TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Řídící systém pásového robotu na platformě ARM

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Autor práce: **Michal Kleman**
Vedoucí práce: Ing. Miroslav Holada, Ph.D.

Liberec 2016

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

The control system of the tracked robot on ARM platform

Bachelor thesis

Study programme: B2646 – Information Technology
Study branch: 1802R007 – Information Technology

Author: **Michal Kleman**
Supervisor: Ing. Miroslav Holada, Ph.D.

Liberec 2016

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2015/2016**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Michal Kleman**
Osobní číslo:	**M13000272**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Řídící systém pásového robotu na platformě ARM**
Zadávající katedra:	**Ústav informačních technologií a elektroniky**

Zásady pro vypracování:

1. Seznamte se s pásovým mobilním robotem na pracovišti školitele, pozornost věnujte stávajícímu řídícímu systému.
2. Seznamte se s vývojovou deskou LPC Xpresso of firmy NXP, která využívá procesor platformy ARM.
3. Navrhněte nový řídící systém s procesorem ARM. Snažte se využít stávající hardwarové vybavení robotu (serva, snímače atd.).
4. Realizujte navržený systém včetně řídícího softwaru mikrokontroléru i vzdáleného ovládacího počítače.
5. Realizaci prezentujte ukázkovou demo úlohou řízení robota například v bludišti a diskutujte možnosti nového řídícího systému.

Rozsah grafických prací: Dle potřeby dokumentace

Rozsah pracovní zprávy: cca 30-40 stran

Forma zpracování bakalářské práce: tištěná/elektronická

Seznam odborné literatury:

[1] NOVÁK, Petr. Mobilní roboty: pohony, senzory, řízení, 1. vyd. Praha: BEN - technická literatura, 2004, 248 s. ISBN 80-730-0141-1.

[2] SUTTER, Herb a Andrei ALEXANDRESCU. C: 101 programovacích technik. Vyd. 1. Brno: Zoner Press, 2005, 231 s. Encyklopedie Zoner Press. ISBN 80-868-1528-5.

[3] JEAN J. LABROSSE, Jean J.Freddy Torres. [Mu]C/OS-III: the real-time kernel and the NXP LPC1700. Weston, Fla: Micrium Press, 2010. ISBN 978-098-2337-554.

Vedoucí bakalářské práce: **Ing. Miroslav Holada, Ph.D.**
Ústav informačních technologií a elektroniky

Datum zadání bakalářské práce: **14. září 2015**

Termín odevzdání bakalářské práce: **16. května 2016**

prof. Ing. Václav Kopecký, CSc.
děkan

Technická univerzita v Liberci - 14 -

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
vedoucí ústavu

V Liberci dne 14. září 2015

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 16.5.2016

Podpis:

Anotace v češtině

Tato práce se zabývá úpravami mobilního pásového robota, pracovně přezdívaného Pásák. Po jeho prozkoumání a prvotních úpravách software pro mikrokontolér PICAXE byl stávající procesor shledán nedostatečným. Přestože jedním z požadavků bylo do hardware zasahovat co nejméně, ukázalo se, že výměna řídící jednotky za výkonnější je nezbytná.

Práce se krátce zabývá historií robotiky a poukazuje na některé známé a průkopnické projekty mobilní robotiky. Dále popisuje důvody k rozhodnutí vyměnit starší řídící jednotku a ukazuje úpravy hardwaru pro možnost osazení novější jednotky.

Primárním zadáním je však aktualizace softwarového vybavení, a proto je značná část věnována popisu nového programového vybavení a zkoumá techniky, principy a návrhové vzory použité k dosažení požadovaného výsledku.

Závěr práce diskutuje možnosti dalšího rozšíření robota, jejich teoretickou náročnost jak po hardwarové, tak po softwarové stránce.

Klíčová slova: robot, LPCXpresso, LPC1769, C#, UART, FreeRTOS

Annotation in English

This thesis deals with modifications of mobile tracked robot, nicknamed Pásák. After first examination and initial modifications of software for PICAXE microcontroller actual processor was found insufficient. Although one of demands was to modify hardware as little as possible, the change of control unit was found necessary.

The thesis briefly deals with history of robotics and shows some well-known and innovative projects of mobile robotics. Further it describes reasons for decision to change the control unit and shows needed hardware modifications.

However, the primary task was actualization of software equipment, so the biggest part of this thesis is dedicated to description of new software and examines techniques, principals and design patterns used to achieve desired result.

Conclusion discusses possibilities of further extension of the Pásák robot, their theoretical severity both from hardware and software point of view.

Key words: robot, LPCXpresso, LPC1769, C#, UART, FreeRTOS

Poděkování

Rád bych poděkoval panu Ing. Miroslavu Holadovi, Ph. D., za odborné vedení bakalářské práce, za poskytnuté informace, cenné rady, součástky a materiály.

Obsah

Seznam obrázků

Seznam vzorců a kódů

Seznam použitých zkratek a symbolů

- Zkratka - Stručné vysvětlení (originální název)
- PC - Osobní počítač (Personal Computer)
- GPS - Globální Polohovací Systém (Global Positioning System)
- WiFi - Standard bezdrátové komunikace (WIreless FIdelity)
- SRI - Stanfordský výzkumný institut (Stanford Research Institute)
- A* - Počítačový algoritmus (A star)
- PWM - Diskrétní modulace (Pulse Width Modulation)
- GPIO - Vstupně/výstupní pin (General-purpose input/output)
- MCU - Jednočipový počítač (Micro Controller Unit)
- LED - Dioda emitující světlo (Light Emitting Diode)

- UART - Zařízení pro sériovou komunikaci (Universal Asynchronous Receiver and Transmitter)
- JTAG - Standard pro testování elektroniky (Joint Test Action Group)
- IDE - Vývojové rozhraní (Integrated Development Environment)
- GNU - Svobodný operační system (rekurzivní GNU's Not Unix)
- GDB - Nástroj pro hledání chyb v software (GNU Debugger)
- RTOS - Operační system reálného času (Real Time Operating System)
- CMSIS - Abstrakční vrstva nad hardwarem procesorů Cortex-M (Cortex Microcontroller Software Interface Standard)
- DAP - Port pro přístup k informacím o debugování (Debug Access Port)
- NVIC - Prioritní ovladač přerušení (Nested Interrupt Vector Controller)
- SRAM- Statická paměť (Static Random Access Memory)
- USB - Univerzální komunikační sběrnice (Universal Serial Bus)
- GPDMA - Přímý přístup do operační paměti (General Purposes Direct Memory Access)
- I2C - Datová sběrnice (Inter-Integrated Circuit)
- I2S - Sběrnice pro přenos zvuku (Intgrated Interchip Sound)
- ADC - Analogově-digitální převodník (Analog/Digital Converter)
- WDT - Časovač hlídající selhání systému (Watchdog Timer)
- PMU - jednotka obstarávající užívání energie (Power Management Unit)
- ID - Jedinečný unifikátor (Identity / Identity Document)
- SVN - Verzovací nástroj (Subversion)
- FIFO - Způsob čtení zásobníku (First In, First Out)
- LIFO - Způsob čtení zásobníku (Last In, First Out)
- CRC - Hashovací funkce (Cyclic Redudancy Check)
- WYSIWYG - Typ editoru rozložení prvků (What You See Is What You Get)
- XML - Počítačový jazyk (Extensible Markup Language)
- PHP - Programovací jazyk (rekurzivní PHP: Hypertext Preprocessor)
- CTS - Unifikovaný typový system (Common Type System)
- GPL - Všeobecná veřejná licence (General Public License)

1 Úvod

Stále častěji se okolo nás setkáváme s pohyblivými, na dálku ovládanými roboty. Tyto produkty vědeckého bádání mají užití v ohromné škále lidských činností, od natáčení sportovních událostí (například kvadrikoptérami se zavěšenými kamerami) po záchranu osob ve zřícených budovách (přesně k tomu by se hodil robot typu Pásák).

Sestavit jednoduchého robota tohoto typu přechází z okruhu doktorských prací spíše k dětským hrátkám, například se stavebnicí Lego Mindstorms. Jedním takovým jednoduchým robotem „na hraní, " nebo zkoušení nepoznaných technologií, je i Pásák. Původní softwarová výbava byla navržena pro mobilní telefony a firmware robota zvládal pouze jednoduché příkazy „otoč se doprava", „otoč se doleva" a jeď. Interaktivita ovládání by měla spočívat v možnosti plynule měnit otáčky jednotlivých motorů a robotem „spojitě" otáčet, jako to umí tanky nebo sněžné rolby.

Aby bylo možné plně využít potenciál robota, bylo nutné aktualizovat nejen software pro nízkoúrovňové řízení (software mikrokontroléru robotu), ale i software vysokoúrovňového řízení (software řídícího PC). Pro maximální efektivitu návrhu je samotné řízení a implementace algoritmů ovládajících robota umístěna ve vysokoúroňové (serverové) části, zatímco nízkoúrovňová (klientaská) část se stará pouze o vykonávání dílčích příkazů a odesílání hlášení zpět na server.

1.1 Mobilní robotika

Z hlediska schopnosti pohybu v prostoru lze roboty rozdělit do dvou základních kategorií. První jso

u stacionární, kteří jsou fixně upevněni k podkladu. Do této kategorie spadají například průmysloví roboti či manipulártory. Druhou kategorii tvoří roboti mobilní, kteří jsou schopní měnit svou polohu v prostoru v závislosti na požadovaném úkolu. Využití mobilních robotů je velmi široké, například v místech, kam se člověk nemůže dostat (průzkumní roboti), nebo je lidská přítomnost na takovýchto místech nebezpečná (pyrotechničtí roboti). V průmyslu se užívají robotická vozítka ve skladech a například společnost Amazon užívá autonomní letecké drony k rychlé distribuci balíčků. K praktické aplikaci však není třeba zacházet do takových extrémů – poslední dobou se automatické vysavače stále častěji objevují i v běžných domácnostech. [1]

1.1.1 Typologie mobilních robotů podle způsobu pohybu

Pohyblivé roboty je možné také dělit, a to dle několika základních charakteristik. Kromě prostředí, ve kterém se robot pohybuje, to může být druh kontaktu pohybového ústrojí s podkladem, nebo stabilita robota.

Stabilita je definovaná počtem a geometrií kontaktních bodů a těžištěm robota. Staticky stabilní jsou takoví roboti, které je možné zastavit v jakýkoliv moment, aniž by robot spadl. Typicky tak jde o zařízení, která mají například tři kola, kterými nejde proložit přímka a jejich základna tedy geometricky definuje rovinu v prostoru, pásová vozidla,

nebo kráčející roboti, kteří se v každé fázi pohybu dotýkají podložky minimálně ve třech místech. Dynamická stabilita je opačný případ, a tak robot nemůže být zastaven v libovolné pozici. Typicky jde o skákající a kráčející roboty, kteří nesplňují kritérium tříbodové opory, nebo dvoukolová vozítka typu Segway.

Z hlediska druhu kontaktu s podkladem můžeme definovat počet kontaktních ploch, jejich velikost, tvar, úhel kontaktu a tření mezi plochami a podkladem. Tyto parametry ovlivňují schopnost robota se hýbat, a to ve smyslu zrychlení (dle koeficientu tření kontaktní plochy s plochou podkladovou), nebo maximálního úhlu podložky, po jaké je robot schopný se pohybovat.[2]

Dle typu prostředí rozeznáváme roboty pohybující se po souši, ve vodě, vzduchem či ve vakuu, případně roboty hybridní. Vlastnosti cílového prostředí ovlivňují, jaký způsob pohybu by měl robot implementovat, například Pásák by zcela jistě daleko nedoplul.

Holonomika robota pak popisuje počet stupňů volnosti, ve kterých se může robot pohybovat. Robot, kterého bychom označili jako holonomický, má stejný nebo vyšší počet stupňů volnosti, než je celkový počet generalizovaných souřadnic nutných k popisu polohy robota, a tedy může změnit směr pohybu bez nutnosti současného pohybu v ostatních osách. Neholomický robot potřebuje pro některé směry nebo typy pohybu zároveň pohyb v ostatních osách. Příkladem holomického zařízení by mohl být vrtulník, který se může otáčet nezávisle na rychlosti pohybu, neholomický je pak automobil, který k otočení potřebuje vykonávat pohyb vpřed.[3]

1.1.2 Typologie mobilních robotů podle senzorového vybavení

Možné užití robota je dáno kromě způsobu pohybu i jeho senzorickou výbavou. Senzory lze rozdělit do dvou kategorií podle dat, která měří – měří-li senzor data o samotném robotu (například otáčky motoru, teplota procesoru, stav baterie), mluvíme o senzoru proprioceptivním. Exteroceptivní senzory čtou data z okolního prostředí a umožňují tak vytvářet jeho model, do této kategorie můžeme zařadit například GPS, kompas, sonar.

Senzory jde zpravidla dále rozdělit podle způsobu získávání dat. Pasivní senzory získávají informaci o měřeném objektu na základě přijaté energie od samotného předmětu. Aktivní senzory pak měří hodnotu tak, že energii vyšlou, zpravidla formou vlnění, a zaznamenají reakci okolí. Do kategorie pasivních řadíme senzory teplotní, kamerové, nebo dotykové. Aktivní jsou pak například ultrazvukové senzory, sonary (mechanické vlnění) a infračervené senzory (elektromagnetické vlnění).[2][3]

1.1.3 Typologie mobilních robotů podle metody řízení

Podle metody řízení lze roboty rozdělit do dvou skupin. Dálkově řízení roboti jsou takoví, kteří jsou naváděni operátorem a vykonávají přímo jeho příkazy. Tyto lze dále rozdělit na teleoperované, kde má operátor přímé vizuální spojení s robotem, nebo

teleprezenční, kde má operátor pouze informace ve smyslu modelu prostředí, ve kterém se robot nachází.

Autonomní robot je takový, který prochází prostředím bez přímého zásahu operátora, lidský vstup je tak omezen pouze na zadání cíle, kterého má robot dosáhnout, a řídící systém robota vyhodnotí, jakými prostředky bude cíle dosaženo. Tento řídící systém může být proveden několika způsoby, v zásadě jej však jde zařadit do jedné ze tří kategorií: reaktivní architektura, funkční dekompozice, nebo hybridní implementace předchozích.

Reaktivní architektura je princip, kdy robot přímo reaguje na jeho bezprostřední okolí. Robot zpracovává aktuálně měřená data a z nich vyvozuje reakci, aniž by znal model problému, jenž řeší. Tento přístup tak dokáže reagovat pouze na aktuální situaci a není schopen jakéhokoliv plánování nebo vytváření modelu.

Řešit složitější úkoly může přístup funkční dekompozice díky vlastnímu modelu okolí, který může robot buď dostat, nebo vytvářet a průběžně zlepšovat pomocí dat získaných z vlastních senzorů. Systém užívající zpětnou vazbu ke zpřesňování modelu úlohy však nemusí být stabilní a v praxi může selhat.[1]

1.1.4 Metody komunikace s mobilními roboty

Existuje několik způsobů, jak dostat instrukci od operátora k cílovému zařízení. Výběr metody komunikace závisí nejen na prostředí, ve kterém se má robot pohybovat, ale i na požadavcích na spolehlivost nebo rychlost spojení.

Asi nejjednodušším způsobem komunikace je kabelové spojení. Pro komunikaci kabelem existuje veliké množství protokolů, ať již sériových, nebo paralelních, synchronních, nebo asynchronních. Hlavní výhodou kabelového spojení je rychlost přenosu, vysoká odolnost vůči rušení a malá pravděpodobnost výpadku komunikace, tedy pokud mezi vámi a robotem nestojí sabotér. Nevýhodou je samozřejmě kabel samotný, který značně omezuje možnou působnost robota, zvyšuje jeho hmotnost a výrazně zvyšuje náročnost režie pohybu, kdy například při průchodu bludištěm by se robot musel vracet a svůj kabel sbírat předtím, než by prozkoumal jinou větev zmíněného bludiště.

Komunikace přes rádiové vlny umožňuje velkou vzdálenost mezi robotem a řídícím zařízením, zpravidla však nebývá příliš rychlá. Rádiové vlny se tak využívají především u robotů, kteří mají implementované robustní autonomní řízení a nepotřebují dostávat dílčí instrukce, pouze komplexnější cíle. Nevýhodou použití rádiových vln v běžném prostředí je možnost vzájemného rušení komunikačních kanálů jednotlivých zařízení.

Další možností, jak předat robotu instrukce je infračervené záření. Stejně jako rádiové vlny, ani tento přístup neexceluje přenosovou rychlostí a je jednoduše rušitelný jakýmkoliv jiným optickým zdrojem emitujícím záření v pásmu, na něž je přijímač citlivý. Infračervené vlny se používají stále častěji jako senzory přiblížení pro detekci překážek, než jako fyzická vrstva komunikačního protokolu.

Bluetooth je komunikační standard, který postupně nahrazuje zmíněný „infrared“ kvůli několika lepším parametrům a vlastnostem. Jednak poskytuje rychlejší přenos dat, má vyšší dosah a je výrazně spolehlivější co se přerušení spojení týče, jelikož nepotřebuje přímý vizuální kontakt mezi vysílačem a přijímačem. Kromě těchto „bonusů“ oproti přenosu infračerveným zářením má Bluetooth další výhodu v jeho rozšíření – tento protokol je obsažený prakticky v každém přenosném zařízení, ať jde o notebook, tablet nebo mobilní telefon.[4]

Asi nejlepší parametry v kategorii bezdrátových spojení poskytuje standard WiFi, a to ať už se bavíme o přenosové rychlosti, dosahu, nebo stabilitě připojení. Takto silné spojení umožní robotu odesílat veškeré informace, včetně vizuálního záznamu, v reálném čase, a tak veškeré metody umělé inteligence, jako je například extrakce informací při zpracování obrazu, mohou probíhat na výpočetně silnějším serveru a šetřit tak robotu procesorový čas a baterii.

1.2 Historie a současnost robotů

Člověkem vytvoření pomocníci se v literatuře vyskytují již dlouho, lze poukázat třeba na golema, slovo „robot“ je však jedno z mála slov, které světové jazyky převzaly z češtiny. Poprvé se objevilo v roce 1920 v divadelní hře Karla Čapka R. U. R., tedy Rossumovi univerzální roboti, která varuje před vlivem moderních technologií na lidskou společnost. Za prvním moderní experiment s robotem je považována doktorská práce H. A. Ernsta, který v roce 1961 na MIT (Massachusetts Institute of Technology) připojil k počítači manipulátor.

Prvním proslulým mobilním robotem je „Shakey“[5], sestavený v centru výzkumu umělé inteligence výzkumného ústavu SRI (Stanford Research Institute) v roce 1969. Tento robot byl průlomový především díky své autonomnosti. Do této doby byly robotům předávány jednotlivé instrukce, které vykonávali, Shakey však byl schopný přijmout zadání komplexnější úlohy a sám jí rozložit na jednotlivé dílčí instrukce, které postuně vykonal. Tento projekt spojil široké spektrum stále aktuálních oborů včetně robotiky, počítačového vidění a zpracování přirozeného jazyka. Výzkum vyústil ve vytvoření mnoha dodnes užívaných přístupů a metod (nebo jejich vylepšení při první praktické aplikaci), za zmínku stojí například A* algoritmus, Houghova transformace nebo viditelnostní graf.

S rozvojem elektroniky a řídících algoritmů postupně roboti přestávali vypadat jako „krabice s koly“, příkladem může být například Wabot-1 vyrobený na univerzitě Waseda v Tokiu v roce 1973. Tento humanoidní robot byl schopný přijímat hlasové příkazy a podle nich zvedat a přemisťovat předměty.[6]

Humanoidní roboti však zcela jistě nejsou vhodní pro všechny účely. Například pro průzkum Měsíce byl navržen robot ATHLETE společnosti JPL, který připomíná spíš šestimetrového pavouka, než člověka. Při extraterestriální aplikaci je potřeba počítat s výrazným zpožděním řídícího signálu, a tak jsou roboti tohoto typu vybavení dostatečnou inteligencí, aby mohli dostat pouze cílové souřadnice a na místo se dostali bez jakékoliv další asistence nebo řízení.[7]

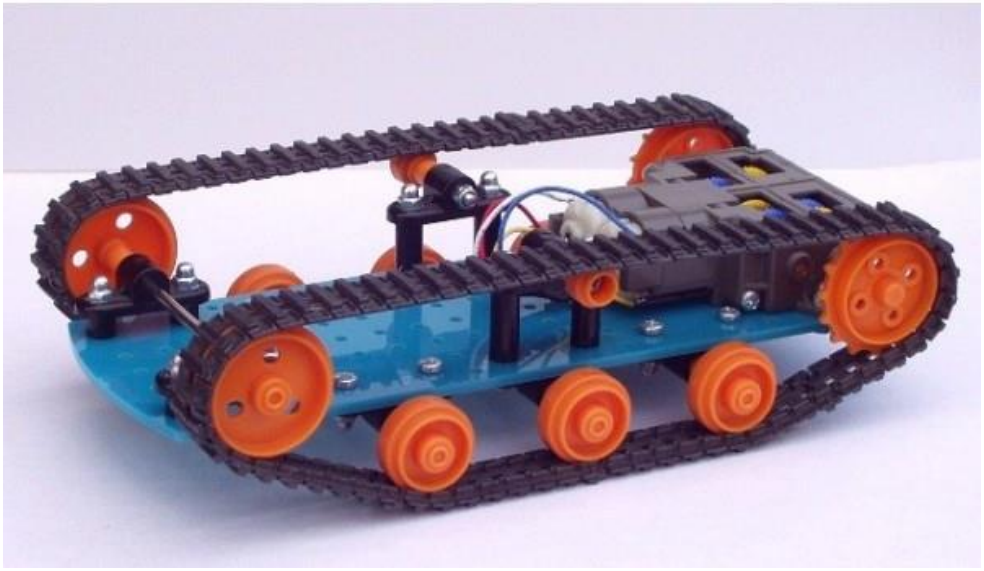
2 Původní sestavení Pásáka

Mobilní robot Pásák byl sestaven jako bakalářský projekt kolegy Psmolou a Štěpánkem v roce 2011. Původní projekt počítal s jednoduchým ovládáním robota přes mobilní telefon a zvládáním jen relativně málo příkazů, největším nedostatkem robotavšak byla neschopnost vykroužit zatáčku o libovolném poloměru. Mikrokontolér PICAXE [8] 18M má pouze jeden PWM (Pulse-Width Modultaion) kanál, navíc vyvedený na dva konkrétní piny, které autoři původního projektu použili jako GPIO (General-purpose input/output) piny a neuvažovali o jejich zapojení pro účel ovládání motorů. Pohybové schopnosti robota tedy končily u rovné jízdy a otáčení na místě. Při mém bakalářské projektu předcházejícím této práci jsem se snažil toto napravit a napsal jsem program, který generoval požadovaný signál na pinech, na kterých byly připojeny motory. Problém však nastal v neschopnosti staršího MCU (Microcontroller Unit) vykonávat současně více programových vláken a nemožnost použít asynchronní přerušení pro řídící komunikaci, takže robot buď jel definovanou trasu, nebo stál na místě a čekal na další příkaz. Na základě těchto informací bylo rozhodnuto, že mikrokontrolér PICAXE bude nahrazen modernější deskou LPCXpresso [9] LPC1769, která má sice také pouze jeden PWM kanál, ale zvládá multitasking a asynchronní přerušení, takže PWM signál pro druhý motor může být dopočítáván a obsluha komunikace nezastavuje vykonávání pohybu.

2.1 Realizace mechanické části robota

Pásák je vybaven dvěma stejnosměrnými motory. Stejnosměrný motor má stator opatřený střídavě opačně orientovanými permanentními magnety. Rotor (kotva) má na sobě umístěn několik vinutí, jejichž konce jsou vyvedeny do komutátoru, který zajišťuje přivedení správně orientovaného proudu do cívek rotující kotvy tak, aby se vytvořilo požadované magnetické pole, působící přitažlivě na magnetické pole statoru a vznikl tak otáčivý pohyb. Stejnosměrný motor má jednodušší řídící elektroniku než alternativa – motor krokový. Jednodušší elektronika však znamená složitější programové vybavení v případě potřeby komplexnějšího pohybu.

Pásový podvozek TANK-02 byl vybrán kvůli vyhovujícím rozměrům, nízké ceně a hlavně nastavitelné čtyřrychlostní převodovce TM70168.



Obrázek 1: Podvozek TANK-02 [23]

2.2 Elektronická výbava robota

Aby bylo možné robotem couvat, potažmo rychleji otáčet, je vybaven dvojitým H-můstkem. Obě hnaná kola jsou pak vybavena černým páskem se stříbřitými tečkami, které fungují jako odrazové plochy pro optické senzory otáček. Rušení vlivem denního světla bylo omezeno použitím fototranzistorů citlivých na infračervené záření. Vedle senzorů je pak umístěna infračervená LED dioda.

Dále je Pásák vybaven bezdrátovým bluetooth modulem ConnectBlue OEMSPA310 [10]. Technologie bluetooth byla zvolena kvůli její dostupnosti z téměř jakéhokoliv zařízení. Zmíněný modul se chová jako sériová linka, takže byl vhodný pro komunikaci s původním mikrokontrolérem PICAXE a lze jej připojit i na UART linku pro komunikaci s LPC1769.

3 Platforma LPCXpresso

LPCXpresso [11] je vývojová platforma vytvořená společností NXP, dříve známou jako Philips Semiconductors. Platforma zahrnuje širokou nabídku desek s procesory architektury ARM, které jsou dodávané s debugovací sondou JTAG (Joint Test Action Group). Desky jsou vyvíjené firmou NXP ve spolupráci se společnostmi Code Red Technologies a Embedded Artist. Softwarové vybavení zahrnuje vývojové prostředí LPCXpresso IDE (Integrated Development Environment) založené na známém prostředí Eclipse, GNU C kompilátor, linker, knihovny a GDB debugger rozšířený pro potřeby debugování vestavěných systémů. V rámci platformy je k dispozici i mnoho ukázkových kódů, podpora je poskytována přes téměř neaktivní fórum, eventuálně si lze samozřejmě doplatit podporu přes email.

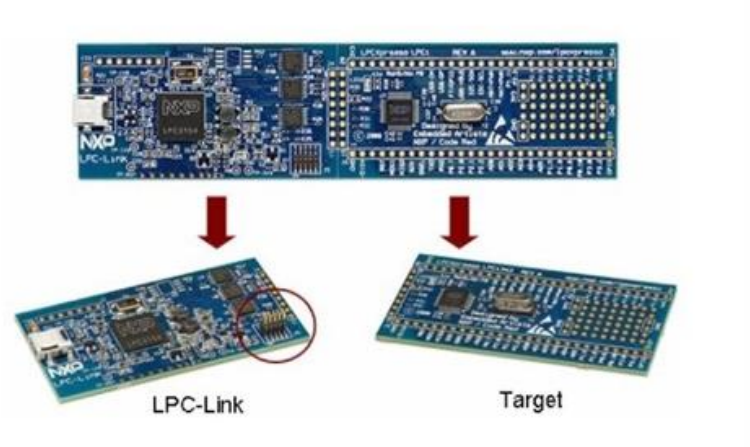
Pro programování desek LPCXpresso je možnost použít buď klasický jazyk C, nebo pokročilejší C++, ale jelikož jsou všechny tutoriály, ze kterých jsem vycházel, psané v C, zůstal jsem při své tvorbě u tohoto staršího jazyka.

3.1 Ukázkové projekty

Jak bylo zmíněno v předchozím textu, platforma poskytuje množství ukázkových projektů, demonstrujících schopnosti vývojových desek. Tyto kódy jsou dostupné v zip archivech na stránkách NXP a jsou okamžitě kompilovatelné a spustitelné. Zmínil bych například projekty implementující FreeRTOS, UART komunikaci, nebo analogově digitální převodník, z jejichž zdrojových souborů jsem ve své práci vycházel.

3.2 Vývojová deska LPCXpresso LPC1769

Deska s procesorem LPCXpresso LPC1769 byla uvedena na trh v roce 2009. Hlavními součástmi desky jsou testovací sonda CMSIS-DAP (Cortex Microcontroller Software Interface Standard Debug Access Protocol) SWD (Single Wire Debugger) a mikrokontrolér LPC1769. MCU je typu ARM Cortex-M3, architektury ARMv7-M. Na obrázku dole je vidět deska v původním stavu a její možné rozdělení v momentě, když už nepotřebujete ladící sondu.



Obrázek 2: Deska LPCXPresso LPC1769 [24]

Následuje seznam hlavních vlastností cílového zařízení [12]:

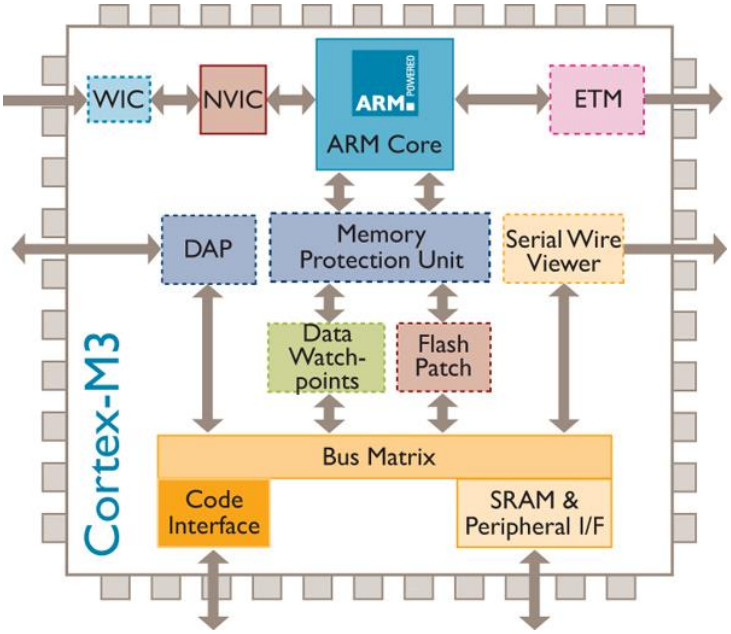
- ARM Cortex-M3 processor, taktovaný na frekvenci 120 MHz
- Zabudovaný Nested Vectored Interrupt Controller (NVIC)
- 512 kB programovatelné flashové paměti
- 64 kB SRAM
- In-System Programming (ISP) a In-Application Programming (IAP)
- Osmikanálový řadič přímého přístupu do operační paměti (GPDMA)
- Ethernet MAC
- USB 2.0 řadič
- 4x UART
- Dvoukanálový CAN 2.0B řadič
- SPI řadič s možnostmi synchronní, sériové nebo plně duplexní komunikace
- Třikrát rozhraní I2C
- I2S (Inter-IC Sound) rozhraní
- 70 obecných vstupně/výstupních pinů (GPIO)
- 12-bitový/8-kanálový Analog/Digital převodník (ADC)
- 4 hardwarové čítače
- Jeden PWM kanál pro kontrolu třífázových motorů
- Nízkonapěťové hodiny reálného času s vlastním oscilátorem
- WatchDog Timer (WDT)
- Čítač pro opakované softwarové přerušení
- Standardní JTAG testovací/debugovací rozhraní kompatibilní s existujícími zařízeními
- Integrovaný PMU (Power Management Unit)
- Čtyři možné úsporné napájecí módy: Sleep, Deep-sleep, Power-down, and Deep power-down
- Jeden zdroj 3.3 V (2.4 V až 3.6 V)
- Čtyři externí přerušení nastavitelná na detekci hrany nebo úrovně
- Vstup pro nemaskovatelné přerušení
- Řadič budícího přerušení (WIC)

- Power-On Reset (POR)
- Krystalový oscilátor s frekvencemi 1 MHz až 25 MHz
- Unikátní ID pro identifikaci zařízení

3.3 Procesor ARM Cortex-M3

Procesor ARM Cortex-M3 [13] je 32 bitový procesor poskytující výborný poměr výkonu k spotřebě energie. Tento nový procesor poskytuje několik nových funkčností oproti svým předchůdcům, například instrukční sadu Thumb-2, hardwarovou děličku, přerušitelné načítací a ukládací (load/store) instrukce, které mohou být dokončeny po konci přerušení, a integrovaný řadič přerušení s řadičem budícího (Wake-up) přerušení.

Procesor je vybaven třívrstvou instrukční pipeline, takže v jednom taktu je jedna instrukce vykonávána, následující je dekódována a třetí je načítána z paměti. Branch prediction obvod pak předpovídá, jak dopadne výsledek operace větvení, například if-then-else struktury nebo konec for cyklu, předtím, než je výsledek opravdu znám a zlepšuje tak tok v instrukční pipeline.



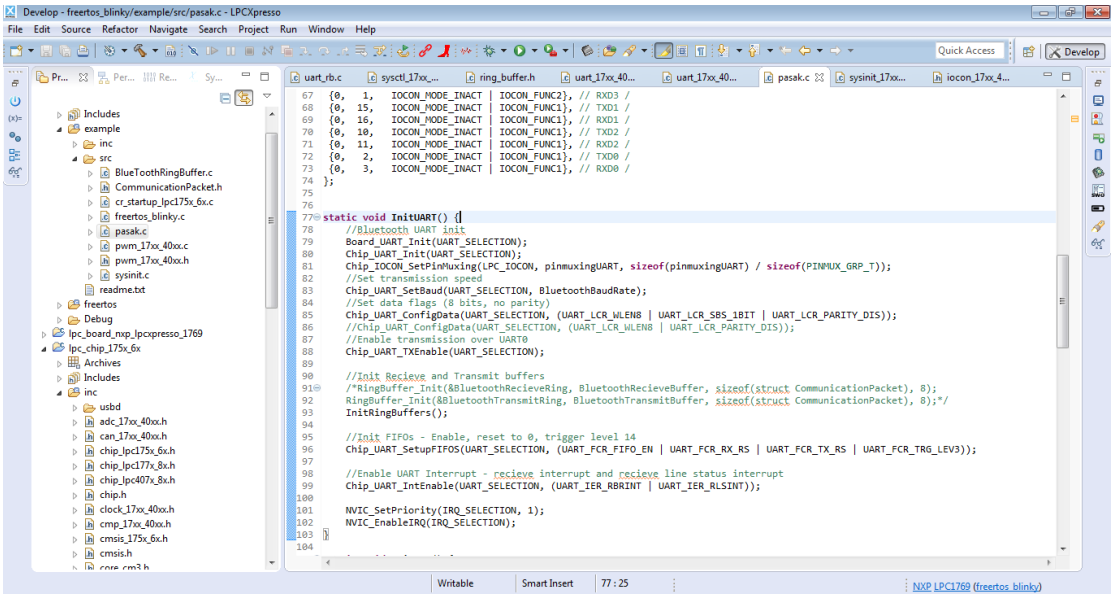
Obrázek 3: Schéma procesoru ARM Cortex-M3. [25]

3.4 LPC-LINK JTAG/SWD debugger

Deska LPCXpresso je dodávána s testovací sondou, kterou lze připojit k PC přes rozhraní USB. Obsluha tohoto rozhraní a debugovacích funkcionalit je zprostředkována pomocí mikrokontroléru ARM9 NXP LPC3154. Po oddělení této části desky lze k cílovému MCU připojit jinou ladící sondu, vice versa lze oddělenou část desky použít jako samostatný debugger.[12]

3.5 Vývojové rozhraní LPCXpresso IDE

Vývojové rozhraní poskytované společností NXP vychází ze známého IDE Eclipse a díky tomu je pro něj dostupná široká škála doplňků, které jsou originálně psané právě pro Eclipse. Po instalaci se toto prostředí musí aktivovat, na výběr je Free licence, nebo placená Pro licence. Hlavním omezením neplacené verze je maximální velikost programu, která je omezena na 256 kB. Pro licence má pak velikost spustitelného kódu neomezenou a předplatitel má k dispozici roční emailovou podporu přímo od inženýrů společnosti NXP.[12]



Obrázek 4: Vývojové prostředí LPCXpresso IDE

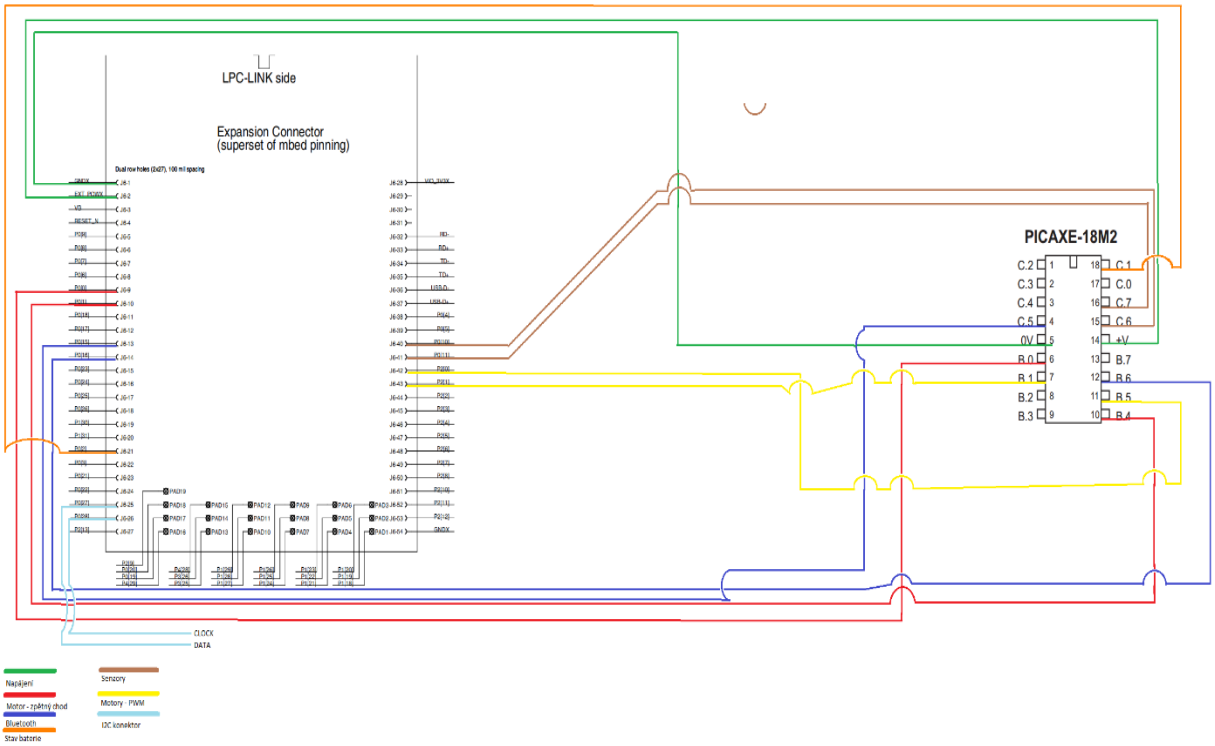
Vývojové rozhraní splňuje požadavky, které uživatel očekává od moderních programovacích nástrojů, jako je barvení syntaxe, formátování kódu, nebo management kódu ve smyslu jeho týmového sdílení a integrace do služeb jako je SVN (Apache Subversion) nebo GIT. Osobně mi při vývoji chyběla funkcionalita kontroly kódu v reálném čase, kdy se mnoho chyb (například nevalidní přetypování) ukázalo až při kompilaci. Vývojová studia jako je například Visual Studio nebo NetBeans jsou schopné tyto chyby detekovat už při psaní kódu, a tak programátorovi ušetří cenný čas.

LPCXpresso IDE podporuje širokou škálu moderních operačních systémů. Od společnosti Microsoft jsou to systémy Windows XP a dále, příznivci jablečných produktů mohou instalovat toto IDE pokud mají systém OS X 10.7.5 nebo novější a uživatelé Linuxu se mohou pustit do vývoje, pokud používají Ubuntu verze vyšší než 9, nebo Fedoru verze vyšší než 14.

4 Návrh nového systému

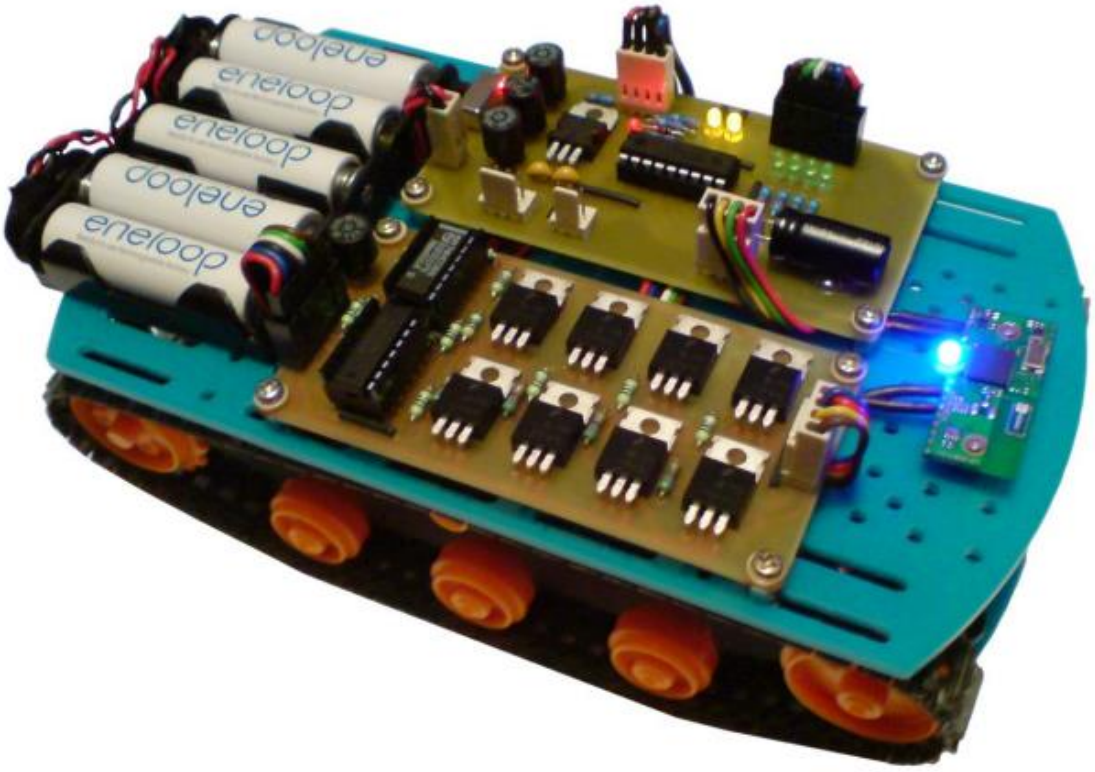
4.1 Osazení desky

Pro osazení nového mikrokontroleru bylo potřeba vytvořit hardwarový interface mezi paticí původního MCU a samotným LPC1769. Jelikož jsme s vedoucím projektu chtěli zachovat hardware robota v co nejoriginálnější formě, rozhodli jsme se piny patice vyvést v originálním rozložení do nového patra a použít jednoduchý tištěný spoj na přemapovaní pinů. Návrh mapování můžete vidět na obrázku níž, o samotnou realizaci tištěného spoje se postaral Ing. Miroslav Holada, Ph. D.

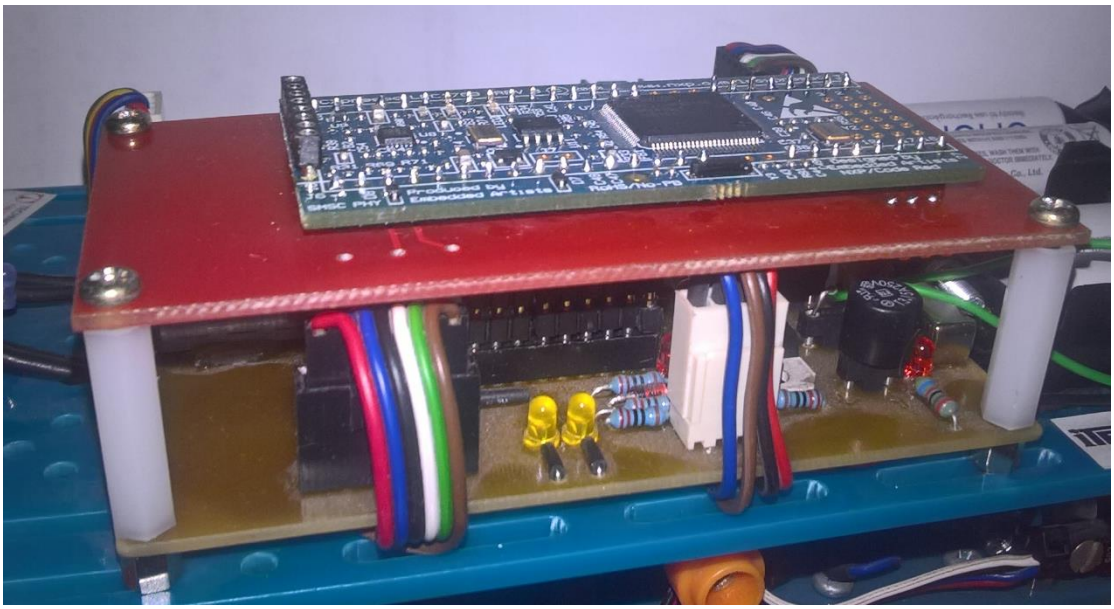


Obrázek 5: Návrh mapování pinů z Picaxe na LPC1769

Kromě komunikace, senzorů a ovládání motorů je z desky vyveden třípinový konektor, přes který je možné připojit další zařízení na sběrnici I2C. Plány budoucího rozšiřování nebyly v době zhotovování desky jasné, a tak je vyveden pouze jeden konektor pro připojení další periferie. Pokud bude požadavek periferií připojit víc, pravděpodobně bude potřeba desku znovu předělat a vyvést vstupů/výstupů více, nebo nové zařízení připojit dráty přímo do desky LPC.



Obrázek 6: Původní sestavení robota [26]

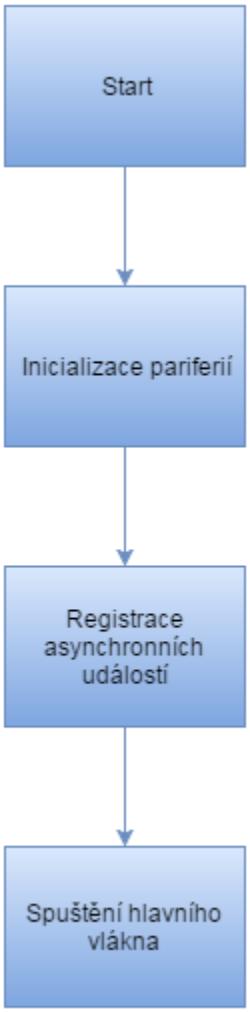


Obrázek 7: Nově vytvořené "patro" pro řídící jednotku LPCXpresso 1769

4.2 Návrh řídícího algoritmu

Main funkce inicializuje periferie desky a nastaví každému pinu jeho požadovanou funkcionalitu. Dále zaregistruje asynchronní obslužné rutiny pro komunikaci (UART) a senzory (GPIO). Nakonec probíhá nastavení priority a spuštění hlavního vlákna.

Algoritmus je popsán v následujícím vývojovém diagramu, jeho jednotlivé součásti jsou rozebrány v následujících podkapitolách.



Obrázek 8: Vývojový diagram funkce Main

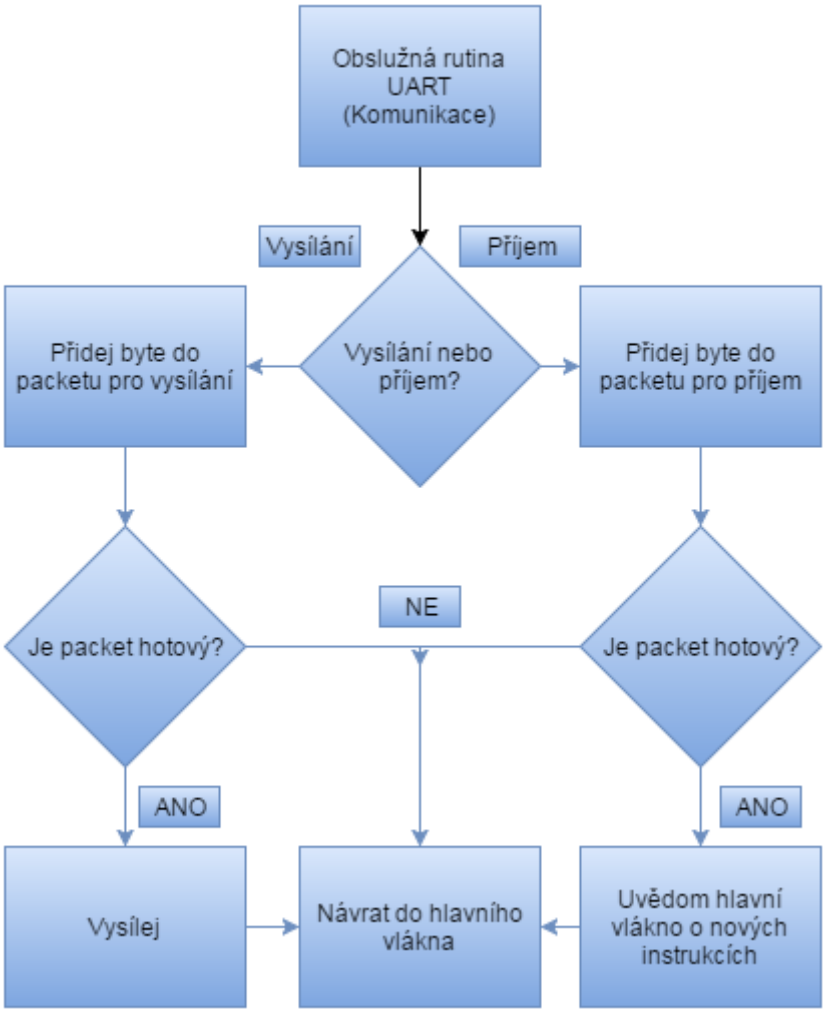
4.3 Obslužná rutina pro UART komunikaci

Pro komunikaci přes bluetooth je zaregistrována jedna asynchronní obslužná rutina, která odpovídá za správné skládání komunikačního paketu, jeho odeslání, nebo upozornění hlavního vlákna, pokud je přijímaný paket hotový.

Komunikace využívá na univerzitě standardizovaný protokol HB-12C, který je popsán v kapitole Použité technologie, podkapitola Protokol HB-12C. Jelikož tento protokol nespecifikuje fixní velikost přenášených dat, obslužná rutina má za úkol v paměti alokovat místo pro další data podle pole z hlavičky udávajícího délku přenosu.

Aby nebylo blokováno hlavní vlákno, aplikace obsahuje dva kruhové FIFO (First In, First Out) buffery držící aktuální pakety, do kterých tato rutina zapisuje, hlavní vlákno

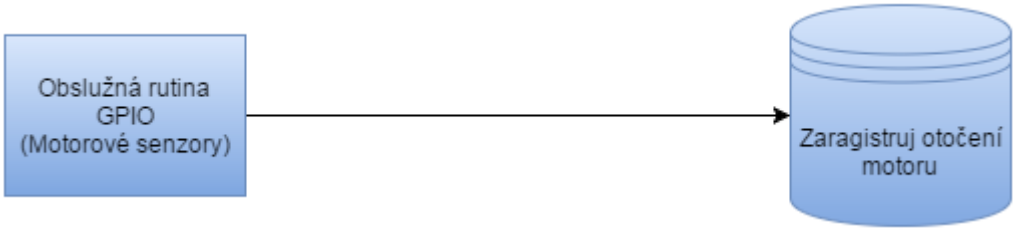
pak z těchto zásobníků pakety maže v momentě, kdy paket zpracuje. Mazání probíhá pouze nastavením „vlajky“ (flag) určující, že paket byl přečten a může být přepsán.



Obrázek 9: Vývojový diagram obslužné rutiny komunikace

4.4 Obslužná rutina pro GPIO přerušení

Senzory otáčení kol jsou připojeny na GPIO (General Purpose Input-Output) piny. Ty mají zaregistrovanou obslužnou rutinu zaznamenávající počet otáček a časovou značku, aby bylo možné zpětně zkonstruovat ujetou trasu. Ta může být případně zobrazena, nebo mohou tyto data být použity k návratu do předchozí pozice při ztrátě signálu s řídícím serverem.



Obrázek 10: Vývojový diagram obslužné rutiny GPIO přerušení

4.5 Hlavní vlákno

Hlavní vlákno programu se stará o vykonávání instrukcí obdržených od serveru. Po obdržení instrukcí zkontroluje CRC (Cyclic redundancy check) součet. Pokud tento souhlasí, přejde dál k vykonání instrukce, jinak odešle zpět na server zprávu o chybě. V současnosti jsou nadefinovány tři základní pokyny: Move, Stop, Report.

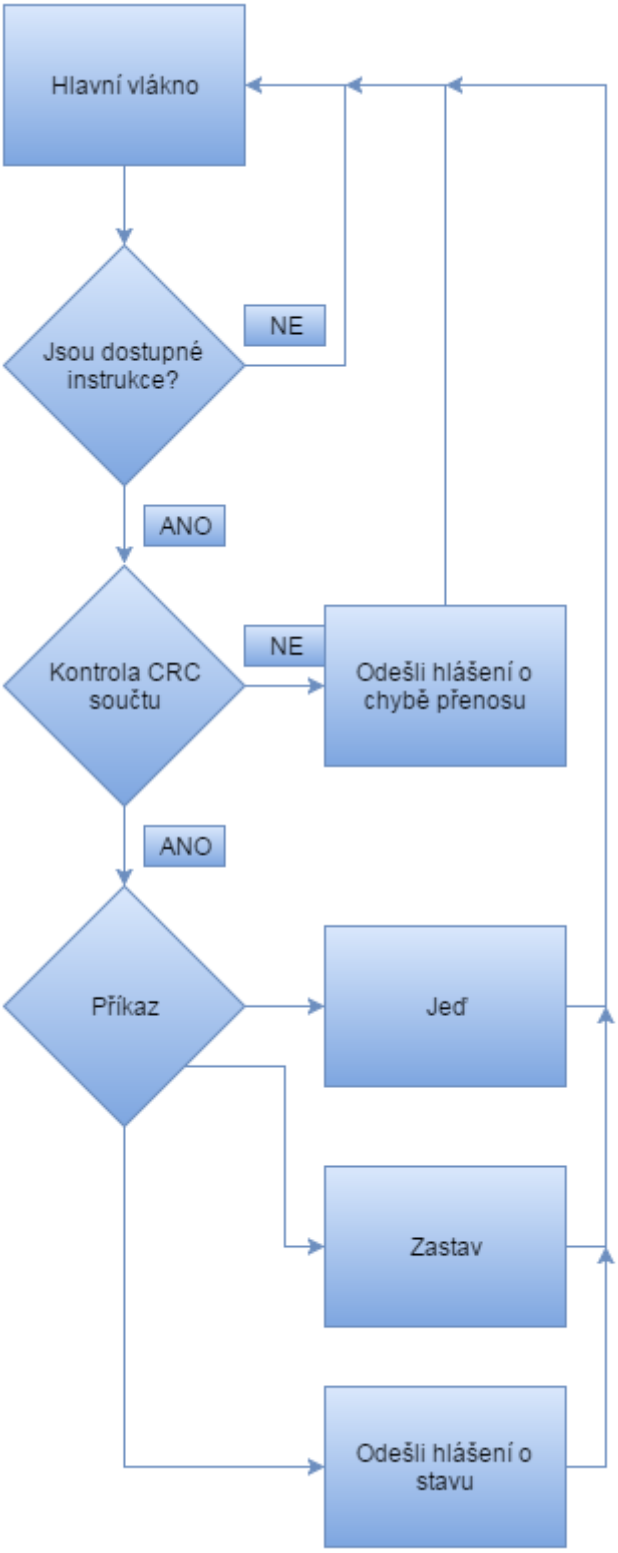
Paket žádající o zahájení pohybu by měl nést data specifikující, jak se má robot hýbat, tedy poměr výkonu motorů, případně dobu vykonávání, nebo maximální ujetou vzdálenost. Tato data si analyzuje z paketu metoda Move sama a pokud žádné neobdrží, zapne oba motory na plný výkon na neurčitou dobu.

Příkaz Stop ukončí aktuální pohyb bez ohledu na předchozí data předané při žádosti o záhájení pohybu, tedy pokud pohyb definuje například dobu jízdy, metoda Stop tato data nebere v potaz a okamžitě zastaví robota. Paket s příkazem Stop by měl přijít bez jakýchkoliv dalších dat.

Report je druhou metodou, která může mít další data specifikující informace, které má Pásák odeslat zpět na server. Defaultně, tedy bez doplňujících dat, robot odešle všechny dostupné informace, jinak je možné odeslat ujetou trasu, nebo aktuální stav baterie. Toto rozložení se může zdát trochu přebytečné vzhledem k množství dostupných dat, ale jde o krůček do budoucna, kdy se počítá, že Pásák bude obohacen o další senzory, případně o kameru.

Vyčlenit řízení do zvláštního vlákna, místo aby zůstalo v Main funkci, jsem se rozhodl hlavně proto, aby byl projekt co nejjednodušeji rozšiřitelný o další periferie. K režii přepínaných vláken je použit operační systém reálného času FreeRTOS. Další vlákna by se mohla starat například o čtení dat z kamery, jejich kompresi, vysílání nebo ukládání, případně by mohla mít na starosti režii uživatelů připojených přes WiFi atd.

Návrat vlákna do volající metody vyústí ve FaultException, která je v rámci defaultní implementace FreeRTOS [14] ošetřena nekonečným prázdným cyklem. Ošetření této výjimky v rámci Pásáku probíhá odesláním informace o chybě včetně stack trace (trasování zásobníku) a pokusu o restart programu. Tento scénář by se však neměl stát, jelikož hlavní vlákno obsahuje nekonečný while cyklus.



Obrázek 11: Vývojový diagram hlavního vlákna

5 Řídící software pro PC

Řídící software na straně ovládajícího počítače hraje v celém řídícím systému důležitou roli. Software zprostředkovává uživateli informace o stavu robota a poskytuje mu rozhraní, kterým může jednoduše odesílat robotu požadované příkazy.

5.1 Výběr prostředí

Aplikaci pro řízení Pásáka z PC je napsána v C#, a to z několika prostých důvodů. Primárně proto, že C# používám již dlouho, zadáním bylo aktualizovat software robota, a tak v této části úlohy nebyl důvod pro experimenty. Zadruhé jde o moderní objektově orientovaný jazyk, který je pro operační systém Windows „šitý na míru." Dále bych chtěl poukázat na robustnost frameworku .NET, širokou komunitu schopnou poradit a výbornou dokumentaci poskytovanou společností Microsoft. [15]

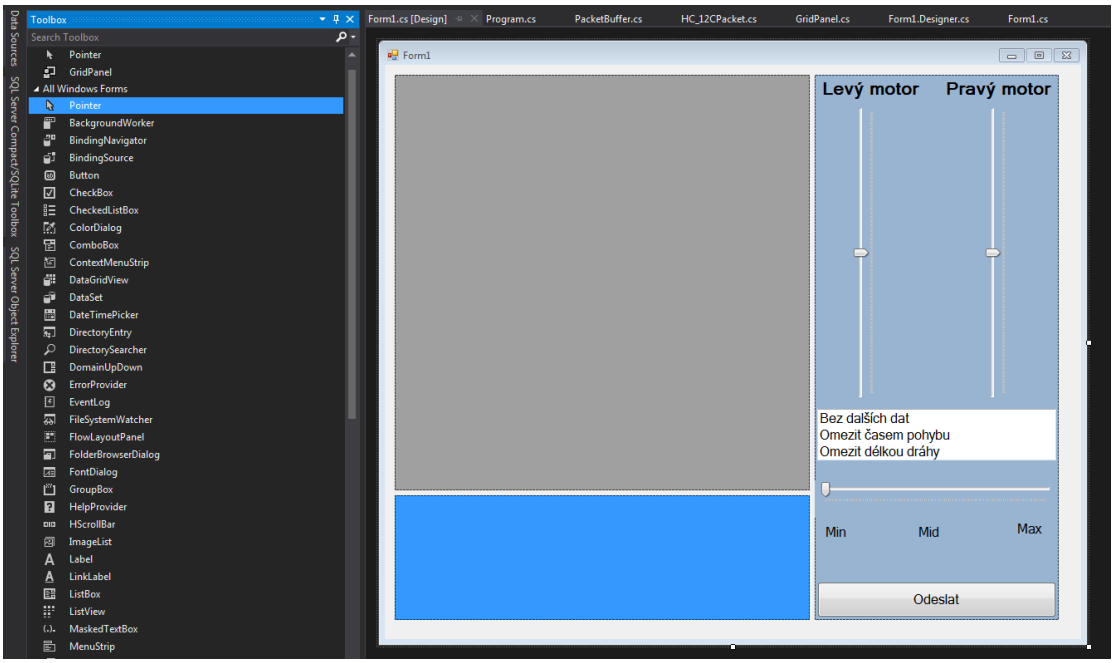
Jeden z návrhů doktora Holady byl, že v budoucnu by mohl být Pásák ovládán z webové aplikace, nebo z mobilního telefonu. K obojímu je C# vhodný, a tak jsem svou prací připravil půdu i pro navazující projekty. Webové aplikace lze v C# psát již dlouho, pro mobilní platformu je určen projekt Xamarin, který umožňuje psát nativní aplikace pro Windows Phone, iOS i Android. Xamarin je kalifornská firma založená v roce 2011 vývojáři projektu Mono, kterou v únoru 2016 koupil Microsoft.

5.2 Aplikace

Samotná aplikace pro řízení Pásáka je rozdělena do dvou částí. Jádro programu se stará o komunikaci s robotem a aplikační logiku, druhá část programu má na starosti grafický interface zobrazovaný uživatelem. Program je takto rozdělen, aby bylo možné grafickou nástavbu jednoduše vyměnit, jak je zmiňováno v předchozí podkapitole, a přitom nezasahovat do logiky ovládání.

5.2.1 Grafické rozhraní

Pro implementaci grafického rozhraní byla užita knihovna Windows Forms, která je standardní součásti frameworku .NET. Skládat uživatelské rozhraní pomocí této knihovny je velmi jednoduché a intuitivní, jelikož součástí vývojového prostředí Microsoft Visual Studio je i WYSIWYG (What You See Is What You Get, co vidíte, to dostanete) editor umožňující „naklikat" layout (plán, rozvrh) přes grafické rozhraní bez potřeby zasahovat do kódu.

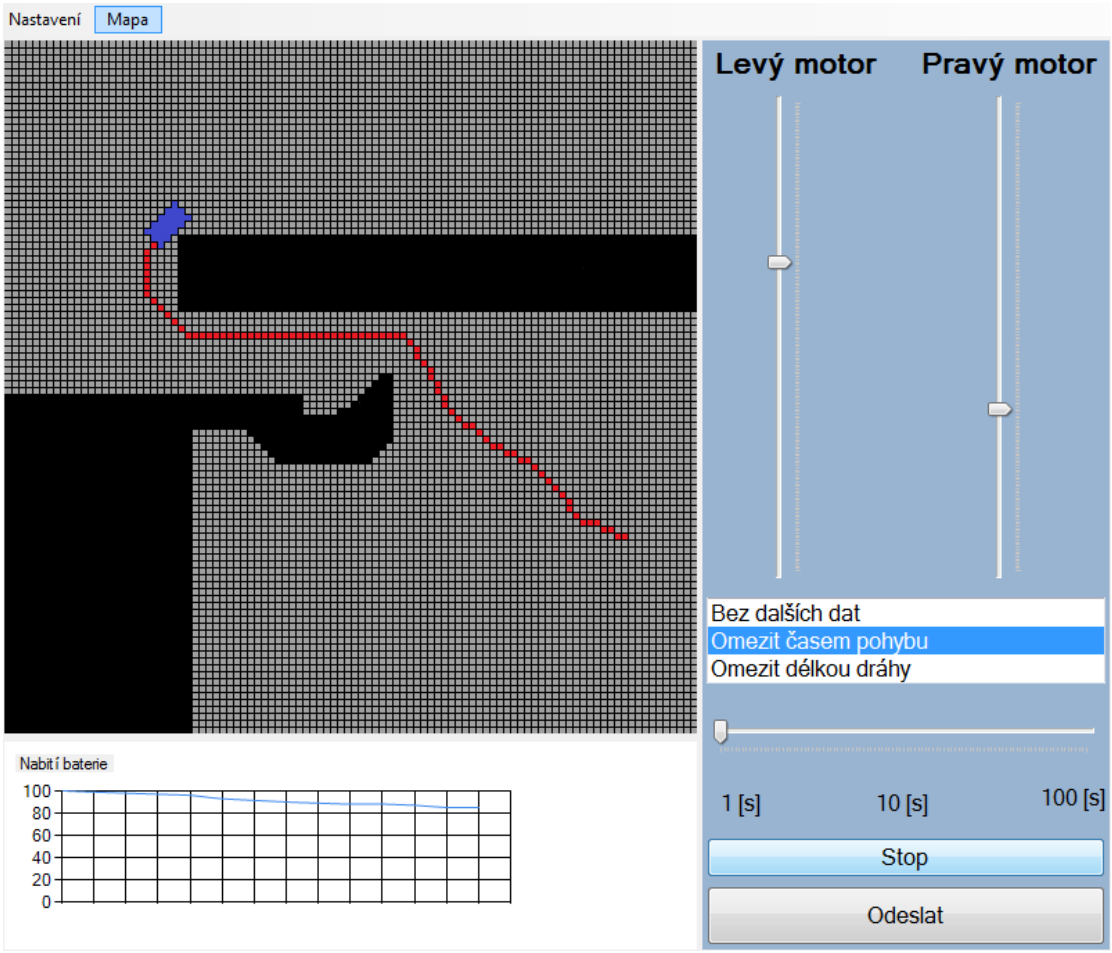


Obrázek 12: WYSIWYG editor Visual Studia

Jmenný prostor Pasak3.Visual obsahuje tři formuláře, kde jeden je určen pro samotné řízení robotu a zobrazení dostupných informací (ControlForm.cs), Druhý umožňuje uživateli definovat mapu, případně překážky, které se okolo nachází, nebo tuto mapu načíst ze souboru XML (MapForm.cs). Poslední formulář slouží k zobrazení historie komunikace, případně k ručnímu zadání příkazu (CommandHistoryForm.cs).

Kromě automaticky vygenerovaného kódu popisujícího rozložení a velikost jednotlivých komponent tato část obsahuje ještě obslužné rutiny pro uživatelské události. Třída ControlForm implementuje událost pro změnu hodnoty v rámci komponenty CheckedListBox, která určuje, jaká další data se mají robotu odeslat. Na výběr jsou možnosti „Bez dalších dat“, „Omezit časem pohybu“ a „Omezit délkou dráhy“. Dále jsou implementovány „event listenery“ (posluchači událostí, metody spuštěné, jakmile subject [vydavatel, nebo také pozorovaný] zavolá metodu Invoke(), více v kapitole 5.9 Návrhový vzor Observer) reagující na tlačítka Odeslat a Stop. Ty asynchronně předávají uživatelem zvolené nastavení jádru programu pro další zpracování. Pro kliknutí do mapy je implementována obslužná rutina, která opět asynchronně předá data jádru programu. Výsledkem této operace je vykreslení nejkratší možné trasy, jakou se Pásák do zvoleného bodu může dostat. Tato trasa je odeslána Pásáku po kliknutí na tlačítko Odeslat, přičemž pokud se změní jiná nastavení, je tato trasa smazána a vykoná se poslední uživatelem zadaný požadavek. Kromě zmíněných událostí má ControlForm, jakožto hlavní okno programu, rozbalitelné menu, ze kterého lze měnit nastavení a spouštět okna ostatní.

Třída MapForm zobrazí pravoúhlou mřížku a několik ovládacích prvků, které mění velikost mřížky, měřítko a počáteční umístění robota. Po kliknutí do mřížky se otevře dialog umožňující příslušnému poli nastavit časovou penalizaci, kterou by stálo Pásáka tímto bodem projet (ve smyslu násobku projetí neoznačenou mřížkou), případně určit, že daný bod je neprůchozí.



Obrázek 13: Vizualizace průjezdu robota uživatelem definovanou mapou

5.2.2 Jádro a logika

Jádro aplikace se skládá z několika částí, kde každá má svůj vlastní jmenný prostor. Pro účely komunikace je vyhrazen jmenný prostor Pasak3.Communication, zde-je singleton třídy CommunicationManager a třída Message. Třída Message je datovou strukturou odpovídající paketu komunikačního protokolu HB-12C, jeho jednotlivé části jsou popsány v kapitole 5.3 Komunikační protokol HB-12C. Kromě datových polí a tří konstruktorů (pro paket tohoto protokolu existují dvě verze hlavičky, třetí konstruktor přijímá pole bajtů a je tedy určen pro zpracování přijaté zprávy) je v rámci třídy Message implementována instanční metoda Serialize(), která vrací pole bajtů pro účel odeslání paketu.

Singleton (více v kapitole 5.10 Návrhový vzor Singleton) CommunicationManager drží dva seznamy instancí třídy Message, jeden pro zprávy přijaté, jeden pro zprávy odeslané. Jedinými veřejnými metodami, kterou tato třída vystavuje, jsou CreateAndSendCommand a RequestReport. Metoda SendCommnd vyžaduje čtyři bajty, první dva určují výkon motorů, kde -100 je plný chod vzad, 100 je plný chod vpřed. Třetí i čtvrtý bajt jsou nullovatelné (jde o rozšíření hodnotového typu tak, aby umožnil i hodnotu null, více v kapitole 5.2 Jazyk C#) a užívají se, pokud operátor robotu specifikuje dobu trvání pohybu, nebo vzdálenost, kterou má robot ujet. Na

základě těchto dat metoda vytváří instanci objektu Message a přidá ji na zásobník k odeslání. Metoda RequestReport nepřijímá žádné vstupní parametry, pouze vygeneruje instanci Message s žádostí o oznámení stavu a přidá ji na zásobník k odeslání.

Konstruktor tohoto singletonu jednak vytváří instance zmiňovaných seznamů pro zprávy, jednak registruje event handler (obsluha události, o rozdílu mezi zmiňovaným posluchačem události [event listener] a obsluhou události se můžete dočíst více v kapitole 5.9 Návrhový vzor Observer), reagující na přijetí určitého počtu bytů přes sériovou linku. Tato obsluha má na starost skládat validní pakety podle informací uvedených v hlavičce, tyto následně ukládat na zásobník přijatých zpráv a přes delegát metody informovat vlákno pro správu grafického rozhraní o přijetí nových informací týkajících se stavu robota.

Logika řízení je implementována ve jmenném prostoru Pasak3.Logic. Zde nalezneme i datové struktury pro držení dat, je to třída Map pro držení informace o mapě, třída Route pro popis pohybu robota a třída Config, držící nastavení programu. Všechny tři zmiňovaná úložiště implementují rozhraní System.Runtime.Serialization.ISerializable z knihovny mscorlib.dll, třídy Map a Route mají ve jmenném prostoru Pasak3.Visual svůj protějšek, zodpovědný za grafickou interpretaci těchto dat (jak třída MapPanel.cs, tak třída RoutePanel.cs, dědí ze třídy System.Windows.Forms.Panel a překrývají metodu OnPaint). Namespace Pasak3.Logic dále obsahuje třídu Controll.cs, která je vybavená několika veřejnými statickými metodami:

- Map LoadMap(string path) na základě řetězce udávajícího lokaci načte xml soubor, který deserializuje a vrátí instanci třídy Map.
- void SaveMap(string path, Map map) přijme řetězec, udávající, kam se má soubor uložit a instanci třídy Map, kterou serializuje do xml a uloží na pevný disk.
- Route LoadRoute(string path) na základě řetězce udávajícího lokaci načte xml soubor, který deserializuje a vrátí instanci třídy Route.
- void SaveMap(string path, Route route) přijme řetězec, udávající, kam se má soubor uložit a instanci třídy Route, kterou serializuje do xml a uloží na pevný disk.
- Config LoadConfig() načte a zpracuje xml konfiguračního souboru umístěného ve složce spustitelného souboru, vrátí instanci třídy Config.
- void SaveConfig() uloží instanci konfigurace do složky, ve kterém je umístěn spustitelný soubor ve formátu xml.
- UpdateRoute(Route route, Message message) přijímá instance tříd Route a Message a na základě informací, uvedených ve zprávě přijaté od robota, aktualizuje datovou strukturu držící informace o dráze robota.
- void CalculateRouteAndSendCommands(Map map, Route Route) je nejsložitější metoda této třídy. Vstupními parametry je instance třídy Map, popisující prostředí Pásáka, a instance třídy Route, která v ten moment drží pouze počáteční polohu robota a cíl, kam má robot dojet. Tato metoda implementuje navigační algoritmus A* (více v kapitole 5.11 A* algoritmus) pro nalezení optimální cesty. Následně nalezenou trasu rozloží na elementární

instrukce, které postupně předává singletonu CommunicationManager pro odeslání.

6 Použité technologie

6.1 Jazyk C

C je programovací jazyk vyvinutý na začátku sedmdesátých let dvacátého století v Bellových laboratořích telekomunikační společnosti AT&T. Používaný je především pro psaní systémového softwaru, lze v něm však psát i aplikace. Tento jazyk je nízkoúrovňový (poskytuje pouze malou abstrakci fungování procesoru a paměti) a kompilovaný (napsaný zdrojový kód je potřeba nejprve přeložit překladačem do strojového kódu a až pak lze program spustit). C umožňuje takzvaný inline zápis assembleru do kódu pro tvorbu funkcionalit, které není možné psát přímo v C. Na rozdíl od assembleru je C výrazně čitelnější a přenositelný na procesory jiné architektury použitím jiného kompilátoru. Z C vychází mnoho moderních programovacích jazyků, jako je například C#, PHP nebo Perl.

C definuje čtyři základní datové typy, jsou to: char pro ukládání znaků, int pro ukládání celých čísel, float drží desetinné číslo s plovoucí řádovou čárkou stejně jako double, ten má však dvojnásobnou přesnost. Součástí C je i sada řídících příkazů, je to příkaz if, který větví chod programu na základě vyhodnocení podmínky, příkaz switch, který na základě hodnoty vybírá jednu z několika možných větví běhu programu a cykly for, while a do-while.

6.2 Jazyk C#

C# je vysokoúrovňový kompilovaný objektově orientovaný jazyk vyvinutý společností Microsoft, jehož vývoj začal v roce 2000. Implementace jazyka ubírá programátorovi svobodu výměnou za bezpečnost programu. Programátor se nemusí zabývat pointery, alokací paměti, hlídáním hranic polí a „uklízením“ nepoužívané paměti, kterou spravuje automatický garbage collector.

C# nedovoluje vícenásobnou dědičnost, každá třída tak může být potomkem pouze jedné rodičovské třídy, zato lze ve třídě implementovat libovolný počet rozhraní (interface). Na rozdíl od C nelze deklarovat globální proměnnou, všechny proměnné musí být součástí nějaké třídy.

V rámci C# je používán unifikovaný typový systém CTS (Common Type System). Všechny datové typy, včetně typů primitivních, dědí od třídy Systém.Object a mají tak dostupné všechny základní metody, jako je nepříklad ToString převádějící objekt na textový výstup, nebo Equals rozhodující, zdali jsou objekty shodné. CTS dělí typy do dvou základních skupin, a to na hodnotové a referenční. Hodnotové typy jsou alokovány na zásobníku a lze je dále dělit do tří skupin: primitivní datové typy, struktury a výčtové typy. Referenční typy jsou ukládány na haldu a uchovávají odkaz na místo paměti, kde je instance objektu uložena.

Od verze 2.0 C# podporuje také generické třídy a metody. Generická je taková třída nebo metoda, která odkládá specifikaci konkrétního typu vstupního parametru do doby, než je tato třída daklarována a instanciována v klientském kódu. Při použití generického typového parametru pak lze napsat jednu třídu, jíž lze vytvořit několik různých instance podle vstupního typu. Zde se raději odvolám opět na příklad. Definujme generickou třídu UniverzálníSeznam, která může držet seznam hodnot jednoho typu, a té definujme metodu Přidej takto:

```
public class UniverzalniSeznam<T>
{
    public void Pridej(T vstup) { }
}
```

Tuto třídu je pak možné využít s libovolným typem, aniž bychom se museli bát chyb vznikajících při přetypování, nebo zabalování a rozbalování hodnot:

```
UniverzalniSeznam<int> seznamIntegeru = new UniverzalniSeznam<int>();
UniverzalniSeznam<Message> seznamZprav = new UniverzalniSeznam<Message>();
seznamIntegeru.Pridej(1);
seznamZprav.Pridej(new Message());
```

V kapitole 4.2.2 Jádro a logika byl zmíněn takzvaný nulovatelný typ. V návaznosti na předchozí odstavce jde pouze o generickou třídu Nullable, která rozšiřuje hodnotové datové typy o možnost mít hodnotu null, tedy nebýt definované. Deklarace této generické třídy vypadá následovně:

```
public class Nullable<T> where T : struct
```

Mou speciálně oblíbenou vlastností tohoto jazyka je možnost použít při deklaraci klíčové slovo var. Typ proměnné je pak určen podle pravé strany výrazu při překladu. Toto klíčové slovo zpřehledňuje zápis, a pokud programátor změní návratový typ metody, nemusí přepisovat deklaraci proměnné, která tuto metodu volá.

```
var inPort = new SerialPort("COM3", 2400, Parity.None, 8, StopBits.One);
```

6.3 Komunikační protokol HB-12C

Pro komunikaci mezi zařízeními je na univerzitě standardizován protokol HB-12C, který obsahuje v hlavičce 12 bytů. HB-12C má dvě verze hlavičky, CC a TCP. TCP se skládá ze čtyř bytů nesoucích informaci o velikosti následujících dat, čtyř bytů obsahujících informaci o požadovaném příkazu. Poslední čtyři byty nesou ID paketu. Toto ID slouží jednak ke kontrole, zdali Pásák, potažmo řídící počítač, obdržel všechny informace a případně pak může vygenerovat požadavek na opakování, jednak pro párování požadavků o informace, kde robot na takovýto paket odpovídá paketem se stejným ID. Složitější verze CC obsahuje navíc verzi protokolu, CRC, a adresáta zprávy, velikosti jsou samozřejmě adaptovány tak, aby bylo využito právě 12 bytů.

6.4 Sériové rozhraní UART

Universal Asynchronous Receiver and Transmitter (universální asynchronní přijímač a vysílač) je zařízení pro převod mezi paralelní a sériovou komunikací. Slovo univerzální

v názvu značí, že formát dat a přenosové rychlosti jsou nastavitelné, stejně tak jako napěťové úrovně signálů, které jsou zpracovávané řadičem mimo samotný UART. Obvykle jde o součást integrovaného obvodu používanou pro sériovou komunikaci. Pro zrychlení přenosu lze tyto linky znásobit, v případě dvou linek jde o DUART, osmi OCTART atd.

Při nastavování UART je potřeba zvolit několik parametrů. Jedním z nich je tzv. baudrate, tedy přenosová rychlost. Jde o počet bitů, který se přenese za vteřinu, je však potřeba mít na paměti, že ne všechny bity jsou posílaná data. Relace je zahájena start bitem, načež následuje několik bitů dat a jeden, jeden a půl, nebo dva stop bity. Dále je potřeba oběma komunikujícím stranám sdělit, jestli bude použit paritní bit a v jakém formátu. Pásák komunikuje při nastavení 2400, 8-N-1, tedy při frekvenci 2,4 kHz s osmi datovými bity, bez paritního bitu a s jedním stop bitem. [16]

6.5 Hashovací funkce CRC

Cyclic redundancy check (cyklický redundantní součet) je speciální hashovací funkce, která slouží k detekci náhodných chyb vzniklých při přenosu. [17] Při přenosu dat je výsledek této funkce (takzvaný kontrolní součet) odeslán společně s daty a po jejich přijmutí znovu nezávisle spočítán.

Na začátku výpočtu se vstupní data reprezentují jako polynom, který je následně vydělen předem domluveným polynomem nižšího řádu. Výsledkem, tedy samotným hashem, je polynomiální zbytek po tomto dělení.

Reprezentace dat jako polynomu probíhá následovně: z N bitů se vygeneruje polynom N-1 řádu tak, že všude, kde je bit nenulový je koeficient u příslušného stupně polynomu roven jedné. Opačně pokud se bit na m-tém místě rovná nule, rovná se nule i koeficient násobící polynom x^m. Jeden příklad na tomto místě vydá za dva tisíce slov: bytu 1001 1011 odpovídá polynom sedmého řádu

$$p(x) = 1 * x7 + 0 * x6 + 0 * x5 + 1 * x4 + 1 * x3 + 0 * x2 + 1 * x1 + 1 * x0.$$

Pokud zvolíme vhodný klíč, i malá změna vstupních dat vyústí ve velkou změnu výsledku této hashovací funkce. Pravděpodobnost odhalení chyby pak logaritmicky roste s šířkou klíče podle vzorce [18]

$$P(Oc) = 1 - \frac{1}{2}^{Dk}$$

kde P(Oc) je pravděpodobnost odhalení chyby a Dk je délka zvoleného klíče. Pro šestnáctibitový klíč je pravděpodobnost odhalení chyby při přenosu přibližně 0,999985.

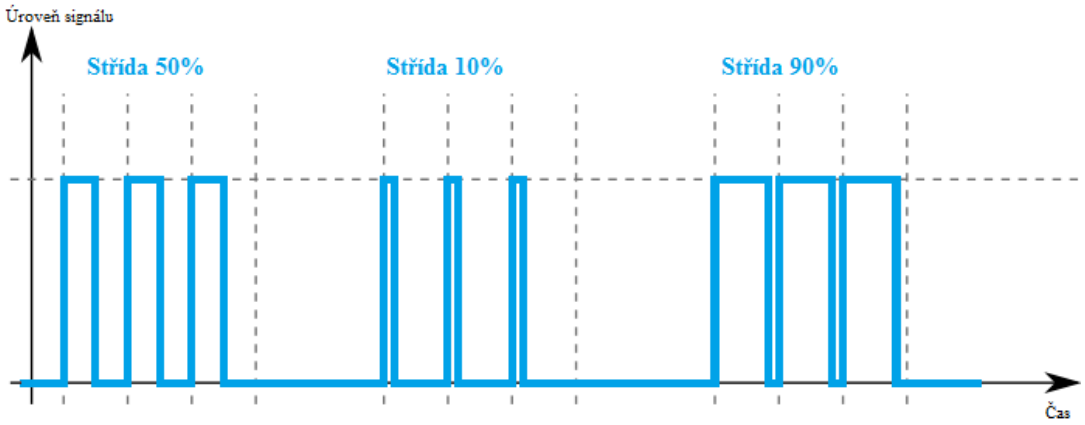
6.6 Regulace výkonu pomocí pulsně-šířkové modulace

Pulse Width Modulation (pulsně šířková modulace) je diskrétní modulace pro přenos analogového signálu pomocí signálu binárního. Obecně může být použito mnoho

veličin, například světelný tok, proud, nebo napětí. Výkon motorů Pásáku je dán poměrem trvání excitovaného stavu k délce jedné periody. Matematicky lze výkon vyjádřit vzorcem:

$$P = \frac{1}{T} \int_0^T f(t)\, dt$$

kde T je perioda a f(t) je funkce průběhu signálu. [19]

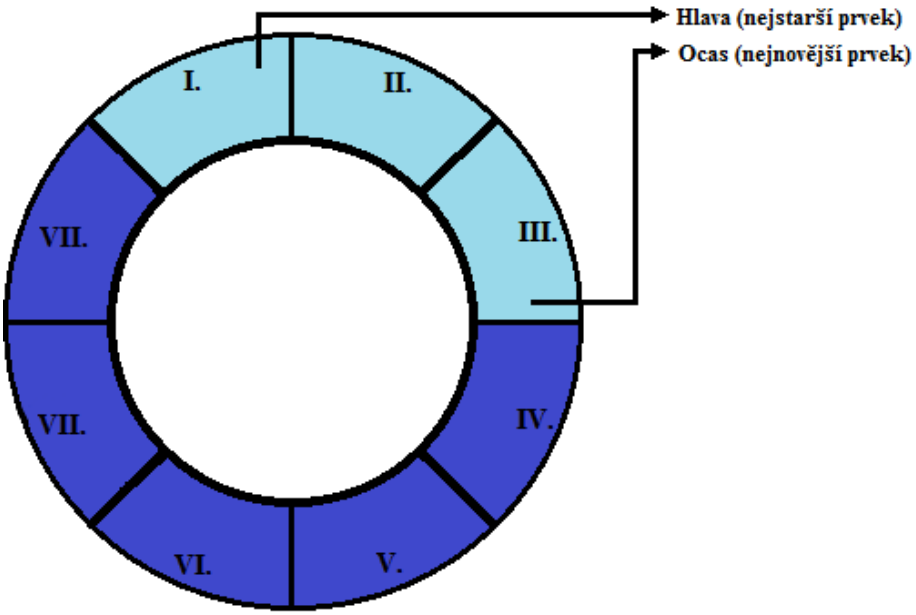


Obrázek 14: Příklad použití PWM.

6.7 Kruhový FIFO Buffer

Kruhový buffer je datová struktura užívající jeden zásobník s pevně danou velikostí, který se chová, jako by byly jeho konce spojeny. Tato struktura užívá dva ukazatele nazvané head a tail (hlava a ocas), kde ocas ukazuje na nejnovější přidaný prvek do zásobníku, hlava naopak na nejstarší nezpracovaný prvek. Při vložení nového prvku se iteruje ocas, při přečtení prvku se iteruje hlava, což je operace v podstatě totožná s nastavením vlajky říkající, že packet je pracován a je ho možné smazat nebo přepsat. Při dosažení velikosti zásobníku se ukazatele nastavují opět na nulu, což vytváří dojem kruhu. V momentě, kdy hlava „předběhne“ ocas, mluvíme o takzvaném přetečení zásobníku, což může vyústit ve ztrátu nebo poškození nezpracovaných dat. [20]

FIFO je zkratka pro First In First Out, tedy první dovnitř první ven. Jde o způsob zpracovávání příchozích dat, kde se nejprve zpracovávají prvně přijatá data. Alternativou je LIFO, tedy Last In First Out (poslední dovnitř první ven), kdy ze zásobníku bereme přednostně data, která byla zapsána později. Tímto způsobem funguje například call stack (zásobník volání) uvnitř procesoru.



Obrázek 15: Vizualizace kruhového bufferu

Implementace kruhového bufferu pro účely Pásáku drží několik instancí struktur odpovídajících protokolu HB-12C. Tato struktura drží hlavičku paketu, pointer na přijatá data a pomocný celočíselní neznaménkový byte, který se používá při skládání paketu v obslužné metodě komunikačního přerušení. Dále můžete vidět definici paketu pro TCP verzi protokolu z hlavičkového souboru CommunicationPacket.h.

```
typedef struct TCPCommunicationPacket{
uint32_t Datasize;
uint32_t Command;
uint32_t PacketID;
uint8_t* AdditionalData;
uint8_t DataPointer;
};
```

6.8 Operační systém FreeRTOS

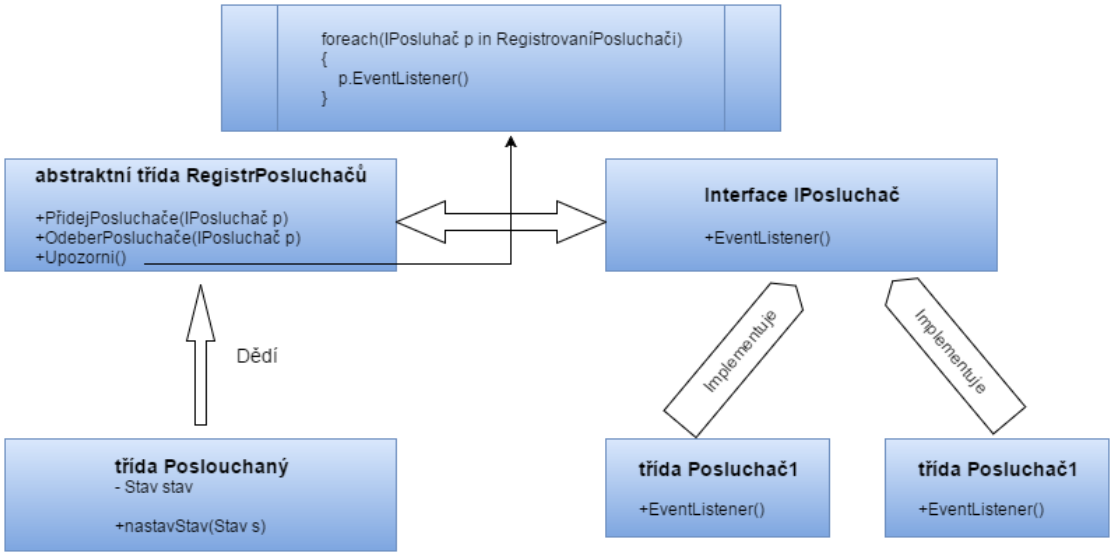
FreeRTOS je open-source jádro operačního systému reálného času distribuovaný pod licencí GNU GPL. Jádro je navrženo tak, aby bylo co nejmenší a nejjednodušší, samotné se skládá z přibližně tří nebo čtyř zdrojových souborů (záleží na konkrétní implementaci) a je napsáno převážně v jazyku C.

FreeRTOS poskytuje metody pro režii souběhu několika vláken (Thread) a úkolů (Task). Dále implementuje algoritmy pro vzájemné vyloučení (Mutex) zabraňující přerušení při vykonávání kritické části kódu, synchronizační semafory a softwarové časovače.

Na rozdíl od komplexních operačních systémů jako Windows nebo Linux FreeRTOS neobsahuje žádné řadiče nebo prostředky pro datovou komunikaci, lze o něm tedy smýšlet spíš jako o knihovně poskytující metody k režii souběhu programových vláken.

6.9 Návrhový vzor Observer

Při návrhu softwaru je jedním z hlavních problémů otázka, jak se vypořádat se závislostmi. Často na nějakém objektu závisí několik dalších objektů, ale z hlediska návrhu není rozumné původní (pozorovaný) objekt zatěžovat seznamem závislých (poslouchajících) objektů, potažmo jejich režií. Tento návrhový vzor původní objekt od sledujících instancí odstiňuje. Implementace na straně pozorovaného většinou probíhá zděděním abstraktní třídy, která definuje metody pro přidání nebo odebrání posluchačů a metodu pro jejich upozornění. Poslouchající se pak registrují u posluchače přidáním delegáta metody, která má zareagovat na změnu stavu. [21]



Obrázek 16: Návrhový vzor Observer

V přechozím textu byl kromě „posluchače události“ (Event Listener) zmíněn i termín „obsluhovač události“ (Event Handler). Stručně řečeno, rozdílem je fakt, že při použití posluchače je implementován vzor popsaný v předchozí události, zatímco Event Handler je pouze jeden a není tedy potřeba vytvářet registr posluchačů reagujících na událost.

6.10 Návrhový vzor Singleton

Singleton lze z Angličtiny přeložit jako unikát. Jedná se o návrhový vzor, používaný v aplikaci v momentě, kdy je potřeba, aby existovala pouze jediná instance třídy a poskytuje k ní globální přístupový bod.[22] V následujícím úryvku kódu je vidět jednoduchá implementace tohoto vzoru:

```
private static CominicationManager instance;
public static CominicationManager Instance
{
    get
    {
        if (instance == null)
            instance = new CominicationManager();
        return instance;
    }
}
```

6.11 A* algoritmus

A* (A star) je počítačový algoritmus využívaný k hledání nejkratších cest v kladně ohodnoceném grafu. Algoritmus vznikl přidáním heuristiky do Djiskrova algoritmu. S tímto vylepšením přišel v roce 1964 Nils Nilsson, jeho optimálnost dokázal v roce 1968 Bertram Raphael za předpokladu, že je použita monotónní heuristika.
A* využívá hladový princip k nalezení cesty z počátečního do koncového uzlu. Během průběhu je každý vrchol grafu ohodnocen funkcí určující pořadí, ve kterém se mají uzly procházet. Tuto funkce lze rozložit na dvě části:

$$f(x) = g(x) + h(x)$$

kde g(x) je funkce udávající vzdálenost mezi počátečním a hodnoceným uzlem. Lze říci, že je to suma dílčích vzdáleností všech uzlů mezi počátkem a právě hodnoceným uzlem. Funkce h(x) pak představuje heuristickou funkci, tedy odhad správnosti postupu. Pro aplikaci v navigačním algoritmu se zpravidla užívá euklidovská vzdálenost, tedy délka „vzdušné čáry“ mezi hodnoceným bodem a bodem cílovým. Aby byl algoritmus konečný, musí navštívit každý uzel maximálně jednou, tato podmínka je ekvivalentní s podmínkou monotónnosti heuristické funkce. Monotónní heuristická funkce je taková, která splňuje nerovnici

$$h(x_n) \leq d(x_n, x_{n+1}) + h(x_{n+1})$$

slovně vyjádřeno musí být hodnota heuristické funkce n-tého vrcholu menší, než součet vzdálenosti n-tého a následujícího vrcholu s výsledkem heuristické funkce následujícího vrcholu. Pokud by tomu tak nebylo, algoritmus by se mohl z některých bodů vracet zpět, což by mohlo vyústit v neoptimální výsledek (trasu, která by nebyla nejkratší), nebo takzvanému zacyklení, kdy by algoritmus nebyl schopný požadovanou úlohu vyřešit.
A* pak probíhá v několika jednoduchých krocích:

- Zařaď startovní uzel do množiny otevřených uzlů
- Dokud není množina otevřených uzlů prázdná:
    - Vyber z množiny otevřených uzlů uzel s nejnižší hodnotou f(x)
    - Jestli je vybraný uzel cíl
        - Konec
    - Přidej vybraný uzel do množiny uzavřených uzlů
    - Pro každý sousední uzel vybraného uzlu
        - Jestli je sousední uzel v množině uzavřených uzlů
            - Přeskoč jej
        - Spočti f(x) sousedního uzlu
        - Jestli není sousední uzel v množině otevřených uzlů
            - Přidej sousední uzel do množiny otevřených uzlů
        - Jestli je sousední uzel v množině otevřených uzlů

        - Aktualizuj jeho f(x) nově spočtenou hodnotou
- Konec

Pokud je v rámci uzlů implementován spojový seznam ukazující, z jakého uzlu jsme do následujícího uzlu dostali, můžeme po konci algoritmu rekonstruovat cestu tak, že se rekurzivně díváme na předchozí uzly, vycházejíc z cíle. Pokud algoritmus skončí, aniž bychom našli cílový uzel, můžeme předpokládat, že požadovaná cesta z počátku do cíle v rámci grafu neexistuje.

7 Závěr

7.1 Shrnutí

Prvním z úkolů práce bylo seznámit se s platformou LPCXpresso a deskou LPCXpresso LPC1769, této vývojové platformě je věnována celá kapitola. Ze svých zkušeností bych se odvážil říct, že dostupná zařízení jsou dostatečně výkonná pro aplikace, jako je řízení robota a přitom stále zůstává mnoho prostoru pro rozšiřování jeho programových funkcionalit.

Na začátku implementačních prací bylo potřeba vytvořit mapovaní pinů z patice PICAXE na piny nové řídící jednotky, vytvořit příslušný tištěný spoj a pomocí distančních sloupků vytvořit na Pásáku nové „patro."

Nový řídící systém robotu Pásák zachovává rozdělení na nízkoúrovňové a vysokoúrovňové řízení. Nízkoúrovňové řízení je zajištěno mikroprocesorem robotu a obstarává obsluhu komunikace ve smyslu přijímání příkazů a odesílání hlášení o stavu, dále se stará o motorický i senzorický subsystém. Vysokoúrovňové řízení je pak implementováno pro řídící počítač a zahrnuje grafickou klientskou aplikaci, která zobrazuje data ze senzorů robota a jeho ujetou trasu. Součástí vysokoúrovňové části systému je i přijetí uživatelských vstupních dat a jejich převod na hodnoty zpracovatelné robotem. Počítačový program také implementuje možnost autonomního řízení, která umožňuje uživateli zadat pouze cílový bod, do kterého se musí robot sám dostat. Způsob dosažení tohoto bodu je pak určen samotným jádrem programu, vykonávání této trasy pak probíhá zcela nezávisle na uživateli.

Komunikace mezi jednotlivými částmi řídícího systému probíhá bezdrátově přes standard bluetooth, kde obě starny implementují komunikační protokol HB-12C. S bluetooth moduly, jak na straně robota, tak na straně PC, je komunikováno po standardní sériové lince. Nově navržený systém umožňuje moderování komunikace, kontrolu konzistence přenášených dat a potvrzení o přijetí a zpracování řídícího paketu.

Oproti původnímu řešení současný řídící systém umožňuje preciznější ovládání robota, a to zprovozněním pulsně-šířkové modulace regulující výkon jednotlivých motorů. Robot je tak schopný vytočit zatáčku o libovolném poloměru. Dále byl implementován standardizovaný protokol HB-12C a řídící systém je tedy mimo jiné schopný hlídat chyby přenosu, nebo fungovat například v rámci multiagentního systému. Nová aplikace pro PC pak rozšířila a zpřehlednila zobrazované informace, zpřístupnila grafické rozhraní pro ovládání nových funkcionalit robota a byla navržena tak, aby bylo možné tuto grafickou nástavbu vyměnit a robota tak ovládat třeba přes webové rozhraní nebo z mobilního telefonu s minimálními zásahy do jádra programu.

7.2 Postřehy

Původně jsem k robotu přicházel s představou, že pro něj budu vytvářet spíše vyšší logiku typu autonomního pohybu nebo zpracování obrazu. Ta se však rozplynula v momentě, když bylo rozhodnuto, že starší řídící jednotka bude vyměněna za novou, tu bude třeba naprogramovat od nuly a já vyměním jazyk třetího tisíciletí C# a vrátím se o čtyřicet let zpět k C. Nakonec to byla pravda jen napůl – C# jsem použil při vytvoření aplikace pro ovládání a tvorbu vyšší logiky jsem si užil při implementaci navigačního algoritmu, navíc jsem si procvičil dlouho nepoužívané zkušenosti s tvorbou frontendu (grafického rozhraní).

Prvotní zkoumání ukázkových kódů dodávaných k platformě LPCXpresso mě poněkud vyděsilo, osobně jde o mou první zkušenost s psaním ovladače pro mikrokontrolér v jazyce C a nulová vrstva abstrakce jazyka od funkčnosti procesoru mě nutila neustále nahlížet do dokumentace a zjišťovat, který registr je odpovědný za kterou funkcionalitu. V porovnání s platformou PICAXE, která byla na robotu použita původně, pro mě bylo výrazně obtížnější těmto kódům porozumět. PICAXE lze totiž na rozdíl od LPCXpresso programovat ve speciálním jazyce založeném na jazyku BASIC a programátorovi je poskytnuta vrstva abstrakce procesoru, která mu umožní nevnímat jednotlivé hodnoty registrů a soustředit se na vytváření samotné funkcionality. V souvislosti s faktem, že samotný jazyk C nikdy nebyl mou silnou doménou, můžu říct, že jsem si značně rozšířil obzory ve směru nízkoúrovňového programování a samotný jazyk jsem si hodně osvěžil a procvičil. Musím se přiznat, že jsem se několikrát vztekal při hledání chyby, jen abych zjistil, že jsem opět prohodil ampersand za hvězdičku a místo hodnoty jsem dostal ukazatel, případně vice versa.

Ačkoliv C poskytuje programátorovi, tedy kromě assembleru a strojového kódu, nejširší možnosti, osobně bych při potřebě vytvořit obdobný řídící systém vyzkoušel platformu, umožňující programování v nějakém vyšším jazyce. Na výběr je například Microsoft .NET Micro Framework, nebo zmenšené verze Pythonu (PyMite, TinyPython).

8 Literatura a použité zdroje

[1] KOŠNAR, Karel. Mobilní robotika, Pandatron.cz Elektrotecnický magazín [online]. Praha: České Vysoké Učení Technické v Praze, 2010 [cit. 2016]. Dostupné z WWW: http://pandatron.cz/?1745&mobilni_robotika

[2] NOVÁK, Petr. Mobilní roboty: pohony, senzory, řízení, 1. vyd. Praha: BEN - technická literatura, 2004, 248 s. ISBN 80-730-0141-1.

[3] SIEGWART, Roland. Introduction to autonomous mobile robots, 1. vyd. Massachusetts: MIT Press, 2004, 321 s. ISBN 02-621-9502-X. Dostupné z: http://www.robotee.com/EBooks/Introduction_to_Autonomous_Mobile_Robots.pdf

[4] The Official Bluetooth® Technology Web Site [online]. 2011 [cit. 2016]. The Official Bluetooth® Technology Web Site. Dostupné z WWW: http://www.bluetooth.com/Pages/Bluetooth-Home.aspx

[5] Sri.com [online]. 2011 [cit. 2016]. Shake0004. Dostupné z WWW: https://www.sri.com/sites/default/files/uploads/publications/pdf/629.pdf

[6] Humanoid.waseda.ac.jp [online]. 2008 [cit. 2016]. WABOT-1-1973. Dostupné z WWW: http://www.humanoid.waseda.ac.jp/booklet/kato_2.html

[7] www-robotics.jpl.nasa.gov [online]. 2009 [cit. 2016]. The ATHLETE Rover. Dostupné z WWW: https://www-robotics.jpl.nasa.gov/systems/system.cfm?System=11

[8] www.picaxe.com [online]. 2010 [cit. 2016]. PICAXE Manual. Dostupné z WWW: http://www.picaxe.com/docs/picaxe_manual1.pdf

[9] www.ncp.com [online]. 2015 [cit. 2016]. LPC1769/68/67/66/65/64/63. Dostupné z WWW: http://www.nxp.com/documents/data_sheet/LPC1769_68_67_66_65_64_63.pdf

[10] Spezial.cz [online]. 2010 [cit. 2016]. Bluetooth RS232 OEM Serial Port Adapter moduly. Dostupné z WWW: http://www.spezial.cz/connectblue/bluetooth-seriove-rs232-oem-moduly.html

[11] Lpcware.com [online]. 2015 [cit. 2016]. LPCXpresso Patform. Dostupné z WWW: https://www.lpcware.com/lpcxpresso

[12] Embeddedartists.com [online]. 2013 [cit. 2016]. LPCXpresso LPC1769 Board. Dostupné z WWW: https://www.embeddedartists.com/products/lpcxpresso/lpc1769_xpr.php

[13] Arm.com [online]. 2004 [cit. 2016]. Cortex-M3 Processor. Dostupné z WWW: http://www.arm.com/products/processors/cortex-m/cortex-m3.php

[14] Freertos.org [online]. FreeRTOS. Dostupné z WWW: http://www.freertos.org/

[15] Msdn.microsoft.com. Overview of the .NET Framework. Dostupné z WWW: https://msdn.microsoft.com/en-us/library/zw4w595w(v=vs.110).aspx

[16] Societyofrobots.com. Microcontroller UART Tutorial. Dostupné z WWW: http://www.societyofrobots.com/microcontroller_uart.shtml

[17] Accuhash.com. What is CRC32? Dostupné z WWW: http://www.accuhash.com/what-is-crc32.html

[18] Barrgroup.com. CRC Series, Part 2: CRC Mathematics and Theory. Dostupné z WWW: http://www.barrgroup.com/Embedded-Systems/How-To/CRC-Math-Theory

[19] J. Huang, K. Padmanabhan, and O. M. Collins, "The sampling theorem with constant amplitude variable width pulses", IEEE transactions on Circuits and Systems, vyd. 58, červen 2011.

[20] Stackowerflow.com. How to implement a circular buffer in C? Dostupné z WWW: http://stackoverflow.com/questions/827691/how-do-you-implement-a-circular-buffer-in-c

[21] Tutorialspoint.com. Design Patterns – Observer Pattern. Dostupné z WWW: http://www.tutorialspoint.com/design_pattern/observer_pattern.htm

[22] Tutorialspoint.com. Design Patterns – Singleton Pattern. Dostupné z WWW: http://www.tutorialspoint.com/design_pattern/singleton_pattern.htm

[23] Snailshop.cz. TANK-02. Dostupné z WWW: http://www.snailshop.cz/pasove/313-tank-02.html

[24] Shop.freertos.org. FreeRTOS NXP LPCXpresso LPC1769 Education Kit. Dostupné z WWW: http://shop.freertos.org/FreeRTOS_NXP_LPC1769_LPCXpresso_Evaluation_Kit_p/lpc1769_education_kit.htm

[25] embeddedinsights.com. ARM Cortex-M3 Dostupné z WWW: http://www.embeddedinsights.com/epd/Diagrams/arm-cortex-m3.jpg

[26] ŠTĚPÁNEK, Jakub. MOBILNÍ ROBOT s mikrokontrolérem PICAXE ROČNÍKOVÝ PROJEKT, 2011

9 Přílohy

- 1 CD ROM obsahující elektronickou verzi práce v PDF, zdrojové kódy pro PC i pro robota a sadu obrázků použitých v této práci.