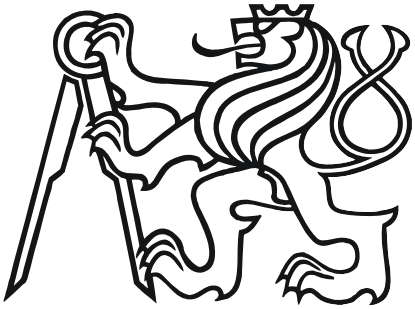
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA ELEKTROTECHNICKÁ

KATEDRA ŘÍDICÍ TECHNIKY



DIPLOMOVÁ PRÁCE

Robotičtí fotbaloví hráči

Praha, 2006 Petr Kovačik

Katedra řídicí techniky *Školní rok:* 2004/2005

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Bc. Petr K o v a č i k

Obor: Technická kybernetika

Název tématu: Robotičtí fotbaloví hráči

Z á s a d y p r o v y p r a c o v á n í :

1. Seznamte se s pravidly robotického fotbalu kategorie MiroSot.
2. Navrhněte schéma zapojení řídicí desky robotu. Řídit se budou dva stejnosměrné motory vybavené IRC snímači. Deska bude komunikovat s okolím pomocí technologie Bluetooth a bude umožňovat připojení volitelných rozšiřujících periferií.
3. Navrhněte, osaďte a oživte plošný spoj.
4. Naprogramujte řídicí software robotu. Ten bude obsahovat regulátory pro řízení motorů a modul pro komunikaci přes Bluetooth.
5. Pro vývoj software použijte vývojový řetězec z GNU nástrojů (gcc, gdb).

Seznam odborné literatury: Dodá vedoucí práce.

Vedoucí diplomové práce: Ing. Michal Sojka

Termín zadání diplomové práce: zimní semestr 2004/2005

Termín odevzdání diplomové práce: leden 2006

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

L.S.

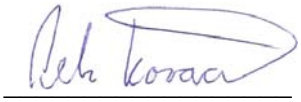
prof. Ing. Vladimír Kučera, DrSc.
děkan

V Praze dne 07.03.2005

Prohlášení

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

V Praze dne 25.5. 2006



Petr Kovačik

Abstrakt

V předkládané diplomové práci je popsán návrh robotu pro robotický fotbal. Robot musí splňovat podmínky kladené organizací FIRA (*Federation of International Robot-soccer Association*). Komunikace mezi robotem a osobním počítačem je řešena bezdrátovou technologií bluetooth. Mezi hlavní části diplomové práce patří návrh elektroniky robotu s možností rozšíření o další periferie, realizování a osazení plošného spoje a napsání knihovny pro obsluhu síťového protokolu technologie bluetooth. Řídicí software robotu, který tvoří regulátory dvou stejnosměrných motorů, byl převážně převzat z dřívějších projektů. Vývoj programového vybavení robotu je tvořen vývojovým řetězcem z GNU nástrojů.

Abstract

This diploma thesis deals with design of a robot specialized on playing robotic soccer. The robot has to satisfy rules of FIRA (*Federation of International Robot-soccer Association*). Communication between the robot and PC is solved by wireless technology bluetooth. The main contributions of the thesis are: design of the robot electronic with possible extensions of additional peripheries, realization and design of the printed circuit board, creation of a Bluetooth protocol library. Control software of the robot, based on control of two DC engines, builds on previous projects. The robot software is developed with the GNU toolchain.

Chtěl bych především poděkovat Michalu Sojkovi,
za jeho pomoc, za cenné připomínky a postřehy, jejichž důležitost a významnost
jsem si často uvědomoval trochu opožděně.
Dále bych chtěl poděkovat těm, v jejichž přítomnosti jsem si
mohl odpočinout a mohl přejít na jiné myšlenky…

Obsah

1 Úvod

Nápad sestavit „fotbalový tým“ složený pouze z malých robotů za účelem hry fotbalu vznikl v Koreji kolem roku 1995. Následně byla založena mezinárodní organizace FIRA (*Federation of International Robot-soccer Association*) definující ke dnešnímu dni dvě rozdílné kategorie. *Small League MiroSot,* ve které hrají v každém týmu tři roboti a *Middle League MiroSot*, kde hraje v týmu robotů pět včetně brankaře. Jednotlivé kategorie se od sebe odlišují, nejenom počtem hráčů, ale i velikostí hřiště.

Samotný fotbalový zápas musí probíhat, až na vyjímky, bez zásahu člověka. Pro splnění tohoto předpokladu je nutné znát polohu všech hráčů-robotů a míče na hřišti v každém okamžiku. Podle pravidel organizace FIRA, informaci o aktuální poloze zajišťuje barevná kamera umístěná nad hřištěm. Kamera pro určení aktuální pozice robotu, snímá čtyři barevné čtverce vyobrazené na horní ploše krychle každého „hráče“ [1][2].

Obr. 1.1 Vzhled hráče - robota s hracím míčkem

Podle ústavu automatizace a měřicí techniky Fakulty elektrotechniky a informatiky VUT v Brně (ÚAMT) [1], několikanásobného mistra ligy, můžeme celý problém strategie řízení rozdělit do čtyř úrovní. Na nejvyšší úrovni použitím metod umělé inteligence a počtu pravděpodobnosti program vybírá nejvhodnější akce, jakou může být např. dotek míče, „kopnutí“, obrana aj. Ve druhé úrovni se naplánuje odpovídající dráha robotu (často s ohledem na spolupráci s dalšími roboty). Ve třetí úrovni je

potřebné najít pro naplánovanou dráhu odpovídající pohyby kol. Poslední nejnižší úroveň potom již zajišťuje realizaci potřebných impulsů pro napájení motorků robotu.

Na základě vymyšlené strategie řídicí počítač musí informovat jednotlivé hráče o změně stavu jejich pohybu. Tato komunikace musí probíhat bezdrátově.

Mozkem celého robotu je mikrokontoler, mezi jehož hlavní úlohy patří řízení pohybu dvojice motorů, určování aktuální rychlosti a polohy robotu a dekódování přijatých povelů. Nevhodná volba mikrokontroléru může výrazným způsobem zhoršit kvalitu celého návrhu, což se může negativně projevit na hracích schopnostech robotu. Vhodný a postačující se nám jeví mikrokontrolér od firmy Hitachi 2638 z rodiny H8S, pracující na kmitočtu do 20MHz a obsahující pro nás vhodné periférie přímo na čipu. V popisované diplomové práci jsou využívány periferie k zajištění všech řídicích a komunikačních činnosti robotu. Je nutné zmínit, že náš robot vybavený mikrokontrolérem H8S /2638 zdaleka nedosahuje maximální výpočetní a obvodové výbavy, která je dostupná na trhu. Z tohoto důvodu nelze ani očekávat, že konkurenční týmy nemůžou mít lépe vybaveného hráče – robota.

Pohyb robotu je zajišťován prostřednictvím dvou stejnosměrných motorků, v našem případě od firmy Maxon, která tvoří světovou špičku ve svém oboru. Každý z dvojice motorů pohání jedno kolo robotu a tak je možno libovolného pohybu po hrací ploše. Maximální napětí připojené na vinutí je voleno kolem 10V. Tato hodnota je limitována počtem akumulátorů, které zabírají spolu s motory většinu objemu robotu, který má předepsané maximální rozměry 73x73x73mm.

Pro napájení každého z motorků je použit nezávislý zesílený signál PWM (*Pulse Width Modulation*) generovaný přímo v použitém mikrokontroléru.

Aktuální rychlost motoru je snímaná pomocí vstupních signálů vyvedených z IRC snímačů uložených v ose každého z motorů. Vzdálenost mezi dvěma pulsy představuje aktuální rychlost. Samotné měření rychlosti provádí i v tomto případě mikroprocesor. Opět je výhodné ušetřit procesorový čas a zjednodušit náročnost programu, použitím vhodné vnitřní jednotky – nabízí se TPU (*Timer Pulse Unit*) [4].

V závislosti na vypočtené aktuální rychlosti a požadované rychlosti, kterou obdržíme od řídicího počítače jsme schopni použitím vhodného regulátoru nastavit výše popsaný výstupní signál PWM. Jako vhodný se jeví regulátor PSD (*Proporcionálně Sumační Diferenční*), který je diskrétní obdobou PID regulátoru [5].

Při výběru součástkové základny byl hlavní důraz kladen na kvalitu a elektrické vlastnosti součástek a teprve na druhém místě se přihlíželo k jejich ceně. Jelikož rok od

roku se úroveň ligy, kterou organizuje FIRA zvyšuje, tak pouze cestou moderního pohledu na řešení se dá obstát ve velké konkurenci. Na zlepšení výsledků v lize má podstatný vliv především program v řídicím počítači vytvářející strategii hry, ale kvalitně pracující hráč-robot k samotnému zlepšení herních vlastností musí nutně také přispět.

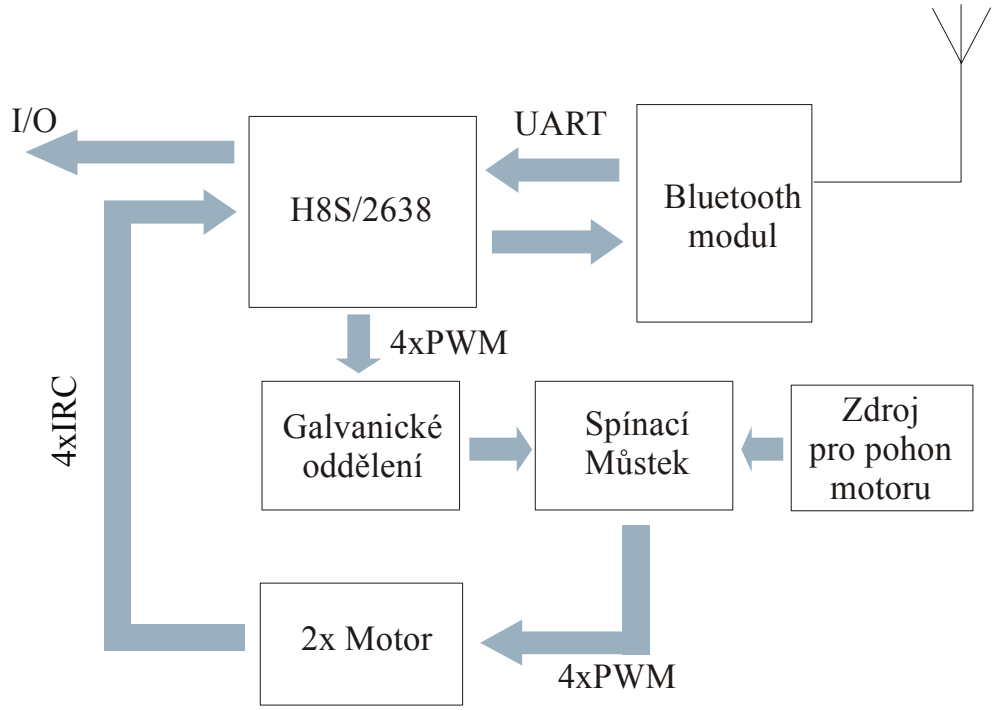
V následujících kapitolách bude popsána elektronika robotu a jeho programová výbava, která vznikla jako součást popisované diplomové práce. Následující kapitola 1 je věnovaná obvodovému řešení robotu. V kapitole 2 jsou popsány registry periférií mikrokontroléru H8S /2638, které se podílejí na generování PWM signálu (*PWM jednotka* kapitola 3.1) a na měření rychlosti motorů robotu (*TPU jednotka* kapitola 3.2). Další kapitoly jsou převážně věnovány komunikaci robotu s okolím pomocí architektury bluetooth. V kapitole 3 budou stručně nastíněny teoretické informace o architektuře bluetooth a dále v kapitolách 4 a 5 bude popsána aplikace, která zajišťuje tuto komunikaci. Zároveň v kapitole 4 budou stručně uvedeny i další knihovny, které jsou použity pro činnost robotu.

2 Obvodový návrh

Následující podkapitoly popisují jednotlivé obvodové části robotu. Uvedené pořadová čísla součástek odpovídají číslům ze schématu uvedeného v příloze A.

2.1 Blokové schéma robotu

Celkové obvodové řešení navrhovaného robotu můžeme vyjádřit blokovým schématem na Obr. 2.1.



Obr. 2.1 Blokové schéma robotu

Hlavní jednotkou celého zapojení je mikrokontrolér H8S/2638, ve kterém je soustředěna většina schopností robotu. Jedná se především o regulační mechanismus pro řízení motorů, zpracování signálu o aktuální poloze motorů, generování PWM signálů a vnější komunikace (bluetooth, SCI) s nadřazeným systémem (PC).

Na blokovém schématu vyobrazeném na Obr. 2.1 není z důvodu čitelnosti vyjádřeno napájecí napětí pro jednotlivé bloky. Většina obvodových prvků je napájena napětím 5V odvozeného od napájecího napětí přímo na plošném spoji. Další hodnota napětí je 3,3V, potřebná pouze pro napájení použitého bluetooth modulu.

V dalších kapitolách budou popsány jednotlivé bloky z návrhového hlediska podrobněji.

2.2 Motory

K pohonu robotu, jsou použity dva kartáčové stejnosměrné motory od firmy Maxon s označením 221013. Průměr motoru je Ø21mm. Jmenovitý výkon každého z motorů je 5W. Napájecí napěti může být maximálně do 18V. Motory jsou osazeny inkrementálním snímačem otáček s označením 201537, jehož maska má 512 inkrementů na jednu otáčku.

Otáčivý moment motoru je na kola robotu převeden pomocí ozubeného kola s následujícími parametry:

- čelní ozubení
- modul 0,4
- počet zubů 72
- tloušťka materiálu cca 2mm
- vnitřní otvor průřezu 9mm

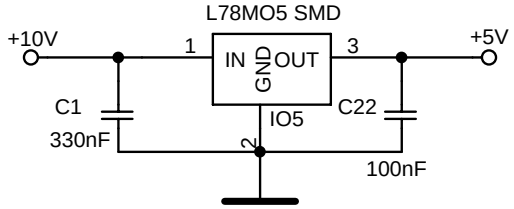
2.3 Ochranné prvky a úprava napětí

Jak již bylo uvedeno výše, tak v návrhu robotu se pracuje se třemi odlišnými hodnotami napětí. Jedná se o napětí vstupní, které je použito k odvození napětí ostatních a pak dále k napájení dvojice motorů. Tato hodnota je volena kolem 10V, ale může být použito vstupní napětí o hodnotě v širším intervalu. Spodní hranice je kolem 8V, od kterých je použitý výkonový můstek již schopen vytvářet spínané napětí pro motory. Horní hranice je dána maximem z hodnot napětí napěťových stabilizátorů, výkonového můstku (52V) a maximálním hodnotou, kterou lze ještě připojit na stejnosměrný motor (18V). Nebudeme-li uvažovat omezení motorů, tak je tato hodnota 18V pokud je plošný spoj osazen napěťovým regulátorem LE33C plnícím funkci napájení bluetooth modulu. Pokud bychom neosadili tento integrovaný obvod, tak se maximální hodnota posune na 35V (dáno obvodem 78M05).

Celé zapojení je možné odpojit od napájení přepínačem T1. Za tímto přepínačem se nachází ochranné diody D7 a D8 proti přepólování. Jejich umístění je pouze ve větvích pro vytvoření 5V a 3,3V, nikoliv v části pro napájení výkonového můstku, kde riziko přepólování potlačuje již použitý můstek. Ve stejných obvodových větvích, jako jsou diody, jsou umístěny i tlumivky L1 a L2, které slouží ke snížení rušení od motorů.

2.3.1 78M05

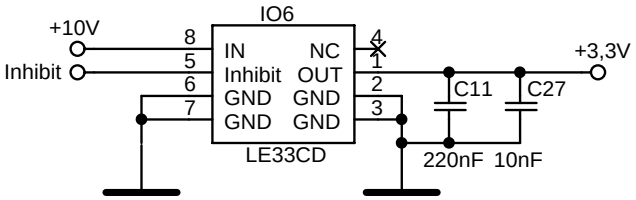
Zapojení napěťového stabilizátoru pro 5V je uvedeno na Obr. 2.2. Součástka je volena v SMD pouzdře. Obvod a okolní součástky jsou zapojeny podle doporučení výrobce. Hodnoty vstupního napětí výrobce by neměly překročit maximální hodnotu 35V.



Obr. 2.2 Zapojení napěťového stabilizátoru L78M05

2.3.2 LE33C

Pro napájení bluetooth modulu je použit obvod LE33C. Zapojení obvodu s doplňujícími součástkami je na Obr. 2.3. Jedná se o SMD integrovaný obvod, u kterého lze napětí na výstupních pinech potlačit přivedením nízké logické úrovně na svorku *Inhibit*. Zapojení je použito podle doporučení výrobce. Maximální hodnota vstupního napětí je 18V při výstupním proudu 100mA.



Obr. 2.3 Zapojení napěťového stabilizátoru LE33C

2.4 Mikrokontrolér H8S/2638

Nejdůležitější částí celého zapojení je mikrokontrolér. Mikrokontrolér se podílí na činnosti všech dílčích částí robotu. Jedná se především o generování PWM signálů, měření rychlostí a směrů otáčení motorů z fázově posunutých IRC signálů, nezávislou komunikaci s bluetooth modulem a osobním počítačem atd.

Z důvodu efektivního řešení robotu bylo nutné zvolit řídicí jednotku, co možná nejvhodněji. Použitý obvod H8S/2638 se jeví jako vhodný především díky vnitřním periferiím a své 32-bitové architektuře.

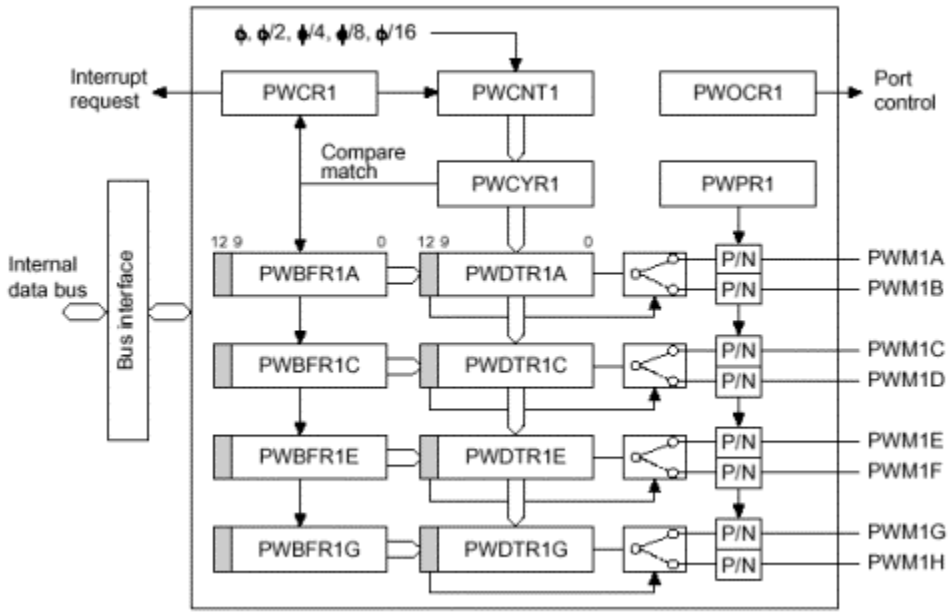
2.4.3 Šířkově pulsní modulátor PWM

Vlastnosti :

- dvě jednotky po osmi 10-bitových PWM kanálech
- každý kanál má svůj 10-bitový čítač
- lze nastavit polaritu u každého výstupu
- možnost pracovat v pěti taktovacích kmitočtech odvozených od frekvence krystalu (θ, θ/2, θ/4, θ/8, θ/16), kde θ představuje kmitočet použitého krystalu
- dva druhy zdrojů přerušení

Jak je uvedeno v seznamu vlastností PWM jednotky, tak na čipu jsou umístěny dva plně nezávislé PWM kanály. Každý z kanálů má osm výstupů. Pro naší aplikaci byl zvolen kanál 1, který má sice menší okruh využití než-li kanál 2, ale plně postačuje našim požadavkům. Rozdíly a popis kanálu 2 je uveden v [4].

Obr. 2.4 je vyjadřuje blokové schéma prvního PWM kanálu. Činnost jednotlivých registru bude naznačena v kapitole 3.1.



Legenda:

PWCR1:PWM řídicí registr 1
PWOCR1:PWM výstupní řídicí registr 1
PWPR1: PWM polarity registr 1
WCNT1:PWM čítač 1
PWCYR1: PWM cycle registr 1
PWDTR1A,1C,1E,1G:PWM duty registry 1A,1C,1E,1G
PWBFR1A,1C,1E,1G: PWM zásobníkové registry

Obr. 2.4 Blokové schéma prvního PWM kanálu

2.4.4 Časovací a pulsní jednotka TPU

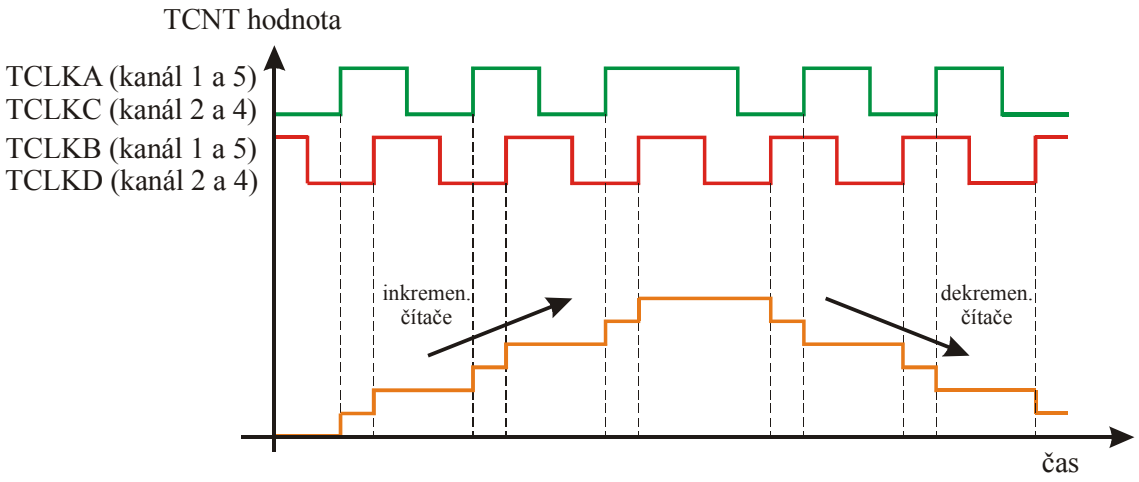
Vlastnosti :

- obsahuje šest 16-bitových časovacích kanálů
- každý z kanálů 0 a 3 má čtyři časovací registry TGRs (záchytný registr) a kanály 1,2,4,5 mají po dvou TGRs registrech.
- TPU umožňuje
    - zachytávat časové hrany vstupních signálů v závislosti na náběžné, spádové, nebo obou hranách
    - nulování čítače je možné při dosažení shody čítače s žádanou hodnotou, nebo vlivem zachycení hrany vstupního signálu
    - umožňuje, aby více čítačů pracovalo vzájemně synchronně
    - je možné současné nulování více čítačů vlivem shody čítače s žádanou hodnotou nebo vlivem zachycení hrany vstupního signálu
    - fázový mód může být použit nezávisle pro každý z kanálů 1,2,4,5
    - je možné inkrementovat resp. dekrementovat TPU čítač podle fázového posunu u dvoufázových signálů (vhodné pro IRC snímače)
    - je možné spojovat časovací kanály a vytvářet tak složitější struktury
- možno generovat až 26 zdrojů přerušení. Kanály 0 a 3 mají čtyři zdroje přerušení od porovnání stavu čítače a žádané hodnoty (i počtem vstupních hran pulsů) a jeden zdroj přerušení od přetečení čítače. Kanály 1,2,4,5 mají dva zdroje přerušení od porovnání stavu čítače a žádané hodnoty (i počtem vstupních hran pulsů) a po jednom zdroji přerušení od přetečení a podtečení čítače

Každý z motorů, který pohání kola robotu je osazen inkrementálním snímačem otáček. Snímač generuje vzájemně fázově posunuté signály s šířkou pulsu odpovídající rychlosti motoru. Tento signál je přiváděn na vstupy TCLKA a TCLKB v případě prvního motoru a TCLKC a TCLKD v případě druhého motoru.

Pro činnost měření rychlosti (změny polohy) motoru jsou využívány tři TPU kanály. Časovací kanál 4 je využit ke generování vzorkovacích period. Dosažení vzorkovací periody je určeno shodou mezi počtem načítaných pulsů vnitřních hodin a představenou hodnotou záchytného registru TPU_TGR4A. Hodnota v registru TPU_TGR4A určuje vzorkovací periodu. Po dosažení vzorkovací periody je vyvoláno přerušení EXCPTVEC_TGI4A.

Spolu s časovacím kanálem 4 pracuje souběžně i časovací kanál 2 pro jeden z motorů a časovací kanál 5 pro druhý z motorů. Kanály 2 a 5 fázově počítají náběžné hrany podle schématu vyobrazeném na Obr. 2.5. U těchto TPU kanálů není generován žádný ze zdrojů přerušení. Stav čítače kanálu 2 resp. 5 (dle motoru) se čte až v obsluze přerušení od TPU_TGR4A.

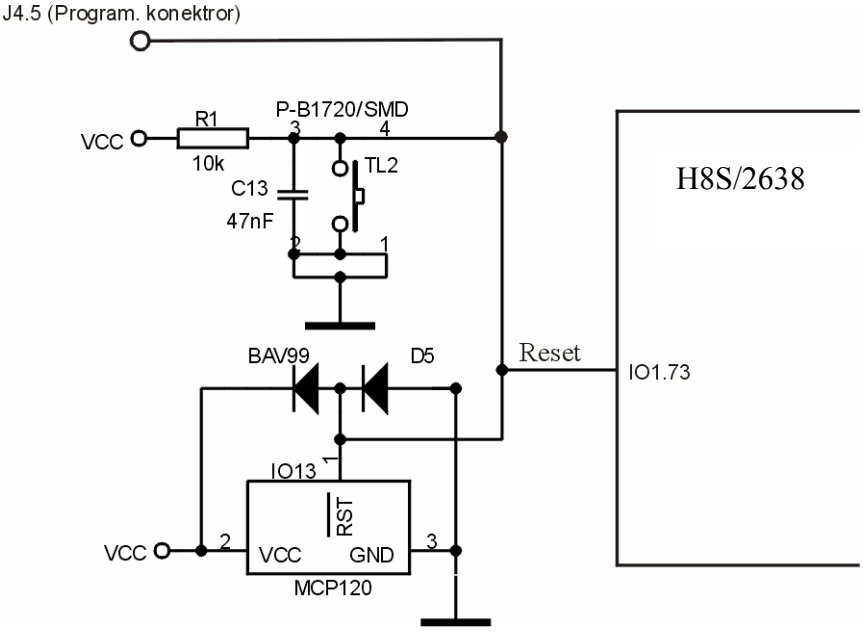


Obr. 2.5 Fázový čítací mód 1

2.4.5 Zapojení RESET a hlídání poklesu napětí

Obvodové zapojení související se vstupem RESET u mikrokontroléru je naznačeno na Obr. 2.6. Obvod MCP120 je hlídající obvod s otevřeným kolektorem. Jeho činnost je založena na držení nízké logické úrovni v době, kdy je napájecí napětí (napětí na pinu 2) nižší než cca 4,5V. Dvoj-dioda BAV99 je použita z důvodu potlačení napěťových špiček při spínání obvodu.

Pro snadnější manipulaci s robotem a především pro snadnější ladění programu, je plošný spoj robotu osazen spínačem, pro možnost resetování mikrokontroléru.



Obr. 2.6 Zapojení RESET

2.4.6 Programování H8S/2638 a nezařazené obvodové řešení u H8S/2638

Převodníky na úpravu napěťových úrovní mezi sériovým rozhranním osobního počítače a mezi sériovým rozhranním mikrokontroléru není osazeno na plošném spoji robotu. Externí převodník je nutné připojit na konektor J4.

Aby mohl obvod H8S/2638 spouštět uživatelský program uložený ve vnitřní paměti FLASH je nutné správně nastavit režim procesoru a úroveň na pinu FWE. To se děje prostřednictvím konektoru J8 (režimy) a jumperu J10 (FWE). V tomto případě postačuje umístit propojku na J10 a konektor J8 nepropojovat. Pokud chceme do vnitřní paměti zavádět bootstrap je nutné propojit piny odpovídající MOD0 (MD0) a MOD2 (MD2). Propojky J10 a MOD1 (MD1) propojeny nejsou.

V následujícím textu budou stručně zmíněny další obvodové součástky, které byly použity při návrhu robotu. Pořadová čísla součástek korespondují se schématem v příloze A.

K mikrokontroléru může být připojen krystal o maximální frekvenci, kterou uvádí výrobce obvodu H8S/2638, 20MHz. Hodnoty kondenzátorů C7, C8, C9, R6, které souvisí s činností krystalu, jsou použity o výrobcem doporučených hodnotách. Hodnoty dalších obvodových součástek C12, C14 jsou také zvoleny podle doporučení.

Rezistory R7, R8, R9 a R19 umístěnými mezi inkrementálním čidlem a mikrokontrolérem slouží ke snížení vstupního proudu a jsou voleny o hodnotě 1kΩ.

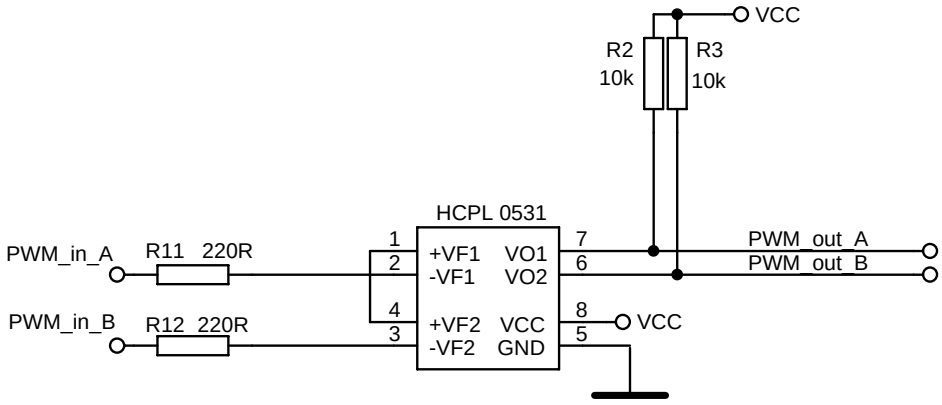
Při rozvrhování plošného spoje byly ke každému napájecímu vstupu u procesorového obvodu H8S/2638 a i u jiných integrovaných obvodů robotu umístěny filtrační kondenzátory o hodnotách 100nF. Jedná se kondenzátory C2, C3, C20, C21, C23, C24.

K mikrokontroléru jsou připojeny čtyři svítivé diody LED, které můžou být volně využívány programátorem. Diody jsou zapojeny proti kladnému napětí, což snižuje proudové namáhání brány obvodu H8S/2638. Diodám jsou předřazeny rezistory R16, R17, R18, R31 o hodnotě 1kΩ ke snížení proudu procházejícího diodami.

2.5 Galvanické oddělení

Na plošném spoji robotu jsou celkem tři optočleny. Dva slouží ke galvanickému oddělení PWM signálů mezi procesorem a výkonovým můstkem. Druhý typ optočlenu je použit k blokovacím/povolovacím účelům - víz dále.

Na Obr. 2.7 je znázorněno zapojení jednoho ze dvou optočlenů HCPL0531, který plní první úlohu. Mezi požadavky na výběr optočlenu bylo umístění dvoucestného galvanického oddělení v jednom pouzdře a možnost přenést na výstup velice úzké pulsy. Jejich šířka závisí na frekvenci PWM signálu a na rozlišení PWM jednotky mikrokontroléru H8S/2638.



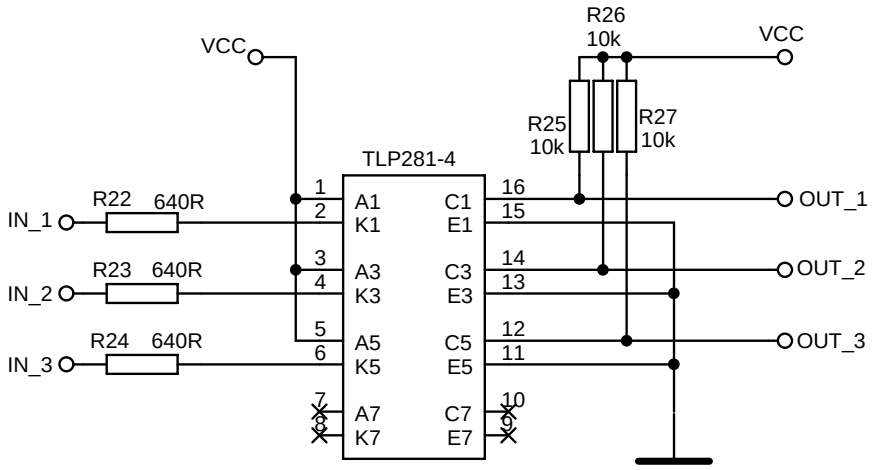
Obr. 2.7 Použité zapojení jednoho z optočlenů HCPL0531

Pokud se rozhodneme použít PWM signál o kmitočtu cca 20kHz, z důvodu potlačení slyšitelnosti, s rozlišením 10 bitů (vlastnost H8S/2638) můžeme vypočítat, že nejužší puls bude mít šířku :

$$\tau_{PWM} = \frac{1}{f_{PWM} \cdot 2^n} = \frac{1}{20000 \cdot 2 \cdot 1024} = 488ns\,, \qquad \text{n–počet bitů PWM (n = 10).}$$

Sled těchto nejtenčích pulsů by odpovídal přenosové rychlosti cca 2Mbit/s. Energie, kterou by dodal puls o vypočtené šířce není schopen motor převést na otáčivý moment. Obvod, který jsme použili pro návrh robotu má přenosovou rychlost 1Mbit/s, což pro naší aplikaci plně postačuje. Diody tvořící vstupní stranu oddělovače HCPL0531 jsou spínány kladnou úrovní vstupního napětí VCC proti hodnotě PWM signálu. Hodnoty vstupních rezistorů R11, R12 (první optočlen HCPL0531), R13, R14 (druhý optočlen HCPL0531), jsou voleny o hodnotě 220Ω. Tato hodnota odpovídá proudu 22mA, který prochází vstupními diodami. Výstupní proud oddělovačů je zesílen proudem tekoucím ze zdroje Vcc přes výstupní rezistory. Hodnota výstupních rezistorů by měla být vyšší než 1,8kΩ s ohledem na rychlost odezvy vůči vstupu. V našem případě jsou hodnoty výstupních rezistorů R2, R3, R4, R5 voleny o hodnotě 10kΩ.

Na Obr. 2.8 je znázorněno zapojení optočlenů TLP281-4. Požadavky na rychlost u tohoto obvodu nebyly kladeny, protože obvod nepřenáší trvale pulsující signál, ale pouze povolení resp. omezení činnosti jiného obvodu.



Obr. 2.8 Použité zapojení pro optočlen TLP281-4

Hlavní výhodou použitého obvodu je, že umožňuje galvanicky oddělit hned čtyři kanály.

V naší aplikaci jsou jednotlivé oddělovací cesty použity pro :

- cesta 1 : potlačení činnosti výkonového můstku pro motor A
- cesta 2 : potlačení činnosti výkonového můstku pro motor B

- cesta 3 : potlačení činnosti pro generování výstupního napětí u napěťového stabilizátoru LE33C
- cesta 4 : nevyužitá

Hodnoty vstupních rezistorů R22, R23, R24, jsou voleny o hodnotě 220Ω. Tato hodnota odpovídá proudu 8mA (doporučeno výrobcem obvodu), který prochází vstupními diodami. Výstupy oddělovačů jsou zesíleny proti kladnému napětí. Hodnoty výstupních rezistorů R2, R3, R4, R5 jsou voleny 10kΩ, tak aby se zamezilo zbytečně vysokému výstupnímu proudu.

2.6 Výkonová část

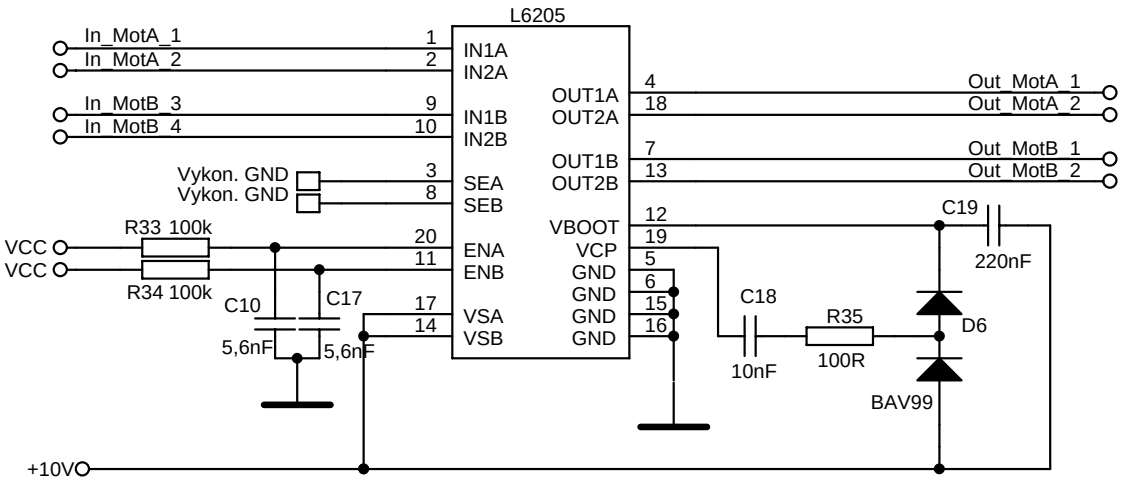
Obvod L6205, tvořící základ výkonové části, je dvojitý můstek, který zesiluje PWM signál generovaný mikrokontrolérem na signál stejného průběhu, ale amplitudově zesílený na hodnotu Vs, v našem případě tedy na cca. 10V.

Hlavní přednosti obvodu L6205 můžeme shrnout do následujících bodů

- dva paralelně pracující můstky soustředěné v jednom integrovaném obvodu
- pouzdro SMD
- frekvence spínání 100kHz
- spínací odpor 0,34Ω
- možnost odpojení výstupu po přivedení logické úrovně 0 na vstup EN.

Na Obr. 2.9 je uvedeno zapojení obvodu L6205. Uspořádání a hodnoty všech obvodových prvků jsou voleny podle doporučení výrobce.

U obvodu je vhodné oddělit výkonovou zem (Výkon. GND) od země digitální (GND), která je společná pro všechny ostatní součástky robotu. V našem případě jsou obě země spojeny v blízkosti napájecího konektoru J1 (viz příloha A). Vodič připadající výkonové zemi není veden na průmyslově vyrobeném plošném spoji, ale je nutné ho vést prostřednictvím drátku napájeném na plošky označené ve schématu "*Vykon. GND"*.



Obr. 2.9 Použité zapojení pro výkonový můstek L6205

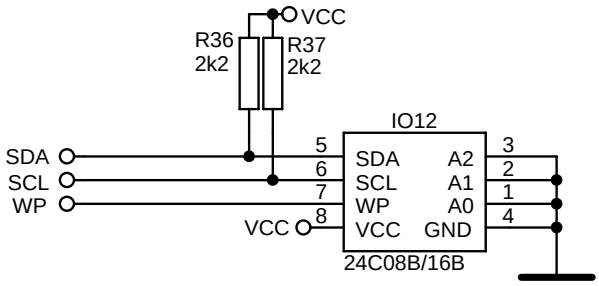
2.7 Sériová paměť EEPROM

Pro možnost ukládání konfiguračních (měnitelných) dat, které je třeba uchovat i po dobu, po kterou robot není připojen k napájení, je možno plošný spoj osadit sériovou pamětí 24C08B/16B. K paměti lze přistupovat prostřednictvím protokolu I^2C. Použitý mikrokontrolér má rozhranní I^2C pouze v rozšířené tzv. *W-mask* verzi. Navíc tyto vývody se překrývají s vývody pro sériové rozhranní SCI. Z tohoto důvodu byla sériová paměť připojena na klasickou I/O bránu a komunikační protokol I^2C je nutné obsluhovat programovou cestou bez využití hardwarových podpor mikrokontroléru.

Na Obr. 2.10 je zapojení sériové paměti. Vodič SDA slouží k zasílání sériové adresy a vstupně výstupních dat, vodič SCL slouží k přivedení synchronizačního signálu, a vodič WP umožňuje při logické úrovni 0 blokovat paměť pro zápis.

Paměť je organizována osmy bloky, kde každý blok má velikost 256 bytů, tj. velikost paměti v bitech je 8x8x256.

Vstupní proud výrobce uvádí 3mA. Z tohoto důvodu jsou hodnoty rezistorů R36 a R37 voleny 2,2kΩ.



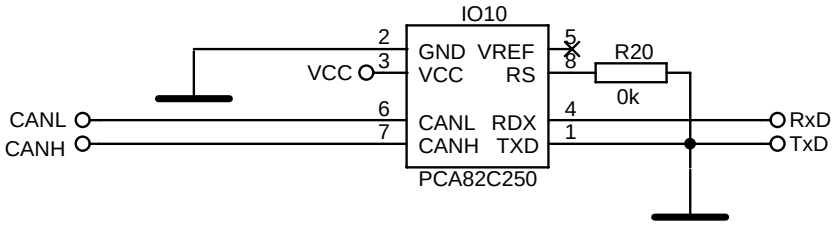
Obr. 2.10 Použité zapojení pro sériovou paměť 24C08B/16B

2.8 Rozhranní CAN

Při navrhování schématu robotu se počítalo s jeho možným pozdějším rozšířením, případně i s využitím plošného spoje k jiným účelům, než-li je využití v robotickém fotbale.

Jelikož mikrokontrolér H8S/2638 obsahuje na čipu dvě CAN rozhranní, byl kladen požadavek na možnost jejich využívání v budoucím rozšíření. Z tohoto důvodu byly do schématu přidány obvody PCA82C250, které tvoří budičem CAN sběrnice.

Na Obr. 2.11 je použité zapojení s budičem PCA82C250. Rezistor R20 (R19) slouží k výběru ze tří druhů činnosti.



Obr. 2.11 Použité zapojení pro budič CAN PCA82C250

2.9 Bluetooth modul a podpůrné obvody

Bluetooth zařízení v zapojení robotu slouží ke komunikaci mezi osobním počítačem (mobilním telefonem atd.) a robotem. V této kapitole bude popsáno pouze obvodové začlenění bluetooth modulu do robotu.

Použité bluetooth zařízení má označení WML-C20. Nejedná se o klasický integrovaný obvod, ale modul, který je již výrobcem osazen na samostatném plošném spoji. Takto opatřený modul je možné napájet na navrhovaný plošný spoj.

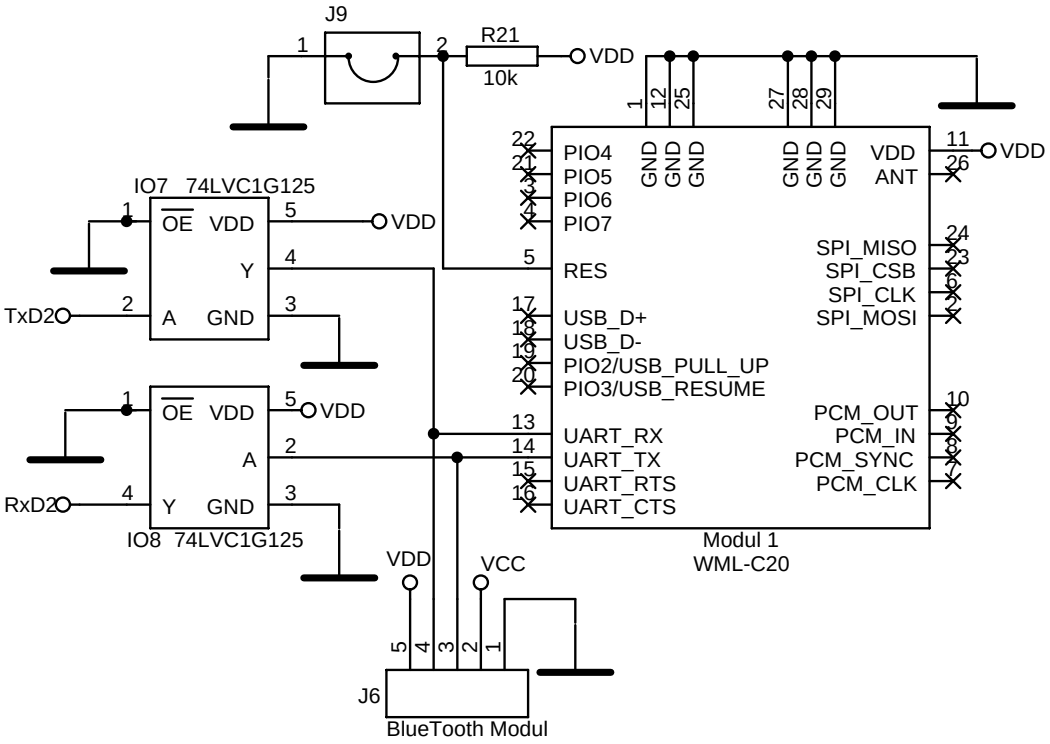
Pro správnou činnost bylo nutné umístit modul na okraj plošného spoje a zajistit tak, co možná nejmenší rušení způsobené provozem elektroniky. Ze stejného důvodu není v místech antény na obou stranách plošného spoje rozlita měď.

Základní vlastnosti modulu WML-C20 jsou

- možnost použití komunikačních rozhranní USB, UART, PCM
- maximální datový přenos pro synchronní režim – 433,9kbps/433,9kbps a pro asynchronní režim 723,2kbps/57,6kbps
- obvod splňuje podmínky standartu bluetooth specifikace verze 1.1. class 1
- maximální odebíraný proud 150mA při 3,3V
- integrovaná anténa
- komunikace po UART 115kbaud

Obvod WML-C20 pracuje s napájecím napětím 3,3V. Z toho důvodu je plošný spoj osazen napěťovým regulátorem LE33C který byl popsán v podkapitole 2.3.2.

Zapojení bluetooth zařízení a podpůrných obvodů je vyobrazen na Obr. 2.12. Datový asynchronní signál TxD z mikrokontroléru je nutné napěťově upravit na hodnotu 0-3V. Úpravu napětí provádí integrovaný obvod IO7 typu 74LV1G125. Pro signál RxD není nutné úroveň zpětně upravovat, na 0-5V, protože hodnota 3V odpovídající již logické jedničce. Plošný spoj je ovšem osazen převodníkem i ve zpětném směru (RxD) a integrovaný obvod IO8 (74LV1G125) zde zajišťuje určité oddělení mezi mikrokontrolérem a zařízením bluetooth. Propojení svorek J9 způsobí hardwarový reset bluetooth zařízení. Konektor J6 slouží k možnosti zapojení jiného vysílacího zařízení nebo k zapojení externího bluetooth zařízení.



Obr. 2.12 Zapojení bluetooth a okolních obvodů

3 Nastavení registrů mikrokontroléru H8S/2638

Mezi hlavní přednosti mikrokontroléru H8S/2638 patří jeho periférie, které umožňují snadněji zpracovávat vnější události a šetřit tak strojový čas procesoru.

V následujících podkapitolách budou popsány registry, které umožňují nastavit PWM a TPU jednotku.

3.1 PWM jednotka

Jedna z periférií mikrokontroléru H8S/2638 je PWM jednotka, která umožňuje generovat pulsní výstupy.

3.1.1 Charakteristické rysy:

- maximálně 16 pulsních výstupů
  - dva 10-bitové PWM kanály, každý s osmi výstupy
  - každý kanál obsahuje 10-bitový čítač *(PWCNT)* a cyklický registr (*cycle register*) *(PWCYR)*
  - Provozní a výstupní polarita může být nastavena pro každý výstup
- buffer registr
  - Data z *duty* (služebního) *registru (PWDTR)* jsou automaticky přesunuta do *buffer registru (PWBFR)* automaticky v každém cyklu
  - Kanál 1 obsahuje 4x *duty registr* a 4x *buffer registr*
  - Kanál 2 má 8x *duty registr* a 4x *buffer registr*
- pět možností taktování čítače PWM jednotky (ϕ, $\phi/2$, $\phi/4$, $\phi/8$, $\phi/16$), kde ϕ představuje rychlost použitého krystalu
- připojení PWM jednotky k 16-bitové vysokorychlostní sběrnici
- dva zdroje přerušení
  - Pro každý kanál může být generováno nezávislé přerušení

3.1.2 Popis registrů

PWCR1, PWCR2 – řídicí registr PWM

7	6	5	4	3	2	1	0
–	-	IE	CMF	CST	CKS2	CKS1	CKS0
0	0	0	0	0	0	0	0

IE	Povolení přerušení : ‚1‘ - přerušení generováno v případě, že načítaná hodnota je shodná s hodnotou v registru PWCYR
CMF	Porovnávací příznak (*Compare Match Flag*) : indikuje případ, když PWCNT=PWCYR (úroveň ‚1‘)
CST	Začátek čítání (*Counter Start*): úroveň ‚0‘ – PWCNT je zastavena, úroveň ‚1‘ - spuštění PWCNT
CKS2 **CKS1** **CKS0**	0 – 0 – 0 : vnitřní čítání ϕ/1 0 – 0 – 1 : vnitřní čítání ϕ/2 0 – 1 – 0 : vnitřní čítání ϕ/4 0 – 1 – 1 : vnitřní čítání ϕ/8 1– * - * : vnitřní čítání ϕ/16

PWOCR1, PWOCR2 – PWM výstupní kontrolní registr 1 a 2 – 8-bitový registr, který umožňuje (zakazuje) PWM výstup. PWOCR1 – výstupy PWM1H až PWM1A, PWOCR2 – výstupy PWM2H až PWM2A

7	6	5	4	3	2	1	0
OEnH	OEnG	OEnF	OEnE	OEnD	OEnC	OEnB	OEnA
0	0	0	0	0	0	0	0

n=1,2

OE ‚0‘(‚1‘) – PWM výstup je zakázán (povolen)

PWPR1, PWPR2 – PWM polarity registr (*PWM Polarity Registers 1 and 2*)
nastavuje polaritu PWM výstupu.

7	6	5	4	3	2	1	0
OPSnH	OPSnG	OPSnF	OPSnE	OPSnD	OPSnC	OPSnB	OPSnA
0	0	0	0	0	0	0	0

n=1,2

OPS ‚0'(‚1') – PWM přímý (inverzní) výstup

PWCNT1, PWCNT2 – PWM čítač 1 a 2 (*PWM Counters 1 and 2*)

10-bitový čítač (up/down) inkrementovaný hodinovým vstupem. Nulování čítače nastává při resetu nebo při nulování bitu CST v registru PWCR. Rychlost čítání je závislá na nastavení bitů CKS2 - CKS0 v registrů PWCR Inicializační hodnota – FC00h. Hodnotu není možno přímo nastavovat programově.

PWCYR1, PWCYR2 - *PWM Cycle Registers 1 and 2*

16-bitový přepisovatelný registr. Hodnota slouží ke komparaci s hodnotou čítače PWCNT. Při shodné hodnotě je PWCNT vynulován a data jsou přesunuta ze zásobníkového (buffer) registru PWBFR do registru PWDTR. Do registru PWCYR by mělo být zapisováno jenom při zastaveném čítači PWCNT.

PWDTR1A, 1C, 1E, 1G - *PWM Duty Registers 1A, 1C, 1E, 1G*

Registry nastavující zdroj výstupu.

PWDTR1A – nastavuje PWM1A a PWM1B, *PWDTR1C* – nastavuje PWM1C a PWM1D, *PWDTR1E* – nastavuje PWM1E a PWM1F, *PWDTR1G* – nastavuje PWM1G a PWM1H.

Do žádného z těchto registrů nesmí být přímo zapsáno. Hodnota, která je uložena v registru PWBFR1, se do registru PWDTR1 kopíruje při shodě hodnot čítače PWCNT1 a registru PWCYR1.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	OTS	-	-	DT9	DT8	DT7	DT6	DT5	DT4	DT3	DT2	DT1	DT0
1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0

OTS	úroveň ‚0' – vybrán pulsní výstup A resp. (C, E, G) úroveň ‚1'– vybrán pulsní výstup B resp. (D, F, H)
DT	Nastavení těchto bitů představuje hodnotu, která při shodě aktuální hodnoty čítače PWCNT1 nastaví požadovaný výstup z úrovně ‚1' do ‚0' (OPS=‚0' registru PWPR) resp. z úrovně ‚0' do ‚1'(OPS=‚1'). Pokud hodnota bitů DT je 0, výstup PWM bude trvale v úrovni ‚0' (OPS=‚0'). Pokud hodnota v registru PWCYR1 je menší než hodnota bitů DT, bude trvale nastaven výstup PWM na ‚1' (OPS=‚0')

PWBFR1A, 1C, 1E, 1G - *PWM Buffer Registers 1A, 1C, 1E, 1G*

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	OTS	-	-	DT9	DT8	DT7	DT6	DT5	DT4	DT3	DT2	DT1	DT0
1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0

Stejně uspořádaný registr jako PWDTR1. Hodnota z PWBFR1 je při každé shodě hodnoty čítače PWCNT1 a hodnoty registru PWCYR1 nahrána do registru PWBFR1A**.**

PWDTR2A až PWBFR2H - *PWM Duty Registers 2A až 2H*

Obdoba registrů PWDTR1. Registry nastavující zdroj výstupu.

PWDTR2A – nastavuje PWM2A, *PWDTR2B* – nastavuje PWM2B, *PWDTR2C* – nastavuje PWM2C, *PWDTR2D* – nastavuje PWM2D, *PWDTR2E* – nastavuje PWM2E, *PWDTR2F* – nastavuje PWM2F, *PWDTR2G* – nastavuje PWM2G, *PWDTR2H* – nastavuje PWM2H.

Do žádného z těchto registrů nesmí být přímo zapsáno. Hodnota, která je uložena v registru PWBFR2, se do registru PWDTR2 kopíruje při shodě hodnoty čítače PWCNT2 a registru PWCYR2.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	DT9	DT8	DT7	DT6	DT5	DT4	DT3	DT2	DT1	DT0
1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0

DT Nastavení těchto bitů představuje hodnotu, která při shodě aktuální hodnoty čítače PWCNT1 nastaví požadovaný výstup z úrovně ‚1' do ‚0' (OPS=‚0' registru PWPR) resp. z úrovně ‚0' do ‚1'(OPS=‚1').

Pokud hodnota bitů DT je 0, výstup PWM bude trvale v úrovni ‚0' (OPS=‚0').

Pokud hodnota v registru PWCYR1 je menší než hodnota bitů DT, bude trvale nastaven výstup PWM na ‚1' (OPS=‚0')

PWBFR2A až PWBFR2H - *PWM Buffer Registers 2A až 2H*

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	DT9	DT8	DT7	DT6	DT5	DT4	DT3	DT2	DT1	DT0
1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0

Stejně uspořádaný registr jako PWDTR2. Hodnota z PWBFR2 je při každé shodě hodnoty čítače **PWCNT2** a hodnoty registru **PWCYR2** nahrána do registru **PWBFR2A.**

3.2 Časovací a pulsní jednotka - TPU (16-bit Timer Pulse Unit)

Jedna z periférií mikrokontroléru H8S/2638 je TPU jednotka, která umožňuje čítat časové úseky, nebo hrany vstupních signálů.

3.2.3 Charakteristické rysy:

- obsahuje šest 16-bitových časovacích kanálů
- každý z kanálů 0 a 3 má čtyři časovací registry TGRs (záchytný registr) a kanály 1,2,4,5 mají po dvou TGRs registrech.
- TPU umožňuje
    - zachytávat časové hrany vstupních signálů v závislosti na náběžné, spádové, nebo obou hranách
    - nulování čítače je možné při dosažení shody čítače s žádanou hodnotou, nebo vlivem zachycení hrany vstupního signálu
    - umožňuje, aby více čítačů pracovalo vzájemně synchronně
    - je možné současné nulování více čítačů vlivem shody čítače s žádanou hodnotou nebo vlivem zachycení hrany vstupního signálu
    - fázový mód může být použit nezávisle pro každý z kanálů 1,2,4,5
    - je možné inkrementovat resp. dekrementovat TPU čítač podle fázového posunu u dvoufázových signálů (vhodné pro IRC snímače)
    - je možné spojovat časovací kanály a vytvářet tak složitější struktury
- možno generovat až 26 zdrojů přerušení. Kanály 0 a 3 mají čtyři zdroje přerušení od porovnání stavu čítače a žádané hodnoty (i počtem vstupních hran pulsů) a jeden zdroj přerušení od přetečení čítače. Kanály 1,2,4,5 mají dva zdroje přerušení od porovnání stavu čítače a žádané hodnoty (i počtem vstupních hran pulsů) a po jednom zdroji přerušení od přetečení a podtečení čítače

3.2.4 Popis registrů

TCR – Řídicí registr TPU (*Timer Control Registr*)

Kanály 0, 3 :**TCR0, TCR3**

7	6	5	4	3	2	1	0
CCLR2	CCLR1	CCLR0	CKEG1	CKEG0	TPSC2	TPSC1	TPSC0
0	0	0	0	0	0	0	0

Kanály 1, 2, 4, 5 :**TCR1, TCR2, TCR4, TCR5**

7	6	5	4	3	2	1	0
-	CCLR1	CCLR0	CKEG1	CKEG0	TPSC2	TPSC1	TPSC0
0	0	0	0	0	0	0	0

TCR je 8-bitový R/W registr, který řídí kanály TCNT. Každá TPU jednotka má svůj TCP registr. Nastavení registru TCR je možné pouze tehdy je-li činnost TCNT pozastavena.

CCLR2 až CCLR0 vybírají zdroj nulování pro čítač TCNT

Kanál	**CCLR2**[1]	**CCLR1**	**CCLR0**	
0, 1, 2, 3, 4, 5	0	0	0	TCNT nulování blokováno
	0	0	1	TCNT je nulován od TGRA2
	0	1	0	TCNT je nulován od TGRB2
	0	1	1	TCNT je nulován po vynulování čítače jiného kanálu, který vykonává synchronní činnost
0, 3	1	0	0	TCNT nulování blokováno
	1	0	1	TCNT je nulován od TGRC
	1	1	0	TCNT je nulován od TGRD

[1] U kanálů 1,2,4,5 se předpokládá, že bit CCLR má hodnotu ‚0' – nemůže být změněn

[2] TGR (A,B,C,D) řídicí časový registr A,B,C,D (*Timer General registr*)

	1	1	1	TCNT je nulován po vynulování čítače jiného kanálu, který vykonává synchronní činnost

CKEG1, CKEG0 Bity nastavují vstupní hodinovou hranu

CKEG1	CKEG1	
0	0	Čítá náběžné hrany
0	1	Čítá spádové hrany
1	x	Čítá obě hrany

TPSC2 až TPSC0 nastavení rychlost čítače TCNT pro každý kanál . Možné nastavit některé z následujících možností θ, θ/2, θ/4, θ/8, θ/16, θ/64, θ/256, θ/1024, θ/4096, TCLKA, TCLKB, TCLKC, TCLKD. Jednotlivý rozpis je uveden v [4].

TMDR – Registr časovacího modů (*Timer Mode Registr*)

8-bitový R/W registr, který se používá pro nastavení časovacího módu pro každý z kanálů. Každý kanál má svůj TMDR registr. Nastavování TMDR registru by mělo být pouze v pozastavené činnosti TCNT.

Kanály 0, 3 :**TMDR0, TMDR3**

7	6	5	4	3	2	1	0
-	-	BFB	BFA	MD3	MD2	MD1	MD0
1	1	0	0	0	0	0	0

Kanály 1, 2, 4, 5 : **TMDR1, TMDR2, TMDR4, TMDR5**

7	6	5	4	3	2	1	0
-	-	-	-	MD3	MD2	MD1	MD0
1	1	0	0	0	0	0	0

BFB vyjadřuje, jestli TGRB pracuje v normální činnosti, nebo zda-li pracuje společně s TGRD k zásobníkovým (buffer) operacím.

BFB	

0	TGRB pracuje normálně
1	TGRB a TGRD jsou použity společně pro zásobníkové (buffer) operace

BFA vyjadřuje, jestli TGRA pracuje normálně, nebo zda-li pracuje společně s TGRC k zásobníkovým (buffer) operacím.

BFA	
0	TGRA pracuje normálně
1	TGRA a TGRC jsou použity společně pro zásobníkové (buffer) operace

MD nastavují mód činnosti časovacího kanálu.

MD3	MD2	MD1	MD0	
0	0	0	0	Normální činnost
0	0	0	1	Rezervováno
0	0	1	0	PWM mód 1
0	0	1	1	PWM mód 2
0	1	0	0	Fázový čítač mód 1
0	1	0	1	Fázový čítač mód 2
0	1	1	0	Fázový čítač mód 3
0	1	1	1	Fázový čítač mód 4
1	x	x	x	Nedefinováno

TIOR – Časovací I/O řídicí registr (*Timer I/O Control Registr*)

8-bitový R/W registr, který řídí TRG registry (*Timer General Registr*). TPU má osm TIOR registrů, dva pro kanály 0 a 3. Je nutno dávat pozor na nastavení registru TMDR, který má na funkci TIOR přímý vliv. Počáteční výstup nastavení podle TIOR je platný, když čítač je zastaven (bit CST v registru TSTR je vynulován).

Kanály 0, 3 : **TIOR0H, TIOR3H**

Kanály 1, 2, 4, 5 : **TIOR1, TIOR2, TIOR4, TIOR5**

7	6	5	4	3	2	1	0
IOB3	IOB2	IOB1	IOB0	IOA3	IOA2	IOA1	IOA0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Kanály 0, 3 : **TIOR0L, TIOR3L**

7	6	5	4	3	2	1	0
IOD3	IOD2	IOD1	IOD0	IOC3	IOC2	IOC1	IOC0
1	**1**	**0**	**0**	**0**	**0**	**0**	**0**

IOA3 až **IOA0**	I/O řízení A3 až A0 (I/O Control A3 to A0) **-** nastavuje funkci TGRA
IOB3 až **IOB0**	I/O řízení B3 až B0 (I/O Control B3 to B0) **-** nastavuje funkci TGRB
IOC3 až **IOC0**	I/O řízení C3 až C0 (I/O Control C3 to C0) - nastavuje funkci TGRC
IOD3 až **IOD0**	I/O řízení D3 až D0 (I/O Control D3 to D0) - nastavuje funkci TGRD

TIER – Povolení přerušení časovacího kanálu (*Timer Interrupt Enable Register*)

Kanály 0, 3 : **TIER0, TIER3**

7	6	5	4	3	2	1	0
TTGE	-	-	TCIEV	TGIEV	TGIEC	TGIEB	TGIEA
0	**1**	**0**	**0**	**0**	**0**	**0**	**0**

Kanál 1, 2, 4, 5 : **TIER1, TIER2, TIER4, TIER5**

7	6	5	4	3	2	1	0
TTGE	-	TCIEU	TCIEV	-	-	TGIEB	TGIEA
0	**1**	**0**	**0**	**-**	**-**	**0**	**0**

TTGE	**(***A/D Conversion Start Request Enable***)** – povoluje (1) nebo zakazuje (0) generování začátku dotazování od TGRA z A/D převodníku.
TCIEU	(*Underflow Interrupt Enable*) – Povoluje (1) nebo zakazuje (0)

	generování přerušení (TCIU) z příznaku TCFV, v případě že je tento příznak v registru TSR nastaven.
TCIEV	(*Overflow Interrupt Enable*) – Povoluje (1) nebo zakazuje (0) generování přerušení (TCIV) z příznaku TCFV, v případě že je tento příznak v registru TSR nastaven.
TGIED	(*TGR Interrupt Enable D*) – Povoluje (1) nebo zakazuje (0) generování přerušení (TGID) z příznaku TFGD, v případě že je tento příznak v registru TSR nastaven v kanálech 0 a 3.
TGIEC	(*TGR Interrupt Enable C*) – Povoluje (1) nebo zakazuje (0) generování přerušení (TGIC) z příznaku TFGC, v případě že je tento příznak v registru TSR nastaven v kanálech 0 a 3.
TGIEB	(*TGR Interrupt Enable B*) – Povoluje (1) nebo zakazuje (0) generování přerušení (TGIB) z příznaku TFGB, v případě že je tento příznak v registru TSR nastaven.
TGIED	(*TGR Interrupt Enable A*) – Povoluje (1) nebo zakazuje (0) generování přerušení (TGIA) z příznaku TFGA, v případě že je tento příznak v registru TSR nastaven.

TSR – Stavový registr TPU (*Timer Status Registr*)

8-bitový R/W registr, který indikuje stavy pro vyvolání přerušení. Jednotlivé bity v registru mohou být pouze nulovány, nastavování je povoleno pouze obvodově. Bit TCFD je pouze pro čtení a nelze do něj zapisovat.

Kanály 0, 3 : **TSR0, TSR3**

7	6	5	4	3	2	1	0
-	-	-	TCFV	TGFD	TGFC	TGFB	TGFA
1	1	0	0	0	0	0	0

Kanály 1, 2, 4, 5 : **TSR1, TSR2, TSR4, TSR5**

7	6	5	4	3	2	1	0
TCFD	-	TCFU	TCFV	-	-	TGFB	TGFA
1	1	0	0	0	0	0	0

TCFD	(*Count Direction Flag*) : Indikace směru čítání TCNT (pouze čtení) TCFD = 0, TCNT čítá směrem dolů, TCFD = 1, TCNT čítač ve směru nahoru
TCFU	(*Underflow Flag*) : Příznak podtečení TCNT Pokud je hodnota bitu TCFU nastavená (1) vyjadřuje to podtečení (změna z 0000h na FFFFh). Bit TCFU=1 musí být nulován.
TCFV	(*Overflow Flag*) : Příznak přetečení TCNT Pokud je hodnota bitu TCFV nastavená (1) vyjadřuje to přetečení (změna z FFFFh na 0000h). Bit TCFV=1 musí být nulován.
TGFn n = D,C,B,A	(*Input Capture/Output Compare Flag* n) : příznak zachycení vstupní hrany resp. porovnávání výstupu Nastavení příznaku když : - TCNT=TGRn, TGRn je porovnávací registr výstupu. - Vlivem zachycení hrany vstupního signálu je hodnota TCNT přemístěna do TGRn registru (pokud TGRn je nastaven jako registr pro zachytávání vstupních hran)

TCNT – Čítač času (*Timer Counter*) - 16-bitový R/W registr

Kanál 0, 3 : **TCNT0, TCNT3**

čítač lze využívat pouze jako inkrementální čítač

Kanál 1, 2, 4, 5 : **TCNT1, TCNT2, TCNT4, TCNT5**

čítač lze využívat k čítání v obou směrech. Čítání v obou směrech smí být využíváno pouze ve fázovém režimu.

TGR – Hlavní časovací registr (*Timer General Registr*) - 16-bitové R/W registry

Činnost TGR

- výstupní porovnávací registry
- registry zachytávání vstupních hran
- TGRC a TGRD u kanálů 0 a 3 můžou být použity v operacích, jako zásobníkové registry

TSTR – Spouštěcí časovací registr (*Timer Start Registr*)

8-bitové R/W registr, který určuje činnost resp. zastavení činnosti čítače pro kanály 0 až 5.

7	6	5	4	3	2	1	0
-	-	CST5	CST4	CST3	CST2	CST1	CST0
0	0	0	0	0	0	0	0

CSTn (*Counter Start 5 to 0*) : Spouštění čítače 5 až 0 – bity zastavují (0) resp. spouštějí (1) příslušný čítač TCNTn
n=0-5

TSYR – Synchronní časovací registr (*Timer Synchro Registr*)

TSYR umožňuje nastavit čítač pracující nezávisle na jiném čítači, nebo umožňuje spojení dvou čítačů (synchronní činnost). Kanál vykonává synchronní činnost tehdy má-li nastaven shodně bit (1) v TSYR

7	6	5	4	3	2	1	0
-	-	SYNC5	SYNC4	SYNC3	SYNC2	SYNC1	SYNC0
0	0	0	0	0	0	0	0

SYNCn (*Timer Synchro 5 to 0*) : SYNCn=0 vyjadřuje, že TCNT pracuje nezávisle na jiném kanálu. SYNCn=1 určuje synchronní operaci s jiným kanálem.
n=0-5

---

4 Rysy bluetooth

Cílem kapitoly není čtenáři poskytnout podrobné informace o bezdrátové síti technologie bluetooth (dále jen bluetooth), ale pouze nastínit hlavní rysy, které bylo nutné mít v podvědomí při skládání programu pro komunikaci mezi robotem a okolním světem.

Na úvod považuji za vhodné zmínit literaturu, kde je možné nabít další poznatky o popisované problematice. Jedná se především o kompletní specifikaci bluetooth [5], nebo lze doporučit i dřívější diplomové práce, které se problematiky bluetooth úzce dotýkají [6] [7].

4.1 Technické řešení

Bluetooth byla navrhována jako osobní bezdrátová síť s nízkým dosahem (ve svém nízko-výkonovém režimu má dosah pouze 10 metrů). Její hlavní využití se nabízí v domácích sítích mezi mobilními zařízeními (PDA, telefon atd.) a periferními zařízeními, zejména pak tiskárnami, nebo ke komunikaci se spotřební elektronikou.

Zařízení bluetooth poprvé představilo na trh sdružení *Bluetooth SIG (Special Industry Group)* v roce 1998. Toto sdružení vzniklo, jako neziskové spojení firem Ericsson, IBM, Intel, Nokia a Toshiba.

Komunikace mezi dvěma bluetooth zařízeními probíhá v bezlicenčním kmitočtovém pásma 2,4 GHz. Použití volně využivatelného pásma zároveň předurčuje použití bluetooth v osobní sféře. Rychlost přenosu dat na fyzické vrstvě je 1Mbit/s, ale skutečná propustnost dat se pohybuje maximálně na hranici 720 kbit/s.

Pro radiovou komunikaci, bluetooth využívá metod rozprostřeného spektra s přeskakováním kmitočtů (*Frequency Hopping Spread Spectrum, FHSS*). To si můžeme představit, jako signál, který 1600 krát/s náhodně změní frekvenční kanál. Každý kanál má frekvenční rozsah 1MHz a bluetooth zařízení jich může využívat celkem 79.

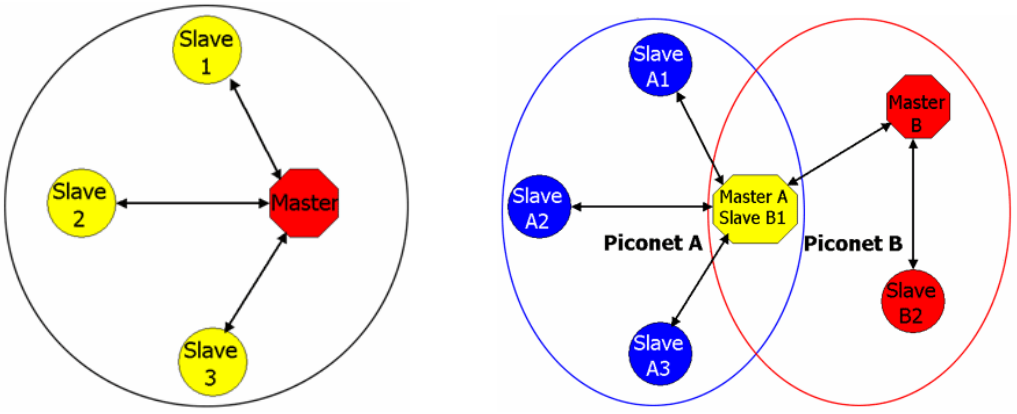
V každé síti bluetooth je vždy pouze jedna hlavní stanice tzv. *master* a několik, maximálně však sedm, podřízených stanic tzv. *slave.* Veškerou komunikace v síti bluetooth řídí hlavní stanice. Podřízená stanice může tedy komunikovat s dalšími podřízenými stanicemi pouze přes stanici hlavní. Komunikace mezi dvěma zařízeními může být buď synchronní nebo asynchronní, přičemž hlavní stanice alokuje časové

úseky prostřednictvím vícenásobných přístupů s časovým dělením (*Time Division Multiple Access, TDMA*). Bluetooth používá stejné kmitočty pro vysílání a příjem s využitím *Time Division Duplexing* (TDD) nebo multi-slot. Časování TDD odpovídá situaci v níž se vysílací a přijímací jednotka postupně střídají ve vysílání. Oproti tomu časování multi-slot odpovídá využití přeskokové sekvence pro přenos tak, že paket může obsadit více než jeden časový rámec (maximálně 5).

Jak už bylo zmíněno, komunikace může probíhat synchronně (*SCO, Synchronous Connection Oriented*) a asynchronně *(ACL, Asynchronous Connectionless).* Oba způsoby komunikace se od sebe výrazně odlišují v přenosových vlastnostech. Kanál ACL používá časování multi-slot, přičemž je možné dosáhnout přenosové rychlosti 721 kb/s v jenom směru a 57,6 kb/s v opačném směru (asymetrický kanál), popř. 433 kb/s v obou směrech (symetrický kanál). Tento způsob komunikace se nejčastěji využívá pro přenos dat mezi stanicemi. Kanál typu SCO dovoluje realizovat přenos dat rychlostí 64 kb/s v synchronním režimu a to až po třech různých kanálech najednou. Tento způsob přenosu dat je vhodnější pro přenos zvuku nebo obrazu.

4.2 Topologie sítě bluetooth

Bluetooth podporuje komunikaci dvoubodovou (*point to point*) nebo mnohabodovou komunikaci (*point-to-multipoint*). Pokud je více stanic vzájemně propojeno vytváří společně tzv. picosíť (*piconet*) viz Obr. 4.1 vlevo. Tato síť může seskupit dohromady až osm zařízení. Zařízení, které spojení vytvořilo, plní úlohu hlavní stanice (*master*) a ostatní stanice jsou hlavní stanici podřízené. Všechna zařízení v pikosíti se synchronizují s taktem hlavní stanice. Specifikace bluetooth umožňuje použití až 10 pikosíti na ploše o dosahu 10 metrů.



Obr. 4.1 Schématické vyjádření sítě piconet (vlevo) a sítě scatternet (vpravo)

V pravé části Obr. 4.1 je patrné, že sítě *piconet* je možno i sdružovat do vzájemně propojených - rozprostřených sítí (*sítí scatternet)*. Vlastností sítí scatternet je, že umožňují vzájemně libovolně sdílet zařízení slave. Zároveň zařízení, které v jedné picosíti zastupuje roli hlavní stanice (*master*) může v jiné picosíti zastupovat pozici stanice podřízené. Specifikace neumožňuje propojit picosítě, tak aby jedna stanice v obou sítích zastávala pozici stanice hlavní.

4.3 Baseband vrstva

Baseband určuje hlavní rysy fyzické vrstvy bluetooth. jeho úloha je umožnit vytvoření vzájemného fyzického spojení několika zařízení v picosíti, vzájemnou synchronizaci mezi nimi, korekci chyb, výběr skokové frekvence pro řízení komunikace a poskytuje i základní funkce kódování. Tato vrstva je shodná pro oba typy kanálů SCO, ACL.

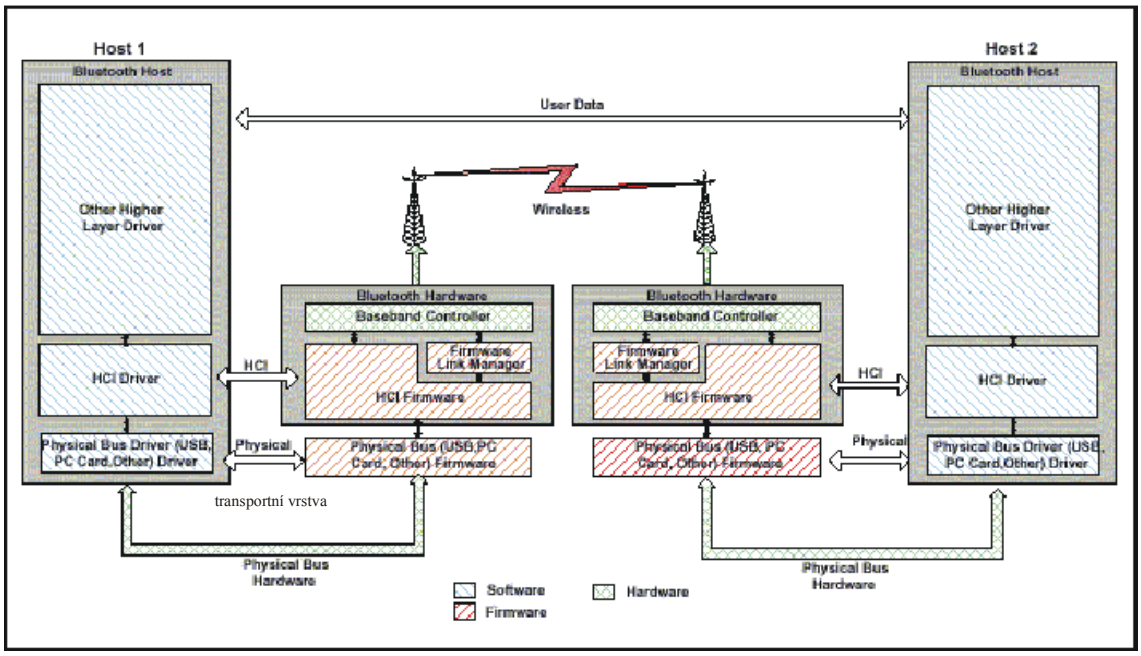
Problematiky vytváření standardních komunikačních baseband paketů se popisovaná diplomová práce nedotýká, protože bluetooth zařízení nabízí vyšší vrstvy, na kterých se komunikuje prostřednictvím zcela odlišných paketů. Tyto vyšší vrstvy již samy spolehlivě zaručují transformaci do paketu baseband. Z tohoto důvodu nebude tato kapitola dále rozvíjena, ale potřebná informace o vrstvě je možné nalézt ve specifikaci bluetooth [5] Part B, nebo v [6] [7].

4.4 rozhranní HCI

HCI (*Host Control Interface*) rozhranní poskytuje metody ke zpřístupnění hardwarových schopností bluetooth. Funkce HCI je tvořena třemi oddělenými částmi: Hostitel, transportní vrstva a Host Controler – viz Obr. 4.2.

- **HCI firmware** – je součást Host Controleru. Jeho úloha je upravovat HCI příkazy do tvaru pro hardware bluetooth (vrstvu baseband). Dále upravuje příkazy pro LM (*link manager* – správce linky), řídicí registry, stavové registry a registry událostí (*event*).
- **HCI driver -** je součást hostitele a zabezpečuje analýzu přijatých událostí, na jejichž základě poskytuje informace vyšším vrstvám.

- **Transportní vrstva** – je mezivrstva mezi HCI driverem na hostujícím systému a HCI firmwarem v bluetooth hardwaru. Je možné komunikovat prostřednictvím několika komunikačních rozhranní – USB, UART, RS232



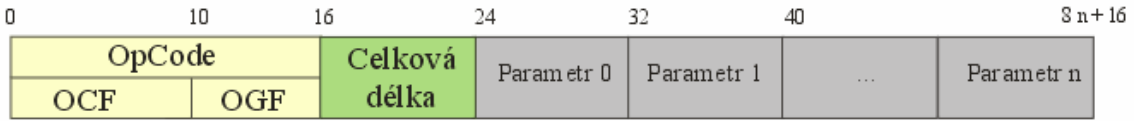
Obr. 4.2 Přehled nejnižších softwarových vrstev při přenosu dat

4.4.1 HCI příkazy

Jedná se o několik skupin příkazů, kterými lze využívat schopnosti hardware bluetooth.

HCI spojovací příkazy (*Link Commands*) nabízí hostiteli možnosti obhospodařovat spojovací vrstvu (*Link layer*) připojenou k dalšímu bluetooth zařízení. Tyto příkazy většinou vyžaduje *Link Manager* (LM) k výměně LMP (*Link Manager Protocol*) příkazů se vzdáleným zařízením.

Pakety s HCI příkazy jsou zasílány hostitelem do Host Controlleru přes transportní vrstvu. Formát paketů je na Obr. 4.3.



Obr. 4.3 Struktura HCI příkazu

Každý příkaz je jednoznačně určen pomocí tzv. *OpCode*, který se skládá ze dvou částí OGF (*OpCode Group Field*) a OCF (*OpCode Command Field*). Část OGF představuje číslo skupiny, do které příkaz patří a část OCF udává číslo příkazu v dané skupině. Za OpCodem následuje 8-bitová hodnota představující délku parametrů.
Jednotlivé skupiny jsou:

- příkazy k řízení spojení (*Link Control Commands*)
- strategické (zásadní) příkazy (*Link Policy Commands*)
- příkazy Host Controller & baseband (*Host Controller & Baseband Commands*)
- příkazy k získávání informací o parametrech (*Informational Parameters*)
- příkazy ke zjištění různých stavů (*Status Parameters*)
- příkazy k testování (*Testing Commands*)

Příkazy Link Control

Skupina je charakterizována hodnotou OGF = 0x01. Příkazy skupiny umožňují Host Contolleru řídit připojení k dalším bluetooth zařízení. Při použití těchto příkazů řídí správce linky (*Link Manager, LM*) vytvoření a udržení spojení v dané picosíti resp. ve scatternetu.

Příkazy Policy Commands

Skupina je charakterizována hodnotou OGF = 0x02. Příkazy této skupiny umožňují, aby hostitel mohl ovlivnit, jak správce linky (*Link Manager, LM*) pracuje s picosítí. Patří sem příkazy, které jsou schopny uvést zařízení bluetooth do stavu hold (*Hold Mode*) nebo do stavu sniff (*Sniff Mode*).

Příkazy Host Controller & baseband

Skupina je charakterizována hodnotou OGF = 0x03. Příkazy umožňují měnit vlastnosti hardwaru bluetooth a tím i jeho chování.

Příkazy k získávání informací o parametrech

Skupina je charakterizována hodnotou OGF = 0x04. Příkazy umožňují získávat informace o bluetooth zařízení a o vlastnostech Host Controlleru.

Příkazy k zjištění různých stavů

Skupina je charakterizována hodnotou OGF = 0x05. Příkazy umožňují získávat stavové informace o Host Controlleru, správci linky (*Link Manager, LM*) a baseband. hostitel nemůže modifikovat tyto parametry.

Příkazy k testování

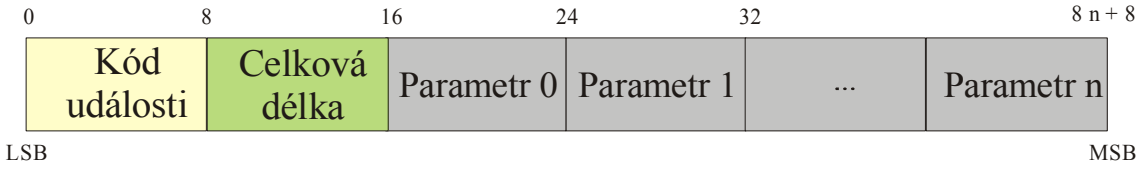
Skupina je charakterizována hodnotou OGF = 0x06. Příkazy umožňují testovat funkčnost hardware bluetooth.

4.4.2 HCI události

Jedná se o pakety (*event*), které oznamují hostiteli, že došlo k nějaké události. Jejich délka je maximálně 255 bytů.

Významná část událostí je hostiteli zasílána jako odpověď na HCI příkaz. Podle specifikace bluetooth je totiž na každý zaslaný HCI příkaz Host Controller povinen odpovědět jednou nebo více HCI událostmi.

Základní formát HCI události je na Obr. 4.4. Stejně tak jako u HCI příkazů, kde je každý příkaz jednoznačně určen OpCodem, tak v případě HCI událostí je každá událost jednoznačné určena 8-bitovým kódem události (*Event Code*). Za kódem události následuje 8-bitová hodnota představující délku parametrů v bytech.



Obr. 4.4 Struktura HCI události

4.4.3 HCI datové pakety

Datové pakety slouží k výměně dat mezi Host Controllerem a hostitelem. Pro možnost zasílání datových paketů, musí být nejdříve navázáno spojení s jiným zařízením bluetooth.

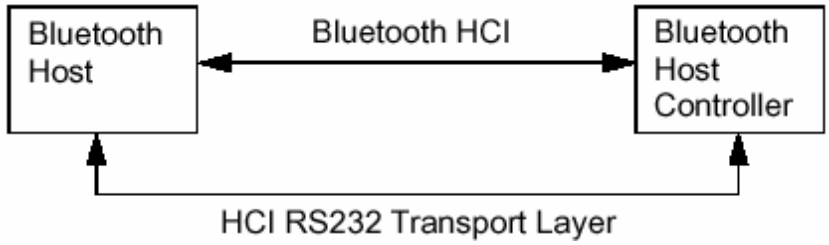
ALC datové pakety

Struktura ALC datového paketu je uvedena na Obr. 4.5. Prvních 12-bitů tvoří tzv. *Connection Handle*. Jedná se o jedinečné číslo, které je automaticky přiděleno v okamžiku, kdy se dvě zařízení domluví na vzájemném spojení. Další dva bity jsou PB Flag (*Packet Boundary Flag*), které určují, zda-li se jedná o první datový paket, nebo zda-li se jedná o paket, který je pokračováním předchozího paketu. Bity 14 a 15 určují způsob zasílání dat (point-to-point, broadcast). Rozdíl mezi ACL a SCO paketem je právě v těchto čtyřech bitech, které mají nulou hodnotu. Následujících 16-bitů určují délku dat v bytech.

Obr. 4.5 Struktura HCI ALC datového paketu

4.4.4 HCI RS232 transportní vrstva

Transportní vrstva slouží k přenosu dat mezi hostitelem a HC (*Host Controller*). Blokově ji lze vyjádřit pomocí Obr. 4.6.



Obr. 4.6 HCI RS232 transportní vrstva

Pravidlem je, že HCI příkazové pakety (*command packet*) jsou odesílány pouze z hostitele do Host Contolleru. Naproti tomu HCI pakety událostí (*event packet*) odesílá pouze Host Controller. Datové pakety lze přenášet obousměrně mezi hostitelem a Host Controllerem.

Aby bylo možné rozpoznat o jaký typ paketu se jedná, umístí se před každý HCI paket, ve tvaru uvedené v kapitole 4.4.1, byte, jehož hodnota jednoznačně specifikuje

typ HCI paketu. Tabulka 4.1 vyobrazuje přípustné hodnoty tohoto bytu a zařazení HCI paketu.

Tabulka 4.1 Typy HCI paketů

Typ HCI patetů	Hodnota
HCI příkaz	0x01
HCI ACL data	0x02
HCI SCO data	0x03
HCI událost	0x04

4.4.5 L2CAP

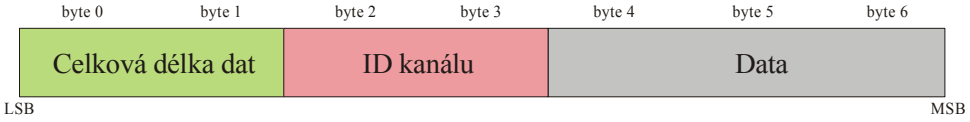
L2CAP (*Logical Link Control And Adaptation Protocol*) je protokol pro řízení a adaptaci spojení. Používá se k přenosu asynchronních dat na vyšší vrstvě. Pro přenos dat slouží pakety ACL HCI pakety.

L2CAP protokol nabízí oproti HCI protokolu řadu vylepšení. Především se jedná o možnost asynchronní (synchronní nikoliv) komunikace mezi dvěma zařízeními na více kanálech vzhledem k jednomu HCI spojení (např. dvě rozdílné aplikace komunikují přes jedno zařízení bluetooth). Dalším důvodem, proč využívat L2CAP protokol je, že některá zařízení (PC, telefon) neumožňují přístup k HCI.

Aby bylo možné komunikovat prostřednictvím L2CAP je nutné předem navázat spojení mezi dvěma účastníky na úrovni HCI. Poté, co je spojení navázáno, obě zařízení si můžou vyměňovat datové pakety ACL a tudíž i údaje potřebné pro spojení na úrovni L2CAP.

4.4.5.1 L2CAP datový paket

Struktura L2CAP datového paketu uvedená na Obr. 4.7 je součástí HCI ACL datového paketu (viz. kapitola 4.4.3) a následuje ihned za ACL hlavičkou v datové části.

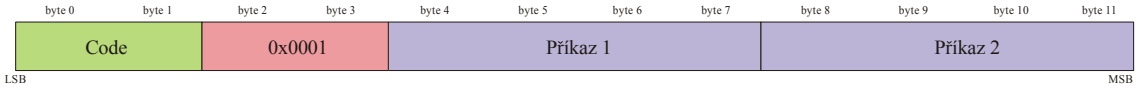


Obr. 4.7 L2CAP datový paket

Celková délka dat je 16-bitová hodnota vyjadřující délku datového pole. *ID Kanálu* je 16-bitová hodnota, vyjadřující smluvený komunikační kanál koncového zařízení. Položka *Data*, je určena pro přenášená data.

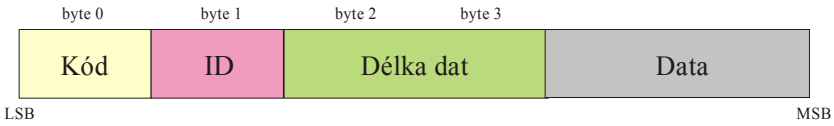
4.4.5.2 L2CAP signalling paket

Struktura L2CAP bezkanálového paketu uvedená na Obr. 4.8 je součástí ACL HCI datového paketu (viz. kapitola 4.4.3) a následuje ihned za ACL hlavičkou v datové části. Tyto pakety si vyměňují zařízení při navazování L2CAP spojení.



Obr. 4.8 L2CAP signalling paket

Celková délka je 16-bitová hodnota, která představuje délku příkazové části. Za touto položkou následuje rezervovaná část s hodnotou 0x0001. Jednotlivé příkazové části mají proměnnou délku, která je minimálně 4-byty. Struktura příkazů je naznačena na obrázku Obr. 4.9.



Obr. 4.9 Příkazový formát signalling paketu

Položka *Kód* určuje druh příkazového paketu. *ID* je identifikátor, který slouží k porovnání žádosti a odpovědi. Zařízení, které odpovídá na žádost je povinné napsat stejný identifikátor, jaký mu byl zaslán. Při dalších žádostech musí být použito rozdílného čísla než v předchozích případech. Položka *Délka dat* určuje délku datové části. Položka *Data* slouží k zasílání doplňující informací.

5 Programové vybavení pro mikrokontrolér

Aby bylo možné plně využívat fyzických možnosti robotu, je nutné ho vybavit inteligencí, v podobě programu, která jeho schopnosti využije, v co možná nejširším rozsahu.

Stručně lze úlohu rozdělit na dvě části. V první části je nutné zajistit, aby se robot mohl pohybovat požadovaným způsobem. Tuto část by měl řešit vhodně zvolený regulátor. Druhá část programového vybavení robotu, by měla robotu zajistit komunikaci s okolním světem, odkud bude přijímat povely pro svůj pohyb. Tuto část v našem případě bude řešit komunikace prostřednictvím bluetooth protokolu a komunikace po sériovém rozhranní.

5.1 Soubory projektu a knihovny

V této kapitole budou uvedeny použité knihovny, jejich stručná charakteristika, seznam zdrojových souborů, které obsahují a způsob jakým lze knihovnu začlenit do programu.

5.1.1 Knihovna pro regulátory motorů - pxmc

jedná se o knihovnu, která byla převzatá z dřívějších projektů[3]. Tato knihovna obsahuje programově řešené regulátory motorů. Součástí knihovny je konfigurace PWM jednotky a TPU jednotky mikrokontroléru.

pxmc_base.c	Jedná se o soubor, který není závislý na typu mikroprocesoru. Soubor zajišťuje řízení pro více pohybových os. Verze podporuje zpětnou vazbu pro krokové motory a pro DC a kartáčové motory.
pxmc_h2638.c	Soubor obsahuje funkce, které jsou závislé na typu mikrokontroléru (H8S/2638). Jedná se například o funkce nastavující registry mikrokontroléru H8S/2638.
pxmc_deb.c	Soubor obsahuje podpory pro ladění programu.
pxmc_ptable.c	Soubor obsahuje funkce pro generování fázových tabulek

[3] Vytvořil: 2001 Pavel Píša pisa@cmp.felk.cvut.cz
2002 PiKRON Ltd. http://www.pikron.com

používaných pouze u krokových motorů.

`pxmc_hh.c`	Soubor tvoří nástroj pro nalezení domácí polohy.
`pxmc.h`	Hlavičkový soubor, který obsahuje struktury a makra nezbytné pro činnost pxmc. Dále soubor obsahuje seznam funkcí, které mají souvislost se zpětnovazebním řízení DC motorů a zpětnovazebním řízením krokových motorů.
`pxmc_h2638.h`	Hlavičkový *s*oubor obsahuje seznam funkcí z pxmc, které jsou přístupné dalším aplikacím. Soubor je závislý na použitém procesoru (H8S/2638).
`pxmc_internal.h`	Hlavičkový soubor obsahuje seznam vnitřních funkcí, které jsou určeny pro činnost celé knihovny pro řízení více os motorů. Funkce obsažené v tomto souboru by měly být používány pouze touto knihovnou.

Začlenění pxmc do projektu

Do hlavního souboru (soubor s funkcí *main*), v našem případě *mirosot_main.c* je nutné vložit hlavičkové soubory, následujícím způsobem.

```
#include <pxmc.h>
#include <pxmc_h2638.h>
```

V dalším je nutné nadefinovat proměnné typu *pxmc_state_t*. Jedná se o strukturu, která nese potřebné informace o vlastnostech motoru. V našem případě takto vznikly globální proměnné *mcsX0* a *mcsX1*, definující vlastnosti jednoho a druhého motoru. Počet motorů je určen konstantou *SUMMOTORS*. Takto nadefinovaná dvojice motorů je posléze sjednocena do pole **pxmc_main_arr*, jehož položky jsou ukazatele na struktury motorů.

Informace o počtu motorů a jejich vlastnostech jsou uloženy ve struktuře *pxmc_main_list* typu *pxmc_state_list_t*.

Nastavení registrů jednotlivých motorů se provede funkcí

```
void pxmc_set_pwm_tpu(int NumMotors),
```

jejíž zdrojový kód je v souboru `pxmc_h2638.c`. Funkce nastavuje registry TPU a PWM jednotky. Prvnímu motoru jsou automaticky přiřazeny výstupní piny *PWM1A*, *PWM1B* a časový kanál TPU1. Druhému motoru v pořadí jsou přiřazeny piny *PWM1C*, *PWM1D* a časový kanál TPU2. Oba motory mají shodný zdroj vzorkovacího kmitočtu v podobě TPU kanálu 4.

Pomocí funkce

void pxmc_add_pservice_and_mode(short mod)

se struktura informací o vlastnostech motoru, doplní o pointery na funkce `pxmc_nofb_inp, pxmc_nofb_noc, pxmc_nofb_out`, které provádí úpravu vstupního signálu, regulaci a generování výstupního signálu.

Druhá úloha funkce *pxmc_add_pservice_and_mode* je nastavit mód motorů. V naší aplikaci je parametrem vstupujícím do funkce číslo „4“. Tato hodnota vyjadřuje, že použité motory jsou stejnosměrné. Od tohoto okamžiku je pxmc plně nastavena.

5.1.2 Knihovna pro obsluhu sériového rozhranní

Jedná se o knihovnu, která byla převzatá z dřívějších projektů[4]. Úloha knihovny je nastavit a zajišťovat obsluhu sériových linek.

sci_rs232.c	Hlavní soubor obsahuje funkce pro čtení resp. zápis dat do vstupních resp. výstupních front. Dále je zde funkce, pro inicializaci sériového rozhranní, funkce vykonávané v přerušeních a doprovodné funkce.
sci_rs232.h	Hlavičkový soubor obsahuje struktury nezbytné pro činnost celé knihovny a seznam funkcí a maker.
sci0.c,sci1.c, sci2.c,	Soubor obsahuje funkce, které se vyvolávají při přerušení od daného sériového kanálu. Tyto funkce slouží pouze k určení čísla kanálu, který vyvolal přerušení. Dále jsou zde nadefinovány struktury charakteristické pro daný kanál.
sci_channels.h	Implicitní nastavení. Nastaví struktury pro sériové kanály 0, 1, 2.
sci_regs.h	Hlavičkový soubor zavádí strukturu, která má položky s přesným uspořádán, jak má H8S/2638 uspořádané registry.

[4] Vytvořil: 2001 Pavel Píša pisa@cmp.felk.cvut.cz
2002 PiKRON Ltd. http://www.pikron.com
2005 Michal Sojka <wentasah@centrum.cz>

Začlenění knihovny sériového rozhranní do projektu

Do hlavního souboru (soubor s funkcí *main*), v našem případě *mirosot_main.c* je nutné vložit následující hlavičkový soubor.

```
#include <periph/sci_rs232.h>
```

s*ci_rs232_setmode(19200, 0, 0, sci_rs232_chan_default)*;	nastavení pro PC
sci_rs232_setmode(115200, 0, 0, 2);	nastavení pro bluetooth

5.1.3 Knihovna textových příkazů pro řízení pxmc

jedná se o knihovnu, která byla převzatá z dřívějších projektů[5]. Prvotní využití knihovny umožňovalo řídit pxmc jednotku pomocí příkazů cmd, zasílaných po sériové lince. V popisované diplomové práci se původní knihovna cmd rozšířila i pro možnost zasílání cmd příkazů přes rozhranní bluetooth.

`cmd_rs232.c`	Zdrojový soubor, který obsahuje funkce na vyčítání resp. zápis znaků z resp. do front sériového rozhranní RS232. Přijaté znaky jsou umisťovány do front knihovny cmd.
`cmd_proc.c`	Zdrojový soubor k porovnávání tvaru příkazů command procesoru a přijatého textového řetězce.
`cmd_proc.h`	Hlavičkový soubor, který obsahuje definice struktur používaných k činnosti *cmd*. Dále obsahuje nadefinovaná makra a seznam funkcí.
`cmd_bth.c`	Obdoba souboru `cmd_rs232.c`, ale komunikace je zajišťována zařízením bluetooth.
`cmd_bth.h`	Hlavičkový soubor obsahující seznam funkcí z `cmd_bth.c.`
`cmd_pxmc.c`	Zdrojový soubor, který obsahuje zdrojové kódy funkcí k řízení nastavování a čtení informací od *pxmc*.

Začlenění cmd do projektu

Do hlavního souboru (soubor s funkcí *main*), v našem případě soubor *mirosot_main.c* je nutné vložit hlavičkové soubory, následujícím způsobem.

---

[5] Vytvořil: 2001 Pavel Píša pisa@cmp.felk.cvut.cz
2002 PiKRON Ltd. http://www.pikron.com

```
#include <cmd_proc.h>
#include <cmd_pxmc.h>
#include <cmd_bth.h>
```

Dále je nutné vytvořit tabulku command příkazů. Tato tabulka je reprezentována strukturou

*cmd_des_t const *cmd_rs232_default[]*.

Jednotlivé položky pole jsou ukazatele na struktury typu *cmd_des_t*, které tvoří seznamy příkazů, které se mohou vykonávat.

Řádky

*cmd_des_t const **cmd_rs232=cmd_rs232_default*
*cmd_des_t const **cmd_bth=cmd_rs232_default;*

nastavují ukazatel na první položku pole seskládaného ze seznamů cmd příkazů. Zápis na prvním řádku je pro obsluhu rozhranní RS232 a druhý řádek pro obsluhu bluetooth rozhranní.

V hlavním souboru (*mirosot_main.c*), kde je využívána *cmd* knihovna je nutné vložit celé zdrojové kódy funkcí

int cmd_rs232_processor_run(void)
int cmd_bth_processor_run(void)

Tyto funkce zajišťují výběr příchozích znaků z front sériového rozhranní resp. bluetooth rozhranní, přesun znaků do front sériového rozhranní resp. bluetooth rozhranní a funkce pro vyhodnocení příchozích textových cmd příkazů.

Tyto funkce je nutné pro správnou činnost cyklicky volat z nekonečné smyčku ve funkci *main*.

5.1.4 Knihovna bluetooth

Bluetooth je knihovna, která slouží ke konfigurování hardwaru bluetooth, k nastavení spojení mezi zařízeními, k balení dat do datových paketů a jejich odvysílání, resp. rozbalování přijatých datových paketů a poskytování dat dalším aplikacím (tj. HCI driver a L2CAP stack).

Knihovna se skládá z následujících souborů.

`bth_main.c` Jedná se o hlavní zdrojový soubor knihovny bluetooth. Soubor nastavuje proměnné nutné pro činnost celé knihovny. Obsahuje funkce pro příjem paketů, odvysílání paketů a

funkce, které slouží k testování.

`bth_command.c`	Soubor obsahuje funkce, které sestavují HCI příkazový (*command*) paket, potřebné ke konfigurování HW bluetooth nebo k upravování spojení mezi zařízeními.
`bth_event_acc.c`	Soubor obsahuje funkce, které dekódují HCI paket událostí (*event packet*) a zpracovávají touto cestou přijatá data, nebo jsou nezbytné k činnosti dekódovacích funkcí. Jednotlivé event funkce jsou volány pointery na funkce ze souboru `bth_command.c.`
`bth_cmd _complete_ev.c`	Soubor obsahuje funkce k dekódování paketů „*command complete*“. Funkce obsažené v souboru jsou volány funkcí *bth_evt_cmd_complete*, která má zdrojový kód v souboru `bth_event_acc.`
`bth_error.c`	Soubor obsahuje funkce na určení chybového stavu z přijatého paketu, nebo k určení chybového stavu vráceného některou z *event* funkcí.
`l2cap.c`	Soubor obsahuje funkce, které slouží k navázání a konfiguraci spojení protokolem L2CAP. Dále obsahuje funkce, které umožňují balit data do L2CAP datových paketů, nebo naopak získávat data z L2CAP datových paketů a poskytovat dalším aplikacím.
`bth_inface.c`	Hlavičkový soubor, který obsahuje funkce, pro práci s datovými frontami.
`hci.h`	Hlavičkový soubor, který definuje proměnné a struktury nezbytné k činnosti celé knihovny bluetooth. Dále jsou zde uvedeny struktury hlaviček HCI paketů a některé inline funkce potřebné i činnosti knihovny bluetooth.
`hci_command.h`	Hlavičkový soubor, který definuje struktury odpovídající command příkazům, velikost struktur v bytech, OGF a OCF kódy charakteristické pro jednotlivé funkce.
`hci_event.h`	Hlavičkový soubor, který definuje struktury odpovídající tělům event paketů, velikost struktur v bytech a Event kód charakteristický pro jednotlivé funkce.

hci_error.h	Obsahuje seznam funkcí v souboru *bth_error.c*
l2cap.h	Obsahuje struktury korespondující s uspořádáním L2CAP paketů, délky paketů v bytech a seznam funkcí obsažených v souboru *l2cap.c.*
inline_fce.h	Hlavičkový soubor, který obsahuje především inline funkce pro převod 16-bitových hodnot do little resp. big endianu.
inface_bth.h	Hlavičkový soubor, který obsahuje inline funkce, struktury, seznam funkcí, které jsou potřebné pro ukládání resp. vyzvedávání z datových front. Knihovna náleží k souboru *bth_inface.c.*
bth_receive.h	Hlavičkový soubor, který definuje strukturu pro orientaci ve vstupní frontě.
bth_fce_out.h	Knihovna obsahuje seznam funkcí knihovny bluetooth které jsou potřebné a postačující pro činnost aplikace využívající knihovnu bluetooth.
h2638_pkt_ controll.c	Soubor obsahuje funkce, pro nastavení a obsluhu třetího kanálu TPU jednotky. TPU jednotka je využívána k měření časové prodlevy mezi příjmem dvou znaků od zařízení bluetooth. Takto lze částečně potlačit chyby vstupní fronty, způsobené neobsloužením některého z přijímaných znaků.
bth_h8s2638.h	Hlavičkový soubor obsahující seznam funkcí závislých na použitém mikrokontroléru (h8s/2638).

Začlenění knihovny bluetooth do projektu

Do hlavního souboru (soubor s funkcí *main*), v našem případě soubor *mirosot_main.c* je nutné vložit hlavičkové soubory, následujícím způsobem.

```
#include <bth_inface.h>
#include <bth_fce_out.h>
#include "bth_h8s2638.h" [6]
```

[6] Není nutné vkládat pokud nechceme využívat TPU jednotku v knihovně bluetooth

Inicializace celého bluetooth zařízení provádí funkce, *void bth_init(void)*. Tato funkce zaznamená do struktury *bth_rs232_que_in* začátek a konec vstupní fronty, který slouží pro příjem bytů z bluetooth zařízení.

Aby měla aplikace od zařízení bluetooth přístupná data, resp. aby mohla data vysílat, je nutné ještě inicializovat datové fronty. To provádí funkce
int bth_inface_setup(intchan), jejíž parametr udává číslo kanálu. Číslo kanálu odpovídá indexu připojeného zařízení. Prvnímu zařízení je přiřazeno číslo nula.

Posléze je vhodné zavolat funkci void *bth_init_pkt_controll(void)*, která nastavuje třetí časovací kanál TPU jednotky. TPU jednotka je využívána k měření časové prodlevy mezi příjmem dvou znaků (ve funkci *bth_recieve_packet* v souboru *bth_main.c*) ze zařízení bluetooth. Takto lze částečně eliminovat chyby vstupní fronty, způsobené neobsloužením některého z přijímaných znaků. Pokud nechceme využívat funkce kontroly správnosti vstupních paketů, stačí nevolat funkci *bth_init_pkt_controll*, zároveň není nutné vkládat hlavičkový soubor *bth_h8s2638.h*.

Dále je nutné nastavit pro zařízení bluetooth jeho úlohu v síti. V současné době lze zařízení nastavit pouze do režimu podřízeného zařízení (*slave*). Sestavení potřebných paketů, které nastaví zařízení do režimu podřízeného zařízení zajistí funkce *void bth_parametr_slave(void).* Možnost odesílání paketů do zařízení bluetooth se spustí funkcí *void bth_start(void).*

Před zavoláním funkce *bth_start* je nutné vyprázdnit frontu sériového rozhranní, která je přidělena pro zařízení bluetooth. K tomu je dobré použít krátkou časovou smyčku. V našem případě je časová smyčka tvořena třetím TPU kanálem již dříve inicializovaného funkcí *bth_init_pkt_controll*. Funkce *bth_parametr_slave* a *bth_start* je také nutné zavolat z hlavního programu.

Po odeslání vytvořených paketů (zajistila funce *bth_parametr_slave*) může bluetooth zařízení, které plní roli hlavní stanice (*master*) lokální zařízení vyhledat, připojit a komunikovat s ním.

Funkce, které jsou již cyklicky volány z nekonečné smyčky programu jsou *int sci_rs232_recch (int chan), int bth_inface_sendch(int val, int chan).* První funkce (součást knihovny obsluhující sériové rozhranní) čte znak z datové fronty určeného parametrem *chan* (přijatá data). Druhá funkce (součást souboru `bth_inface.c`) zapisuje přečtený znak (parametr *val*) do datové fronty (určeného parametrem *chan*) k odeslání. Pokud je návratová hodnota –1, tak v případě první zmíněné funkce to

signalizuje, že nová data nebyla přijata a v případě druhé funkce to signalizuje, že se nepovedlo uložit znak do datovéhé fronty.

Další funkce, kterou je nutné volat cyklicky z nekonečné smyčky hlavního programu je *int l2cap_send_data(int inx_handle, int inx_chanal),* která v případě, že datová fronta, jednoznačně určená parametry *inx_handle* (index zařízení připojeného k bluetooth, počítáno od nuly) a *inx_chanal* (index připadající k navázanému komunikačnímu kanálu, počítáno od nuly) obsahuje data k odeslání zajistí vytvoření datového paketu a ten zařadí do fronty k odeslání.

Poslední funkcí, kterou je nutno volat také cyklicky z nekonečné smyčky hlavního programu je *int bth_send_queue(void)* (součást souboru `bth_inface.c`). Funkce zjistí zda-li ve frontě paketů k odeslání jsou nové pakety, pokud ano, a zařízení bluetooth je volné, pakety odešle.

Podrobněji bude činnost knihovny bluetooth popsána v kapitole 5.

6 Funkce knihovny bluetooth

Kapitola má za cíl popsat činnost knihovny bluetooth, způsob zpracovávání příchozích znaků tvořících pakety, ukládání znaků a vybírání událostí, které odpovídají příchozím paketům. Dále zde bude naznačen způsob skládání HCI paketů a metoda, která zajišťuje jejich odvysílání. Budou zde uvedeny i důležité proměnné k činnosti celé knihovny a struktura front k uchování dat.

Kapitola si neklade za cíl popisovat jednotlivé příkazové (*Command*), událostní (*Event*) či L2CAP funkce, které jsou pouze přepis specifikace bluetooth, stejně tak zde nebude uvedena posloupnost, kterou je nutno vykonat pro navázání spojení (uvedeno v příloze C).

6.1 Proměnné a struktury pro uchování informací o zařízeních

Bluetooth sdružuje několik síťových protokolů pro komunikaci mezi dvěmi či více zařízeními. Pro navázání spojení nebo skládání či dekódování paketů je nutné mít uchovány v paměti údaje, které si zařízení vzájemně vyměnily v průběhu připojování. Všechny struktury, které tyto údaje umožňují uchovávat jsou nadefinovány v souboru *hci.h*.

6.1.1 Informace o lokálním zařízení

```
typedef struct {
  char       name[8];
  bdaddr_t   bdaddr;
  uint32_t   flags;
  uint8_t    type;
  uint8_t    features[8];
  uint32_t   pkt_type;
  uint32_t    link_policy;
  uint32_t   link_mode;
  uint16_t   acl_mtu;
  uint16_t   acl_pkts;
  uint8_t    sco_mtu;
  uint16_t   sco_pkts;
  uint8_t    busy;
}bths_dev_info;
```

Položky použité v programu :

bdaddr – BD adresa lokálního BT zařízení

busy – vyjadřuje stav BT zařízení. Proměnná určuje, zda-li je zařízení schopné odvysílat HCI paket, nebo jestli právě paket zpracovává

pkt_type – typy paketů, které lze odesílat

Obr. 6.1 struktura pro uchování informací o lokálním zařízení

V programu je struktura Obr. 6.1 použita pro globální proměnnou *bth_local_info* nadeklarovanou v souboru `bth_main.c`.

Hlavním cílem proměnné je uchovávat údaje o lokálním zařízení nebo o jeho stavu v čase. Většinu z těchto údajů je možné získat po zaslání vhodného příkazového HCI paketu (*command packet*).

6.1.2 Informace o vzdáleném zařízení

```
typedef struct con_bluet {
  bdaddr_t  bdaddr;
  uint16_t   handle;
  uint8_t     link_type;
  uint8_t     encr_mode;
  uint16_t   max_slots;
  uint16_t   ptype;
  uint16_t   ident;
  uint16_t   scid[L2CAP_NUM_OF_CANAL];
  uint16_t   dcid[L2CAP_NUM_OF_CANAL];
}bths_connect_bluet;
```

Položky použité v programu :

- **bdaddr** – BD adresa vzdáleného BT zařízení
- **handle** – číslo, používané k přenosu dat s tímto zařízením
- **scid** – seznam zdrojových kanálů (maximálně L2CAP_NUM_OF_CANAL) které jsou použity ke komunikaci
- **dcid -** seznam vzdálených kanálů (maximálně L2CAP_NUM_OF_CANAL) které jsou použity ke komunikaci
- **max_slots –** maximální počet časových slotů vzdáleného zařízení
- **p_type –** druhy typů paketů, které umožňuje vzdálené zařízení přijmout

Obr. 6.2 struktura pro uchování informací o vzdáleném zařízení

V programu je tato struktura Obr. 6.2 použita pro globální proměnnou *bth_connected* nadeklarovanou v souboru `bth_main.c`. Jelikož zařízení bluetooth může udržovat spojení s více zařízeními souběžně, je proměnná volena jako pole o maximálním indexu 7 (odpovídá osmi zařízení včetně lokálního).

Hlavním cílem proměnné je uchovávat údaje o vzdáleném zařízení. Tyto informace lokální zařízení získává při navazování spojení.

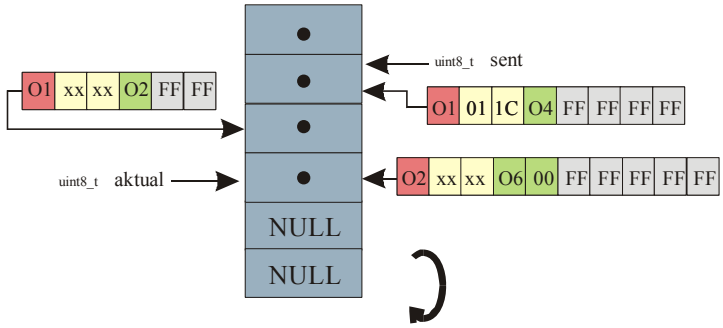
6.2 Fronty a jejich organizace

Knihovna pro bluetooth obsahuje několik různých front paketů, které plní úlohy pro řazení paketů pro vysílání, řazení paketů pro potvrzení, jako vstupní datová fronta nebo jako výstupní datová fronta. Další dvě fronty obsahuje knihovna obsluhy sériového rozhranní pro uchovávání přijatých znaků resp. pro uchovávání znaků pro vysílání po UART. Knihovna sériového rozhranní je využívaná pouze k přesunu znaků, tvořících

paket, pro odvysílání do bluetooth a ke kontrolním výpisům paketů na terminál PC. Při použití sériového rozhranní k bluetooth se čtení znaků ze vstupní fronty UART děje v hlavní nekonečné smyčce programu (funkce *Main*) a přijaté znaky jsou předávány (jako parametr) do funkce *bth_recieve_packet.*

6.2.3 Fronta paketů k odeslání a k potvrzení

Na Obr. 6.3 je znázorněn fronta paketů k odesílání (modrá část) a k němu ukázkově přiřazeny již některé pakety.




```
typedef struct {
    int          len;
    uint8_t  aktual;
    uint8_t  sent;
} bths_command_buf_info;
```

Obr. 6.3 Fronta paketů k odesílání

Deklarace proměnné, tvořící frontu (v souboru **bth_main.c**)

```
uint8_t *bth_pole_adrr_comm_packet[LENCOMMAND]
```

Deklarace proměnné informující o pozicích ve frontě (v souboru **bth_main.c**)

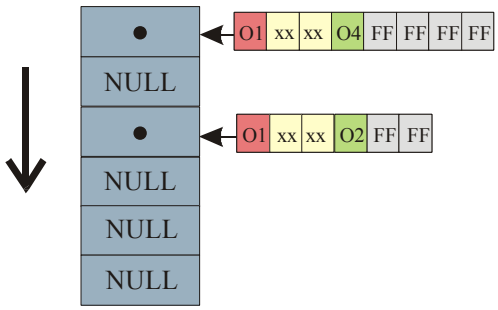
```
bths_command_buf_info bth_com_buf_info=
    {LENCOMMAND,0,0};
```

Fronta paketů je tvořena jako pole ukazatelů. Proměnná zastupující tuto frontu je *bth_pole_adrr_comm_packet*. Velikost tohoto pole je dána konstantou LENCOMMAND. Aby bylo možno se orientovat ve frontě, je deklarovaná proměnná *bth_com_buf_info*. Jedná se o proměnnou typu *bths_command_buf_info*, jejíž struktura je naznačena na Obr. 6.3 vpravo. Položka *len* koresponduje s délkou pole, která tvoří frontu, položka *aktual* uchovává index v poli, kde je uložen poslední paket k odeslání a položka *send* uchovává index v poli, kde je uložen poslední odeslaný paket. Pokud se hodnoty v položkách *send* a *aktual* rovnají, signalizuje to, že všechny pakety byly již odeslány.

Jednotlivé položky jsou do fronty ukládány v kruhu – tj. po položce s indexem LENCOMMAND-1 následuje položka s indexem 0.

Pokud je paket odeslán do zařízení bluetooth, ukazatel na pozici *send* se přepíše hodnotou NULL.

Na Obr. 6.4 je naznačena fronta paketů k potvrzení, která je tvořena jako pole ukazatelů. Proměnná, která frontu paketů k potvrzení reprezentuje je *bth_pole_adrr_check_packet.* Její struktura je stejná, jako v případě fronty k odesílání paketů.



Obr. 6.4 fronta paketů k potvrzení

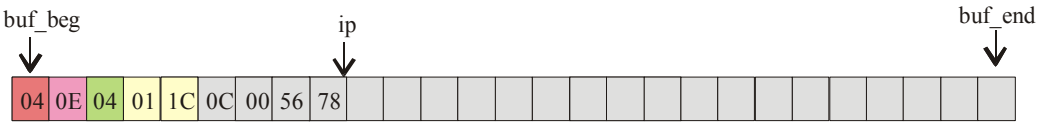
Deklarace proměnné, tvořící frontu (v souboru **bth_main.c**)

```
uint8_t *bth_pole_adrr_check_packet[LENCOMMAND]
```

Rozdíl od fronty paketů k odesílání je, že pro potvrzovací frontu není potřeba struktura, která v odesílací frontě zajišťovala orientaci. Například, pokud by se do zařízení bluetooth zaslalo více paketů najednou, tak není pravidlem, že odpovědi na jednotlivé pakety budou chodit ve stejném pořadí v jakém byly odeslány. Proto v případě potvrzování se vždy pakety kontrolují od indexu 0 po index daný opět konstantou LENCOMMAND-1. V případě ukládání adresy na začátek paketu, do potvrzovací fronty, se prohledávají položky pole od indexu 0 po index LENCOMMAND-1. Pokud je hodnota ukazatele rovna NULL, je ukazatel nastaven na adresu začátku paketu.

6.2.4 Vstupní fronta

Vstupní fronta má za úkol ukládat příchozí znaky tvořící HCI pakety. Proměnná *bth_pole_char_in* tvořící frontu je deklarována jako pole o BTH_BUF_LEN prvcích, kde každý z nich má velikost 1 byte. Aby bylo možno se orientovat ve frontě, je deklarovaná proměnná *bth_rs232_que_in*. Jedná se o proměnnou typu *bth_que_t*. Tato struktura uchovává informaci o velikosti pole a o pozici naposledy uloženého znaku. Položka *buf_beg* resp. *buf_end* ukazuje na začátek resp. na konec fronty. Položka *ip* ukazuje na pozici posledního uloženého znaku. Položka *op* není využívaná a slouží k budoucímu účelům.



Obr. 6.5 Vstupní fronta

Definice struktury pro orientaci ve frontě (v souboru **bth_receive.h**)

```
typedef struct{
    uint8_t *buf_beg;
    uint8_t *buf_end;
    uint8_t *ip;
    uint8_t *op
}bth_que_t;
```

Deklarace proměnné, tvořící frontu (v souboru **bth_main.c**)

```
uint8_t  bth_pole_char_in[BTH_BUF_LEN];
```

Deklarace proměnné informující o pozicích ve frontě (v souboru **bth_main.c**)

```
bth_que_t bth_rs232_que_in;
```

Prvotní nastavení položek proměnné *bth_rs232_que_in* je v souboru **bth_main.c**, funkce **bth_init**..

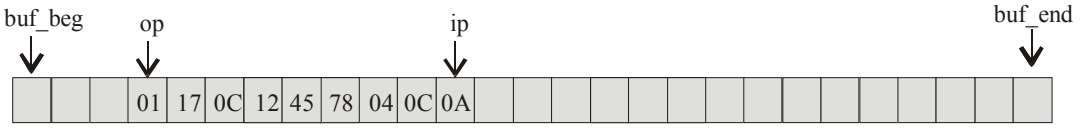
Ukládání znaků do fronty je organizováno tak, že s každý příchozí paket se začíná ukládat do fronty od jejího začátku tj. od pozice *buf_beg*. Pokud se z hlavičky paketu zjistí, že příchozí paket je celý, přiřadí se položce *ip* hodnota *buf_beg* v opačném případě je pointer *ip* inkrementován.

6.2.5 Vstupní a výstupní datová fronta

Organizace vstupní a výstupní datové fronty je znázorněná na Obr. 6.6. Vstupní fronta slouží k ukládání přijatých dat z datových L2CAP paketů, pro aplikaci využívající knihovnu bluetooth. Ukládaná data již neobsahují hlavičky přijatých paketů.

Výstupní fronta je použita k ukládání dat, které aplikace chce odvysílat. V knihovně bluetooth se data opatří hlavičkou paketu a odvysílají.

Pro orientaci ve frontě je nadefinovaná pomocná struktura *bth_inface_que_t*. Organizace fronty je volena jako kruh. Po dosažení konce fronty *buf_end* se začíná zapisovat resp. číst z pozice *buf_beg*.



Obr. 6.6 Organizace vstupní a výstupní datové fronty

Definice struktury pro orientaci ve frontě(v souboru **bth_inface.h**). Proměnné pro orientaci jsou vytvořeny spolu s frontami ve struktuře *bth_inface_info_t* (viz níže).

```
typedef struct{
    uint8_t *buf_beg;
    uint8_t *buf_end;
    uint8_t *ip;
    uint8_t *op;
}bth_inface_que_t;
```

vstupně/výstupní fronty mají přidělenou pamět ve struktuře *bth_inface_info_t*.

```
typedef struct bth_inface_info{
    bth_inface_que_t bth_inface_que_in;
    bth_inface_que_t bth_inface_que_out;
    uint8_t  bth_inface_buff_in[BTH_INFACE_BUF_LEN];
    uint8_t bth_inface_buff_out[BTH_INFACE_BUF_LEN];
}bth_inface_info_t;
```

Deklarace proměnné, tvořící fronty a orientační ukazatele je v souboru **bth_inface.c**. Konstanta BTH_INFACE_CHANAL udává počet komunikačních kanálů a zároveň počet datových front.

```
bth_inface_info_t bth_inface_chan_array[BTH_INFACE_CHANAL];
```

Inicializace fronty s číslem *chan* se provede po zavolání funkce *int bth_inface_setup(int chan),* funkce je součástí **bth_inface.c**.

S datovými frontami úzce spolupracují následující funkce :

1. `int bth_inface_recch(int chan);`
   funkce vrací znak ze vstupní datové fronty komunikačního kanálu *chan* (odpovídá pořadovému číslu připojeného zařízení). Pokud ve frontě není nový znak funkce vrací hodnotu –1.
2. `int bth_inface_sendstr(const char *s, int chan);`
   funkce ukládá do výstupní fronty kanálu *chan* (odpovídá pořadovému číslu

připojeného zařízení). znak *s. Pokud je fronta zaplněná, funkce vrací hodnotu –1.

3. `int bth_inface_r_isr(int chan, int val);`
   interní funkce knihovny bluetooth. Funkce ukládá do vstupní fronty kanálu *chan* (odpovídá pořadovému číslu připojeného zařízení) přijatý znak *val*. Pokud je fronta zaplněná, funkce vrací hodnotu –1.
4. `int bth_inface_t_isr(int chan);`
   interní funkce knihovny bluetooth. funkce vrací znak z výstupní datové fronty kanálu *chan* (odpovídá pořadovému číslu připojeného zařízení). Pokud ve frontě není nový znak funkce vrací hodnotu –1.

6.3 Málo přehledné zápisy v programu

V této kapitole budou uvedeny některé zápisy, které se v programu hojně využívají, a zároveň se můžou zdát na první pohled nepřehledné.

1. zápis pomocí inline funkce *__bthtomc16.*
   Tato inline funkce má zdrojový kód uvedený v souboru *inline_fce.h*. Činnost funkce je, že kopíruje 16-bitová data z adresy *adr1* na *adr2*. Funkce je využívána pro možnost ukládání 16-bitových dat i na liché adresy. Funkce se používá ve tvaru
   ```
   __bthtomc16(uint16_t *adr1, uint16_t *adr2);
   ```
2. zápis pomocí inline funkce *__ store_le16.*
   Tato inline funkce má zdrojový kód uvedený v souboru *inline_fce.h*. Činnost funkce je, že na adresu *adr* uloží 16-bitovou hodnotu `value` ve tvaru little endian. Funkce se používá ve tvaru
   ```
   __ store_le16(uint16_t *adr, uint16_t value);
   ```
3. zápis pomocí inline funkce *memcpy.*
   Pro využívání této funkce je nutné nejdříve do zdrojového kódu vložit hlavičkový soubor *string.h*. Činnost funkce je, že z adresy *adr1* na adresu *adr2* kopíruje blok paměti o velikosti *n*. Funkce se používá ve tvaru
   ```
   void *memcpy(void *adr1, const void adr2, size_t n);
   ```

4. zápis pomocí makra *htobs* (*host to bluetooth*) resp. btohs (*bluetooth to host*). Makro se používá pro 16-bitový vyjádření konstant v *little endianu* resp. *big endianu*. Makro se používá ve tvaru

```
htobs(A);
```

A je 16-bitová konstanta

6.4 Popis organizace knihovny bluetooth

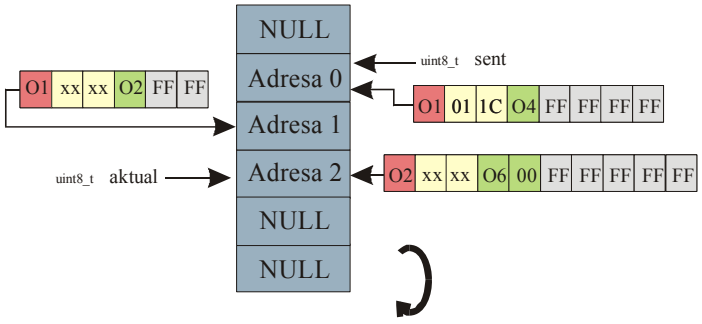
Jak bylo naznačeno v kapitole 5.1.4 knihovna bluetooth je rozčleněna do několika souborů. Teprve jako celek knihovna umožňuje obsluhovat protokol technologie bluetooth.

Cílem této kapitoly je naznačit vazby jednotlivých částí programu, jejichž pochopení povede i k pochopení celé konstrukce programu. Nebudou zde popisovány jednotlivé *Command* funkce resp. *Event* funkce, které neplní jinou úlohu než-li seskládat paket z dostupných dat resp. rozložit paket a data uložit nebo poskytnout dalším funkcím.

Soubor *bth_main.c* tvoří API (*application program interface*) knihovny bluetooth a zajišťuje využívání funkcí z dalších souborů knihovny.

6.4.6 Sestavení paketu, zařazení do fronty paketů a odvysílání do bluetooth zařízení

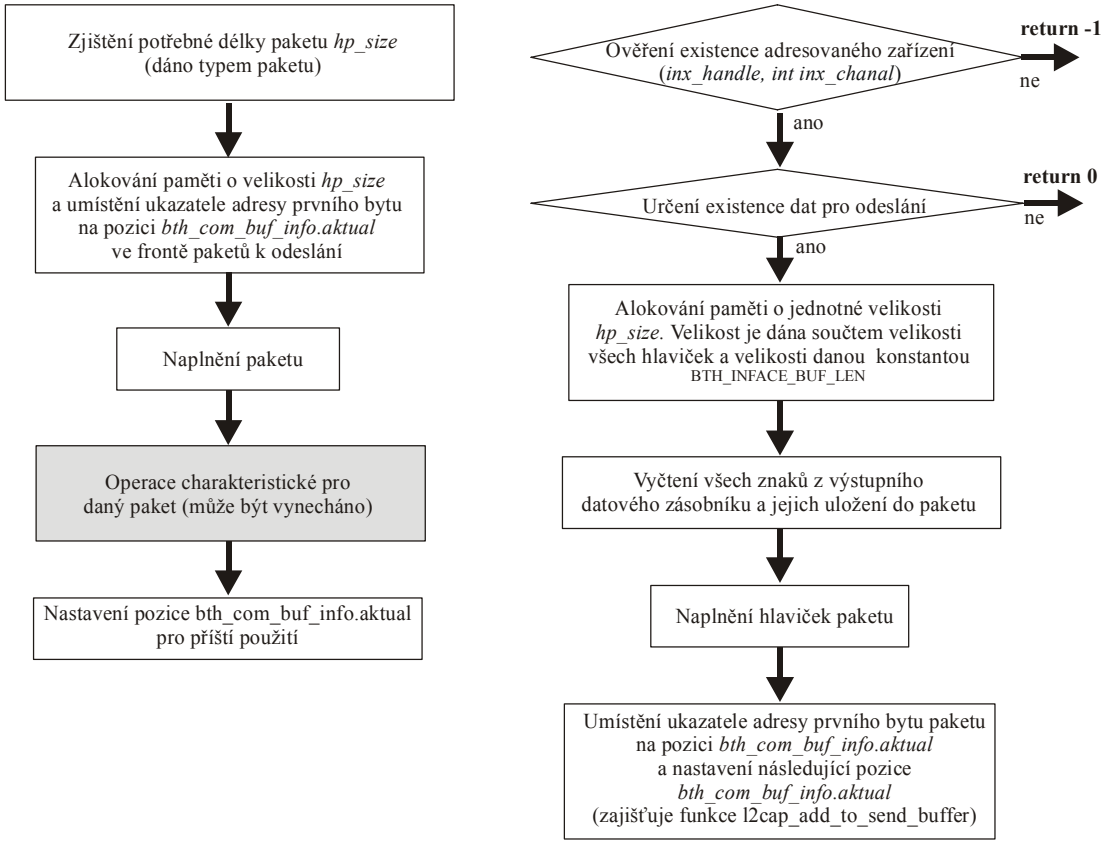
Tato kapitola se bude úzce dotýkat fronty paketů k odesílání a fronty paketů k potvrzení. Jednotlivé fronty a jejich činnost je detailně popsána v kapitole 6.2.3.



Obr. 6.7 Fronta paketů k odesílání

Na Obr. 6.7 je uvedena fronta paketů k odesílání. Fronta je vytvořena jako pole ukazatelů. Po inicializaci mají všechny položky pole hodnotu NULL. Pokud chceme odeslat příkazový resp. datový paket musíme pro něj nejprve alokovat potřebné množství paměti, v této paměti sestavit paket a na závěr ukazatel adresy prvního bytu

alokované paměti (paketu) uložit do fronty paketů k odeslání. Na příkladu z Obr. 6.7 jsou vytvořeny a ve frontě zařazeny tři pakety. Tyto pakety se budou odesílat do zařízení bluetooth v našem případě v pořadí ze shora dolů.



Obr. 6.8 Postup sestavování příkazových (vlevo) a L2CAP datových (vpravo) paketů

Sestavování příkazových paketů (první byte paketu označen 01) zajišťují funkce ze souboru `bth_command.c`. Způsob sestavení příkazového paketu naznačuje Obr. 6.8 vlevo.

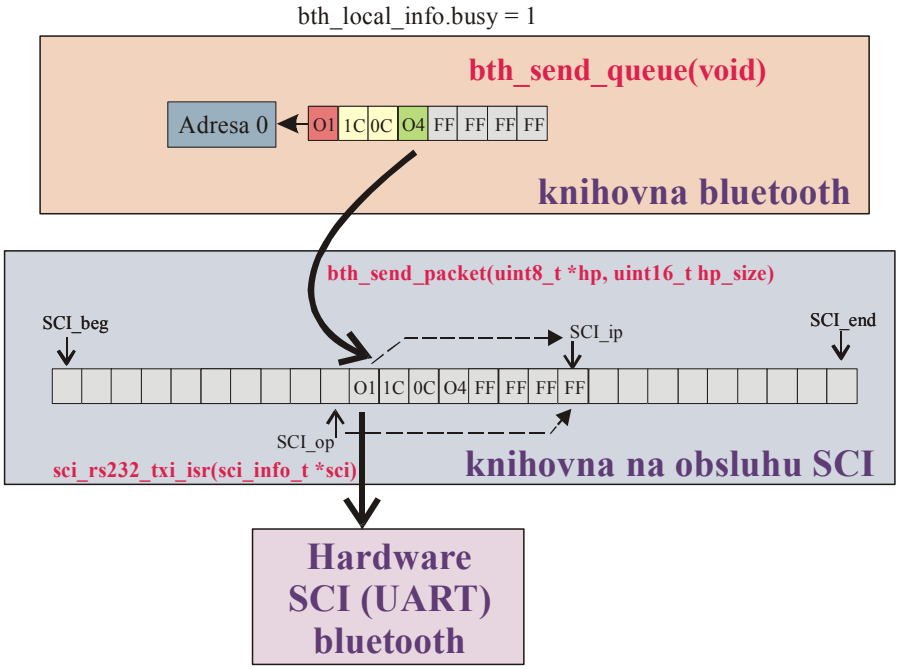
Sestavování datových paketů (první byte paketu označen 02) zajišťují funkce *l2cap_send_data* ze souboru `l2cap.c`. Tato funkce je cyklicky volána z nekonečné smyčky v hlavní funkci *main*. Funkce *l2cap_send_data* má tvar :

```
int l2cap_send_data(int inx_handle, int inx_chanal)
```

Parametr *inx_handle* určuje pořadové číslo (index) připojeného zařízení, a *inx_chanal* určuje pořadové číslo (index) domluveného L2CAP kanálu. Dohromady oba parametry určují adresáta pro odesílaná data.

Způsob sestavení paketu naznačuje Obr. 6.8 vpravo.

Odesílání již sestavených a zařazených paketů zajišťuje funkce *bth_send_queue*, která je cyklicky volána z hlavní nekonečné smyčky ve funkci *main*. Na Obr. 6.9 je znázorněn celý způsob odesílání paketu od okamžiku kdy je funkce *bth_send_queue* vyvolána až po okamžik, kdy se data tvořící příkazový HCI nebo datový HCI paket dostanou na sériové rozhraní spojující mikrokontrolér a zařízení bluetooth.

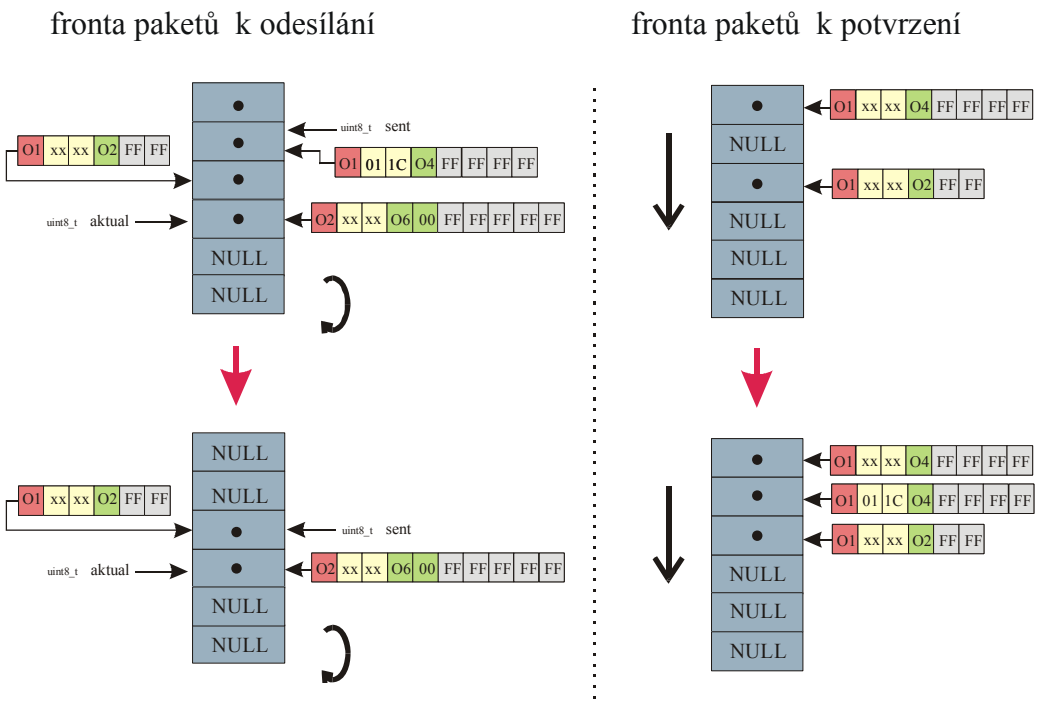


Obr. 6.9. Způsob odesílání paketu

Po vyvolání funkce *bth_send_queue* se nejdříve určuje zda-li je zařízení bluetooth volné pro příjem dat. Zaneprázdněnost bluetooth zařízení je indikována proměnnou *bth_local_info.busy*. Pokud je hodnota rovna 0 a ve frontě paketů k odesílání je zařazen nový paket signalizuje to, že je možné odesílat data. Ještě před samotným odesíláním je nutné zjistit, kolik znaků se bude odesílat, nebo-li kolik znaků tvoří paket. Informaci o délce paketu se přečte z odesílaného paketu. Ze znalosti délky dat (uvedeno v hlavičce paketu) a ze znalosti délky hlavičky paketu příslušející příkazovému resp ACL datovému paketu se určí celá délka paketu.

Následuje zavolání funkce *bth_send_packet*, které se dá jako parametr adresa prvního bytu paketu a celkový počet bytů paketu. Funkce *bth_send_packet*, pak data přesune do vysílací fronty sériového rozhraní odkud se v přerušeních průběžně znaky odesílají po sériové lince do zařízení bluetooth.

Po odeslání příkazového HCI paketu se hodnota ukazatele z indexu *sent*, který přísluší odeslanému paketu přesune do fronty paketů k potvrzení na pozici prvního výskytu hodnoty NULL. Ve frontě paketů k odeslání se na indexu *sent* (pozice odeslaného paketu) zapíše hodnota NULL. Na závěr se hodnota *send* nastaví na následující index fronty paketů k odeslání. Popisovaný děj je naznačen na Obr. 6.10.

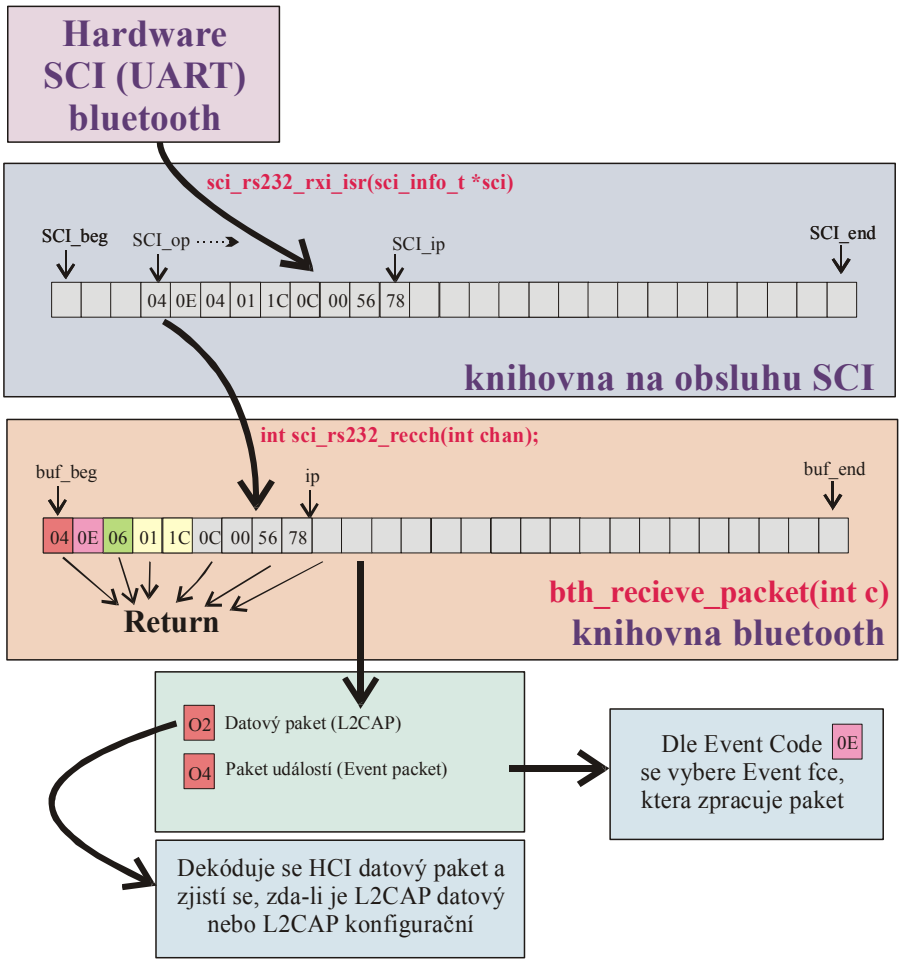


Obr. 6.10 Vývoj fronty paketů k odeslání a fronty paketů k potvrzení po odeslání příkazového HCI paketu

6.4.7 Příjem paketů a jejich rozklad

Tato kapitola se bude úzce dotýkat vstupní fronty, fronty paketů k potvrzování a přijímací datové fronty. Jejich činnost byla detailně popsána v kapitolách 6.2.4, 6.2.3 a 6.2.5.

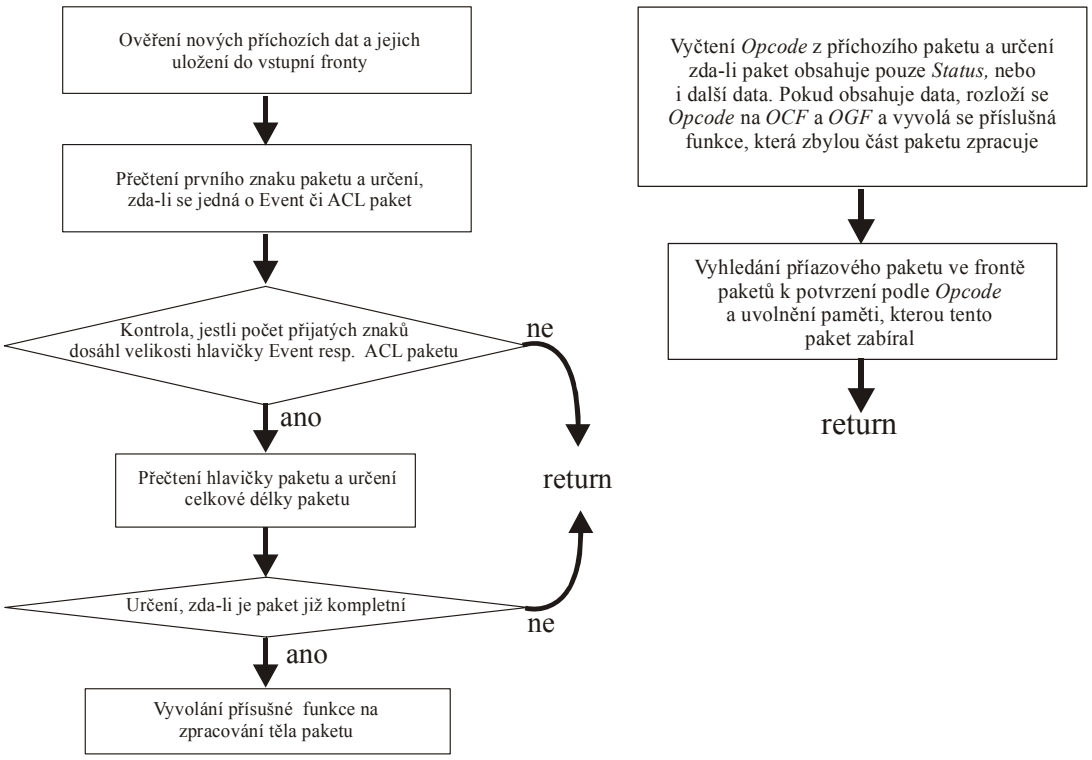
Příjem znaků a určení, zda-li je paket celý či nikoliv, zajišťuje soubor *bth_main.c.* Při postupném rozkladu přijatého paketu a zjištění, že se jedná o paket události (*Event*) se volají funkce obsažené v souboru *bth_event_acc.c,* případně ještě v souboru *bth_cmd_complete_event.* Pokud se jedná o L2CAP datový paket volají se funkce ze souboru *l2cap.c.*



Obr. 6.11 Způsob příjmu a zpracování vstupních dat

Na Obr. 6.11 je blokově naznačen mechanizmus příjmu znaků a jejich vyhodnocování. Znaky, které zasílá zařízení bluetooth do mikrokontroléru jsou sbírány v přerušeních od UART a ukládány do přijímací fronty. Tuto část zajišťuje knihovna pro příjem znaků z rozhranní SCI.

Funkce *int bth_recieve_packet(int c)*, která je cyklicky volána z hlavní smyčky programu (funkce *main*), ověřuje zda-li byl přijat nový znak z bluetooth zařízení. Pokud ano, uloží tento nový znak do vstupní fronty a inkrementuje ukazatel *ip*. Z dosud příchozích dat se určí typ paketu a podle délky hlavičky a počtu znaků ve vstupní frontě se rozhodne, jestli příchozí paket má již hlavičku celou. Pokud je hlavička paketu celá, lze z ní vyčíst kolik bytů obsahuje tělo paketu. Opět se udělá porovnání mezi počtem přijatých znaků a očekávanou délkou paketu a v případě, že si délky odpovídají se začne vykonávat navazující funkce. V případě, že se v některém popisovaném kroku zjistí, že paket není celý, funkce *bth_recieve_packet* se ukončí. Popisovanou činnost funkce *bth_recieve_packet* ilustruje i Obr. 6.12 vlevo.



Obr. 6.12 Blokové schéma funkce *bth_recieve_packet* a činnost funkce *bth_evt_cmd_complete*

Navazující funkce, která má za úkol zpracovávat tělo paketu se vyvolá u paketu události (*Event*) pomocí Event kódu. Samotné vyvolávání je v programu řešeno pomocí ukazatele na funkce. Většina Event funkcí (mimo *bth_evt_cmd_complet)* má za úkol pouze zpracovat příchozí data případně vyvolat reakci v podobě odeslání příkazového (*Command*) paketu.

Pokud má ovšem Event kód hodnotu 0E (*Command complet*), který volá funkci *bth_evt_cmd_complete* v souboru *bth_event_acc.c*, provedou se kroky vyobrazené na Obr. 6.12 vpravo. Paket *Command complet* vyjadřuje, že dříve odeslaný příkazový paket byl již zařízením bluetooth zpracován. Ve funkci *bth_evt_cmd_complet* se nejprve určí zda-li paket obsahuje pouze informaci o správnosti zpracování paketu, nebo jestli paket mimo informace *Status* obsahuje i návratové hodnoty (např. BD adresa …). Pokud paket obsahuje i návratové hodnoty přeloží se *Opcode* (součást těla *Command complet*) na *OCF* a *OGF*. Vzápětí se vyvolá funkce ze souboru *bth_cmd_complete_event.c* v závislosti na *Opcode*. Tato funkce zpracuje přiložená data. Pokud paket *Command complete* neobsahuje přiložená data (vrací pouze status), nebo tato data byla již zpracována (v závislosti na *Opcode*), vyhledá se ve frontě paketů k

potvrzení podle hodnoty *Opcode* příkazový (*Command*) paket, který předcházel právě zpracovávanému paketu. Paměť, kterou zabíral příkazový paket se uvolní a na pozici ve frontě paketů k potvrzení se zapíše hodnota NULL.

Knihovna je vytvořena tak, že v případě, že *Status* přijatého *Command complete* paketu signalizuje chybu, je možné odpovídající příkazový paket vyhledat ve frontě paketů k potvrzení a opět ho zařadit do fronty paketů k odeslání. Funkce, která by zajišťovala opětovné odesílání není v knihovně napsána.

V případě ACL datového paketu se z hlavičky vyčte *Connection Handle*, ze kterého se určí typ spojení (point-to-point, broadcast…). V případě paketu point-to-point se vyvolá funkce *bth_pkt_type_pointopoint*. Funkce které zajišťují ostatní druhy spojení nemají dopsaná těla funkcí. Ve funkci *bth_pkt_type_pointopoint* se určí, zda-li je paket datový, v tomto případě se data zkopírují do vstupní datové fronty pro další aplikace. Pokud je L2CAP konfigurační, volá se funkce *l2cap_signaling*, která je uvedena v souboru *l2cap.c.* Zde následně v příslušných funkcích probíhá rozklad L2CAP konfiguračního paketu a sestavování nových L2CAP paketů podle specifikace bluetooth naznačené v kapitole 4.4.5.2. Tyto pakety jsou následně zařazeny do fronty k odeslání paketů.

7 Zhodnocení a závěr práce

Cíl diplomové práce můžeme rozdělit na dvě části. První část je tvořena návrhem a realizací elektroniky hráče robota, který může komunikovat s okolím prostřednictvím rozhranní bluetooth. Druhá část je tvořena sestavením programu pro mikrokontrolér H8S/2638, který zajišťuje pohybové a komunikační schopnosti robotu.

Při návrhu elektronické části robotu byly kladeny požadavky na možnost budoucího rozšíření robotu o další schopnosti v podobě přídavných desek. Výsledný plošný spoj, který je výsledkem první části diplomové práce, ke dnešnímu dni funguje spolehlivě.

Druhá část, která je tvořená programovou výbavou pro H8S/2638 je tvořena souborem několika různých knihoven. Většina využívaných knihoven byla převzatá ať již bez úprav kódu nebo s úpravami, z dřívějších projektů.

Knihovna, která celá vznikla jako součást popisované diplomové práce, spravující síťový protokol využívaný architekturou bluetooth. Knihovna nemohla plně pokrýt tento síťový protokol z důvodu jeho rozsáhlosti, ale byla napsána tak, aby případné rozšíření knihovny nemuselo zásadním způsobem modifikovat její současnou verzi. Knihovna v současnosti umožňuje, aby připojené zařízení bluetooth bylo pouze podřízené hlavní stanici tj. plnilo roli *slave*. Možnost využívat zařízení jako řídicí stanice (*master*) je možné po dopsání funkcí, které řídicí stanice vyžaduje k činnosti. Jedná se o funkce pro skládání některých z příkazových paketů a zpracovávání jim odpovídajících událostí. Začlenění těchto funkcí do knihovny je velmi jednoduché a nemělo by být nutné modifikovat funkce na příjem a odvysílání paketů. Stejně tak by neměly být měněny struktury charakterizující vlastnosti lokálního zařízení či vzdáleného zařízení.

8 Použitá literatura

[1] Petr Vavřín , Fotbalové roboty z VUT v Brně mistry Evropy, 2001, http://www.odbornecasopisy.cz/automa/2001/au090172.htm

[2] Bohumil Honzík , Robotičtí fotbalisté z Brna obhájili titul, 2002, http://www.odbornecasopisy.cz/automa/2002/au070207.htm

[3] Zdenek Bradáč, Bezdrátové komunikace v automatizační praxi II: standard Bluetooth, 2003, http://www.odbornecasopisy.cz/automa/2003/au070338.htm

[4] Tým autorů, Renesas 16-Bit Single-Chip Microcomputer Hardware Manual H8S/2639, H8S/2638, H8S/2636, H8S/2630 Group, 2004.

[5] Tým autorů, Specification of the Bluetooth System, 2001.

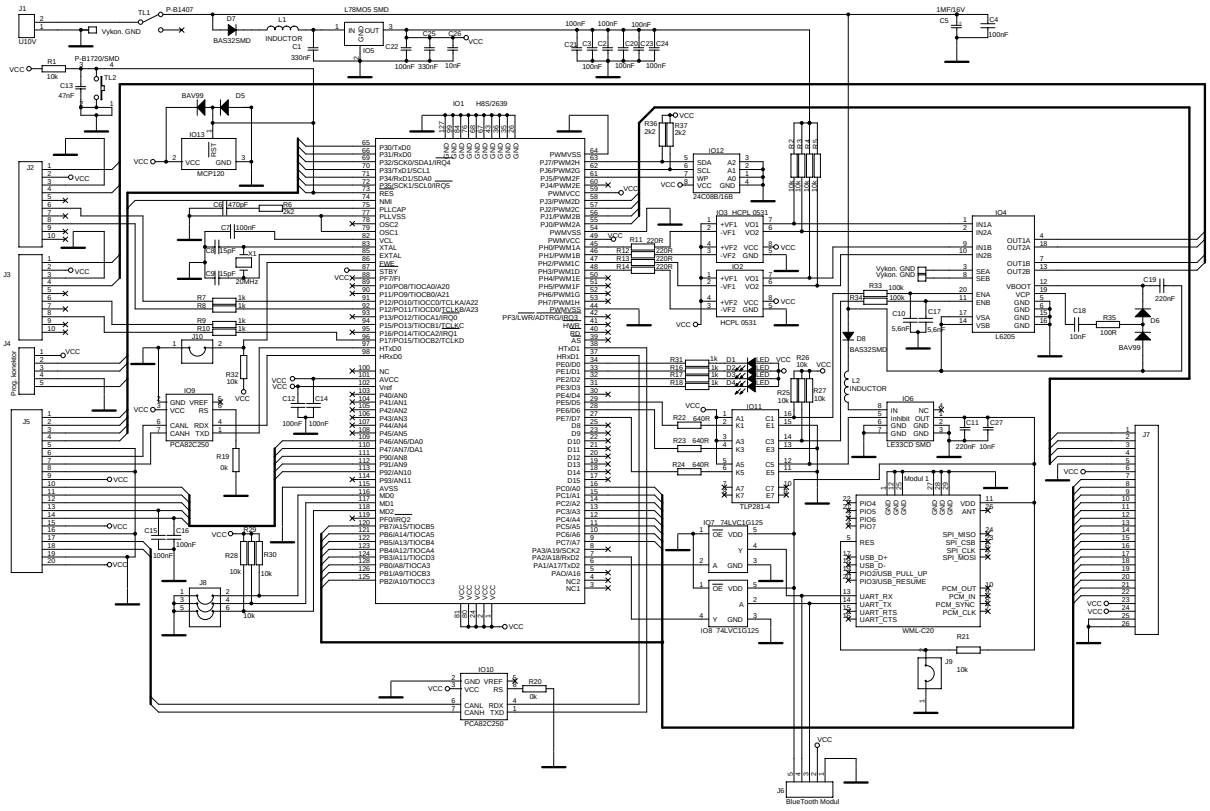
[6] Petr Mužíček, Dálkové řízení modelu, 2005 – diplomová práce

[7] Pavel Perman, Dálkové řízení modelu, 2005 – diplomová práce

[8] Rita Pužmanová, Osobní sítě -- Bluetooth a IEEE 802.15, 2002 http://www.lupa.cz/clanky/osobni-site-bluetooth-a-ieee-802-15/

[9] Tým autorů, Dual Channel, High Speed Optocouplers HCPL-0531, 1998

A. Příloha Schéma zapojení



B. Příloha Seznam součástek

Sum	Označení	Hodnota	Pouzdro	Funkce v obvodu
2	C25	330nF	SMD 0805	5V pro logiku
	C1	330nF	SMD 0805	5V pro logiku
6	C2	100nF	SMD 0805	Kondenzátor MC napájení
	C3	100nF	SMD 0805	Kondenzátor MC napájení
	C20	100nF	SMD 0805	Kondenzátor MC napájení
	C21	100nF	SMD 0805	Kondenzátor MC napájení
	C23	100nF	SMD 0805	Kondenzátor MC napájení
	C24	100nF	SMD 0805	Kondenzátor MC napájení
1	C4	100nF	SMD 0805	Filtrační kond. k napájení
1	C5	1MF/16V	SMD 0805	Tantal. kond. k napájecí
1	C6	470pF	SMD 0805	Kondenzátor MC
1	C7	100nF	SMD 0805	Krystal MC
2	C9	15pF	SMD 0805	Krystal MC
	C8	15pF	SMD 0805	Krystal MC
2	C17	5,6nF	SMD 0805	Výkonová část
	C10	5,6nF	SMD 0805	Výkonová část
1	C11	220nF	SMD 0805	3V pro Bluetooth
2	C14	100nF	SMD 0805	Kondenzátor MC ref. nap.
	C12	100nF	SMD 0805	Kondenzátor MC ref. nap.
1	C13	47nF	SMD 0805	Reset MC
2	C16	100nF	SMD 0805	Kondenzátor AD převodník
	C15	100nF	SMD 0805	Kondenzátor AD převodník
1	C18	10nF	SMD 0805	Výkonová část
1	C19	220nF	SMD 0805	Výkonová část
1	C22	100nF	SMD 0805	5V pro logiku
1	C26	10nF	SMD 0805	5V pro logiku
1	C27	10nF	SMD 0805	3V pro Bluetooth
4	D1	LED	SMD 1206	LED
	D2	LED	SMD 1206	LED
	D3	LED	SMD 1206	LED
	D4	LED	SMD 1206	LED
1	D5	BAV99	SOT23	Reset MC
1	D6	BAV99	SOT23	Výkonová část
2	D8	BAS32SMD	SMD 1206	3V pro Bluetooth
	D7	BAS32SMD	SMD 1206	5V pro logiku
1	IO1	H8S/2639	FP-128B	MC - H8S/2638
2	IO2	HCPL 0531	SO8	Galv. oddělení PWM signálu
	IO3	HCPL 0531	SO8	Galv. oddělení PWM signálu
1	IO4	L6205	SO20	Výkonová část
1	IO5	L78MO5 SMD	D-PAK	5V pro logiku
1	IO6	LE33CD SMD	SO8	3V pro Bluetooth
2	IO7	74LVC1G125	SOT753	Úprava napětí Bluetooth
	IO8	74LVC1G125	SOT753	Úprava napětí Bluetooth
2	IO9	PCA82C250	SO8	CAN

	IO10	PCA82C250	SO8	CAN
1	IO11	TLP281-4	SO16	4-vstup. optočlen
1	IO12	24C08B/16B	SOIC8	Paměť EEPROM
1	IO13	MCP120	SOT23	Reset MC
1	J1	U10V	100/TM1/2	Napájecí konektor
1	J2		100TM2/10	Konektor PWM kanál A,B
1	J3		100TM2/10	Konektor PWM kanál C,D
1	J4	Prog. konektor	100/TM1/5	Konektor pro programovací SCI1
1	J5		100/TM2/20	Univerzální konektor
1	J6	Bluetooth Modul	100/TM1/5	Externí bluetooth
1	J7		100/TM2/26	Univerzální konektor
1	J8		100/TM2/6	Nastaveni modu MC
1	J9		JUMPER100	Bluetooth
1	J10		JUMPER100	FWE MC
2	L1	Tlumivka	SMD 0805	5V pro logiku
	L2	Tlumivka	SMD 0805	Úprava napětí Bluetooth
1	Modul 1	WML-C20	WML-C20(anténa)	Bluetooth
1	R1	10k	SMD 0805	Reset MC
4	R2	10k	SMD 0805	Galv. oddělení PWM signálu
	R3	10k	SMD 0805	Galv. oddělení PWM signálu
	R4	10k	SMD 0805	Galv. oddělení PWM signálu
	R5	10k	SMD 0805	Galv. oddělení PWM signálu
1	R6	2k2	SMD 0805	Rezistor MC
4	R7	1k	SMD 0805	Ochranné rezistory IRC
	R8	1k	SMD 0805	Ochranné rezistory IRC
	R9	1k	SMD 0805	Ochranné rezistory IRC
	R10	1k	SMD 0805	Ochranné rezistory IRC
4	R11	220R	SMD 0805	Ochranné rezistory PWM
	R12	220R	SMD 0805	Ochranné rezistory PWM
	R13	220R	SMD 0805	Ochranné rezistory PWM
	R14	220R	SMD 0805	Ochranné rezistory PWM
4	R16	1k	SMD 0805	Ochranné rezistory LED
	R17	1k	SMD 0805	Ochranné rezistory LED
	R18	1k	SMD 0805	Ochranné rezistory LED
	R31	1k	SMD 0805	Ochranné rezistory LED
2	R20	0k	SMD 0805	CAN
	R19	0k	SMD 0805	CAN
1	R21	10k	SMD 0805	Bluetooth
3	R22	640R	SMD 0805	Ochranné rezistory 4-vs. optočlen
	R23	640R	SMD 0805	Ochranné rezistory 4-vs. optočlen
	R24	640R	SMD 0805	Ochranné rezistory 4-vs. optočlen
3	R25	10k	SMD 0805	4-vstup. optočlen
	R26	10k	SMD 0805	4-vstup. optočlen
	R27	10k	SMD 0805	4-vstup. optočlen
3	R28	10k	SMD 0805	Nastaveni modu MC
	R29	10k	SMD 0805	Nastaveni modu MC
	R30	10k	SMD 0805	Nastaveni modu MC

1	R32	10k	SMD 0805	FWE MC
2	R33	100k	SMD 0805	Výkonová část
	R34	100k	SMD 0805	Výkonová část
1	R35	100R	SMD 0805	Výkonová část
2	R36	2k2	SMD 0805	Paměť EEPROM
	R37	2k2	SMD 0805	Paměť EEPROM
3	R38	Vykon. GND	SMD 0805	Výkonová část
	R39	Vykon. GND	SMD 0805	Výkonová část
	R40	Vykon. GND	SMD 0805	Výkonová část
1	TL1	P-B1407	Conrad	Hlavni spínač napětí
1	TL2	P-B1720/SMD		Reset MC
1	X1	20MHz		Krystal MC

C. Příloha Výměna HCI paketů k propojení dvou bluetooth zařízení

Následující řádky demonstrují proces připojení mezi dvěmi zařízeními bluetooth, zaslání datového L2CAP paketu a ukončení vzájemného spojení. Výpis paketů je z pohledu zařízení, které v uzavřeném spojení zaujímá roli podřízeného zařízení.

Seznam níže uvedených paketů byl generován programem *hcidump*.

```
> 04 04 0A 31 16 CA 72 02 00 00 01 3E 01
      HCI Event: Connect Request (0x04) plen 10
      bdaddr 00:02:72:CA:16:31 class 0x3e0100 type ACL

< 01 09 04 07 31 16 CA 72 02 00 01
      HCI Command: Accept Connection Request (0x01|0x0009) plen 7
      bdaddr 00:02:72:CA:16:31 role 0x01
      Role: Slave

> 04 0F 04 00 01 09 04
      HCI Event: Command Status (0x0f) plen 4
      Accept Connection Request (0x01|0x0009) status 0x00 ncmd 1

> 04 03 0B 00 0B 00 31 16 CA 72 02 00 01 00
     HCI Event: Connect Complete (0x03) plen 11
     status 0x00 handle 11 bdaddr 00:02:72:CA:16:31
     type ACL encrypt 0x00

< 01 0D 08 04 0B 00 0F 00
      HCI Command: Write Link Policy Settings (0x02|0x000d) plen 4
      handle 11 policy 0x0f
      Link policy: RSWITCH HOLD SNIFF PARK

> 04 1B 03 0B 00 05
      HCI Event: Max Slots Change (0x1b) plen 3
      handle 11 slots 5

> 04 0E 06 01 0D 08 00 0B 00
      HCI Event: Command Complete (0x0e) plen 6
      Write Link Policy Settings (0x02|0x000d) ncmd 1
```

status 0x00 handle 11

< **01 0F 04 04 0B 00 18 CC**
HCI Command: **Change Connection Packet Type** (0x01|0x000f) plen 4
handle 11 ptype 0xcc18
Packet type: DM1 DM3 DM5 DH1 DH3 DH5

> **02 0B 20 0C 00 08 00 01 00 02 01 04 00 01 10 40 00**
ACL data: handle 11 flags 0x02 dlen 12
L2CAP(s): **Connect req**: psm 4097 scid 0x0040

< **02 0B 20 10 00 0C 00 01 00 03 01 08 00 40 00 40 00 00 00 00 00**
ACL data: handle 11 flags 0x02 dlen 16
L2CAP(s): **Connect rsp**: dcid 0x0040 scid 0x0040 result 0
status 0 Connection successful

> **04 0F 04 00 01 0F 04**
HCI Event: **Command Status** (0x0f) plen 4
Change Connection Packet Type (0x01|0x000f) status 0x00 ncmd 1

> **04 1D 05 00 0B 00 18 CC**
HCI Event: **Connection Packet Type Changed** (0x1d) plen 5
status 0x00 handle 11 ptype 0xcc18
Packet type: DM1 DM3 DM5 DH1 DH3 DH5

> **02 0B 20 0C 00 08 00 01 00 04 02 04 00 40 00 00 00**
ACL data: handle 11 flags 0x02 dlen 12
L2CAP(s): **Config req**: dcid 0x0040 flags 0x00 clen 0

< **02 0B 20 0E 00 0A 00 01 00 05 02 06 00 40 00 00 00 00 00**
ACL data: handle 11 flags 0x02 dlen 14
L2CAP(s): **Config rsp**: scid 0x0040 flags 0x00 result 0 clen 0
Success

< **02 0B 20 0C 00 08 00 01 00 04 01 04 00 40 00 00 00**
ACL data: handle 11 flags 0x02 dlen 12
L2CAP(s): **Config req**: dcid 0x0040 flags 0x00 clen 0

> **02 0B 20 0E 00 0A 00 01 00 05 01 06 00 40 00 00 00 00 00**
ACL data: handle 11 flags 0x02 dlen 14
L2CAP(s): **Config rsp**: scid 0x0040 flags 0x00 result 0 clen 0

Success

> **04 13 05 01 0B 00 03 00**
HCI Event: **Number of Completed Packets** (0x13) plen 5
handle 11 packets 3

> **02 0B 20 09 00 05 00 40 00 61 61 61 61 0A**
ACL data: handle 11 flags 0x02 dlen 9
L2CAP(d): cid 0x0040 len 5 [psm 4097]
0000: 61 68 6f 6a 0a = ahoj.

> **04 13 05 01 0B 00 01 00**
HCI Event: **Number of Completed Packets** (0x13) plen 5
handle 11 packets 1

> **02 0B 20 0C 00 08 00 01 00 06 03 04 00 40 00 40 00**
ACL data: handle 11 flags 0x02 dlen 12
L2CAP(s): **Disconn req**: dcid 0x0040 scid 0x0040

< **02 0B 20 0C 00 08 00 01 00 07 03 04 00 40 00 40 00**
ACL data: handle 11 flags 0x02 dlen 12
L2CAP(s): **Disconn rsp**: dcid 0x0040 scid 0x0040

> **04 13 05 01 0B 00 01 00**
HCI Event: **Number of Completed Packets** (0x13) plen 5
handle 11 packets 1

> **04 05 04 00 0B 00 13**
HCI Event: **Disconn Complete** (0x05) plen 4
status 0x00 handle 11 reason 0x13
Reason: Remote User Terminated Connection

D. Příloha Přiložené CD

Na následujících řádcích budou popsány jednotlivé adresáře z přiloženého CD disku.

`DiplomkaText`	Obsahuje pdf soubor s textem popisované diplomové práce
`BhtSpecifikace`	Obsahuje pdf soubor s kompletní specifikací bluetooth, ze které bylo čerpáno při psaní knihovny bluetooth.
`Katalogy`	Obsahuje systematicky řazené katalogové listy součástek použité při konstrukci robotu
`Schema`	Obsahuje soubory programu OrCAD 9.2. tvořící schéma, plošný spoj, seznam součástek, knihovnu kreslených součástek, knihovnu kreslených pouzder atd.
`Aplikace`	Obsahuje aplikace pod operační systém Linux. Aplikace *l2captest* umožňující komunikaci prostřednictvím technologie bluetooth se vzdáleným zařízením. Aplikace *hcidump* umožňuje zachytávat a vypisovat HCI příkazy, které lokální zařízení bluetooth (v PC) zasílá resp. přijímá od vzdáleného zařízení.
`SW_2638`	Obsahuje knihovny a programy pro H8S/2638.

Na následujících řádcích budou popsány některé důležité podadresáře adresáře *SW_2638* na přiloženém CD disku.

`\SW_2638\app\Mirosot_complet\`

Zdrojové a hlavičkové soubory programu pro řízení a komunikaci hráče-robotu Mirosot. Přeložení knihovny se provede příkazem „*make*“ v aktuálním adresáři. Nahraní celého programu do mikrokontroléru se provede příkazem „*make load*“

`\SW_2638\app\bth_complet\`

Zdrojové a hlavičkové soubory knihovny bluetooth. Přeložení knihovny se provede příkazem „*make*“ v aktuálním adresáři.

`\SW_2638\libs4c\pxmc\`

Zdrojové a hlavičkové soubory knihovny pxmc. Přeložení knihovny se provede příkazem „*make*“ v aktuálním adresáři.

`\SW_2638\libs4c\misc\`

Zdrojové a hlavičkové soubory knihovny textových příkazů pro řízení pxmc. Přeložení knihovny se provede příkazem „*make*“ v aktuálním adresáři.

Použití aplikací *l2captest* a *hcidump.*

`./hcidump -i hci1 -X –V`

Zachytává a vypisuje HCI příkazy, které jsou vysílány resp. přijímány bluetooth zařízením (*hci1*)

`./l2captest -i hci1`

Zařízení bluetooth (*hci1*) čeká na připojení od jiného zařízení.

`./l2captest -i hci0 00:02:72:C9:F6:AC`

Zařízení bluetooth (*hci0*) vyhledává a snaží se připojit vzdálené zařízení s BD adresou `00:02:72:C9:F6:AC.`