je při sériové komunikaci (1 start bit, 8 datových bitu, 1 paritní bit a 1 stop bit) rychlostí 57600 Baud vyslán za 7,3 ms. Jde samozřejmě o nejhorší případ, hodnoty o velikosti 6 bajtů se v praxi nevyskytují. Při čtení dat každých 50 ms může myš s rozlišením 1600 bodů na palec postoupit maximálně o ±80 bodů, což jsou v ASCII řetězci celkem 3 bajty. Vlastní řídicí algoritmu však často pracuje se svoji vzorkovací periodou. Výše uvedené zpoždění sériového přenosu se proto do celkového zpoždění nezapočítává, neboť obecně platí, že vzorkovací perioda musí být delší než doba odběru vzorku.



Obr 14: Dopravní zpoždění při přenosu informace od senzoru k algoritmu v Matlabu , vlevo je nejhorší a vpravo nejlepší případ. S - senzor, ED - koncové zařízení, AP - sběrné zařízení, Zpoždění může činit až 4 periody, pokud Má Matlab stejnou periodu jako bezdrátová komunikace

Vlastní zpoždění v bezdrátové části je dáno použitou periodou přenosu informace. Pokud činí perioda komunikace $T_p = 50\,ms$, je jisté, že AP odesílá data do PC, která byla v koncovém uzlu nanejvýš před 50 ms. Koncový uzel odesílá ihned poslední data ze snímačů. Podle typu snímače zde dochází k dalšímu zpoždění. To je patrné zejména u optické myši s PS/2 komunikací. Tato myš posílá inkrement typicky každých 10 ms a vlastní přenos jí trvá 3 ms. U inkrementálního snímače a optické závory je prodleva minimální resp. neuvažuji takové rozlišení polohy, že je nutné považovat jeden inkrement za nová data. Zpoždění akčního členu je také zanedbatelné. Pochopitelně stejné prodlevy platí i pro obrácený přenos dat. Zpoždění ve smyčce s regulátorem je pak dáno způsobem komunikace mezi AP a PC a vzorkovací periodou řídicího algoritmu. Velikost dopravního zpoždění se může teoreticky pohybovat od $2T_p < T_c < 4T_p$.

Shrneme-li získané poznatky, dospějeme k závěru, že:

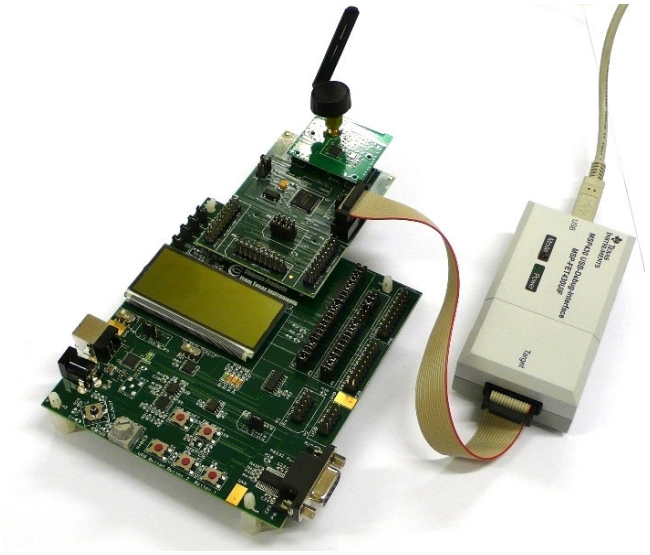
- vzorkovací perioda v aplikaci je dána přímo periodou bezdrátové komunikace T_p , která je z period přenosového řetězce nejdelší
- Dopravní zpoždění celého komunikačního řetězce ve zpětné vazbě je dáno způsobem předávání informace mezi sběrným uzlem (Access point) a nadřazeným řídicím algoritmem v PC. Činí minimálně $2T_p$. Tohoto zpoždění se dá dosáhnout synchronizací period vysílání (vzorkování) AP a Matlabu po přijmutí posledních dat z koncových zařízení v periodě. Zároveň musí Matlab načíst a zapsat výsledky do AP dřív, než AP zahájí nové vysílání.

3 Hardwarové řešení

3.1 Přehled hardwarového řešení

Úkolem této kapitoly je seznámit čtenáře s navrženým hardwarem. Zejména pak s klíčovými částmi jednotlivých typů uzlů bezdrátové senzorové sítě. Jedná se o společnou část pro všechny uzly tj. zvolený mikrokontrolér MSP430, transceiver CC2520 a jemu příslušnou vysokofrekvenční část. Následně pak typové prvky uzlu jako je např. řešení zálohového napájení a optických závor a myši. Výsledkem hardwarového řešení jsou desky plošných spojů vhodné pro univerzální zástavbu do jednoho z typů jednotek - pohyblivé, stacionární a sběrné.

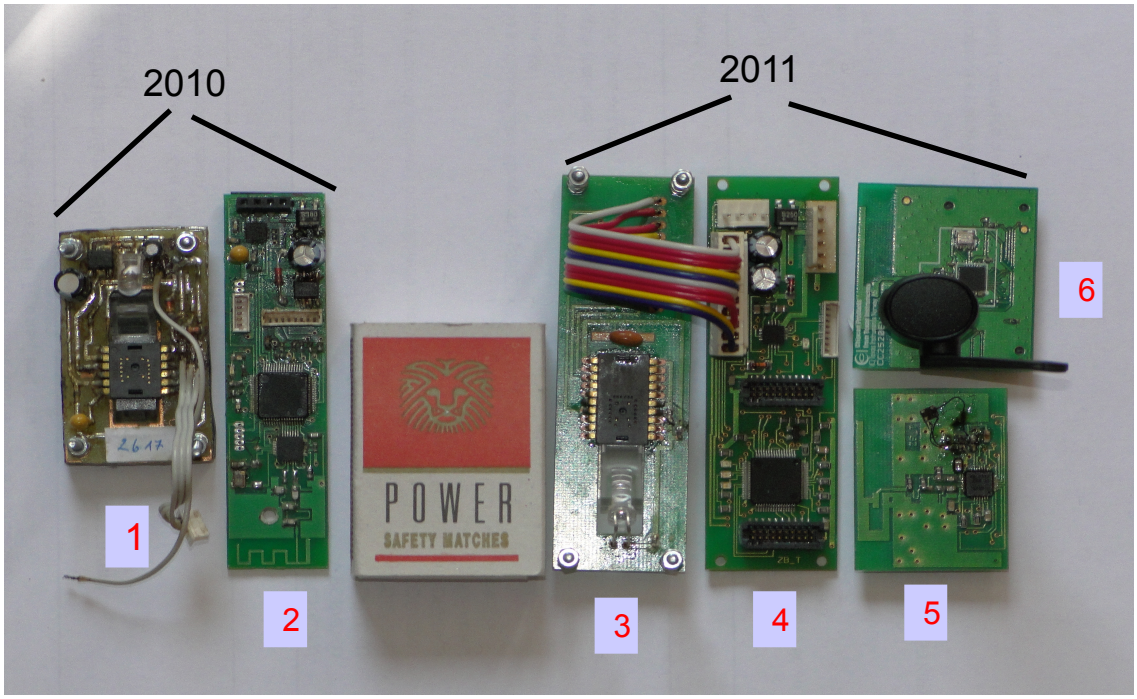
Na základě zvážení dostupných obvodových řešení (mikrokontrolér integrovaný s transceiverem Motorola MC1321x, moduly TelosB a MICAz firmy Crossbow), byl vybrán mikrokontrolér MSP430 firmy v kombinaci s rádiovým čipem CC2520. Tento mikrokontrolér zvítězil díky 16-bitové architektuře s dobře vystavěným instrukčním souborem na pomezí RISC/CISC, nízkou spotřebou, a rozsáhlými možnostmi periferních modulů. Rovněž rozhodla softwarová podpora a zkušenosti s tímto mikrokontrolérem na katedře. Obvod CC2520 je novou (v roce 2009) generací transceiverů firmy Texas Instruments. Je zde proto zájem porovnat ho se staršími obvody CC2420, které používá ve svých modulech firma Crossbow. Pro první otestování možností kombinace MSP430 + CC2520 byl zvolen vývojový kit CC2520DK, který pro účely projektu poskytlo české zastoupení firmy Texas Instruments. Na původním základu tohoto hardwaru jsem navrhl svou první desku v rámci bakalářské práce. Po otestování kitu jsem navrhl vlastní prototypové hardwarové řešení vycházející z doporučeného zapojení otestovaného na kitu. Vlastní návrh pak přidává výkonový stupeň s kompaktním obvodem H-můstku DRV8800 a rozhraní pro další použité periferie.



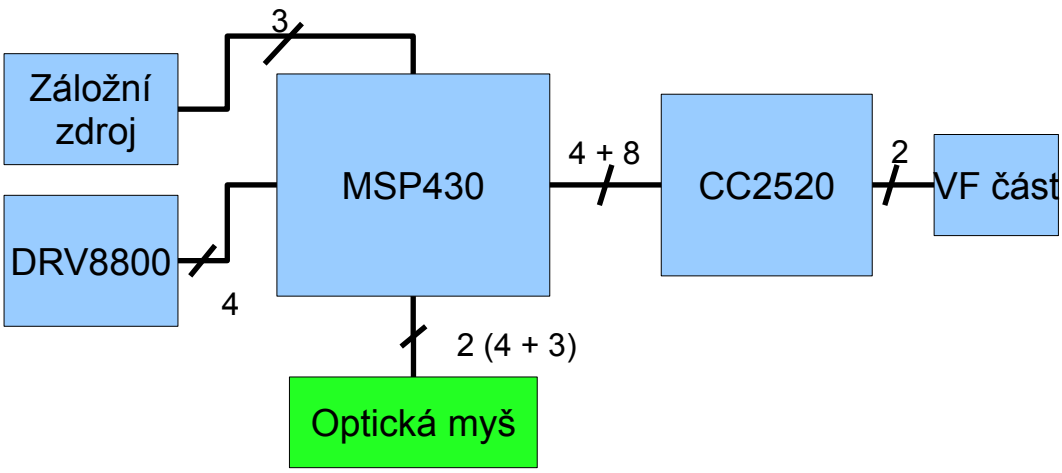
Obr 15: Sestavený modulární vývojový kit Texas Instruments CC2520DK, vlastní mikrokontrolér MSP430 je na prostřední desce, transceiver je na malé desce, vpravo je programovací rozhraní JTAG/USB MSP-FET430UIF

Konstrukce univerzálního hardwaru pro bezdrátovou komunikaci má v sobě řadu úskalí. Především je třeba vyhovět požadavkům na rozměry. Často se požaduje zástavba do malých rozměrů. Naštěstí výrobci integrovaných obvodů vymýšlejí stále nová řešení, která návrhářům desek plošných spojů usnadní práci. Důležitá je zde nejen miniaturizace obvodů pro zpracování a přenos informace, ale i výkonových prvků a propojovacích konektorů. Vývojový kit CC2520DK je příkladem i další důležité vlastnosti návrhu desek plošných spojů - vzájemné interoperability. Skládá se ze 3 samostatných modulů. Modulu periferií se zobrazovacími a ovládacími prvky, modulu mikrokontroléru a modulu transceiveru. Toto řešení sice není nejlepší pro vestavbu do motorové jednotky modelového kolejiště (alespoň ne ve stejné velikosti), poskytuje však inspiraci k dalšímu vývoji. Postupný vývoj mého hardwaru je ilustrován na následující fotce. V průběhu návrhu desek plošných spojů pro vestavěné jednotky jsem v roce 2010 navrhl desku rozměrů 20 x 80 mm (číslo 2). Tato deska je osazena konektory s roztečí 1,25 mm. Deska se dala použít i pro vestavbu do velmi úzké motorové jednotky. Nicméně manipulace s ní je problematická, nemá možnost pevného uchycení, navržené řešení vf části se ukázalo jako mezní a pro jiné účely je naprosto nevhodná. Nelze ji však upřít zásluhu na splnění požadavků aplikace (kromě zálohování napájení). Proto jsem v roce 2011 navrhl desku s modulárním přístupem (deska č.4). Je jen nepatrně větší (26 x 84 mm), její možnosti jsou však nesrovnatelně širší. Spolupracuje s profesionálním modulem (CC2520EM č. 6) nebo mnou navrženým modulem (deska č. 5) s transceiverem CC2520 a je dostatečně robustní pro zástavbu i do jiných aplikací (standardní konektory s roztečí 2,54 mm a otvory pro spojovací materiál). Velikosti profesionálního řešení modulu transceiveru jsou 35 x 39 mm. Můj modul má pak velikost 30 x 39 mm. Zároveň jsem provedl náhradu snímače myši (modul č. 1, osazen PAN3401) za laserový typ (deska č. 3 osazena PAW3601DH-NF), který vykazuje lepší kompatibilitu s použitým mikrokontrolérem MSP430. Rozměry celé modulové sestavy (č. 3, 4 a 5) jsou:

- délka 84 mm
- šířka 31,5 mm (vzdálenosti od podélné osy hlavního modulu k okrajům jsou 13 a 18,5 mm)
- celková výška od povrchu (optická soustava 2 mm od povrchu) 33 mm

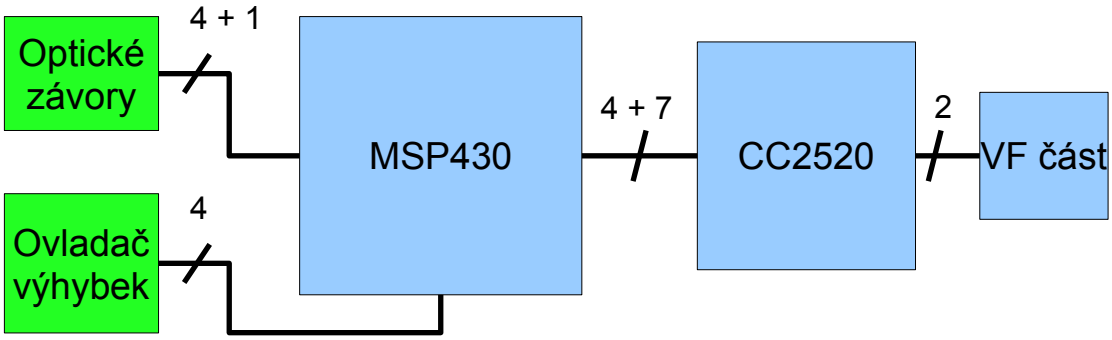


Obr 16: Navržené desky plošných spojů, 1 - modul optického senzoru PAN3401, 2 - vlastní deska model 2010, 3 - modul optického senzoru PAW3601, 4 - hlavní deska model 2011, 5 - modul s transceiverem, 6 - modul od TI CC2520EM



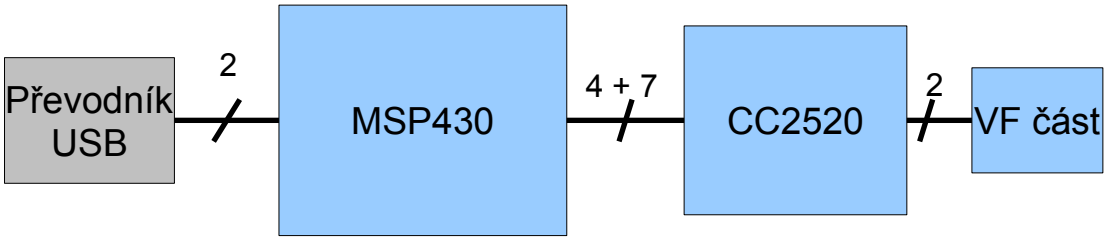
Obr 17: Blokové schéma bezdrátového uzlu pohyblivé jednotky

Blokové schéma ilustrovaného hardwaru ve funkci pohyblivé jednotky je na výše uvedeném obrázku. Mikrokontrolér MSP430 komunikuje s transceiverem CC2520 převážně pomocí čtyřvodičového SPI (serial peripheral interface). Navíc využívá dalších 6 programovatelných vstupů/výstupů transceiveru pro doplňkovou komunikaci. Zbývající 2 vodiče slouží k resetu a odpojení napájení transceiveru. Sám transceiver je diferenciálním výstupem připojen k vysílací části desky plošných spojů. Ta se skládá z tištěné antény pro pásmo 2.4 GHz a přizpůsobovacího členu, který provádí impedanční přizpůsobení anténky a transceiveru. Na levé straně obrázku je část specificky využívaná u pohyblivé jednotky. Záložní zdroj zde představuje vypínatelný zdroj proudu sloužící k napájení baterie. Součástí tohoto zdroje je rovněž detekce vstupního napětí (napětí kolejiště) a napětí sady záložních akumulátorů. Dalším prvkem obsluhovaným mikrokontrolérem integrovaný H-můstek DRV8800. Tento výkonový obvod převádí signál směru a pulsně šířkové modulace z mikrokontroléru na napětí na svorkách hnacího motoru jednotky. Jako doplňkové signály jsou využity i signalizace chyby (přehřátí obvodu, zkrat na výstupu) a vstup pro úsporný režim obvodu. Posledním prvkem pohyblivé jednotky je optická myš umístěná na samostatné desce. K propojení myši s mikrokontrolérem je potřeba podle typu buď 2 nebo 4 (pro plnou kontrolu 7) signálů. V případě PS/2 komunikace s obvodem PAN3401 postačí právě 2 signály pro data a hodiny (oboje je generováno myší). Pro obvod PAW3601DH-NF jsou potřeba minimálně 4 signály. Jedná se o kvadraturní inkrementální výstupy myší pro oba směry pohybu. Doplňkovými signály pak jsou data a hodiny pro sériovou komunikaci s myší (podobné PS/2) a signál pro úsporný režim myši.



Obr 18: Blokové schéma stacionární jednotky

V případě stacionární jednotky zůstává kombinace mikrokontroléru a transceiveru stejná. Navíc je zde propojení s infračervenými závorami. Každá infračervená závora představuje jeden signál vstupující do mikrokontroléru. Jediným výstupem mikrokontroléru pro tento modul je obdélníkový signál (36 kHz) sloužící k buzení infračervených LED. Ovladač výhybek, což není nic jiného než čtveřice výkonových tranzistorů, spíná celkem 8 cívek přestavníků 4 výhybek.



Obr 19: Blokové schéma sběrné jednotky (také koordinátor sítě, AP - Access point)

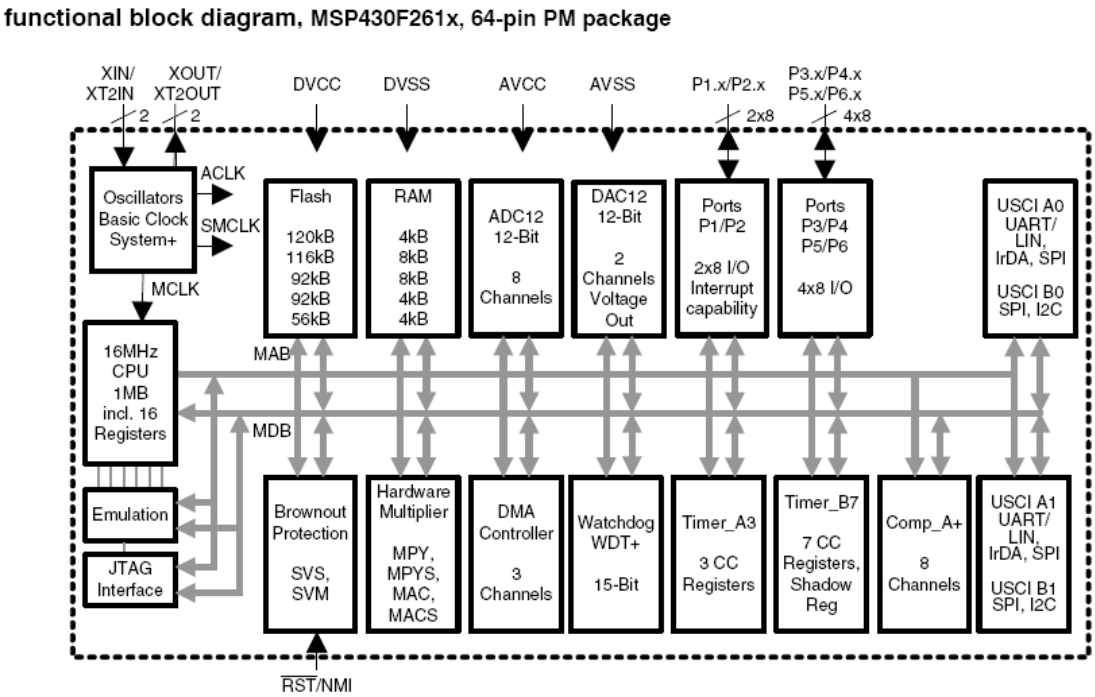
Poslední možností použití hardwaru je ve funkci sběrného uzlu. Využívá se UART modulu mikrokontroléru (Universal asynchronous receiver/transmitter). Jeden vodič pro příjem a jeden pro vysílání tvoří sériovou linku, která je napojena na externí převodník na signál USB. Tento převodník není součástí mnou vytvořeného hardwaru.

Nyní již následují sekce s popisem klíčových obvodů, kompletní schéma zapojení modulárních desek (nové desky 2011) je v příloze B.

3.2 Mikrokontrolér MSP430F2618

Mikrokontrolér [9] je výrobkem firmy Texas Instruments cíleným na jednoduché aplikace, kde se hlavní důraz klade na nízkou spotřebu energie. Pochází z rodiny mikrokontroléru, která je postavena na 16 bitovém RISC procesoru MSP430X s adresní sběrnicí rozšířenou na 20 bitů. V uvažovaném zapojení je použit mikrokontrolér MSP430F2618, který je použit i ve vývojovém kitu CC2520DK. Je ovšem možné použít i obdobné typy MSP430F2617, MSP430F2418 a MSP430F2417. Typy s 2x17 se liší menší flash pamětí (92 kB místo 116 kB). Typy 241x se liší absencí DMA a DAC převodníku. Společné je pro všechny 4 mikrokontroléry to, že mají ze všech MSP430x2xxx, které Texas Instruments vyrábí, největší paměť RAM (8 kB). Mikrokontroléry jsou vybaveny celou řadou modulů, pro činnost v uvažovaném zapojení lze vyzdvihnout vlastnosti:

- 2 x 16b čítače/časovače s celkem 10 záchytnými/porovnávacími (capture/compare) registry (čítač A obsahuje 3 registry, čítač B pak 7 registrů)
- 2 moduly univerzální sériové komunikace (SPI, I2C, UART)
- 12 bitový A/D převodník s 8 vstupy, multiplexovaný
- celkem 16 vstupů se schopností vyvolat přerušení (po 1 vektoru přerušení od portu P1 a P2)
- celkem 3 možné zdroje hodinového signálu včetně interního oscilátoru (až 16 MHz)
- rozsah napájecího napětí 1.8 V až 3.6 V
- rozsáhle možnosti řízení spotřeby mikrokontroléru, má celkem 4 módy činnosti
- pouzdro LQFP64



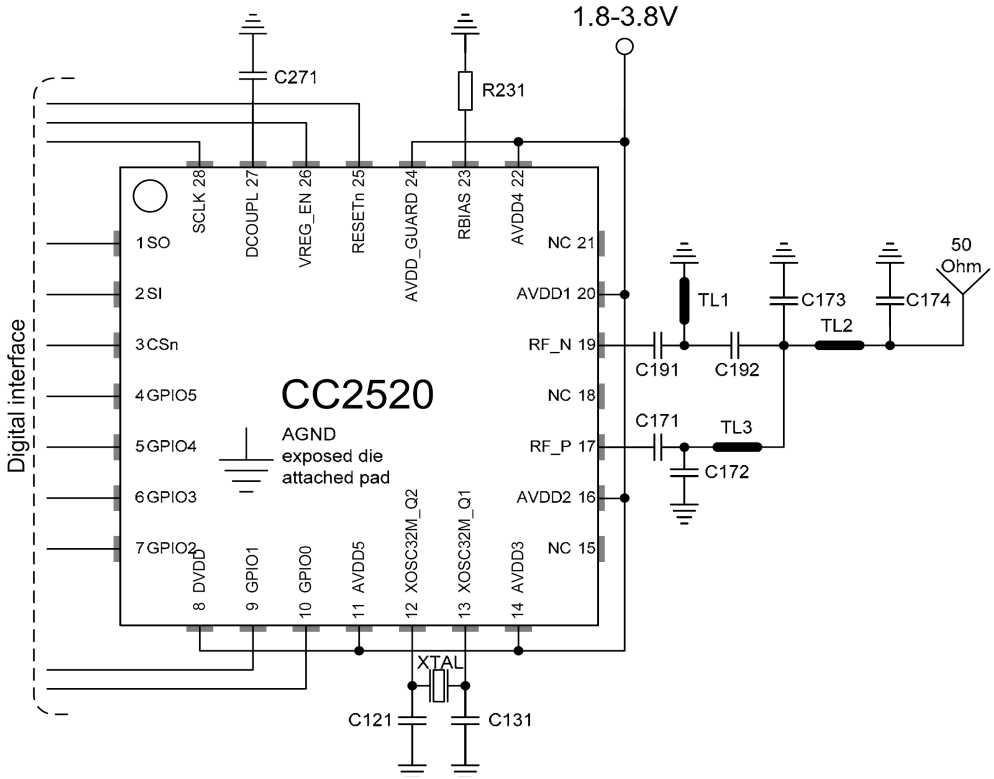
Obr 20: Blokové schéma mikrokontroléru řady MSP430F261x, převzato z [9]

K popisu mikrokontroléru z hlediska programátora je určen průvodce programátora [10]. Obsahuje podrobný popis práce se všemi moduly MSP430, které lze v mikrokontroléru této řady nalézt. Co se týče CPU, tak je popsán pod částí MSP430X (extended – 20 bitová adresní sběrnice). Mnohé popisy jsou však „device specific“, takže pro popis konkrétních přerušovacích vektorů a konfiguračních registrů periférií je nutné mít při ruce datasheet [9].

3.3 Transceiver CC2520

CC2520 [8] je 2.4 GHz IEEE 802.15.4 kompatibilní transceiver od firmy Texas Instruments 2. generace (první generací je myšlen obvod CC2420). Mezi jeho hlavní přednosti patří:

- rozsáhlá podpora IEEE 802.15.4 rámce:
  - úplná podpora fyzické vrstvy
  - podpora MAC vrstvy – detekce a filtrování podle FCF (frame control field) a adresy, automatická kontrola podle FCS (frame check sequence), automatické posílání potvrzovacích rámců, CSMA-CA, AES-128 kódování (bez dekódování) – vše volitelné
- regulovatelný vysílací výkon až + 5 dBm, citlivost až -98 dBm
- základní komunikace s MCU pomocí 4 vodičového SPI + 6 dalších volitelných GPIO
- vyhrazená paměť pro vysílaný a přijímaný rámec spolu s maskou adres (celkem 768 B)
- rozsáhlý systém výjimek pro zachytávání významných událostí při činnosti obvodu
- maximální spotřeba 34 mA, pod 1 µA v úsporném režimu
- přídavné funkce - hodinový výstup a teplotní senzor
- pouzdro VQFN28 (5x5 mm)



Obr 21: Referenční zapojení transceiveru CC2520, převzato z [8]

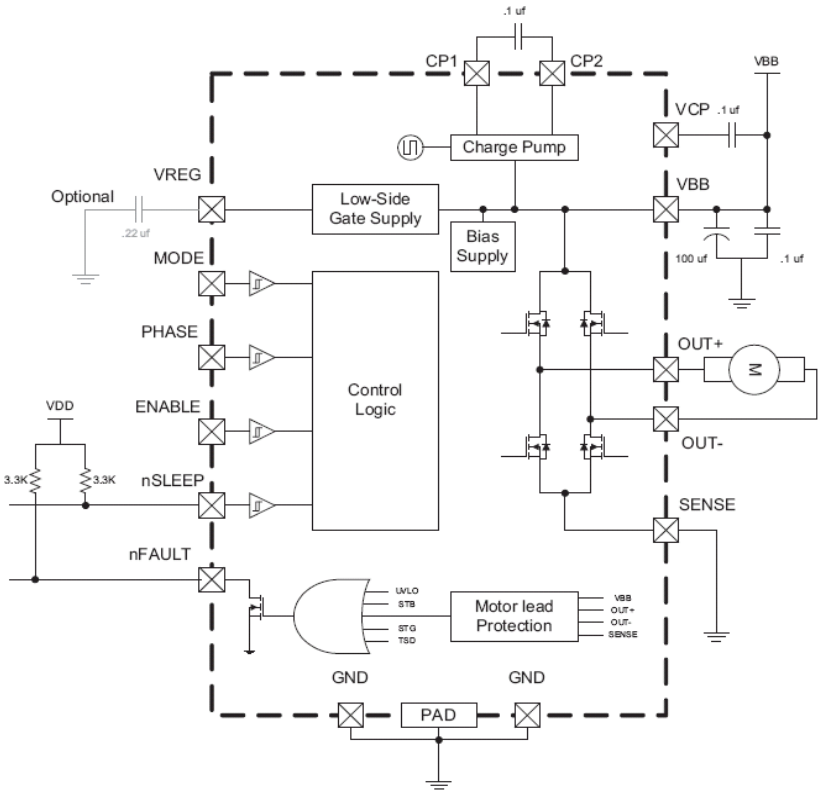
Pro správnou činnost obvodu je v první řadě nutný krystal či oscilátor (připojen na XOSC32M_Q1) o frekvenci 32 MHz. Nejběžněji dostupný typ nevyhoví, neboť osciluje na 3. harmonické, proto je třeba zvolit typ pracující ve fundamentálním oscilačním módu. Navržený hardware obsahuje SMD krystalový oscilátor, takže tento problém není aktuální. Další kritickou součástkou je R231 na vývodu RBIAS, který musí být 56 kΩ s tolerancí 1 %, neboť slouží k regulaci proudu v analogové části transceiveru. Na místě kondenzátoru C271 vyhoví běžný keramický 100 nF. Pasivní prvky připojené na diferenciální výstupy vysílací části rádia (RF_N a RF_P) nejsou povinné, nicméně ve většině zapojení nutné. Tvoří součást vysokofrekvenčního obvodu, o kterém pojednává samostatný oddíl o návrhu anténky.

Skupina pinů označená Digital interface slouží k propojení s mikrokontrolérem MSP430. Základ komunikace tvoří čtyřvodičové SPI. Rádio je při komunikaci vždy podřízená jednotka, na straně mikrokontroléru tvoří výstup MO (master output) a vstup MI (master input), u rádia pak vstup SI (slave input) a výstup SO (slave output). Mikrokontrolér vysílá SPI hodiny do vstupu (maximálně 8 MHz) SCLK a aktivuje komunikaci signálem do vstupu CSN. Propojení GPIO s mikrokontrolérem nejsou teoreticky nutná, avšak významně ulehčují jeho činnost, neboť se s jejich pomocí dají přímo vyvolat přerušení např. při přijmutí platného rámce do paměti rádia. Zvláště výhodné je zapojení GPIO0, jelikož po resetu rádia poskytuje hodinový signál 1 MHz, což ušetří krystal pro mikrokontrolér. Vstup VREG_EN slouží k přechodu do „ hlubokého spánku“ (LPM2 - odběr 30 nA, jedná se fakticky o vypnutí celého obvodu). Pokud je VREG_EN připojen na napájení, rádio zůstává stále aktivní a lze použít mód „lehkého spánku“ (LPM1 – odběr 0.2 mA). Vstup RESETn slouží k resetu rádia. Opět při trvalém napojení na napájení zůstává rádio stále v činnosti. I tak je však možno provést ekvivalentní reset pomocí dvou SPI příkazů.

3.4 Ovladač motorku DRV8800

DRV8800 [13] je DMOS integrovaný můstkový ovladač DC motorků. Hodí se pro spínání stejnosměrného napětí až 36 V při proudu ±2.8 A. Zahrnuje v sobě klasický tranzistorový H-můstek s měřením protékajícího proudu, spořící a ochranné funkce se signalizací. To vše v pouzdru QFN16 (půdorys pouze 4x4 mm). Rozhraní ovladače je následující:

- PHASE — určuje směr toku proudu (ovládá polaritu výstupů OUT+ a OUT-)
- ENABLE — sepnutí tranzistorů ve větvi můstku určené signálem PHASE
- MODE — způsob brzdění motoru, je na výběr z rychlého brzdění (protiproudem končí při poklesu proudu k nule) a pomalého brzdění (volná cirkulace proudu přes otevřenou větev a ochranou diodu v zavřené větvi)
- nSLEEP — odpojení výkonové části, spotřeba klesne na 10 µA
- SENSE — napětí na tomto výstupu je do velikosti 500 mV úměrné proudu zátěží, zatěžovací odpor tohoto výstupu musí být dostatečně malý tak, aby nedošlo k překročení 500 mV, obvod se jinak vypíná
- nFAULT — signalizující poruchu činnosti. Jedná se o zkratování výstupu, přehřátí obvodu nebo pokles výkonového napětí pod 8 V



Obr 22: Blokové schéma ovladače DC motorku DRV8800, převzato z [13]

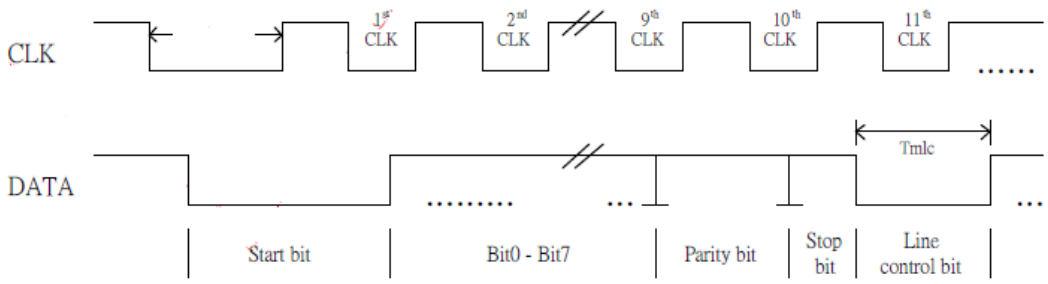
Měření proudu připojeným motorem jsem vyhodnotil jako nadbytečné, proto je výstup SENSE uzemněn. Chlazení obvodu jsem vzhledem k jeho předimenzování pro účely pohonu motorové jednotky modelového kolejiště zanedbal. Odpor obvodu v sepnutém stavu je totiž řádově 0,5 Ω, což při maximálním proudu motorku 0,2 A představuje zanedbatelný výkon 20 mW. S dynamickými ztrátami při spínání tranzistorů můstku bude ztrátový výkon sice vyšší, nicméně stále v řádu stovek miliwattů.

3.5 Optická myš PAN3401 (PAW3402)

Jako první jsem ověřil snímání polohy pomocí optické myši. Obvod PAN3401 [20] je kompletní integrovaný obvod zajišťující funkci třítlačítkové myši s kolečkem (vyskytuje se v běžně dostupných PS/2 myších Genius NetScroll100). Obvod samotného snímače je CMOS optický systém v upraveném pouzdru DIP12 snímající obrazy pojížděcí plochy, který matematicky porovnává a určuje tak velikost posunu. Rozlišení je až 1000 bodů na palec při vzorkovací frekvenci 100 Hz. K činnosti myši je navíc potřeba pouze přisvětlovací červená ovládaná přímo snímacím obvodem LED. Komunikační rozhraní myši obsahuje 2 signály - data a hodiny. Logické úrovně jsou zde bohužel posunuté na rozsah 0 až 5 V. Výhodou ovšem je to, že úroveň log. 1 je uvnitř obvodu realizována pull-up odporem o velikosti 5 kΩ. Pokud chceme dosáhnout kompatibility s mikrokontrolérem MSP430 pracujícím pří napájecím napětí 3,3 V, je potřeba omezit velikost napětí právě na tuto hodnotu. Úroveň 3,3 V totiž obvodu myši nevadí a spolehlivě ji rozliší. Jednou z možností, jak dosáhnout snížení napětí je použití Zenerovy diody na 3,3 V, jinou pak zatížení odporem o velikosti kolem 10 kΩ, kdy dostaneme napětí 3,33 V.

K popisu vlastního formátu komunikace:

- Hodinový signál generuje vždy myš. Jeho frekvence není pevně stanovena, pohybuje se v řádu 10 kHz. Datový signál je platný při spádové hraně hodinového signálu
- Datový přenos je klasický sériový - 1 start bit, 8 datových bitů, paritní bit a stop bit
- Standardně probíhá přenos z obvodu myši k mikrokontroléru. Pokud chce mikrokontrolér uskutečnit přenos do obvodu myši, musí mu uzemnit hodinový signál nejméně po dobu 100 mikrosekund.
- Pak mikrokontrolér předá hodinový signál zpět myši a nastaví start bit na datovém vodiči. Myš pokračuje ovládáním hodinového signálu a po každé jeho náběžné hraně očekává platný bit na datovém vodiči
- Komunikace se ukončí posledním bitem po stop bitu tzv. line control bit

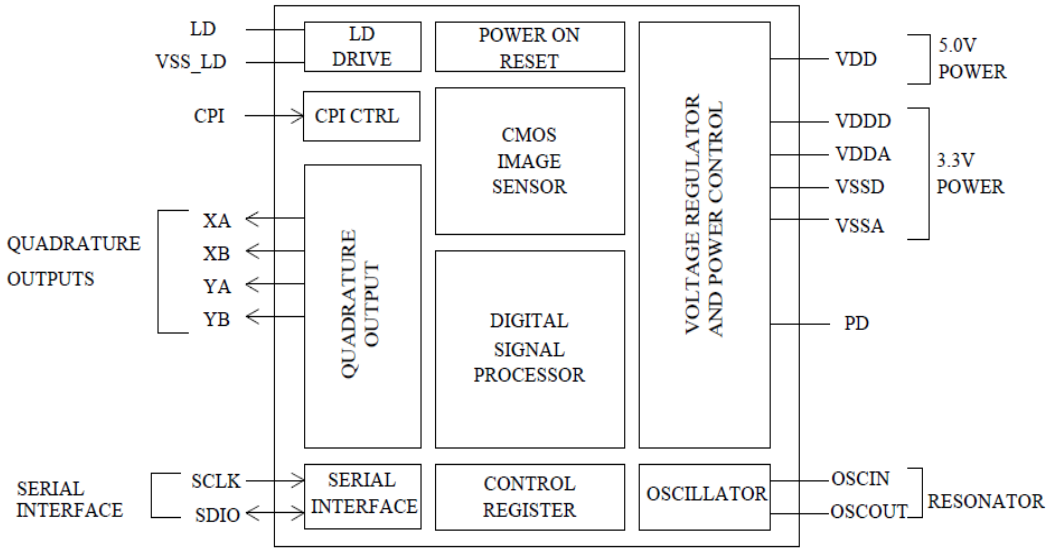


Obr 23: PS/2 komunikace od mikrokontroléru k obvodu optického snímače myši, převzato z [20]

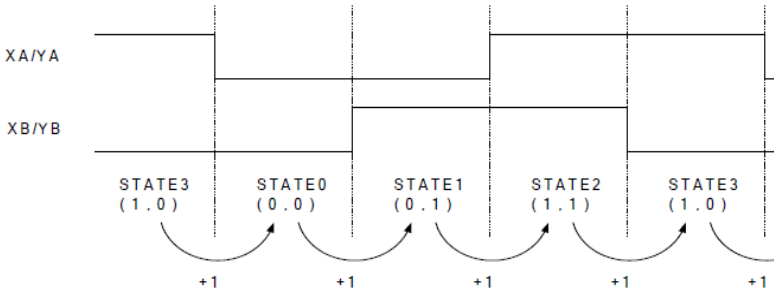
3.6 Laserová myš PAW3601DH-NF

Laserová myš s obvodem PAW3601DH-NF [19] se vyskytuje v myších typu Genius Netscroll200. Jedná se opět o optický snímač pohybu s CMOS snímačem, který na základě matematického zpracování obrazů povrchu určuje změnu polohy. Tento obvod ačkoliv je fyzicky větší a obsahuje více vývodů (upravený DIP20), obsahuje pouze samotný optický snímač. Jeho hlavní výhodou je možnost volby napájení senzoru 3,3 nebo 5 V. Dále jsou zde simulované

kvadraturní inkrementální výstupy, které na základě zvoleného rozlišení předávají mikrokontroléru přímo údaj o pohybu.



Obr 24: Blokové schéma senzoru PAW3601DH-NF, převzato z [19]



Obr 25: Kvadraturní inkrementální výstupy, převzato z [19]

Doplňkovým způsobem komunikace je sériové rozhraní SCLK a SDIO. Pomocí něho lze také vyčítat polohu přímo z registrů obvodu. Sériovou komunikaci zde v kontroluje mikrokontrolér. Po 8 adresních bitech následuje 8 datových bitů. Při zápisu do registru je na SDIO zapisuje všechny mikrokontrolér. Při čtení po zapsání adresy přepne mikrokontrolér svůj datový vodič do stavu vysoké impedance. Obvod senzoru následně zapisuje na SDIO svých 8 bitů dat z příslušného registru. Pro synchronizaci a vypnutí myši se pak hodí vstup PD. Připojením tohoto pinu k napájení přejde obvod do vypnutého stavu (odběr typicky 100 µA). Posledním zajímavým pinem je CPI. Zapojením tohoto vstupu se ovlivňuje rozlišovací schopnost obvodu 800 nebo 1600 bodů na palec. Uzemněním se získá vyšší z obou hodnot.

Externí součástky nutné k funkci obvodu jsou zde celkem 3 - laserová osvětlovací dioda s příslušným odporem a krystal. Myš NetScroll200 obsahuje krystal 27 MHz. Zajímavá je přesná hodnota odporu použitá s laserovou diodou PNDR-00004 (přesný název funkce je "Vertical-cavity surface-emitting laser"), udávající výkon diody. Velikosti odporu, 270 Ω odpovídá proud diodou 5,5 mA a vyzařovaný výkon kolem 0,4 mW.

3.7 Návrh zálohovaní napájení

Jedním z požadavků na konečnou verzi bezdrátového uzlu (model 2011) pro činnost jako pohyblivé jednotky je možnost zálohového napájení. Zálohový systém se skládá z těchto hlavních částí:

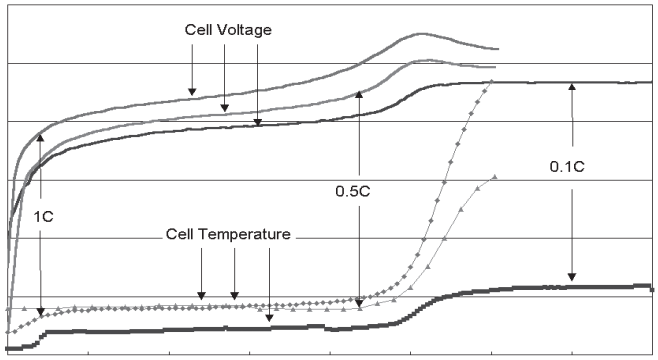
- zásobník energie (sada akumulátorových článků)
- obvod nabíjení akumulátorů
- odpojovač spotřebičů od zásobníku energie

V podrobném zadání byl zásobník energie již specifikován - 10 akumulátorových článků v sérii typu NiMh (Nikl-Metal-hydrid), každý o jmenovité kapacitě 300 mAh. Sada těchto akumulátorových článků by měla zajistit řádově hodinový provoz pohyblivé jednotky bez vnějšího napájení. Pro instalaci na plošinový vůz k pohyblivému koncovému uzlu je vhodný typ 1/3 AA (třetina tužkové baterie) o průměru 14,3 a délce 17 mm. Z 10 článků lze poskládat velké množství tvarů vhodných k obsazení volného místa. K spojování článků do série se používá poniklovaný plech.

Další součástí zálohového napájení je obvod nabíjení akumulátorů. Pro návrh nabíjení série akumulátorů, které pracují jako záloha, je nutné zvolit řadu možností. Prvním je způsob nabíjení:

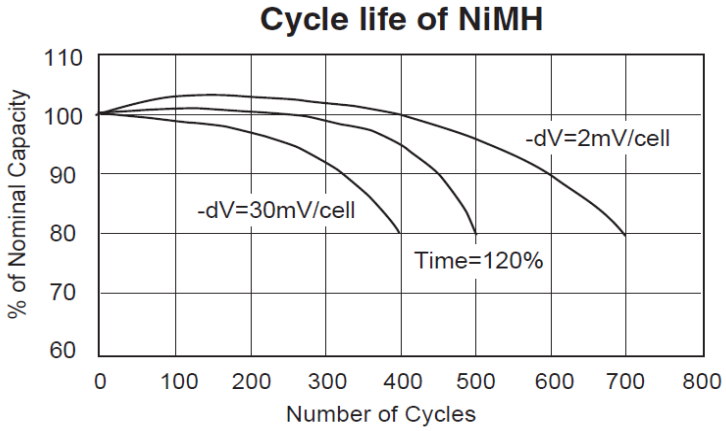
- konstantním proudem s odměřováním času
- trvalé nabíjení malým proudem
- u niklových akumulátorů existují dva speciální případy rychlo-nabíjení
    - nabíjení do poklesu napětí tj. -dV/dt
    - nabíjení do vzrůstu teploty tj. dT/dt

Akumulátory typu NiMh se nejjednodušeji nabíjejí konstantním proudem po přesně stanovený čas. Nejčastější je případ nabíjení proudem o velikosti desetiny kapacity (0,1C) po dobu přibližně 14 hodin. Podstatou funkce nabíjení konstantním proudem jsou nicméně vybité články na počátku odměřování času. Uvádí se [22], že NiMh akumulátory snesou bez problému trvalé přebíjení proudem 0,05 C po dobu až jednoho roku a proudem 0,1 C v řádu dní, takže tato nevýhoda je v podstatě nevýznamná. Skutečnou nevýhodou nabíjení konstantním proudem je tak jeho doba, kdy se akumulátory vybíjí řádově za 1 hodinu proudem 1 C a znovu nabity budou až příští den. Naopak nabíjení na pokles napětí, neboli jeho zápornou derivaci, případně na vzrůst teploty se používá při nabíjení proudem 1 C. Jakmile dojde k plnému nabití akumulátoru, vznikne vlivem zvýšení tlaku přebytečného kyslíku teplo. Se vzrůstem teploty, který lze sám o sobě považovat za signál pro ukončení nabíjení, klesá napětí na vývodech akumulátoru. Konečné nabíjecí napětí (maximální dosažené napětí) je 1,62 V na článek, z vlastní zkušenosti však mohu říci, že je kolem 1,55 V, neboť při nabíjení série 10 článku se po 14 hodinách proudem 0,1 C prakticky ustálí na této hodnotě.



Obr 26: Vliv nabíjecího proudu na nabíjecí křivku NiMh akumulátoru, převzato z [22]

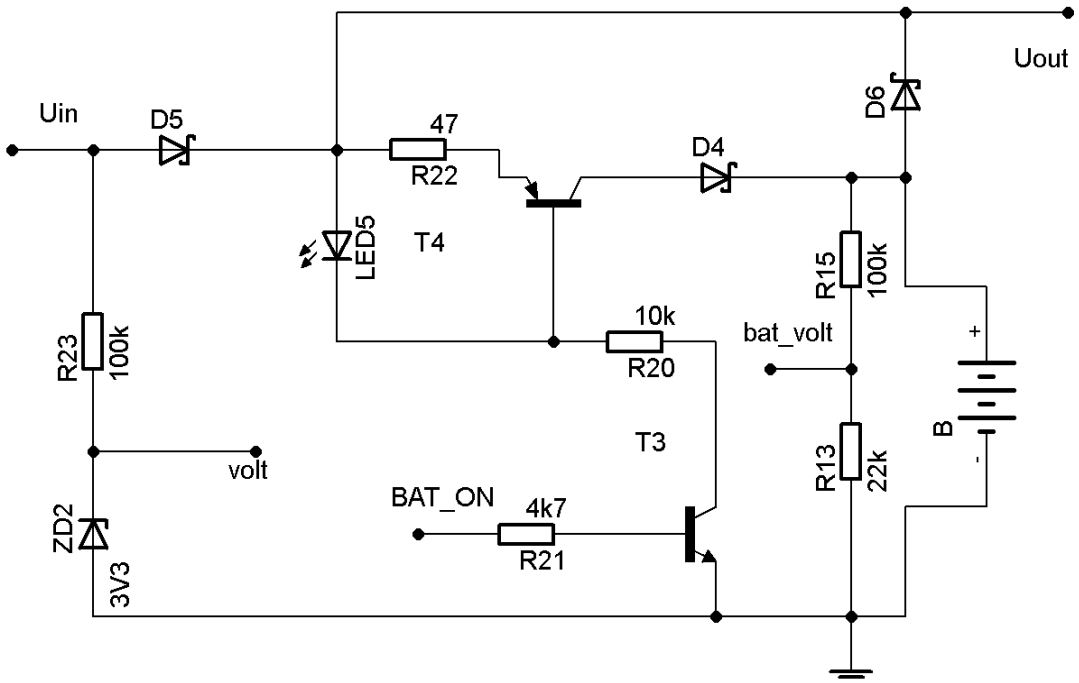
Jako porovnání možností nabíjení uvádím ze stejné publikace porovnání počtu vybíjecích cyklů akumulátoru při různém způsobu jeho nabíjení. Nabíjení konstantním proudem po 0.2 C po dobu 6 hodin (tj. 120 % kapacity článku) vede na praktickou životnost kolem 500 nabíjecích cyklů.



Obr 27: Porovnání životnosti NiMh akumulátoru podle způsobu nabíjení, převzato z [22]

Pro nabíjení v mém obvodu jsem zvolil nabíjení konstantním proudem. Tento proud vytváří proudový zdroj vytvořený z tranzistoru T4. Pokud sepne tranzistor T3, protéká diodou LED5 proud, vznikne na ni úbytek napětí 1,7 až 2 V. Tento úbytek se rozdělí také na napětí báze - emitor tranzistoru T4 a odpor R22. Vzhledem k tomu, že úbytek napětí na R22 musel způsobit proud který vytéká z kolektoru T4, dostaneme pro jeho velikost:

$I_c = \frac{U_{R22}}{R_{22}} = \frac{U_{LED5} - U_{BE}}{R_{22}} = \frac{1{,}8 - 0{,}7}{47} \approx 23\ mA$ tj. nabíjecí proud 0,08C. Tento proud by měl při napájecím napětí Uin kolem 18 V poskytovat až do konečného nabíjecího napětí série článků 15,5 V.



Obr 28: Schéma nabíječe akumulátoru, Uin je napájení z kolejiště, Uout napájí spotřebiče

Další součástí zálohové zdroje je odpojovač spotřebičů, využije se v okamžiku, kdy jsou akumulátory vybité a nelze z nich získat žádnou energii. Akumulátory typu NiMh se totiž nesmějí nechat vybít pod konečné vybíjecí napětí. Podle výrobce se toto napětí udává na úrovni 0,9 V až 1 V na článek. Sériové spojení akumulátorů má nevýhodu v neznalosti tohoto napětí, neboť jej neměříme. Měříme pouze napětí celé série článků. Články jsou jako nové stejné, postupným stárnutím se jejich napětí rozcházejí. Proto k zabránění hlubokého vybití některého z článků se uvažuje v literatuře [22] jako konečné vybíjecí napětí pro sérii více jak 8 článků napětí 1,2 V na článek vynásobeno počtem článků zmenšeným o jeden. V případě 10 článkového akumulátoru je činí velikost konečného vybíjecího napětí 10,8 V. Napěťový dělič R13, R15 toto napětí zmenší pro A/D převod na hodnotu:

$$U_{vybP} = U_{vyb} \frac{R13}{R13+R15} = 10{,}8 \frac{22}{122} = 1{,}95\ V$$

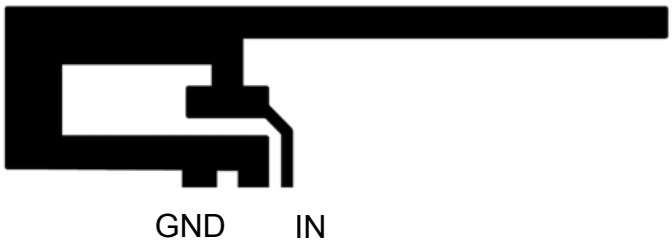
Když známe konečné vybíjecí napětí a dojde k poklesu pod jeho hodnotu, měla by se všechna zařízení od akumulátorů odpojit. Všechny integrované obvody firmy Texas Instruments se dají přepnout do úsporného režimu. Rovněž spotřeba laserové myši se dá snížit na úroveň 100 µA. Největším odběratelem energie zůstanou stabilizátor pro vytváření napětí 3,3 V a krystalový oscilátor taktující transceiver CC2520. Prvně jmenovaný má klidový proud bez zatížení do 1 mA. Oscilátor je sám o sobě největším spotřebitelem energie a podle datasheetu, potřebuje až 20 mA. To je však hodnota maximální, nepodařilo se mi k oscilátoru najít podrobnější specifikaci, zda se jeho spotřeba sníží např při vypnutí výstupu, či zůstane stejná. K objasnění této skutečnosti jsem změřil úbytek napětí na odporu o velikosti 6,6 Ω zapojeném mezi napájecí napětí a napájecí vývod oscilátoru při vypnutém transceiveru. Velikost změřeného úbytku je 21 mV, což odpovídá odběru proudu 3,2 mA. Sečteme-li všechny spotřeby, dostaneme se tak k hodnotě řádově 5 mA. Každopádně odpojením integrovaných obvodů obsluha velice rychle zjistí, že pohyblivá jednotka nefunguje a buď ji připojí na dostatečně napájené kolejiště nebo mechanickým vypínačem odpojí akumulátory.

3.8 Návrh vf části

Nedílnou součástí bezdrátové komunikace je vyzařování elektromagnetického vysokofrekvenčního pole. K tomu slouží různé druhy zářičů (antén), které spolu s přizpůsobovacím členem převádějí signál do prostoru. Výrobce transceiveru poskytuje pro pásmo 2,4 GHz celou řadu referenčních řešení antének. Jejich přehled je v dokumentu [14]. Bohužel jsem tento dokument objevil resp. jeho poslední revize vyšla až po návrhu starší desky (model 2010). Jako první jsem vyzkoušel meandrovitou anténku [15]. Tento typ je nejmenší referenční návrh anténky na plošném spoji. Jeho vnější rozměry jsou pouze 15.2 mm x 5.7 mm. Nevýhodou, která se uvádí v dokumentaci [14], je obtížné impedanční přizpůsobení anténky transceiveru resp. velké nároky

Obr 29: Meandrovitá anténka, 50 Ω, použita v modelu 2010

na přesnost jeho provedení. Praktický dosah s navrženým přizpůsobovacím členem je kolem 4 metrů. Jako alternativu pro starší generaci hardwaru jsem také navrhl desku se skládaným dipólem [16]. Zde není potřeba žádný přizpůsobovací člen a vstupy anténky se připojí rovnou na výstupy transceiveru. Tato anténka má rozměry 46 mm x 9 mm. Bohužel není určena přesně pro impedanci transceiveru CC2520, takže její výkon je ještě horší než v případě meandrovité anténky s přibližným impedančním přizpůsobením. Praktický dosah činí pouze 2 m. Nový hardware (modul transceiveru, model 2011) využívá anténku [17] a impedanční přizpůsobení přímo z modulu CC2520EM. Cely modul transceiveru je vlastně zmenšenou kopií profesionálního modulu. Některé prvky jsou jiné, zejména obvod impedančního přizpůsobení, což je způsobeno zejména nedostupnosti kondenzátorů disponujících kapacitou s přesností na desetiny pF a rozměrové kategorie SMD0402. Rozměry samotné anténky jsou 25,7 x 7,5 mm. Praktický dosah v otevřeném prostoru jsem nezjišťoval. Pokrýval celou místnost (při uvažování nepřímé viditelnosti místa vysílání s místem příjmu záleží na odrazivosti, která může být různá).



Obr 30: Anténka modulu transceiveru, 50 Ω, použita v modelu 2011

K citlivosti anténky na navržené desce plošných spojů mohu uvést porovnání s modulem CC2520EM. Využitím vývojového kitu (s odpojenou koaxiálně připojovanou anténkou pro snížení kvality signálu) v zapojení pro detekci paketů pomocí aplikace Packet Sniffer [18] jsem odečetl údaj LQI o velikosti 130. Pokud jsem ze stejného místa vysílal z modulu CC2520EM (s připojenou anténkou) se shodným výkonovým nastavením, Packet Sniffer ukázal hodnotu LQI kolem 145. Přepočteme-li údaje LQI na údaj o síle signálu v dBm dostaneme rozdíl síly signálu RSSI mezi oběma moduly:

$$RSSI_1 - RSSI_2 = (LQI_1 - LQI_2)\frac{RSSI_{max} - RSSI_{min}}{255} = (130-145)\frac{10-(-85)}{255} \approx -6\,dBm \quad .$$

Mnou navržený modul tak v tomto scénáři prokázal, že výkon anténky spolu s obvodem impedančního přizpůsobení je poloviční oproti profesionálnímu řešení.

4 Softwarové řešení

4.1 Softwarové volby řešení bezdrátové komunikace

Základní volbou při realizaci bezdrátové komunikace je použitá vývojová platforma. V dnešní době je pro tvorbu programů jasnou volbou programovací jazyk C, který zcela potlačil přímé psaní strojového kódu, neboli jazyk symbolických adres. V tomto jazyku poskytuje firma TI pro své vývojové kity také implementovány komunikační protokoly ke svým integrovaným obvodům:

- plný ZigBee protokol v projektu s názvem „Z-Stack“
- pouze samotný protokol IEEE802.15.4 v projektu „TIMAC“
- jednoduchý proprietární protokol „SimpliciTI“

Všechny protokoly jsou k dispozici spolu s funkčními příklady pro použití ve vývojových kitech a hardwaru TI pro nekomerční účely. Z jmenovaných možností obsahují „Z-Stack“ a „TIMAC“ nejen implementaci ovladače transceiveru, ale pro usnadnění funkce i jednoduchý operační systém (fronta úloh, správa paměti) a ovladače periferii mikrokontroléru jako je např. A/D převodník a sériovou komunikaci. Nevýhodou je používání uzavřených knihoven pro realizaci vlastního komunikačního protokolu, kdy zpracování dat probíhá pomocí volání knihovních funkcí, které následně volají buď funkce nižší vrstvy nebo přímo ovladač hardwaru. Pro správnou funkci pevně stanoveného komunikačního protokolu, jakým je IEEE802.15.4, není přepisování kódu nutné, nicméně neexistence této možnosti je i tak velkou nevýhodou pro další použití. Protokol SimpliciTI je volně dostupný včetně plných zdrojových kódů. Ve zkratce by se dalo říci, že se ani nejedná o plnohodnotný komunikační protokol, ale spíše o ovladač transceiveru, kdy přímo pomocí volání jedné funkce dochází k přesunu dat do transceiveru a jejich následnému odeslání. Navíc zde nejsou implementovány ani ovladače periférií a funkcionality operačního systému.

Pro vestavěné systémy existuje ovšem navíc i další volba – jazyk NesC, který je používán open source operačním systémem navrženým pro bezdrátovou senzorovou síť s operačním systémem TinyOs [23]. NesC je programovací jazyk podobný jazyku C. Není s ním však vzájemně kompatibilní. Systém TinyOs je spravován komunitou vývojářů, kteří sdílejí svou práci na serveru http://code.google.com/p/tinyos-main/. V systému TinyOs jsou již hotové jednotlivé komponenty pro protokol IEEE802.15.4 a zprovozněno několik druhů komerčních bezdrátových zařízení (vývojových platforem). Bohužel mezi nimi není ani jedna, které by se shodovala s mým hardwarem (stav březen 2011). Jsou zde např. spojení staršího transceiveru CC2420 s MSP430 nebo mikrokontroléry ATMEL. Je zde sice platforma s CC2520 a mikrokontrolérem typu ARM3, nicméně je použita k jednoduché testovací aplikaci, která nepředstavuje použitelné komunikační schéma. Obecně implementace nové platformy pro TinyOs je složitá, neboť je třeba propojit velké množství modulů tvořících jednotlivé funkce kódu. Relizaci v systému TinyOs jsem po seznámení

se s jeho složitostí a odlišností od profesionálního řešení firmy TI opustil. Přenos dosavadního C kódu do nesC sám o sobě je dost náročný a ovladač pro použitý transceiver CC2520, který je sice v hlavní vývojové větvi, není dostatečně zapracován do ostatních komponent systému TinyOs.

Pro vlastní realizaci jsem již z bakalářské práce měl zkušenosti se ZigBee protokolem Z-Stack. Postupně jsem řešení postavené nad knihovnou Z-Stack vylepšoval až do jeho konečné podoby, která provádí přenos dat podle navrženého komunikačního scénáře. Tento komunikační scénář jde také řešit pomocí jednoduchého protokolu SimpliciTI. Poslední implementovaná verze je tak vytvořena právě s jeho pomocí. Následující kapitoly popisují klíčové části obou komunikačních protokolů spolu se zakomponováním obsluhy měřicích a akčních prvků. Mnou vytvořené zdrojové kódy jsou umístěny na přiloženém CD včetně popisu jejich kompilace ve vývojovém prostředí IAR EW430. K jejich kompilaci je potřeba plná nebo demoverze vývojového prostředí. Demoverze je dostupná po registraci na stránkách firmy IAR. Původní implementace protokolového zásobníku ZigBee a SimpliciTI je po registraci dostupná na stránkách Texas Instruments. Kompletní a aktuální kódy práce včetně odkazů na potřebné komponenty třetích stran jsou uvedeny na adrese http://rtime.felk.cvut.cz/hw/index.php/CC2520_radio.

4.2 Vývojové nástroje

Základním vývojovým nástrojem je překladač jazyka C. V současné době se pro práci s mikrokontroléry MSP430 používají 3 překladače:

- překladač firmy IAR, součást vývojového prostředí Embedded Workbench for MSP430 (zkráceně IAR EW430)
- překladač firmy Texas Instruments, součást vývojového prostředí Code Composer Studio
- open-source překladač MSPGCC4

Standardně se v komerční sféře používá překladač IAR. Dosahuje nejlepších parametrů optimalizace kódu. Vlastní vývojové prostředí v sobě integruje překladač, textový editor, správu projektů a debuggovací rozhraní. Zároveň obsahuje ovladač programovacího modulu MSP-FET430UIF s rozhraním JTAG pod prostředím Windows. Komerční verze je ze 3 jmenovaných nejdražší. IAR nicméně poskytuje zkušební verzi, která je po registraci funkční po dobu 30 dní nebo omezenou verzi, která neslinkuje dohromady soubory větší než 4 kB. Druhou možností je vývojové prostředí Code Composer Studio. Obsahuje všechny prvky vývojového prostředí firmy IAR a je cenově dostupnější. Rozdíl ve velikosti zkompilovaného kódu se podle experta firmy TI pohybuje kolem 2 %. Toto vývojové prostředí se nabízí také v omezené verzi - 16 kB strojového kódu. Poslední možností je kompilátor MSPGCC4. Jedná se o pouhý kompilátor tj. využívá klasický makefile, který obsahuje veškeré instrukce pro překladač a linker. Pro nahrání programu přes JTAG lze použít open-source ovladač programátoru MSP-FET430UIF MSPDebug. Ten ve spojení s DDD (Data Display Debugger) vytvoří plnohodnotnou grafickou debuggovací podporu. Jednotlivé debuggovací příkazy jsou volány přes gdb rozhraní (příkaz typu "monitor xxx" v příkazovém řádku DDD).

Spouštěcí skript pro MSPDebug a DDD:

```
#!/bin/sh
# start mspdebug and wait for GDB at localhost:2000
xterm -e /home/vaclav/msp430/mspdebug-0.13/mspdebug -d /dev/ttyUSB0 -j uif gdb &
MSPDEBUG_PID=$!
echo "MSPDEBUG_PID = $MSPDEBUG_PID"
#-x /opt/msp430-debug/share/msp430-ddd/msp430-connect.gdb
```

ddd --debugger /opt/msp430-gcc-4.4.5/bin/msp430-gdb -x /opt/msp430-debug/share/msp430-ddd/msp430-connect.gdb "$@"

kill $MSPDEBUG_PID

Jelikož obě dvě řešení komunikace Z-Stack i SimpliciTI jsou od výrobce k dispozici jako hotové projekty pod IAR EW430 a Code Composerem, vyžil jsem toho a vyvíjel jsem software pod IAR EW430, neboť jsem s ním pracoval od začátku již při bakalářské práci. Code Composer totiž nebyl v roce 2009 ještě funkční pro mikrokontrolér typu MSP430X.

Doplňkovým vývojovým nástrojem je zobrazovač paketů Packet Sniffer [18] rovněž od Texas Instruments. Jde o jednoduché grafické rozhraní, které zobrazuje vysílané pakety. Spolupracuje s vývojovým kitem CC2520DK. Tento kit v konfiguraci bez středního modulu zachytává veškerou komunikaci v prostředí a přes USB posílá data počítačové aplikaci. Packet Sniffer je kompatibilní jak s protokolem ZigBee, tak i se SimpliciTI.

4.3 Kompilační parametry

Podíváme-li se na přehled zhotoveného hardwaru, máme zde celkem 3 typy. Původní vývojový kit CC2520DK, starší verzi desky s integrovaným transceiverem a konečně novou modulární verzi. K tomu navíc ještě přistupuje dělení fyzických jednotek podle funkce v bezdrátové senzorové síti na koncové jednotky a sběrnou jednotku. Koncové jednotky se dále dělí na jednotky pohyblivé (motorové) s optickou nebo laserovou myší a jednotky stacionární (obsluha kolejiště). Když vyčíslíme počet všech kombinací, dosáhneme čísla 9 nebo i vyššího zohledníme-li nutnost naprogramovat každé jednotce jinou MAC adresu (v mé implementaci i pořadí vysílání). Skutečný počet nicméně není takto vysoký např. vývojový kit používám výhradně ve funkci sběrného uzlu. K použití společného kódu jsou proto při překladu nutné kompilační parametry. Jedná se v podstatě o symboly s direktivou *#define* nebo o parametry definované jinde. Při použití prostředí IAR EW430, které používá vlastní překladač, se definují direktivy preprocesoru v jednotlivých konfiguracích. Každá konfigurace, kterých může mít každý projekt několik, obsahuje vlastní sadu direktiv preprocesoru. Může zde být např. konfigurace pro původní kit "*TIdevelKitCoordinator".* Tato konfigurace je určena pro sběrný uzel sítě (koordinátor bezdrátové sítě ZigBee). Nakonec posledním zdrojem konfiguračních parametrů jsou konfigurační soubory v adresářové struktuře projektu. Tyto soubory jsou převzaty z originálních implementací Z-Stack a SimpliciTI. Obsahují parametry bezdrátové komunikace jako je číslo používaného komunikačního kanálu a struktury sítě (maximální počet přeposílaní paketů, nastavení dotazování koordinátoru na data). Konfigurační soubory jsou vždy vázány na konkrétní konfiguraci souboru. Pro přehlednost uvádím všechny potřebné parametry pro funkci požadovaného typu uzlu. Ve všech konfiguracích projektu se navíc vyskytuje jméno procesoru "*MSP430F2618*", i když je už definováno v nastavení cílového mikrokontroléru projektu.

Požadovaná činnost uzlu	Konfigurace projektu	Direktivy definované v konfiguraci projektu	Používaný konfigurační soubor
ZigBee koordinator CC2520DK	*TIdevelKitCoordinator*	*USB, TI_DEVEL_KIT*	*f8wConfig, f8wCoord*
ZigBee koordinator, model 2010	*vencasBoardCoordinator*	*USB*	*f8wConfig, f8wCoord*
ZigBee koordinátor model 2011	*vencasBoardCoordinator*	*USB, MODULAR_BOARD*	*f8wConfig, f8wCoord*
ZigBee motorová jednotka, model 2010, PS/2, pořadí vysílání x	*vencasBoard*	*OSM_PAN3401, SENS_IDx*	*f8wConfig, f8wEndev*
ZigBee stacionární jednotka, model 2010, pořadí x	*vencasBoard*	*TRACK, SENS_IDx*	*f8wConfig, f8wEndev*
ZigBee motorová jednotka, model 2011, laserová myš	*vencasBoard*	*OSM_PAW3601, MODULAR_BOARD, SENS_IDx*	*f8wConfig, f8wEndev*
ZigBee stacionární jednotka, model 2011	*vencasBoard*	*MODULAR_BOARD, TRACK, SENS_IDx*	*f8wConfig, f8wEndev*

Tabulka 3: Kompilační parametry Z-Stack pro zařízení

U SimpliciTi je situace obdobná, hlavní kompilační parametr je MRFI_CC2520 udávající, že se překlad provádí pro transceiver CC2520.

Požadovaná činnost uzlu	Konfigurace projektu	Direktivy definované v konfiguraci projektu	Používaný konfigurační soubor
SimpliciTI Access point, CC2520DK	*CC2520-Access Point*	*USB, TI_DEVEL_KIT*	*Access_Point\ smpl_config.dat*
SimpliciTI Access point, modulární deska	*CC2520-Access Point*	*USB, MODULAR_BOARD*	*Access_Point\ smpl_config.dat*
SimpliciTI motorová jednotka, modulární deska, pořadí vysílání x	*CC2520-End Device*	*OSM_PAW3601, MODULAR_BOARD, SENS_IDx*	*End_Device\ smpl_config.dat*
SimpliciTI stacionární jednotka, modulární deska	*CC2520-End Device*	*MODULAR_BOARD, TRACK, SENS_IDx*	*End_Device\ smpl_config.dat*

Tabulka 4: Kompilační parametry SimpliciTI pro zařízení v síti

4.4 Použití hodinového signálu

Mikrokontrolér MSP430 disponuje 3 základními hodinovými signály MCLK (Main clock), ACLK (Auxiliary clock) a SMCLK (Subsystem clock). Hodiny MCLK jsou funkční vždy tj. po restartu mikrokontroléru jsou nastaveny na interní oscilátor. Hodiny ACLK jsou typicky řízeny externím (na vstupu XT1IN) nebo interním pomalým oscilátorem. Hodiny SMCLK pak mají jako svůj zdroj opět externí (tentokrát vstup XT2IN) nebo interní oscilátor. Pro zajištění funkce protokolu Z-Stack i SimpliciTI je nutné mít zdroj přesného hodinového signálu, zejména pro periferní obvody. Přesnost samotného interního oscilátoru se pohybuje v řádu jednotek procent, neboť je teplotně závislá. Původní verze Z-Stacku i SimpliciTI pro vývojový kit využívá jako

zdroj hodinového signálu kalibrování interního oscilátoru na požadovanou frekvenci pomocí krystalu 32,768 KHz na vstupu XT1IN. Tento postup je možno vysledovat v makru *HAL_CLOCK_INIT(x)* , které pro x = 0 volá volá v assembleru implementovanou funkci *TI_SetDCO()*. Toto platí pro vývojový kit z výroby s neosazeným krystalem na místě XT2IN (6 MHz). Připájením krystalu lze získat přesnější hodinový signál. Vývojový kit CC2520DK používá jako zdroj hodinového signálu pro modul sériové komunikace (SPI, UART) hodiny SMCLK a pro časovač (TimerA) pak hodinový signál ACLK. K správné funkci mého hardwaru je tento postup nutno zcela zavrhnout. Má koncepce využívá navíc pouze hodiny ACLK, jejichž zdrojem je transceiver CC2520. Aby bylo možno spouštěcí sekvenci mikrokontroléru použít jak pro starou desku (VREGn transceiveru vždy zapnuto), tak i pro novou desku (VREGn transceiveru vypnuto), je nutné zabránit použití ACLK před tím než bude jejich hodinový zdroj řádně nastaven. Po restartu hodiny mikrokontroléru běží na interní oscilátor (přibližně 1 MHz) a hodiny ACLK jsou nastaveny pro signál z XT1IN, nicméně nejsou zatím funkční - modul SPI běží z SMCLK. Poté je možno přes SPI nastavit výstup GPIO0 transceiveru na signál o frekvenci až 16 MHz. K funkci ACLK je 16 MHz příliš velká hodnota, SPI modul je možno provozovat maximálně na frekvenci 8 MHz. Sekvence zapnutí transceiveru je implementována ve funkci Mrfi_TurnOnRadioPower (zde příklad funkce ze SimpliciTI v Z-Stacku je shodná funkce dualchipTurnOnRadioPowerVREG):

```
 static void Mrfi_TurnOnRadioPower(void)
{
#ifdef TI_DEVEL_KIT                                    //platí pro originální hardware CC2520DK
  /* put radio chip into reset */
  MRFI_DRIVE_RESETN_PIN_LOW();
  /* enable the voltage regulator */
  MRFI_DRIVE_VREG_EN_PIN_HIGH();
  /* wait for the chip to power up */
  Mrfi_DelayUsec(MRFI_VREG_SETTLE_TIME_USECS);
  /* release from reset */
  MRFI_DRIVE_RESETN_PIN_HIGH();
  /* wait for the radio crystal oscillator to stabilize */
  MRFI_SPI_SET_CHIP_SELECT_ON();
  while (!MRFI_SPI_SO_IS_HIGH());
  MRFI_SPI_SET_CHIP_SELECT_OFF();
#endif
#ifdef MODULAR_BOARD                          //uzel model 2011
    uint16 count = 0xFFFF;
    /* put radio chip into reset */
    MRFI_DRIVE_RESETN_PIN_LOW();
    /* enable the voltage regulator */
    MRFI_DRIVE_VREG_EN_PIN_HIGH();
    //wait for stabilize
    while(count){
      count--;
    }
    /* release from reset */
    MRFI_DRIVE_RESETN_PIN_HIGH();
    MRFI_SPI_SET_CHIP_SELECT_OFF();
    MRFI_SPI_SET_CHIP_SELECT_ON();
    while (!MRFI_SPI_SO_IS_HIGH());
    MRFI_SPI_SET_CHIP_SELECT_OFF();
    /*Change EXTCLOCK frequency to 16 MHz, output to GPIO0 is default*/
    mrfiSpiWriteReg(EXTCLOCK, 0x03E);         //0011 1100 - 8 MHz, 0011 1110 - 16 MHz
    /*setting clock from radio*/
    asm("BIC.W #020h,SR");            //clear OSCOFF
    asm("BIS.B #0D1h,&0057h");      // BCSCTL1 , XT2OFF,high frequency mode, divider 2,
    asm("MOV.B #030h,&0053h");    // BCSCTL3 , digital external source
    asm("BIC.B #002h,&0002h");      // clear oscilator fault interrupt flag
    count=0x0FFFF;          //wait for sure
```

```
  while(count){
    count--;
  }
  asm("BIS.B #0C0h,&0058h");     //BCSCTL2 , MCLK is sourced from external source or
                                  //VLOCLK
  /*blink test for ensuring current clock frequency, if it's slow, MSP is
  **running on DCO*/
  BSP_BLINKING_TEST_LED1();
  /*Change SPI clock to ACLK*/
  MRFI_SPI_ACLK_CLOCK();
#endif
}
```

Pokud požadujeme vypnutí celého uzlu v případě zjištění malého napětí baterie a odpojení od napájení lze mikrokontrolér uspat do jednoho z úsporných módů. ACLK i MCLK nastavíme na VLOCLK a pomocí časovače budeme periodicky kontrolovat stav napájení. Pokud je detekováno vnější napájení uzlu, restartujeme mikrokontrolér zapsáním vynulováním ochranné hodnoty v registru watchdogu.

```
WDTCTL = ~WDTPW;
```

4.5 Ovladač transceiveru CC2520

Ovladač transceiveru CC2520 používá pro komunikaci s mikrokontrolérem především sériové rozhraní SPI. Dále je zde možnost využít 6 konfigurovatelných pinů pro doplňkovou komunikaci. Přehled o zapojení vývodů transceiveru na mé desce dává následující tabulka. Názvy funkcí jsou převzaty ze zdrojového kódu.

Pin CC2520	pin MSP430	Funkce
SO	*P5.2*	*SPI slave output, master input*
SI	*P5.1*	*SPI slave input, master output*
SCLK	*P5.3*	*SPI clock*
CSn	*P5.0*	*SPI chip select*
VREGn	*P5.4*	*povolení napájení analogové části transceiveru*
RESETn	*P6.2*	*reset transceiveru*
GPIO0	*XT1IN*	*hodinový signál pro mikrokontrolér*
GPIO1	*P2.7*	*EXCEPTION_TX_FRM_DONE a EXCEPTION_TX_ACK_DONE*
GPIO2	*P2.6*	*EXCEPTION_RFC_FIFOP*
GPIO3	*P4.5*	*EXCEPTION_RFC_SNIFFER_CLK nebo GPIO_CMD_STXON nebo GPIO_CMD_STXONCCA pro Z-Stack*
GPIO4	*P4.6*	*EXCEPTION_RFC_SFD_SYNC pro Z-Stack nebo EXCEPTION_RFC_FIFO pro SimpliciTI*
GPIO5	*P4.7*	*EXCEPTION_RFC_CCA nebo EXCEPTION_RFC_SNIFFER_DATA pro Z-Stack*

Tabulka 5: Rozhraní transceiveru CC2520

Význam nakonfigurovaných GPIO funkcí je následující:

- *EXCEPTION_TX_FRM_DONE* je výstupní signál transceiveru říkající, že vysílací buffer byl celý odvysílán
- *EXCEPTION_TX_ACK_DONE* opět výstup, signalizuje, že potvrzovací rámec byl odvysílán
- *EXCEPTION_RFC_FIFOP* indikuje platný rámec ve vstupním bufferu transceiveru případně, že je vstupní buffer naplněn nad definovanou úroveň

- *EXCEPTION_RFC_SNIFFER_CLK* je výstup hodin 250 kHz pro přímé vzorkování přijatého rámce
- *EXCEPTION_RFC_SNIFFER_DATA* jsou data přímo přímo vysílaná/přijímaná transceiverem
- *EXCEPTION_RFC_SFD_SYNC* udává přesný okamžik vysílání/příjmu SFD čili počátku vysílání/příjmu
- *EXCEPTION_RFC_CCA* signalizuje stav CCA.
- *GPIO_CMD_STXON* a *GPIO_CMD_STXONCCA* jsou vstupní signály transceiveru spouštějící vysílání

Definice propojení pinů MSP430 a CC2520 jsou v souboru *hal_mac_config.h* pro Z-Stack a *mrfi_board_defs.h* pro SimpliciTI, kde je i vlastní zapsání hodnoty do příslušného kontrolního registru transceiveru. Z-Stack provádí konfiguraci pinů na straně transceiveru v *mac_dualchip.c.* Zdrojové soubory vlastního ovladače jsou v případě SimpliciTI ve zdrojovém kódu souborů *"bsp_external/mrfi_board.c" , "radios/family3/mrfi_spi.c", "radios/family3/mrfi_radio.c".* Tyto zdrojové soubory patří k transceiveru CC2520, nicméně funkce skupiny MRFI (minimum RF interface) jsou shodné pro všechny podporované transceivery firmy Texas Instruments. MRFI obsahuje tyto funkce, jejichž název je většinou samovysvětlující:

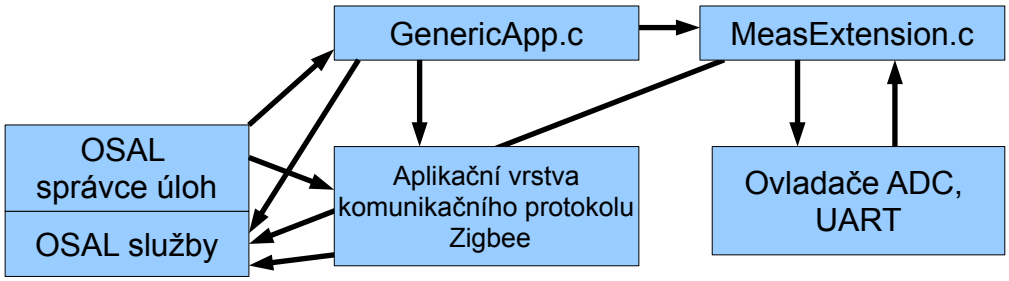
```
void    MRFI_Init(void);
uint8_t MRFI_Transmit(mrfiPacket_t *, uint8_t);
void    MRFI_Receive(mrfiPacket_t *);
void    MRFI_RxCompleteISR(void); /* populated by code using MRFI */
uint8_t MRFI_GetRadioState(void);
void    MRFI_RxOn(void);
void    MRFI_RxIdle(void);
int8_t  MRFI_Rssi(void);
void    MRFI_SetLogicalChannel(uint8_t);
uint8_t MRFI_SetRxAddrFilter(uint8_t *);
void    MRFI_EnableRxAddrFilter(void);
void    MRFI_DisableRxAddrFilter(void);
void    MRFI_Sleep(void);
void    MRFI_WakeUp(void);
uint8_t MRFI_RandomByte(void);
void    MRFI_DelayMs(uint16_t);
void    MRFI_ReplyDelay(void);
void    MRFI_PostKillSem(void);
void    MRFI_SetRFPwr(uint8_t);
```

Z této skupiny je zajímavá zejména funkce *MRFI_transmit(mrfiPacket_t *, uint8_t),* která slouží k vysílání dat. Tato funkce provádí algoritmus CCA (kontrola volného přenosového média), a pro skončení vysílání využívá nakonfigurovanou funkci transceiveru *EXCEPTION_TX_FRM_DONE.* Příjem dat se pak děje pomocí přerušení v transceiveru *EXCEPTION_RFC_FIFOP*, které je navázáno na přerušení v mikrokontroléru s voláním funkce *Mrfi_FiFoPIsr().* Jako doplňkový signál pak přerušovací funkce využívá signál *EXCEPTION_RFC_FIFO* informující, jestli došlo k přetečení v přijímacím bufferu transceiveru. Implementace ovladače v Z-stacku je složitější, obecně se dá říci, že využívá stejného způsobu vysílání běžných rámců. Pokud se provádí slotted CSMA-CA a tedy beacon enabled komunikace, používají se signály *EXCEPTION_RFC_CCA, GPIO_CMD_STXON a GPIO_CMD_STXONCCA.* Příjem je pak obdobný, znovu se využívá přerušení mikrokontroléru od *EXCEPTION_RFC_FIFOP* . Vlastní vyčtení dat z transceiveru se však neprovádí v samotném přerušení, ale po částech daných parametrem *MAX_PAYLOAD_BYTES_READ_PER_INTERRUPT.* Zdrojový kód příjmu paketu z transceiveru je v souboru *mac_rx.c.*

4.6 Aplikační vrstva v Z-Stack

Aplikační vrstva Z-Stack je tvořena obecnou částí aplikace pro bezdrátovou komunikaci *GenericApp.c* , abstraktní vrstvou operačního systému pro aplikaci *OSAL_GenericApp.c* a moji implementací vlastní aplikace složenou z doplňkových funkcí *MeasExtension.c.* pro obsluhu

hardwaru uvedeného v kapitole o něm pojednávajícím. V souboru *GenericApp.c*, který je modifikován pro vysílání a příjem vlastních dat ze senzorů pro koordinátor i koncový uzel, je hlavním prvkem smyčka zpracování zpráv představována funkcí *GenericApp_ProcessEvent* . Další funkcí obecné aplikace je zpracování přijatých zpráv a časované odesílání dat. Tato obecná část aplikace přímo spolupracuje s aplikační vrstvou protokolu ZigBee.



Obr 31: Diagram volání funkcí mezi moduly aplikační vrstvy v projektu Z-Stack

```
UINT16 GenericApp_ProcessEvent( byte task_id, UINT16 events )
{ ....
   if ( events & SYS_EVENT_MSG )
  {
   MSGpkt = (afIncomingMSGPacket_t *)osal_msg_receive( GenericApp_TaskID );
    while ( MSGpkt )
    {
      switch ( MSGpkt->hdr.event )
      {
        case ZDO_CB_MSG:
          GenericApp_ProcessZDOMsgs( (zdoIncomingMsg_t *)MSGpkt );break;
        case AF_DATA_CONFIRM_CMD:
          .........
          break;
        case AF_INCOMING_MSG_CMD:
          GenericApp_MessageMSGCB( MSGpkt ); break;
        case ZDO_STATE_CHANGE:
          GenericApp_NwkState = (devStates_t)(MSGpkt->hdr.status);
          if ( (GenericApp_NwkState == DEV_ZB_COORD)
              || (GenericApp_NwkState == DEV_ROUTER)
              || (GenericApp_NwkState == DEV_END_DEVICE) )
          {
#ifdef USB          //same as  GenericApp_NwkState == DEV_ZB_COORD
          osal_start_timerEx(GenericApp_TaskID,USB_SENSOR_ANSWER_TIMEOUT_EVT  ,0);
#endif
          }
          break;
        default: break;
      }
      // Release the memory
      osal_msg_deallocate( (uint8 *)MSGpkt );
      // Next
      MSGpkt = (afIncomingMSGPacket_t *)osal_msg_receive( GenericApp_TaskID );
    }
    // return unprocessed events
    return (events ^ SYS_EVENT_MSG);
  }
  .... //next events
}
```

Zprávy *events* jsou posílány ostatními úlohami, které jsou inicializovány v *OSAL_GenericApp.c* nebo přímo funkcemi modulů "operačního systému". Vlastní funkce meziprocesní komunikace jsou pak napsány v *OSAL.c*. Smyčka zpracování zpráv může zpracovat až 16 rozdílných zpráv. Zprávu

typu *SYS_EVENT_MSG* posílají nižší vrstvy Z-Stacku, aby upozornili na příchozí zprávu nebo změnu stavu zařízení v síti. Podle toho následně proběhne volání *GenericApp_MessageMSGCB(MSGpkt)* , která obsahuje vlastní zpracování přijatých dat. Speciální funkce *GenericApp_ProcessZDOMsgs((zdoIncomingMsg_t *)MSGpkt)* obsahuje implementaci ZDO zprávy. ZDO (Zigbee device object) je část v aplikační vrstvě ZigBee, která se stará o správu spojení mezi koncovými aplikacemi. Zpracovává například odpověď na volání funkce *ZDP_MatchDescReq(...)* , sloužící k nalezení adres zařízení se stejnou aplikací v celé síti. Další zajímavou vlastností systémových zpráv je zpráva typu *ZDO_STATE_CHANGE* hlásící, že došlo k vytvoření nebo k připojení k síti. Pokud ji obdrží koordinátor, nastaví milisekundový časovač operačního systému *osal_start_timerEx(GenericApp_TaskID, USB_SENSOR_ANSWER_TIMEOUT_EVT , 0);* , aby neprodleně nastavil současné smyčce událostí další událost k obsloužení *USB_SENSOR_ANSWER_TIMEOUT_EVT* , což je událost odeslání dat. Další pokračování periody posílání dat koncovým uzlům se pak ve smyčce obsluhy události zajistí dalším voláním časovače, tentokrát s nastaveným zpožděním *wirelessCycle.*

```
if (events & USB_SENSOR_ANSWER_TIMEOUT_EVT )    //timer has expired
 {
  if(sendEnable)
  {
   broadcastPacket=getAllPwmValues();
   MeasExtension_SendTheValue(); //transmitt data to all nodes
   osal_start_timerEx(GenericApp_TaskID,USB_SENSOR_ANSWER_TIMEOUT_EVT , wirelessCycle);
  }
  return (events ^ USB_SENSOR_ANSWER_TIMEOUT_EVT);
 }
```

Volání funkce pro vlastní odeslání dat pak provádí předání datového pole do nižší části aplikační vrstvy.

```
if ( AF_DataRequest( &GenericApp_DstAddr, &GenericApp_epDesc,
                GENERICAPP_MEASEXTENSION_CLUSTERID,
                7,  //number of bytes 7 int8
                (byte *) broadcastPacket,
                &GenericApp_TransID,
                AF_DISCV_ROUTE, AF_DEFAULT_RADIUS ) == afStatus_SUCCESS )
  HalLedSet(HAL_LED_1,HAL_LED_MODE_OFF)
 else
  HalLedSet(HAL_LED_1,HAL_LED_MODE_ON);
}}
```

Pokud se odeslání nezdaří (např. byl obsazen přenosový kanál nebo není je zaplněna fronta zpráv čekající na odeslání), rozsvítí se jedna z LED.

Kromě vysílání zpráv má *GenericApp.c na* starosti také zpracování přijatých zpráv. V popisu Zigbee jsem psal, že funkčnost aplikační vrstvy pro směřování zpráv nepoužívám, resp. že nemá žádný vliv. Tady v přijímací funkci zprávy je rozhodováno podle čísla clusteru zprávy. Nicméně jediný cluster, který se obsluhuje je právě *GENERICAPP_MEASEXTENSION_CLUSTERID.*

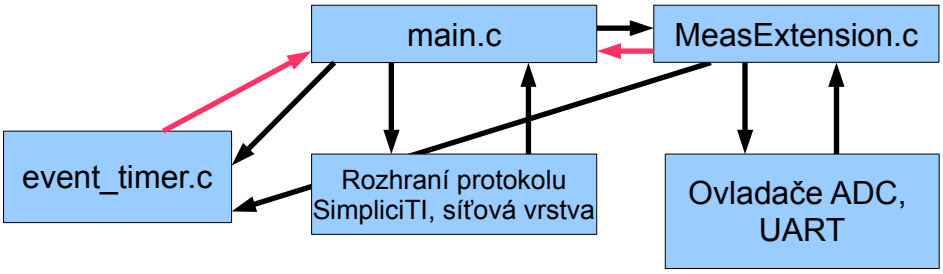
```
GenericApp_MessageMSGCB( afIncomingMSGPacket_t *pkt )
{
 int16 hodnotaX,hodnotaY;
 int16 *bpacket;
 dataPacket *pack;
 uint8 i,ok;
 byte *ch;
 switch ( pkt->clusterId )
 {
case GENERICAPP_MEASEXTENSION_CLUSTERID:
#ifdef USB
    //data comes to coordinator
    pack =(dataPacket*)pkt->cmd.Data;
    hodnotaX=pack->valX;
    hodnotaY=pack->valY;
```

```
    i=pack->node_id;
    setSensorValue(hodnotaX,hodnotaY ,i-1);
#else
    bpacket=(int16 *)pkt->cmd.Data;          //older version, int8 is enough
    #ifdef OSM_PAN3401
        setSpeed(*(bpacket+sensorID-1));
    #endif
    #ifdef TRACK
        setSwitch((uint8)*(bpacket+sensorID-1),SWITCH_BOTH); //set front switch
    #endif
    osal_stop_timerEx( GenericApp_TaskID,NODE_NO_DATA_TIMEOUT);    //stop timer
    //start watchdog timer for comunication
    osal_start_timerEx(GenericApp_TaskID,NODE_NO_DATA_TIMEOUT ,disableTime);
    //start timer for sending answer
    osal_start_timerEx(GenericApp_TaskID,NODE_ANSWER_TIMEOUT ,timestamp);
#endif
    break;
 }
}
```

Sběrné zařízení (s USB) si zde uloží odpověď koncového zařízení. Velikosti obou inkrementů a identifikátor se předají funkci *setSensorValue(....)*, která je definována v souboru *MeasExtension.c* . Pro koncové zařízení je funkce jiná. Z přijaté zprávy musí extrahovat data, která jsou určena jen pro něj. opět záleží na identifikačním čísle koncového zařízení. Toto číslo určuje pořadí správné hodnoty v poli hodnot, pro všechna zařízení. Podle toho, jestli je koncové zařízení typu pohyblivé nebo stacionární jednotky volá příslušnou funkci z *MeasExtension.c.* Navíc koncový uzel obsahuje pojistku proti výpadku komunikace. Při obdržení zprávy restartuje časovač, který spouští zprávu *NODE_NO_DATA_TIMEOUT.* Pokud tento časovač doběhne, přejdou výkonové výstupy koncového zařízení do bezpečného stavu. Dalším časovačem, který se ve stejný okamžik spouští je časovač odpovědi *NODE_ANSWER_TIMEOUT* koncového zařízení. Opět podle identifikačního čísla uzlu se nastaví časovač k spouštění této události. Po vypršení času zahájí koncový uzel vysílání svých dat.

4.7 Aplikační vrstva v SimpliciTI

Aplikační vrstva komunikačního protokolu SimpliciTI je v zdrojovém kódu SimpliciTI umístěna přímo ve funkci *main(),* neboť SimpliciTi neobsahuje žádnou funkcionalitu "operačního systému" jako Z-Stack, nejsou zde žádné úlohy jednotlivých vrstev komunikačního protokolu. Tato skutečnost znamená, že veškerý fyzický příjem a vysílání transceiveru se buď uskutečňuje při přerušení nebo jako funkční volání přímo z funkce *main()*. Vzhledem k tomu, že jsem projekt SimpliciTI pro bezdrátové řízení motorových jednotek, začal psát až po vyřešení funkce pod Z-Stackem, snažil jsem se funkci *main()* přizpůsobit funkci smyčky obsluhy zpráv z *genericApp.c.* Výsledkem je možnost použít téměř identický soubor *MeasExtension.c a* oddělil jsem tak vlastní aplikaci od bezdrátové komunikace. SimpliciTI neobsahuje podporu periferních modulů mikrokontroléru MSP430, zejména práci se sériovým portem, hardwarový časovač a A/D převodník. Je možnost buď převzít zdrojový kód z projektu Z-Stack nebo napsat si vlastní jednoduché ovladače. Zvolil jsem druhou možnost, jelikož využívám pouze několik jednoduchých funkcí periférií MSP430.



Obr 32: Diagram volání funkcí v aplikační vrstvě SimpliciTI, červeně je vyznačeno nastavení událostí do hlavní smyčky

Funkce *main()* začíná voláním inicializačních příkazů. Pro nastavení transceiveru a slouží funkce *SMPL_Init(sCB)*. Tato funkce dostává jako parametr ukazatel na funkci, kterou bude volat po zpracování datového paketu získaného z transceiveru. Po skončení této funkce je již transceiver připraven k použití. Tato funkce rovněž způsobí volání Mrfi_TurnOnRadioPower(void), která nastavuje zdroj hodinového signálu na hodinový signál transceiveru. Jako další se inicializují podpůrné moduly, především však *initTimers(&events)*. Tato funkce inicializuje modul milisekundového časovače identického s časovačem v modulu *OSAL_Timers.c* projektu Z-Stack. Jelikož jsou všechny události zpracovávané pouze v jedné smyčce zpráv, není nutné provést další dělení. Zůstává zde 16 druhů zpráv, které muže hlavní smyčka zpracovat. K obsluze časovače se pak používají již jen funkce pro přidání *addTimer(uint16 eventNum, uint16 timeout_milisec)* a odebání časovače *deleteTimer(uint16 eventNum)*. Následuje funkce pro zapnutí trvalého příjmu transceiveru *SMPL_Ioctl(IOCTL_OBJ_RADIO, IOCTL_ACT_RADIO_RXON, 0)*. Tato funkce projde rozhraním transceiveru na funkci *MRFI_RxOn(void)*, která pošle po SPI příkaz *SRXON*. Pokud je potřeba provést připojení k síti, tj. spouštěné zařízení je koncový uzel, je třeba volat neblokující funkci *SMPL_Link(&lid)*, která se pokusí o navázání spojení se sběrným uzlem. Proměnná *iid* je číslo linky obousměrného spojení se sběrným uzlem (AP). Funkce uzlu AP je totožná s uzlem koordinátoru v síti ZigBee, tj tvoří centrální uzel v síti. Ovšem o vlastní vytvoření linky na straně AP je třeba se také postarat, protože také není automatické jako v případě protokolu ZigBee. Uzel AP musí po příjmu paketu s číslem linky 0, které vysílá koncový uzel provádějící funkci *SMPL_Link(&lid)*, volat neblokující funkci *SMPL_LinkListen(&sLID[sNumCurrentPeers])*. Tato funkce vytvoří i pro AP číslo linky shodné s číslem, jaké obdrží koncový uzel. Ve funkci *main()* pak již následuje hlavní smyčka zpráv:

```
while (1)
 {
   if ( events & SIMPL_BCAST_MES_EVT )
   {
     processMessage();
     events&=~SIMPL_BCAST_MES_EVT;
   }
   ......
 }
```

Tato nekonečná smyčka obsahuje zpracování stejných zpráv jako funkce *GenericApp_ProcessEvent()* z aplikační vrstvy Z-Stack. Funkce, jež smyčka zpracování zpráv provádí jsou samozřejmě lokalizovány do prostředí SimpliciTI. Funkce pro odeslání dat je *SMPL_SendOpt(lid, (uint8*)&data, sizeof((uint8*)&data), SMPL_TXOPTION_NONE);* Proměnná *lid* je zde opět číslo linky s AP, který jediný ze všech zařízení v síti příjme tuto zprávu. Tento příjem je nicméně znovu kvůli jednoduchosti SimpliciTI propagován až do funkce *main()* všech uzlů v dosahu, kde je nutné ho ošetřit. Získání dat po po indikování příjmu funkcí *sCB(linkID_t lid)* z přijaté zprávy se provede zavoláním funkce *SMPL_Receive(SMPL_LINKID_USER_UUD, msg, &len)* ve funkci *processMessage()* . Jinak je úkol funkce *processMessage()* je totožný s funkcí *GenericApp_MessageMSGCB(MSGpkt)* v aplikační vrstvě projektu Z-Stack.

4.8 Soubor MeasExtension

Soubor *MeasExtension.c* obsahuje zdrojové kódy vlastní aplikace, přesněji výkonou část aplikace. Nachází se zde veškerý kód pro obsluhu fyzických obvodů všech typů bezdrátových uzlů. Je zde implementována komunikace s PC, generování PWM signálu, nastavování výhybek, vyčítání dat z optických senzorů a správa zálohového napájení. Rozdělení funkcí v zdrojovém souboru je podle okamžiku použití následující (verze ze Z-Stacku, v SimpliciTI je název funkcí odlišný, jejich význam však zůstává):

- při inicializaci - *void MeasExtension_Init(uint8 genericAppTaskId)*
- obsluha přerušení - *HAL_ISR_FUNCTION(ps2_clk_isr,PORT1_VECTOR)*
- volání přes ukazatel - *void UART_callback(uint8 port, uint8 ev)*

- volání z aplikační vrstvy projektu - všechny ostatní funkce

Funkce *MeasExtension_Init()* si uloží odkaz na registr zpráv obecné části aplikace a dále provádí inicializaci jednotlivých hardwarových modulů. Z tohoto pohledu je nejsložitější nastavení optického senzoru myši pomocí PS/2 komunikačního protokolu. Implementoval jsem čistě softwarové řešení tj. stylu

```
#define PS2_WAIT_CLK()          st(volatile uint8 portcpy=0;   \
                                while(!portcpy)                \
                                portcpy=PS2_CLK_READ;          \
                                while(PS2_CLK_READ);) ,
```

neboť nastavení se provádí pouze jednou a hodinový signál a tím i platnost dat určuje senzor myši s negarantovanou rychlostí. U senzoru se musí minimálně nastavit tzv. stream mód *PS2_SET_STREAM_MODE* hodnota *0xEA*. Ten povolí automatické vyslání dat z optického senzoru při detekování pohybu. Kdyby se totiž nepoužíval stream mód, musel by se do myši neustále posílat příkaz na vyčítání inkrementu. Další nastavení zde prováděná se tykají sériového portu, PWM a AD převodu napětí akumulátoru. Signál PWM se získá z časovače A mikrokontroléru:

```
#define PWM_INIT()  st( TACCR0 = 0; TACTL = 0; TACCTL1 = 0;TACCR1=0; \
            TACCR0 |= PWM_TACCR0; TACCR1|=PWM_DEFAULT_TACCR1; \
            TACCTL1|=0x00E0; TACTL|=0x0110;P1DIR |=0x0004;P1SEL|=0x0004;)
```

Používám zde hodnotu *PWM_TACCR0 a PWM_DEFAULT_TACCR1* jako výchozí nastavení činitele plnění PWM signálu pro obvod DRV8800 připojeného na vývod P1.2. Stejné makro lze použít i pro generování signálu o frekvenci 36 kHz pro buzení infračervených závor. Při frekvenci hodin *ACLK* 8 MHz, dostaneme zapsáním hodnot *TACCR0 = 222 a TACCR1 = 111 na* vývodu P1.2 symetrický obdélníkový signál o frekvenci 36,036 kHz.

Dalším skupinou funkcí jsou obsluhy přerušení. Jedná se o funkci

```
 HAL_ISR_FUNCTION(ps2_clk_isr,PORT1_VECTOR)
{
.....
}
```

Poněkud netypický zápis funkce přerušení je běžně používán v implementaci Z-Stacku, proto ho také používám. Vyjadřuje pouze zkrácený zápis, kdy kompilátor nahradí implementaci následujícími makry:

```
#define _PRAGMA(x)                          _Pragma(#x)
#define HAL_ISR_FUNC_DECLARATION(f,v)       _PRAGMA(vector=v) __interrupt void f(void)
#define HAL_ISR_FUNC_PROTOTYPE(f,v)         _PRAGMA(vector=v) __interrupt void f(void)
#define HAL_ISR_FUNCTION(f,v)               HAL_ISR_FUNC_PROTOTYPE(f,v);
                                            HAL_ISR_FUNC_DECLARATION(f,v)
```

Samozřejmě, že přímočařejší přístup je použít následující kód, ale horní řešení je zase kratší.

```
void ps2_clk_isr(void);
#pragma vector= PORT1_VECTOR
__interrupt void ps2_clk_isr(void)
{
}
```

V přerušení probíhá odečítání stavu datového signálu z optického senzoru. Přerušení je totiž nastaveno na spádovou hranu hodinového signálu. Postupně tak dochází k vyčítání hodnoty z optické myši. Při nesouhlasu paritního bitu nebo nesouhlasu stavového bajtu, dojde k odmítnutí přijatých dat podržením signálu hodin. Optický senzor přenesení tohoto bajtu opakuje. Senzor laserové myši je obdobně obsluhován v přerušení od jeho inkrementálních výstupů při náběžné hraně jednoho ze signálů. Pokud se signály pro jeden směr při každém přerušení jimi způsobeném střídají, pohybuje se myš stále stejným směrem a inkrement se zvyšuje resp. snižuje. Následuje-li přerušení od stejného signálu, inkrement se začne snižovat resp. zvyšovat.

Další skupinou funkcí jsou funkce volané ovladači hardwaru. Je to především signál o přijmutí ukončovacího znaku do bufferu modulu sériového přenosu. Používám znak *LF* neboli 0x0A. Jakmile je tento znak přijat, volá se funkce *UART_callback()*. Tato provede na základě prvního

znaku, který vyčte z přijímacího bufferu rozhodnutí o dalším postupu. Pokud je tento znak hodnoty *USB_RECEIVE_X*, vyvolá událost typu *USB_INCOMING_DATA_EVT*. Tato událost je ve smyčce zpracování událostí vyhodnocena tak, že zpětně zavolá funkce *receiveDataUsb() a sendDataUsb()* z *MeasExtension*.

Tyto funkce už patří ke skupině volané přímo z aplikační vrstvy projektu Z-Stack nebo SimpliciTI. Provádějí vyčtení dat z bufferu, jejich zapsání do pole pro odesílané hodnoty, převod dat ze senzorů do ASCII znaků a zapsání do vysílacího bufferu UART modulu. Z dalších funkcí v *MeasExtension* stojí za zmínku přestavování výhybek funkcí *setSwitch(uint8 state,uint8 front)*. Při parametru *front = SWITCH_BOTH se* přepnou naráz všechny výhybky. Je to nejvýhodnější způsob, neboť jen částečné přepnutí je nedůvodné. K přepnutí dalších výhybek k postavení smysluplné vlakové cesty by stejně muselo v nejbližší době dojít. Důležité je zde provést vypnutí proudu do cívek přestavníku pomocí události *SWITCH_TIMEOUT_EVT* po krátkém čase. Nedojde-li k dosednutí jazýčku výhybky do krajní polohy z důvodu jejího zaseknutí, cívkou stále protéká proud, poněvadž nebyl přerušen rozpínací kontakt spojený s přestavníkem. Výhybky nejsou stavěny na trvalý průchod proudu a hrozí jejich zničení.

5 Závěr

V diplomové práci jsem navrhl hardware a upravil software pro realizaci bezdrátového řízení modelové železnice. Oproti mé bakalářské práci (2009) jsem navrhl novou desku plošných spojů a ověřil její funkčnost spolu s optickou snímačem zástavbou do vozu modelové železnice (model desek 2010). Dále jsem přidal další typ optického snímače polohy, zavedl podporu zálohového napájení i zlepšil možnosti propojení mého hardwaru použitím robustnějších konektorů a modulového uspořádání (model desek 2011). V softwaru jsem zavedl ovladače pro výše zmíněné obvody a vyzkoušel použití IEEE802.15.4 a jeho odvozených variant - protokolu zásobníku ZigBee a komunikačního protokolu SimpliciTI. Hlavní ponaučení z vývoje hardwaru pramení z chyb, kterých jsem se dopustil. Jedná se zejména o přehlédnutí v návrhu desek plošných spojů a důležitost zvolit správný kompromis rozměrů, výkonu a odolnosti hardwaru pro vestavbu do omezeného, ale z části volného prostoru. Tato skutečnost vynikne na rozdílu mezi modely desek z let 2010 a 2011. Ponaučení z vývoje softwaru se týká hlavně použité bezdrátové komunikace. Protokol ZigBee, resp. jeho implementace Z-Stack je vhodná pro velmi rozsáhlé senzorové sítě, kde se předpokládá bohaté větvení stromové struktury sítě a komunikace navzájem mezi mnoha uzly. Pro jednoduchou síť s hvězdicovou strukturou je vhodnější protokol SimpliciTI, který používá přímo pouze fyzickou a nepatrnou část funkčnosti MAC vrstvy protokolu IEEE802.15.4, což v jednoduché síti s několika zařízeními stačí. Z druhé strany ovšem vystupuje hledisko vlastní implementace těchto protokolů. Protokol SimpliciTi způsobem, jímž je napsán ve zdrojových souborech od Texas Instruments, je kvalitativně hůře zpracován oproti Z-Stacku. Je optimalizován na jednoduchou komunikaci bateriově napájených uzlů s předpokladem malého vytížení přenosového kanálu. Na druhou stranu se jeho implementace dá snadno přizpůsobit řešené úloze, poněvadž protokol je jednoduchý a všechny zdrojové kódy jsou k dispozici. Licence pro použití zdrojového kódu není sice typu open-source, která by umožňovala volné použití, ale vývoj nekomerčních aplikací s obvody firmy Texas Instruments povoluje.

6 Literatura

[1] Václav Rychnovský, "Bezdrátové řízení motorových jednotek modelové železnice", ČVUT – FEL , červen 2009, http://support.dce.felk.cvut.cz/mediawiki/images/a/a4/Bp_2009_rychnovsky_vaclav.pdf

[2] Petr Jurčík, "Real-time Communication over Cluster-tree Wireless Sensor Networks", ČVUT – FEL, leden 2010, http://www.open-zb.net/publications/Thesis_Petr.pdf

[3] Ricardo Augusto Rodrigues da Silva Severino, „On the use of IEEE 802.15.4/ZigBee for Time-Sensitive Wireless Sensor Network Applications“, **P**olytechnic Institute of Porto School of Engineering, říjen 2008, http://www.cooperating-objects.eu/fileadmin/dissemination/2009-thesis-award/severino.pdf

[4] Mario Naval Escartín, „INTRA-VEHICULAR SENSOR ZIGBEE NETWORK“, ČVUT – FEL , prosinec 2008, http://www3.fs.cvut.cz/web/fileadmin/documents/12241-BOZEK/publikace/2008/2008_067_01.pdf

[5] ZigBee Alliance (2008), „ZigBee Specification“ ZigBee Document 053474r17, January 17, 2008 11:09 am, http://www.zigbee.org/

[6] IEEE-TG15.4, „Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR- WPANs)" IEEE Std 802.15.4™-2006, IEEE standard for Information Technology, 2006

[7] Texas Instruments, webová stránka SimpliciTI, http://focus.ti.com/docs/toolsw/folders/print/simpliciti.html

[8] Texas Instruments, „CC2520 Datasheet”, http://focus.ti.com/lit/ds/symlink/cc2520.pdf

[9] Texas Instruments, „MSP430F2618 Datasheet”, http://focus.ti.com/lit/ds/symlink/msp430f2618.pdf

[10] Texas Instruments, „MSP430 User guide“, http://focus.ti.com/lit/ug/slau144h/slau144h.pdf

[11] Texas Instruments, „MSP430™ Programming Via the JTAG Interface User's Guidet”, http://focus.ti.com/lit/ug/slau320b/slau320b.pdf

[12] Texas Instruments, „CC2520DK user guide“ , http://focus.ti.com/lit/ug/swru138/swru138.pdf

[13] Texas Instruments, „DRV8800, DRV8801 DMOS FULL-BRIDGE MOTOR DRIVERS“, http://focus.ti.com/lit/ds/symlink/drv8800.pdf

[14] Texas Instruments, „Antenna Selection Guide”, http://focus.ti.com/lit/an/swra161b/swra161b.pdf

[15] Texas Instruments, „Small Size 2.4 GHz PCB antenna“, http://focus.ti.com/lit/an/swra117d/swra117d.pdf

[16] Texas Instruments, „Folded Dipole Antenna for CC25xx“, http://focus.ti.com/lit/an/swra118/swra118.pdf

[17] Texas Instruments, „2.4 GHz Inverted F Antenna“, http://focus.ti.com/lit/an/swru120b/swru120b.pdf

[18] Texas Instruments, „SmartRF™ Packet Sniffer User Manual“, http://focus.ti.com/lit/ug/swru187e/swru187e.pdf

[19] PixArt Imaging Inc., „ PAW3601DH-NF CMOS LASER MOUSE SENSOR”, http://www.pixart.com.tw/upload/PAW3601DH-NF_SPEC_V23_20100303_20100319091518.pdf

[20] PixArt Imaging Inc., „PAW3402 /PAW3412 PS/2 OPTICAL MOUSE SOC”, http://www.pixart.com.tw/upload/PAW3402_PAW3412_SPEC_V11_20100319091714.pdf

[21] Adam Chapweske, „The PS/2 Mouse/Keyboard Protocol”, http://www.computer-engineering.org/ps2protocol/

[22] GP Batteries, „Nickel Metal Hydride Technical Handbook”, http://www.gpbatteries-cz.com.hk/html/pdf/NiMH_technical.pdf

[23] Podpůrné stránky TinyOs, http://docs.tinyos.net/index.php/Main_Page

[24] Podpůrné stránky MSPDebug, http://mspdebug.sourceforge.net/

Příloha A, tabulky propojení

Funkce	Číslo pinu na 2x7 pin konektoru JTAG vývojového kitu	Číslo pinu na 8 pin konektoru hlavní desky
TCK - test clock	7	1
TMS - test mode select	5	7
TDI/TCLK - test data input or test clock input	3	3
TDO/TDI - test data output	1	4
RST/NMI - reset input, nonmaskable interrupt	11	5
GND - ground terminal	9	6
VDD - napájení 3V3 (není nutné)	2 nebo 4	2
Pozn. Číslování pro 2x7 zástrčku (2,54 mm) JTAG platí pro zámek vlevo; pro 8 pin malý konektor (1,25 mm) je zámek vpravo		

Tabulka propojení programovacích vodičů MSP430

Konektor	Číslo pinu	Funkce
6 pinový	1	Napájení z kolejiště
	2	Napájení z kolejiště
	3	Záporný pól akumulátoru (GND)
	4	Kladný pól akumulátoru, (dělič 0,18 na P6.4)
	5	motorek
	6	motorek

Tabulka zapojení výkonového konektoru

Konektor	Číslo pinu	Funkce
4 pinový	1	GND
	2	P3.4, TXD
	3	3V3
	4	P3.5, RXD, výhybka 4

Tabulka propojení konektoru sériové komunikace

Konektor	Číslo pinu	Funkce
10 pinový	1	3V3
	2	P3.2, UCB0SOMI, závora 1
	3	P3.1, UCB0SIMO, závora 2
	4	P3.3, UCB0CLK, závora 3
	5	P1.6, PD, závora 4
	6	P1.7, YA, buzení infrazávor
	7	P1.5, YB, výhybka 1
	8	P1.1, XA, výhybka 2
	9	P1.0, XB, výhybka 3
	10	GND

Tabulka propojení konektoru pro senzory

Příloha B, schémata zapojení

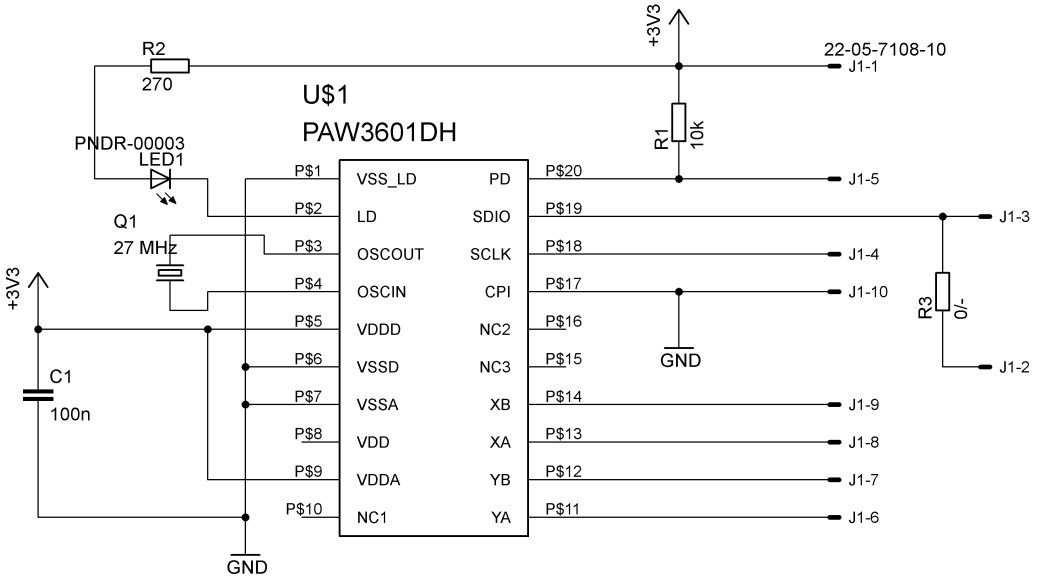


Schéma zapojení desky optického snímání pohybu, senzor laserové myši

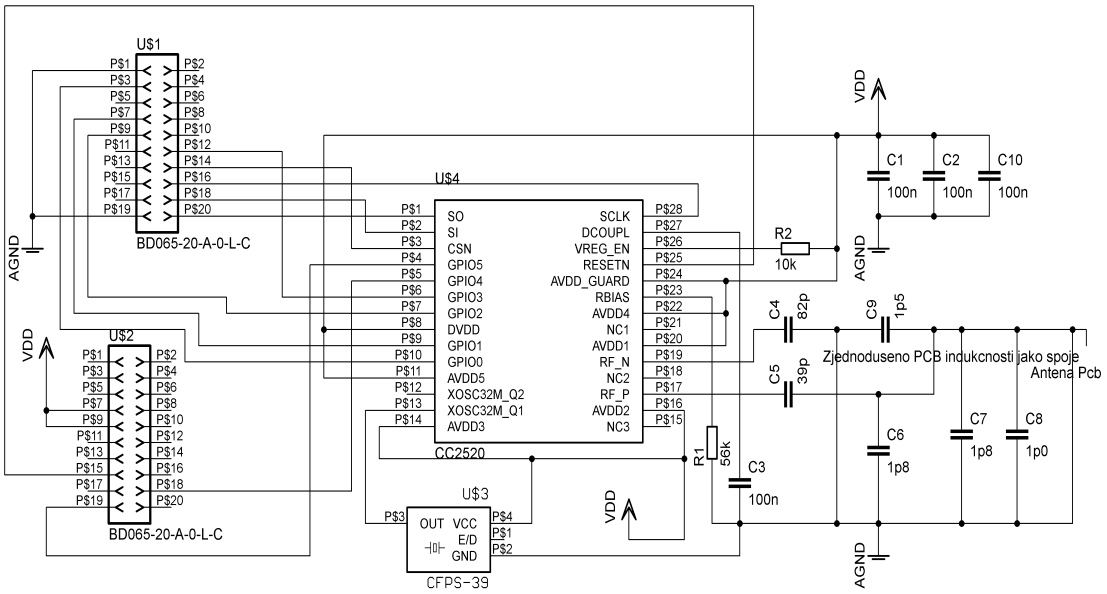


Schéma zapojení modulu transceiveru CC2520

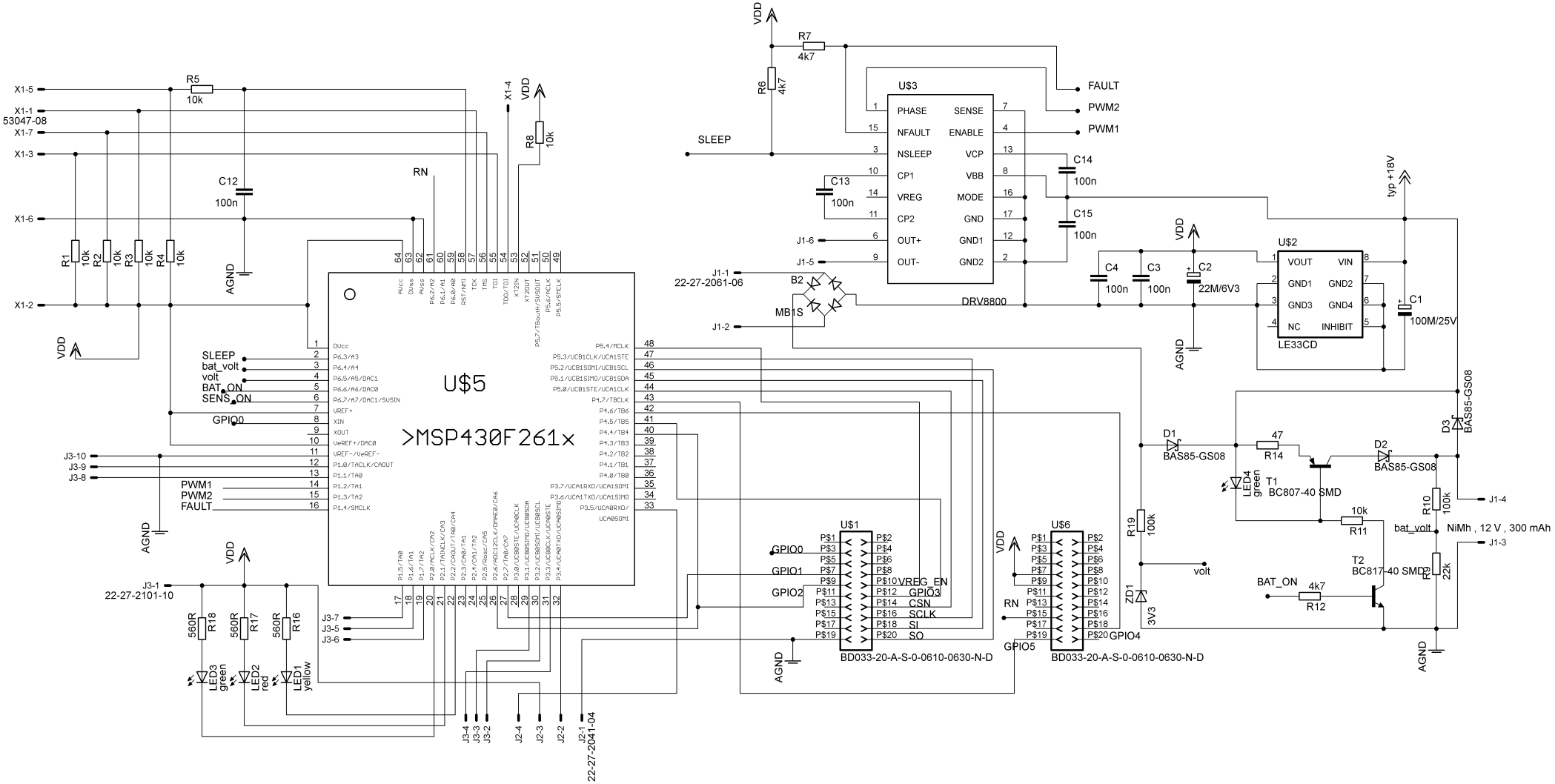
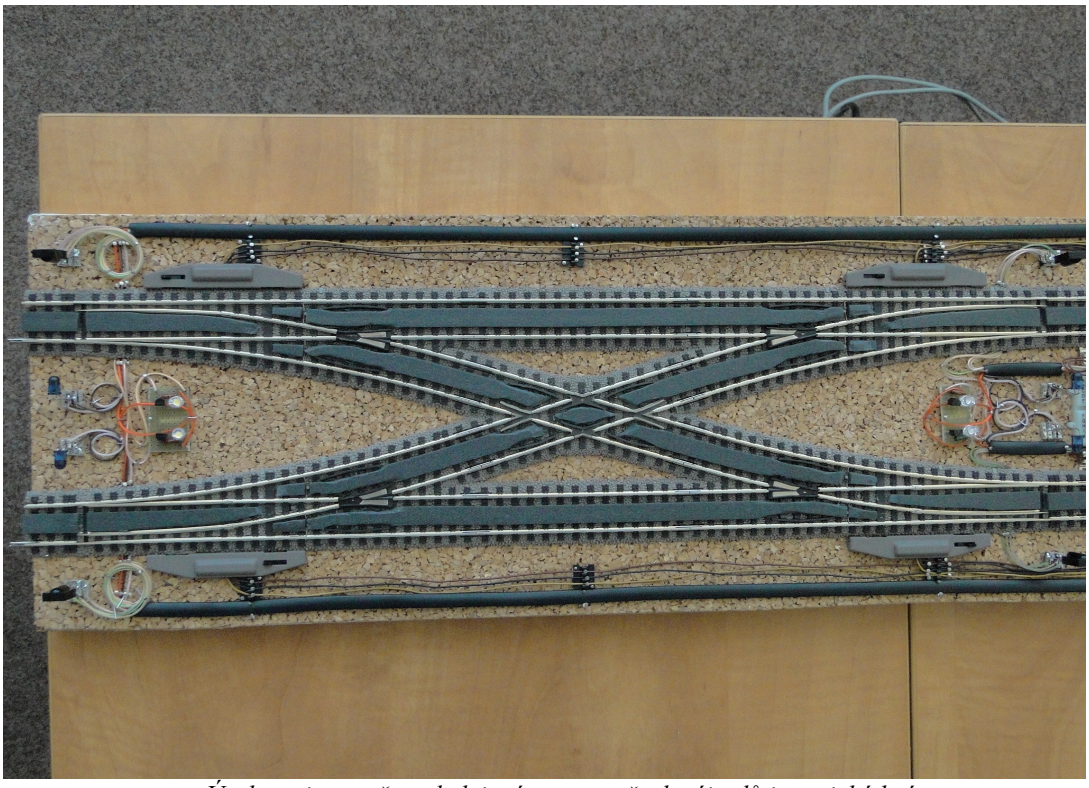
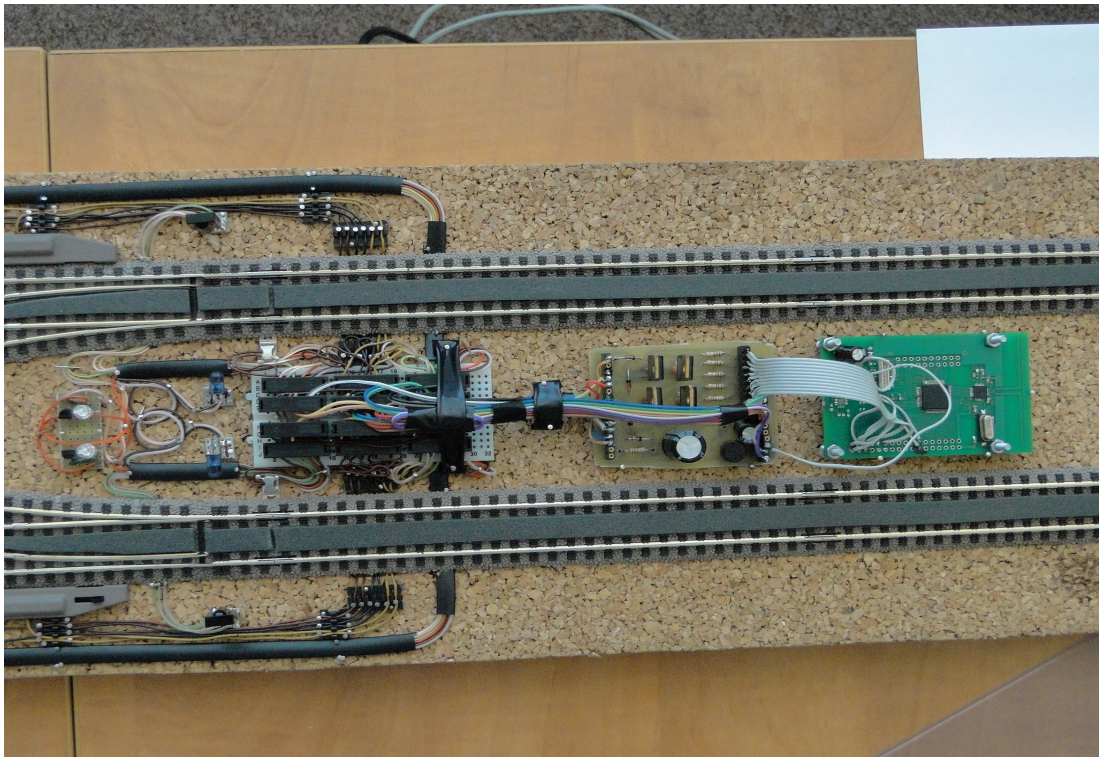


Schéma zapojeni hlavní desky

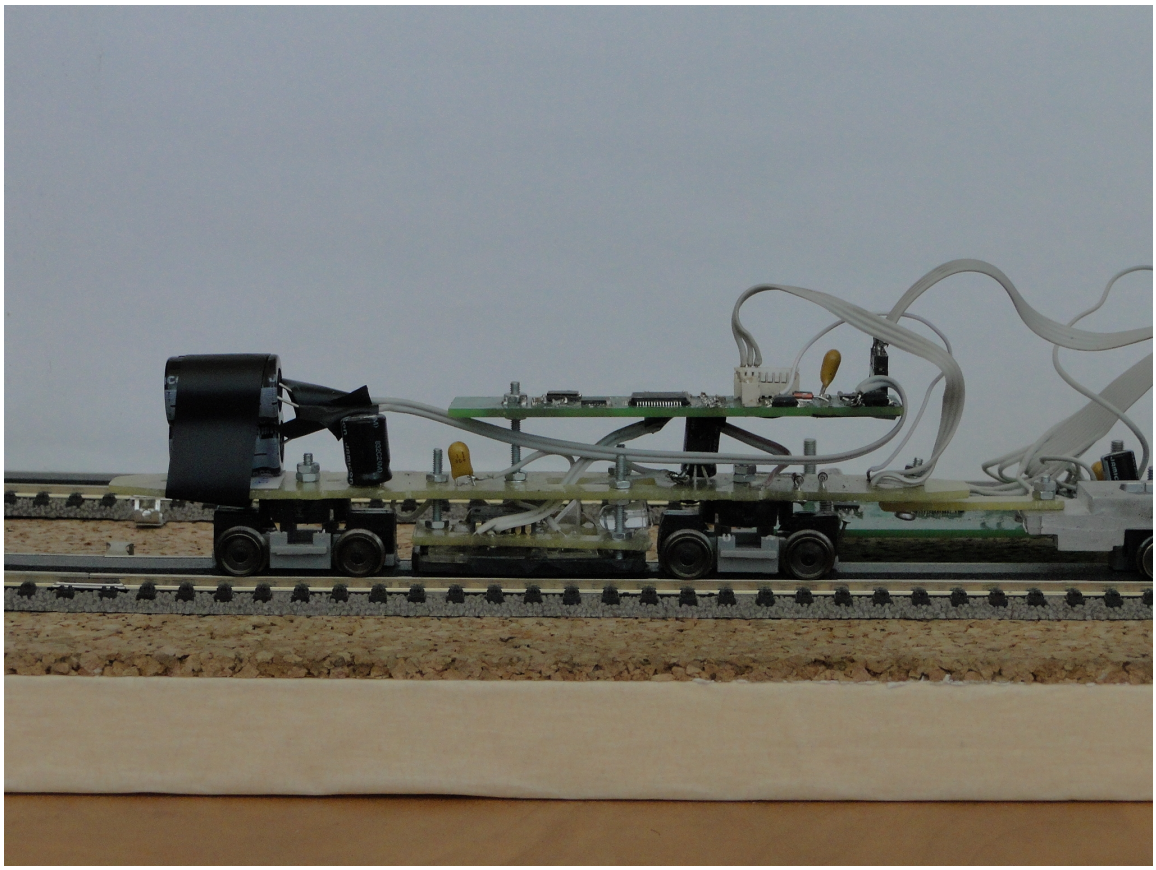
Příloha C, fotogalerie



Úsek trati se změnou kolejové cesty, u všech výjezdů je optická brána



Stacionární jednotka model 2010, transceiver bezdrátové sítě se skládaným dipólem



Snímací vagonek model 2010, původně zamýšlen jako podvozek modelové tramvaje

Snímací vůz model 2010, akumulátory jsou provizorně přímo napojeny na napětí 17,5 V