Řídicí systém mobilního dvoustopého robota

Bc. Ondřej Petr

Diplomová práce
2006

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

nascannované zadání s. 1

nascannované zadání s. 2

ABSTRAKT

Cílem diplomové práce je návrh a konstrukce mobilního robota, sestaveného ze čtyř servomotorů s časově ovládaným inkrementálním řízením, jeho oživení a vývoj podpůrných algoritmů, které umožňují vedení sestavy mobilního robota po zadané trajektorii. V první části se pojednává o vybraných variantách mobilních robotických systémů, o současném stavu ve vývoji robotických systémů a o některých typech používaných pohonů. Dále jsou popsány vybrané normy pro drátovou a bezdrátovou komunikaci, např. RS232, Bluetooth a WiFi. Druhá část pojednává o použitých mechatronických komponentách, použitých v diplomové práci. Dále je popsán navržený algoritmus pro řízení trajektorie mobilního robota a jeho komponent v C++. V návaznosti s tím je popsána funkční činnost použitých servomechanizmů včetně funkčních modulů pro ovládání servomotorů.

Klíčová slova: mobilní robot, servomotor, časově ovládané inkrementální řízení, komunikační standardy

ABSTRACT

The aim of master thesis is the proposal and construction of the mobile robotic set, consisting of four servo motors with time instant incremental control, its activating and the development of supporting algorithms for the tracing of mobile robotic system on defined trajectory. The first part deals with various types of mobile robotic systems, advancements in the development of robotic systems in the world and with some types of actuators used in robotics. Further there are described selected standards for both wire and wireless communication. The second part deals with the modern state-of-the-art components used in master thesis. Master thesis continues by the description of developed algorithms for the control of the mobile robot and its components in C++ language. There is described functional activity of servo-systems including supporting modules for the time instant incremental control.

Keywords: mobile robot, servo, time instant incremental control, communication standards.

Poděkování, motto:

Tímto vyjadřuji poděkování vedoucímu diplomové práce Mgr. Ing. Milanu Kvasnicovi, CSc za svědomité vedení diplomové práce.

Drahá slečno Gloryová, Roboti nejsou lidé. Jsou mechanicky dokonalejší než my, mají úžasnou rozumovou inteligenci, ale nemají duši.

K. Čapek, R. U. R. (předehra)

OBSAH

ÚVOD

Cílem této diplomové práce je návrh, sestavení a oživení mechanických a elektronických částí, vývoj a odladění algoritmů pro řízení mobilního robotického systému, který umožňuje pohyb po zadané trajektorii.

Systém je tvořen servomotory AI-MOTOR 1001 od firmy Megarobotics co. Ltd., Korea. Servomotory AI-MOTOR 1001 vzhledem ke své stavbě a spojovacím dílům umožňují plánování konfigurace stavů koncového efektoru pro více stupňů volnosti. K ovládání robota je možno použít sériové rozhraní RS232, po kterém jsou přenášena data, řídící a nastavovací instrukce servomotorů. Systém je dále rozšířen o bezdrátové komunikační moduly, sestávající z radiových modemů, připojitelných na sériový port osobního počítače a radiového modulu firmy Hőft & Wessel. Radiové modemy a komunikační modul pracují v nelicencovaném pásmu 1880-1900MHz. Robotickou sestavu je možno napájet jak ze sítě, tak za použití šesti nabíjecích baterií o napětí 1,5V. V této práci je popsána funkční činnost všech navržených modulů a mechanizmů. Jako programové prostředí byl zvolen programovací jazyk Borland C++ Builder.

Části této diplomové práce jsou sestaveny tak, aby mohly být využity jako instruktážní manuál pro laboratorní měření z předmětů Základy robotiky a Elektronické zabezpečovací systémy II na Fakultě aplikované informatiky UTB ve Zlíně.

I. TEORETICKÁ ČÁST

1 MOBILNÍ ROBOTY

Výrazný rozvoj elektroniky a výpočtové techniky v posledních dvou desetiletích významně ovlivnil rozvoj robotiky, která je plně závislá na možnosti zpracování údajů ze snímačů v reálném čase. Robot chápeme jako počítačem řízený integrovaný systém, který je schopen autonomní cílově orientované interakce s reálným prostředím v souladu s instrukcemi operátora, které mu umožňují

- vnímat a rozeznávat prostředí
- vytvářet si a průběžně aktualizovat vnitřní model prostředí
- na základě tohoto modelu v souladu se zadanou úlohou rozhodovat o své činnosti
- ovlivňovat prostředí a manipulovat s předměty
- komunikovat s operátorem přirozeným nebo umělým jazykem

Roboty s těmito vlastnostmi se nazývají *kognitivní roboty*, v současnosti častěji *adaptivní roboty* zejména pro odlišení od tzv. *programových robotů*, které pracují podle předem zadaného programu, a to jen za pomoci jednoduché zpětné vazby, nezbytné pro jeho provedení.

Systém, který zabezpečuje vlastní pohyb robota a působí na okolní prostředí robota se nazývá motorický systém a systém, který přijímá informace o prostředí se nazývá senzorický systém. Údaje o pohybu robota, informace o prostředí a rozhodování o posloupnosti a výkonu jednotlivých operací vyhodnocuje a zpracovává řídicí (nebo též kognitivní) systém robota. K zabezpečení správné činnosti robota je potřebné, aby měl pod kontrolou nejen vlastní pohyb, ale aby na základě informace o okolním prostředím koordinoval vykonávání jednotlivých operací. Podle prostorového vztahu k podnětům začleňujeme snímače požívané v senzorickém systému robota na snímače vnitřní a vnější zpětné vazby. Snímače vnitřní zpětné vazby robota se používají na sledování stavu vlastního manipulačního systému, např. na určení polohy polohovacího servomechanizmu. Snímače vnější zpětné vazby se používají na lokalizaci předmětů v okolí, zjištění jejich tvaru a velikosti, navigaci, orientaci a komunikaci s operátorem. Tyto informace, zpracované v řídicím systému robota, umožňují vytvořit vnitřní model okolního prostředí, což v závislosti na zadané úloze rozhoduje o jeho změně polohy, pří-

padně koncového efektoru. Tyto snímače, umožňující na základě senzorické zpětné vazby polohovou adaptivitu mají velký vliv na zvyšování efektivnosti a to s využitím jak pro řízení svářecích, tak i montážních robotů. Mohou být založeny na různých fyzikálních principech, ale z hlediska požadované přesnosti, spolehlivosti, životnosti a rychlosti zpracovávané informace bylo přistoupeno k vývoji optoelektronických snímačů.

1.1 Rozdělení mobilních robotů podle podvozku

Mobilita je specifickou vlastností, která se může vyskytovat u robotů s podvozkem nebo jiným systémem, který umožní pohyb robota.

Trendy současného světového vývoje v oblasti mobilních a servisních robotů ukazují na značně široké spektrum možností jejich uplatnění. Narozdíl od průmyslových robotů nacházejí využití i v netradičních oblastech. Je to zřejmé už z toho, že tyto roboty jsou definovány jako technická zařízení, která se podílejí na nevýrobních činnostech. V poslední době se uplatňují zejména v oblasti služeb a asistenčních technologií. VZP, SBS.

Základním faktorem, který bude mít vliv na konstrukční parametry a chování servisního robotu, je prostředí, ve kterém se bude pohybovat. Zde lze uvést dva základní typy prostředí: venkovní (*outdoor*) a vnitřní (*indoor*).

Pro venkovní prostředí je charakteristické, že se robot bude pohybovat mimo budovy v členitém terénu, jako jsou pole, lesy, skály či krátery, nebo může jít také o městský terén apod. Ve většině případů je to terén, ve kterém se člověk pohybuje velmi obtížně. Tuto oblast je možné rozšířit o pohyb robotu v zamořeném prostředí, podmořský výzkum, průzkum planet apod. Pro vnitřní prostředí je charakteristické, že se robot bude pohybovat pouze v budovách, rozsáhlých nádržích, vzduchovodech, potrubích a podobných prostorách.

Mobilní roboty jsou obecně určeny pro vykonávání úloh v částečně známých nebo zcela neznámých prostředích s měnícími se scénami. V mnohých případech aplikací se již dnes požaduje autonomní chování takového robotu. Autonomní činnost robotu je podmíněna využitím propracovaných rozhodovacích procesů a supervizorového nebo adaptivního řízení. Autonomní roboty jsou technická zařízení schopná plánovat, interpretovat a provádět zadanou úlohu na základě činnosti řídicího systému a informací získaných ze senzorického systému vnější zpětné vazby.

Činnost mobilního robota ovlivňuje zejména polohovací a orientační ústrojí, efektor a také podvozkovou část. Orientační ústrojí a efektor jsou přizpůsobeny pro požadovaný úkon - na rozdíl od robotů průmyslových se často jedná o práci s křehkými předměty (sklenice, jemná zařízení, manipulaci s pacienty, ...) - konstrukční řešení efektorů jsou proto většinou osazena větším počtem senzorů nežli je tomu u průmyslových robotů - jedná se především o taktilní snímače zjišťující uchopovací sílu. Konstrukce efektoru se také často přibližuje podobě lidské ruky s několika (většinou 2-3) prsty a jedním palcem. Činnost servisního robotu také značně ovlivňuje podvozková část. Jedná li se o činnost s potřebou velké manévrovatelnosti, je volen podvozek s více stupni volnosti nebo podvozek se všesměrovými koly, je-li vyžadován rovnoměrný chod robotu (při transportu pacientů, vzorků, nebezpečných předmětů) volí se podvozek s více koly, popř. pásový.

Z hlediska konstrukce podvozků můžeme mobilní roboty rozdělit do několika základních kategorií :

Obrázek 1-Rozdělení mobilních robotů podle druhu podvozku

1.1.1 Lokomoční ústrojí

Lokomočním ústrojím se rozumí technický prostředek (podvozek) umožňující pohyb. Konstrukční řešení podvozků se odvozuje od různých principů a řešení jednotlivých detailů v závislosti na aplikaci robotu a vlastností prostředí, do něhož je určen.

Mobilní asistenční roboty mohou být dráhového nebo terminálového typu, tj. s lokálním nebo globálním řízením v reálném čase. Podle toho je nutné připravit matematický model pro dané prostředí, ve kterém se bude robot pohybovat, a pro daný čas činnosti. Od moderních mobilních robotů se vyžadují schopnosti jako vyhýbání se náhodným překážkám, přemístění se do cílového bodu s požadovanou přesností, kopírování terénu určité třídy, počítačová analýza daného prostředí včetně cílového bodu apod. Ke splnění těchto požadavků je zapotřebí, aby podvozek robotu měl dobré manévrovací schopnosti (otáčení na místě okolo svislé osy, zatáčení v ostrém úhlu apod.). Pro různé oblasti jsou charakteristické různé typy podvozku robotu.

Co se týče nestrojírenských oborů, lze uvést např. zdravotnictví, stavebnictví, vojenství, zemědělství, lesnictví apod. Specifickou oblastí z hlediska požadavku na konstrukci a parametry mobilních robotů je průzkum povrchu planet (kosmický prostor) a průzkum

pod vodní hladinou. Každá z těchto oblastí má svá specifika a uplatnění v nich mohou najít mobilní servisní roboty s různým lokomočním ústrojím.

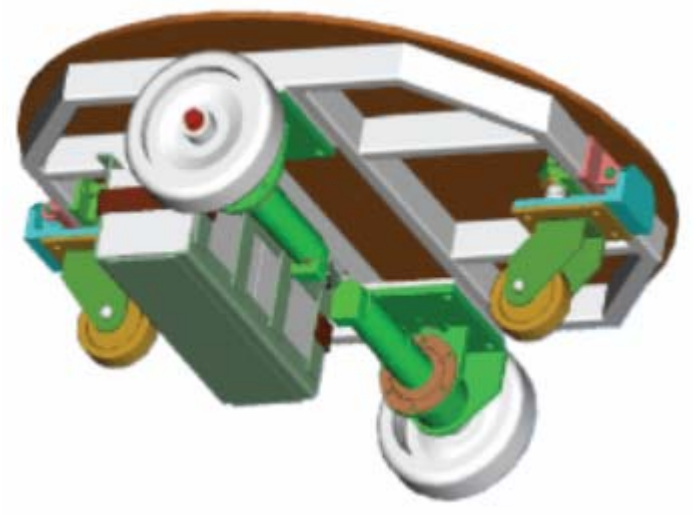
1.1.2 Mobilní roboty na kolovém podvozku

Roboty na kolovém podvozku se vyznačují dobrou schopností manévrovat a překonávat různé druhy překážek. Kolové podvozky jsou nejčastěji tříkolové, čtyřkolové nebo šestikolové. Dvoukolové a více než šestikolové servisní roboty se využívají zřídka, spíše pro výzkumné účely.

1.1.2.1 Tříkolové a čtyřkolové podvozky

Tříkolové a čtyřkolové mobilní roboty lze rozdělit na diferenčně řízené roboty, roboty s více stupni volnosti, roboty řízené Ackermanovým způsobem a synchronně řízené roboty.

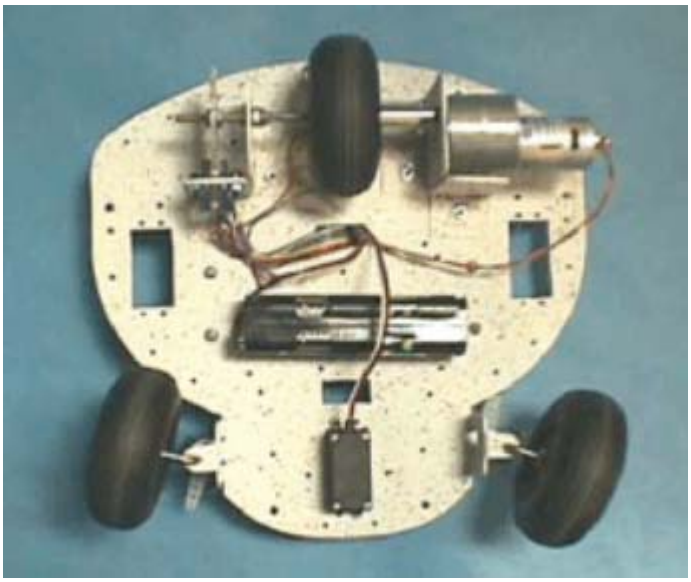
Diferenčně řízené roboty se vyznačují tím, že mají nezávisle poháněná dvě kola a vpředu (popř. vzadu) volně otočné nepoháněné směrové kolo (popř. kola).



Obrázek 2-Diferenčně řízený robot

Roboty s více stupni volnosti mají uprostřed dvě řízená a poháněná kola a vpředu a vzadu po dvou pomocných kolech. Roboty tohoto typu mohou vykonávat i atypické pohyby.

Tříkolový robot řízený Ackermanovým způsobem má jedno poháněné kolo a dvě kola řiditelná nebo dvě poháněná a jedno řiditelné. Pro čtyřkolový robot řízený tímto způsobem platí, že má dvě poháněná kola, která musí být vybavena mechanickým nebo elektrickým diferenciálem, a vpředu (popř. vzadu) otočná nepoháněná kola(např. jako u auta).



Obrázek 3-Ackermanův způsob řízení

Synchronně řízené roboty jsou opatřeny třemi nebo čtyřmi koly, přičemž všechna jsou poháněna a řízena tak, že mají navzájem stále stejné natočení a rychlost. Pro natáčení kol se používá společný hnací řetěz a jeden společný řetěz k pohonu kol. Mobilní roboty s těmito typy podvozků nacházejí uplatnění ve venkovním prostředí.

1.1.2.2 Šestikolové podvozky

Šestikolové podvozky se pro vnitřní prostředí v podstatě nepoužívají. Technickou a technologickou převahu v této oblasti zcela jednoznačně má americká společnost pro výzkum vesmíru NASA. Jako příklad univerzity, která se zabývá výzkumem a vývojem šestikolových mobilních robotů, lze uvést francouzskou Laboratory for Analysis and Archi-

tecture of Systems (LAAS), která zkonstruovala několik těchto robotů. Příkladem může být mobilní robot Lama. V současnosti jsou mobilní roboty na šestikolovém podvozku využívány také v jaderném nebo chemickém průmyslu a v neposlední řadě i v těžebním průmyslu.

Obrázek 4-Příklad šestikolového robota

Šestikolové mobilní roboty lze zařadit do zvláštní skupiny, protože jsou stejně jako pásové podvozky řízeny smykem, popř. mají řízena přední dvě a zadní dvě kola. Tyto konstrukce podvozků se používají převážně pro venkovní prostředí, protože dokážou zdolávat členitější terén a nerovnosti. Jelikož jsou sestrojovány pro venkovní prostředí, vyžadují i odpovídající pohony. Roboty se šestikolovým podvozkem se úspěšně využívají také k průzkumu povrchu planet.

1.1.2.3 Speciální kolové podvozky pro mobilní asistenční roboty

Do skupiny kolových podvozků pro asistenční roboty se zahrnují také podvozky, které využívají speciální kola, jako jsou všesměrová kola, Weinsteinova kola a článkové pojezdy, kola MaxWheel, šnekové podvozky nebo kola robotů zdolávajících svislé překážky.

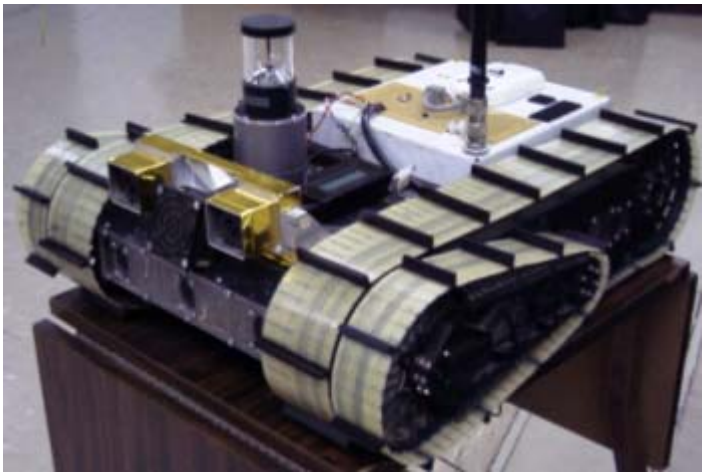
Nyní se ve stručnosti zmíním ještě o několika dalších konstrukčních řešení robotů, využívajících ke svému pohybu jiné prostředky než kola.

1.1.3 Pásové roboty

Pásové roboty nacházejí uplatnění v celé řadě konstrukcí mobilních robotů - slouží pro speciální aplikace a využívají tzv. řízení smykem.
Bývají využívány jako podvozky různých tančíků, buldozerů, vojenských nosičů a jiných konstrukcí - převážně pro použití v outdoor prostředí. V prostředí indoor se uplatňují pro jízdu do a ze schodů a pro aplikace s potřebou vysoké stability. Smykový způsob řízení je pro autonomní roboty značně nepřesný a proto se pásové roboty používají především ve spojení s teleoperátorem.

Oblasti aplikace pásových robotů:

- jaderný průmysl
- stavebnictví
- vojenské a policejní
- zemědělství a lesnictví
- likvidace požárů
- práce s nebezpečným odpadem
- průzkum neznámého terénu
- asistenční technologie a rehabilitační robotika.



Obrázek 5-Pásový robot

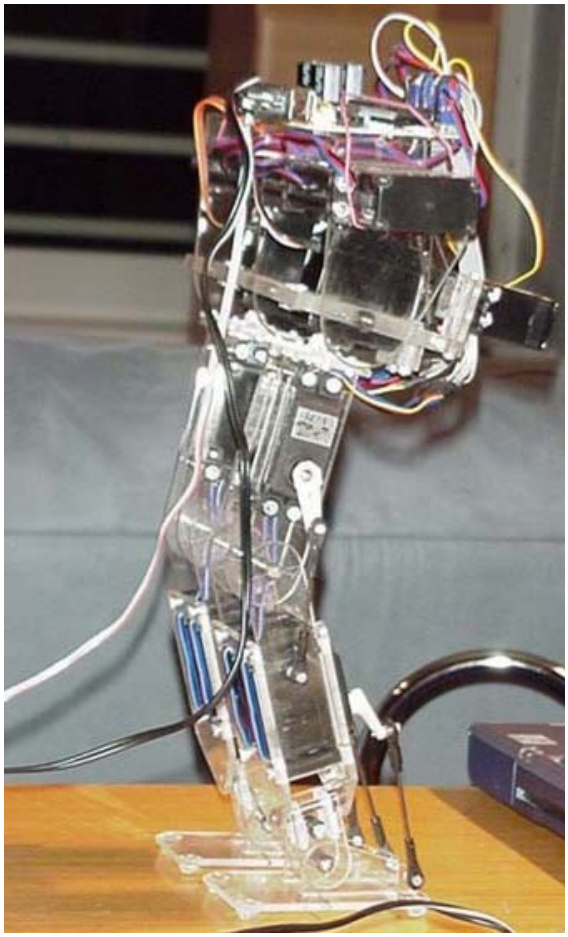
1.1.4 Kráčející roboty

Kráčející roboty lze dělit dle několika hledisek:

- dle počtu nohou - 2, 4, 5, 6, 8 a vícenohé roboty
- dle provedení nohy - typ savec, typ hmyz
- dle kinematických vazeb - rotační a translační.

Nejčastější aplikace kráčejících robotů jsou provedeny na dvou jako humanoid a na čtyřech, či šesti nohou jako kráčející plošiny. Dvounohé roboty se takřka ve všech případech snaží napodobit člověka. Humanoidní roboty jsou schopné překonávat schodiště, relativně vysoké překážky (oproti kolovým a pásovým robotům). Současné špičkové roboty jsou již osazeny dokonalým hardwarovým a softwarovým vybavením, které robotům umožňují plynulou hladkou chůzi, širokou manévrovatelnost a vynikající stabilizační schopnosti. Šestinohé roboty jsou kompromisem mezi 4nohými (levnější HW, jednodušší SW) a osminohými s plynulým hladkým pohybem. Šestinohé roboty (a kráčející všeobecně) jsou používané pro překonávání vysoce členitého terénu - kameny, štěrk, skaliska - tam kde kolový nebo pásový podvozek vůbec nelze použít. Pro zvolení typu nohy kráčejícího robotu je třeba správně odhadnout činnost robotu a prostor ve kterém se robot bude pohybovat. Pro pohyb ve velice složitém terénu je vhodné aby se noha dostávala do kontaktu s

podložím svislým a ne rotačním pohybem - přesnější ustavení nohy, stabilnější, jednodušší aplikace snímačů.



Obrázek 6-Kráčející robot

1.1.5 Plazivé mobilní robotické systémy

Plazivé mobilní roboty se vyznačují specifickou štíhlou podlouhlou konstrukcí. Pro jejich pohyb se vychází z modelů pohybů živočichů a používají se dva typy:

- had

- žížala

Obrázek 7-Plazivý robot

Dle stylu pohybu, předpokládané aplikace a terénu pohybu je volena koncepce robotů, počet článků a typ pohonů. Počet článků se pohybuje v širokém intervalu. Od několika jednotek do několika desítek kusů. Roboty se používají převážně k inspekci a servisní činnosti v potrubích a úzkých prostorách - plazivé roboty pro tuto činnost se vyznačují vysokým počtem článků. Další aplikací je pohyb (průzkum, inspekce,...) ve velice členitém prostoru (kameny, skaliska), kde ani pásové ani kráčející roboty nemají naději na úspěch - roboty se vyznačují nižším počtem členů s tuhou konstrukcí a jsou schopny překonávat překážky vyšší než polovina délky robotu.

Nyní ještě několik vysoce specializovaných typů.

1.1.6 Speciální mobilní robotické systémy

1.1.6.1 Skákající roboty

Skákající roboty slouží pro výzkum kinematických a dynamických poměrů jako výchozí bod pro konstrukci složitějších kráčejících robotů (zejména výzkum stability a udržení přímeho postoje). Tyto roboty bývají většinou osazeny jednou nohou poháněnou pneumatickým válcem. Princip pohybu je takový, že robot musí vyskočit dostatečně vysoko aby se noha mohla překmitnout do nové pozice a robot se posunul o krok (skok) vpřed. Roboty jsou konstrukčně řešeny s hmotným tělem a lehkou kyvnou nohou. Tato koncepce zjednodušuje udržení stability. Příčná stabilita robotu bývá zajištěna vodítkem a robot tak při svém pohybu opisuje kružnici.

1.1.6.2 Kráčející mikroroboty

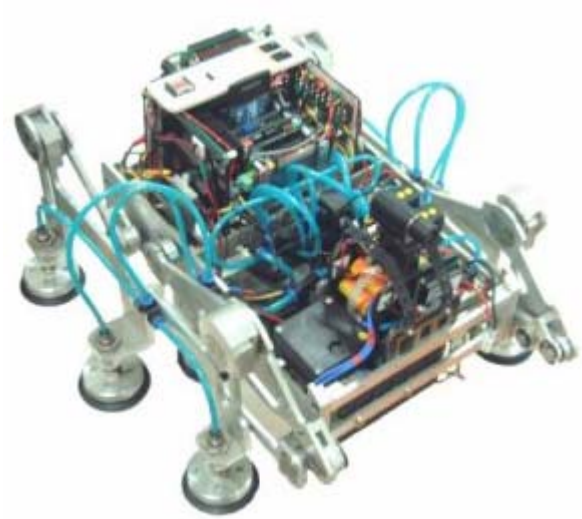
Jedná se o roboty s rozměry několika desítek milimetrů. Jsou vybaveny třemi nohami - dvěmi poháněnými a jednou opěrnou. Pohon robotu je zajištěn pomocí piezoelektrických akčních členů, které rozkmitají nohy robotu s frekvencí 800Hz a amplitudou několik mikrometrů.

Roboty vyvinou rychlost až 80 cm/s.

Nevýhodou robotu je neurčitost jeho pohybu - závisí na povrchu a jeho nerovnostech.

1.1.6.3 Šplhající roboty

Bývají postaveny na kráčející nebo kolové platformě. Kola nebo chodidla robotu bývají osazena přísavkami, které jim umožní pohyb po opláštění budov a kolmých objektech. Užívají se pro kontrolu povrchů, čištění a údržbu výškových budov.



Obrázek 8-Šplhající robot

1.1.6.4 Hybridní roboty

Podvozky robotů vznikají kombinací předchozích platforem.

- kolo-pás

- kolo-noha

- pás-noha

Speciálním druhem podvozku je pak robot který se pohybuje bruslením. Je osazen nohou s kolem.

1.1.6.5 Plavající robotické systémy

Primárním důvodem pro vznik plavajících mobilních robotů byla těžba ropy. První roboty byly konstruovány jako teleoperátory s omezeným dosahem, s komunikací užívající vodiče, později optické kabely. Tyto roboty byly určeny pro kontrolu, údržbu a opravy těžních

ropných věží. Roboty jsou osazeny manipulátory s potřebnými efektory a silným kamerovým vybavením. Další aplikací podmořských robotů jsou průzkumné činnosti. Jedná se o průzkum povrchu mořského dna, anomalit, vraků lodí... Roboty jsou koncipovány jako autonomní s jistým prvkem volného rozhodování. Řízení je však nadále udáváno teleoperátorem. Roboty jsou vybaveny nejrůznějšími snímacími systémy - od termovizí a teplotních čidel pro detekci teplých proudů a sopečné činnosti, přes ultrazvukové sonary, TV kamery, atd. Podmořské roboty jsou navrženy tak aby vydržely pod vodou co nejdelší dobu z jedné zásoby zdrojů energie. V oblasti vodních mobilních robotů se objevují i roboty plovoucí po hladině. Jedná se například o roboty veslující, které slouží k výzkumným účelům. Plovoucí roboty mohou sloužit pro ostrahu přístavů nebo pobřeží.

Obrázek 9-Plavající robot

1.2 Používané pohony pro mobilní robotické systémy

Nejčastějšími druhy pohonů pro mobilní robotická zařízení jsou pohony elektrické, méně již pneumatické nebo hydraulické.

1.2.1 Elektrické pohony

Mezi nejběžnější druhy pohonů patří elektrické synchronní, asynchronní a krokové motory.

V automatizaci a především v robotice se však spíše používají mnohá speciální řešení, kde kritériem pro konstrukční návrh není jen účinnost pohonu, ale také jeho velká přesnost, dobrá zatížitelnost a malé rozměry, rychlý rozběh brždění. Klasické servopohony se stejnosměrnými motory s permanentními magnety ustupují v této oblasti motorům bezkomutátorovým. Velkého rozšíření doznaly krokové motory. Pro dynamicky náročné aplikace s výkonem do 10 kW lze použít servopohony se synchronními motory s permanentními magnety. Mají sice nižší účinnost a vyšší cenu než klasické servopohony s asynchronními motory, ale v automatizačních aplikacích uživatelé ocení především malé rozměry a dobré dynamické vlastnosti.

Stále častější jsou integrovaná řešení, kdy jeden modul plní funkci řízení pohybu i programovatelného automatu. Příkladem může být Servo PLC 9300 (*Lenze*), modul, který kombinuje v jediném hardwaru měnič frekvence a PLC a umožňuje nejen nastavovat parametry pohonů, ale i programovat pohybové úlohy podle IEC 61131 a má k dispozici širokou paletu již připravených bloků a technologických funkcí.

1.2.1.1 Krokový motor

Krokový motor je impulsně napájený motor, jehož funkční pohyb je nespojitý a děje se po jednotlivých úsecích (krocích). K řízení krokového motoru slouží ovladač krokového motoru.

Funkce krokového motoru

Základní princip krokového motoru je úplně jednoduchý. Proud procházející cívkou statoru vytvoří magnetické pole, které přitáhne opačný pól magnetu rotoru. Vhodným zapojováním cívek dosáhneme vytvoření rotujícího magnetického pole, které otáčí rotorem.

Podle požadovaného kroutícího momentu, přesnosti nastavení polohy a přípustného odběru volíme některou s variant jeho řízení. Kvůli přechodových magnetickým jevům je omezena rychlost otáčení motoru a to na několik stovek kroků za sekundu (závisí na typu motoru

a jeho zatížení). Při překročení této maximální rychlosti (nebo velikosti zatížení) motory začínají ztrácet kroky. Základní varianty řízení krokových motorů jsou unipolární a bipolární řízení.

Při **unipolárním** řízení prochází proud v jednom okamžiku právě jednou cívkou. Motor s tímto druhem buzení má nejmenší odběr, ale také poskytuje nejmenší kroutící moment. Výhodou je jednoduché zapojení řídicí elektroniky – v podstatě stačí jeden tranzistor na každou cívku. Pro menší motory lze s výhodou použít integrovaný obvod ULN2803. V jednom pouzdře je dostatek budičů pro řízení dvou motorů.

Při **bipolárním** řízení prochází proud vždy dvěma protilehlými cívkami. Ty jsou zapojené tak, že mají vždy opačně orientované magnetické pole. Motor v tomto režimu poskytuje větší kroutící moment, ovšem cenou za to je větší spotřeba. Pro řízení jsou zapotřebí 2 H-můstky : pro každou větev jeden. To v důsledku znamená jednak složitost zapojení a větší počet kontrolních linek (jejichž počet lze zredukovat pomocí přídavné logiky). Vhodným integrovaným obvodem pro bipolární řízení je H-můstek L293D.

Dalším rozdělením je řízení jednofázové nebo dvoufázové.

Při **jednofázovém** řízení je magnetické pole generováno pouze jednou cívkou (příp. dvěma při bipolárním buzení).

Při **dvoufázovém** řízení generují shodně orientované magnetické pole vždy dvě sousední cívky. Daní za vyšší kroutící moment je dvojnásobná spotřeba oproti řízení jednofázovému.

Mikrokrokování

Při některých aplikacích je požadováno velice jemné krokování. Úhel kroku je dán konstrukcí motoru a bývá v rozmezí 0,36 až 15 stupňů. Další zmenšení kroku umožňuje metoda nazvaná „mikrokrokování“, při které je možno každý krok rozdělit na několik mikrokroků stejné délky. V současné praxi je možno krok rozdělit maximálně na 64 až 128 mikrokroků. Při klasickém buzení teče jednotlivými fázemi proud vždy stejné velikosti. Vhodnou volbou velikosti proudu v jednotlivých fázích můžeme dosáhnout libovolné rovnovážné polohy mezi dvěma sousedními „normálními“ kroky. Pro buzení je nutno použít dvouhladinový napájecí zdroj a zajistit potřebné řídicí impulsy pro řízení tohoto zdroje. V případě většího dělení kroků rostou požadavky na napájecí a spínací obvody.

Zmíním se i o nevýhodách krokových motorů. Ten nejzávažnější je pravděpodobně trvalý odběr proudu i když se motor netočí. Nepříliš výhodný je i poměr výkonu (kroutícího momentu) vůči hmotnosti motoru. Na druhou stranu právě krokové motory nacházejí široké uplatnění díky jejich snadné obsluze. Pro precizní řízení rychlosti nepotřebujeme naprogramovat komplexní PID kontrolér, a pokud motory příliš nepřetěžujeme, lze se obejít bez zpětné vazby o změně natočení – stačí počítat kroky.

1.2.2 Pneumatické pohony

Použití pneumatických pohonů u mobilních robotických systémů je spíše pro jejich manipulační nástavby (pohyb efektoru, atd.) než pro vlastní pohyb robotu. Nejčastěji se ale pneumatické pohony využívají u statických průmyslových robotů (svářecí apod.). Jejich velkou výhodou je vysoký výkon vzhledem k hmotnosti, dále plynule regulovatelná síla a zdvih. Nevýhodou bránící rozšíření do mobilních systémů je nutnost zdroje stlačeného plynu (kompresor, tlaková láhev,…).

1.2.2.1 Umělý sval

Umělý sval představuje nový trend v aplikaci pohonů, založený na kombinaci pružného poddajného materiálu a elektronických prvků. Svou konstrukcí se podobá lidskému svalu. Jako pohon nachází uplatnění všude tam, kde z různých důvodů nelze použít běžný typ

pohonu (např. rozměr, hmotnost, tvarová přizpůsobitelnost atd.). S rozměrem pohonu také úzce souvisí jejich počet, který se dá na manipulační nástavbě či servisním robotu použít. Malý rozměr pohonu nesmí být na úkor vyvozované síly. Velkou výhodou umělého svalu je jeho tvarová přizpůsobitelnost danému mechanismu.

Z hlediska využívaných typů umělých svalů je nejrozšířenější pneumatický přetlakový umělý sval. Ten se v současné době využívá v celé řadě konstrukcí mobilních robotů majících manipulační nástavbu. Tlak vzduchu v umělém svalu lze plynule regulovat, což umožňuje jednoduše řídit vyvozovanou sílu a zdvih. Některé konstrukce manipulačních nástaveb pro mobilní roboty využívají jako pohony pouze umělé svaly. S výhodou se používají umělé svaly také v kombinaci s běžným typem pohonu.

Základní vlastnosti pneumatických svalů

Pneumatické svaly se vyznačují mimořádně vysokým poměrem síly a výkonu k hmotnosti a objemu. Mohou být vyrobeny prakticky v libovolné délce a průměru a jejich vlastnosti, tvar a chování jsou srovnatelné s lidskými svaly (přirozená pružnost), což umožňuje jejich snadné vzájemné propojení (protézy, rehabilitace apod.). Maximální dosažitelné zkrácení se pohybuje na hranici 30 % jmenovité délky svalu, což je opět srovnatelné se svaly živočichů. Dosud vyvinuté regulátory jsou schopny regulace polohy s přesností lepší než 1 % z rozsahu pohybu a umožňují dosáhnout mezní frekvence více než 10 Hz. Tažná síla na jednotku plochy průřezu dosahuje až 300 N/cm2 v porovnání s 40 N/cm2 u svalu živočicha. Mezi další přednosti pneumatických svalů patří:

- přesný a plynulý chod svalu mezi krajními polohami;
- nízká cena, vysoká spolehlivost, minimální údržba;
- hermeticky uzavřená konstrukce odolná proti prachu;
- vysoká bezpečnost – možnost použití ve výbušném a vlhkém prostředí.

Pneumatický sval v robotice

Z uvedených vlastností je zřejmé, že umělý pneumatický sval je téměř ideálním pohonem zejména pro robotické aplikace. Stále však nedošlo k jejich většímu rozšíření. Masovému nasazení pneumatických svalů brání především následující problémy:

Pro pohon svalu je potřeba zdroj stlačeného plynu. Konveční řešení napájení z tlakové lahve nebo kompresoru jsou nákladná a hmotná. Netradiční řešení například získávání plynů chemickými reakcemi se potýkají problémy s řízením reakce, chlazením a bezpečností provozu.

Tření vnitřní struktury pneumatického svalu lze obtížně modelovat. Dosud nepřesně modelované tření vylučuje možnost precizního řízení pneumatického svalu.

Z těchto důvodů se pneumatické svaly zatím uplatňují pouze ve stacionárních robotech s nízkými nároky na přesnost polohování nebo v aplikacích, kde je žádoucí přirozená pružnost pohonu, jako jsou exoskeletony a rehabilitační pomůcky.

2 KOMUNIKACE POMOCÍ RS-232, WIFI A BLUETOOTH

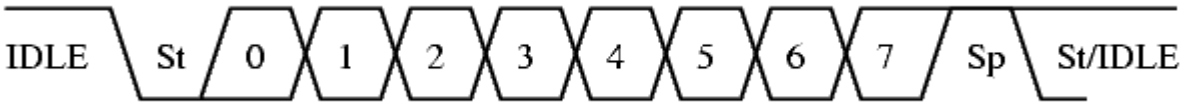
2.1 RS-232

2.1.1 Co je RS-232

Původně byl standard navržen ke komunikaci mezi dvěma zařízeními (DTE-Data Terminal Equipment, DCE-Data Communication Equipment) vzdálenými do 20m. Pro větší odolnost proti rušení je informace po propojovacích vodičích přenášena větším napětím, než je standardních 5 V. Přenos informací probíhá asynchronně, pomocí pevně nastavené přenosové rychlosti a synchronizace sestupnou hranou startovacího impulzu.

RS-232 znamená Recommended Standard číslo 232. Jedná se o relativně dlouho přežívající způsob komunikace, který vznikl v roce 1969 a na většině počítačů ho nalezneme v podobě devíti-pinového konektoru dodnes.

RS-232 komunikuje pomocí rámců (frames). Pokud se nic neděje, tak je linka v klidovém (IDLE) stavu, pro který se používá kladné napětí. Každý rámec začíná start bitem (St), což je změna na záporné napětí na dobu danou rychlostí komunikace (např. pro 9600baud je to 1/9600s, tj. cca 104μs). Následují datové bity, kdy logická jednička odpovídá zápornému napětí a logická nula kladnému. Vysílá se od nejméně důležitého bitu. Celý rámec je zakončen stop bitem (Sp), kdy je linka zase v klidovém, tedy kladném napětí. Po stop bitu může následovat pauza (IDLE) nebo hned start bit (St).



Obrázek 10-Taktovací signál RS-232 sériové komunikace

Frame rámce mohou po sobě hned následovat, takže pokud používáme přenosovou rychlost 9600 baud, tak za 1s můžeme poslat maximálně 9600/10=960 bajtů (číslo 10 odpovídá jednomu start bitu, 8 datovým bitům a jednomu stop bitu).

2.1.2 Zapojení konektorů pro RS-232

2.1.3 Maximální délka vedení

Standard RS-232 uvádí jako maximální možnou délku vodičů 15 metrů, nebo délku vodiče o kapacitě 2500 pF. To znamená, že při použití kvalitních vodičů lze dodržet standard a při zachování jmenovité kapacity prodloužit vzdálenost až na cca 50 metrů. Kabel lze také prodlužovat při snížení přenosové rychlosti, protože potom bude přenos odolnější vůči velké kapacitě vedení. Uvedené parametry počítají s přenosovou rychlostí 19200 Bd.

Texas Instruments uvádí jako výsledek pokusných měření následující délky vodičů v závislosti na přenosové rychlosti(viz Tab.1.). Jedná se o údaje naměřené v laboratorních podmínkách, v praxi je třeba počítat s rušením atd.

Tab. 1. Maximální délka vedení

Přenosová rychlost [Bd]	Maximální délka [m]
19 200	15
9 600	150
4 800	300
2 400	900

Pro přenos dat na větší vzdálenosti je výhodnější používat rozhraní RS-422, RS-485, či proudovou smyčku.

Baud je jednotka používaná pro měření rychlosti přenosu dat. Přenosová rychlost definuje rychlost přenosu dat z datového média na jiné datové médium. Baud rate udává počet změn signálu za sekundu. Jako základní jednotka informace v moderních počítačových systémech se bere jeden bit (nabývá hodnoty 0 nebo 1). Do jedné signálové změny lze zakódovat i více než jeden bit. A proto nelze slučovat pojem bit za vteřinu s pojmem baud.

Rozhraní RS-232 je relativně málo odolné proti rušení, neboť přenos dat je realizován napěťovou úrovní na vodičích (vůči GND) na zatěžovacím odporu 3,7kΩ při šumové imunitě 3V. Mnoho zařízení má ale vstupní impedanci mnohem vyšší (až 30kΩ) a šumovou imunitu nižší (1V), takže dochází ke zvýšenému rušení, a tím ke zmenšenému možnému dosahu linky. V každém případě se doporučuje použít stíněný kabel a věnovat pozornost způsobu provedení signálové země a země zařízení.

2.1.4 Synchronní a asynchronní přenos informace

SYCHRONNÍ přenos informací znamená, že na některém vodiči nebo vodičích se nastaví určitá úroveň, která přenáší informaci a validita informace se potvrdí impulzem, nebo změnou úrovně synchronizačního signálu. Synchronizačním signálem se tedy informace kvantují.

Základní vlastnosti SYCHRONNÍHO přenosu :

- Výhodné pro velké objemy dat, přenášené po více vodičích.
- Nutno jednoznačně určit, kdo vysílá synchronizační impulzy.
- Nutnost synchronizačního vodiče.
- Na straně zařízení nepotřebuje nijak složitou elektroniku.

ASYNCHRONNÍ přenos dat přenáší data v určitých sekvencích. Data jsou přenášena přesně danou rychlostí a uvozena startovací sekvencí, na kterou se synchronizují všechna při-

přijímací zařízení. Všechny strany obsahují vlastní přesný oscilátor, díky kterému odečítají data v přesně definovaných intervalech.

Základní vlastnosti ASYNCHRONNÍHO přenosu :

- Nevýhodné pro velké objemy dat, ale vhodné pro dlouhá vedení, na nichž by synchronizační vodič činil nezanedbatelné finanční náklady.
- Lze použít pro komunikaci mezi mnoha zařízeními.
- Celkem složitá a drahá elektronika, nutno použít krystalové oscilátory.
- Až o 20% menší přenosová rychlost užitečných dat při stejné rychlosti komunikace, vzhledem k nutnosti startovacích a paritních bitů.

RS-232 používá asynchronní přenos informací. Každý přenesený byte konstantní rychlostí je proto třeba synchronizovat. K synchronizaci se používá sestupná hrana tzv. Start bitu. Za ní již následují posílaná data.

Obrázek 11-Synchronizace dat u RS-232

Používá-li se v zařízení TTL nebo CMOS obvody, bude nutno jejich logickou RS-232 linku napěťově upravit před připojením do PC, protože napěťové úrovně RS-232 nejsou přímo slučitelné z žádnou logikou.

2.2 WiFi

WiFi (Wireless Fidelity) je bezdrátová, síť určená primárně k náhradě kabelového ethernetu. Samotný název WiFi vytvořilo WECA(Wireless Ethernet Compatibility Alliance) a v principu jde o bezdrátovou technologii v bezlicenčním nekoordinovaném pásmu 2,4GHz, což přibližně odpovídá vyšším frekvencím vln rádia nebo nižší frekvenci televizních vln. Technologie je založena na protokolu 802.11b. WiFi je pouze komerční název, který je

fakticky pouze podmnožinou 802.11b, nicméně jsou občas tyto dva pojmy používány jako synonymum.

Hlavní výhodou této technologie je její nízká cena, způsobená mimo jiné tím, že certifikovaná zařízení jsou k dispozici ve velkých sériích. Protože požadavky na certifikaci zařízení jsou běžně dostupné a norma 802.11b dokonce volně k dispozici na webu.

Většina sítí založených na WiFi funguje na buňkovém principu, kdy centrální přístupový bod zprostředkovává připojení všem stanicím v dosahu a body dohromady tvoří jakousi plástev - analogicky s GSM sítí.

2.2.1 Historie

Bezdrátové sítě pro běžný trh existují v podstatě od května 1993, kdy firma NCR (tehdejší součást gigantu AT&T) uvedla na trh svou WaveLAN technologii. Tato technologie byla čistě bez standardu, nabízela rychlosti max. 2Mb/s. Už tehdy využívaly pásmo 2,4GHz, v Americe pak ještě pásmo 900 MHz. Téhož roku se také začalo pracovat na standardizaci bezdrátových sítí pod patronací organizace IEEE, první standard byl hotov až v červenci roku 1997 a dostal název IEEE 802.11. Tento standard definoval tři různé fyzické vrstvy.

V té době už ale bylo jasné, že rychlost 2Mb/s nebude postačovat, proto už nějakou dobu se v IEEE pracovalo na vylepšení těchto standardů a to hned na dvou frontách - výzkumná skupina A se zabývala využití jiného frekvenčního pásma, výzkumná skupina B se snažila nalézt způsob jak lépe využít existujícího pásma.

První svou práci dokončila skupina B a dala tak vzniknout standardu 802.11b a to roku 1999. Tento standard se už nezabýval neperspektivními technologiemi a zaměřil se pouze na DSSS. Přidal podporu dvou dalších modulačních schémat, díky kterým dokázal s využitím stejných 20MHz dosáhnout rychlosti 11Mb/s. Tento standard si ale vybral jednu daň - přinesl několik volitelných součástí, které mohl výrobce zvolit a tudíž zařízení podle tohoto standardu nemusela být navzájem kompatibilní.

V roce 2002 skončila výzkumná skupina A hlavní činnost tím, že uvedla na trh standard 802.11a - standard pro bezdrátové sítě v pásmu 5GHz. MAC vrstva je shodná s 802.11, tudíž implementace čipsetů je velmi levná, modulační rychlost je díky nové modulaci na-

výšena na 54Mb/s. Bohužel ale Evropa (včetně ČR) produkty podle tohoto standardu nemohla (a v ČR ještě nemůže) používat, neboť jsou zde na pásmo 5GHz kladeny požadavky, které nebyly v 802.11a zohledněny.

I proto vznikly další dvě skupiny. "G" uvedla roce 2003 standard 802.11g, který nabízí stejnou modulaci, ovšem používá běžnější pásmo 2,4GHz a tak jsou tyto produkty prodávané úspěšně i u nás. Výhoda těchto produktů je, že jsou kompatibilní se zařízeními dle 802.11b (na rychlostech do 11Mb/s). Skupina "H" uvedla standard 802.11h, který řeší nedostatky požadované evropským telekomunikačním úřadem.

2.2.2 Vrstva přístupu k médiu

Specifikace této vrstvy ve standardu 802.11 má určité společné prvky se standardem 802.3 pro klasický drátový Ethernet. Elektrikářsky řečeno, tato vrstva ověřuje před zahájením přenosu dat, zda na komunikačním médiu už nevysílá někdo jiný.

Standard 802.11 používá protokol CSMA/CA. Tento protokol používá techniku předcházení kolizí oproti technice detekce kolize, kterou používá standard 802.3, a to z důvodu obtížnosti detekce kolizí v sítích používajících bezdrátové medium při vysokofrekvenčním přenosu.

2.2.3 CSMA/CA

Omezením bezdrátových LAN je problém tzv. “skrytého uzlu”, který může omezit komunikaci na síti až o 40 a více procent. Jedná se o uzel, který není schopen detekovat používání přenosového média a může se tak pokoušet k němu přistupovat právě v okamžicích, kdy je již síť používána. Tento problém řeší následující postup.

Protokol CSMA/CA zajišťuje minimum kolizí použitím čtyř rámců:

- RTS (Ready to send),
- CTS (Clear to send),
- ACK (Acknowledge)
- NAV (Network allocation vector)

Komunikace pak probíhá následujícím způsobem: jeden z uzlů bezdrátové sítě vyšle požadavek na komunikaci zasláním rámce RTS s udáním adresy příjemce a délkou zprávy. Na základě RTS se v každém uzlu vypočítá NAV, ostatní uzly jsou tak upozorněny, že síť je na nejbližší dobu již používána. Adresát zprávy na RTS odpovídá zasláním CTS, čímž dává na vědomí, že je schopen přijímat. Neobdrží-li vysílací uzel CTS, je to považováno za kolizi a celý proces začíná znovu. Po úspěšném přijetí dat zasílá přijímací stanice potvrzení o přijetí (ACK).

2.2.4 Fyzická vrstva

Fyzická vrstva je nejnižší vrstvou referenčního modelu OSI a dělí se na podvrstvu konvergence fyzické vrstvy PLCP (Physical Layer Convergence Protocol), která je do určité míry nezávislá na použitém přenosovém médiu (rádiový kanál, optický kanál, metoda rozprostření pásma, modulace) a podvrstvu závislou na fyzickém médiu PMD (Physical Media Dependent) specifikující vlastní přenosový kanál.

Původní specifikace 802.11 uváděla tři samostatné fyzické vrstvy pro DSSS, FHSS a IR. Norma 802.11b přidává podporu vysokorychlostní (HR) DSSS.

2.2.4.1 Možnosti fyzické vrstvy

Fyzická vrstva v jakékoliv síti definuje modulační a signalizační charakteristiky přenosu dat. Provozování bezdrátových LAN v nelicencovaných pásmech požaduje modulaci s rozprostřeným spektrem, které jsou v 802.11 definovány dvě: FHSS (Frequency Hopping Spread Spectrum) a DSSS (Direct Sequence Spread Spectrum). Obě tyto architektury pracují na frekvenci 2,4GHz s šířkou pásma 83MHz (tedy od 2,400 GHz do 2,483 GHz). Jako modulační metodu používá FHSS dvou- až čtyřúrovňovou modulaci GFSK (Gaussien Frequency Shift Keying), DSSS pak diferenční BPSK a DQPSK. Modulační rychlost na fyzické vrstvě s použitím systému FHSS je 1,6 nebo 3,2Mb/s, u systému DSSS může být 1, 2, 5.5 a 11Mb/s. Výběr mezi těmito dvěma systémy záleží na nárocích kladených na koncovou aplikaci a také na prostředí, ve kterém bude aplikace provozována.

2.2.4.2 DFIR (Diffused infrared)

Přenos infračerveným zářením. Infračervená varianta lokální datové komunikace je zásadně omezena na jedinou kancelář nebo jiný souvislý prostor, neboť infračervené paprsky neprocházejí pevným materiálem, a naopak dochází k odrazu.

Standard pro infračervené bezdrátové spojení pracuje v pásmu 850 až 950 nm s maximálním výkonem 2W. Pro infračervené spojení jsou podporovány přenosové rychlosti 1 i 2Mb/s. Řešení na bázi infračerveného záření je podstatně dražší než u rádiových sítí, takže se takže se tento standard vůbec neujal.

2.2.4.3 DSSS (Direct Sequence Spread Spektrum

Technika přímého rozprostřeného spektra. Je jednou z metod pro rozšíření spektra při bezdrátovém přenosu dat.

Pracuje tak, že každý jednotlivý bit určený k přenosu, je nejprve nahrazen určitou početnější sekvencí bitů (tzv. chipů). Tyto sekvence mají nejčastěji pseudonáhodný charakter. Každý datový bit je reprezentován známou sekvencí 11-ti bitů a ne všechny chipy jsou tudíž potřebné pro správnou demodulaci. DSSS používá 11 kanálů o šířce 22MHz, povolené pásmo na frekvenci 2,4GHz má ovšem šířku jen 83,5MHz. Použití odlišných sekvenčních kódů pak umožňuje umístění více DSSS systémů v jednom místě. Skutečně přenášena je pak tato sekvence bitů. Jde tedy vlastně o umělé zavedení nadbytečnosti (redundance). Důvodem je, že signál je rozprostřen do větší části radiového spektra a je méně citlivý vůči rušení. Signál se ostatním uživatelům jeví jako náhodný šum, a bez znalosti mechanismu vytváření původní pseudonáhodné sekvence, je pro ně obtížné zpět získat přenášená data. Jedná se o modulační techniku používanou například v bezdrátové technologii či v navigačním systému GPS.

Vzhledem k typické šířce DSSS kanálu mohou v přiděleném bezlicenčním pásmu 2400 – 2483,5MHz pracovat vedle sebe nezávisle 3 kanály DSSS. Jejich středové kmitočty musí být voleny tak, aby se vzájemně nedotýkaly ani okraji zabraných pásem

DSSS umožňuje tři různé modulační metody, přičemž každá z těchto modulačních metod zajišťuje různou přenosovou rychlost. Verze pro 1Mb/s používá diferenciální binární klí-

čování s fázovým posunem a verze pro 2Mb/s používá diferenciální kvadraturní klíčování fázovým posunem.

2.2.4.4 FHSS (Frequency Hopping Spread Spectrum)

Fyzická vrstva, založená na FHSS, má k dispozici 22 modelů (skokové sekvence). Na této fyzické vrstvě je definováno 79 kanálů v okolí frekvence 2,4GHz. Každý z těchto kanálů zabírá šířku pásma 1MHz a "přeskakuje" minimálně 2,5krát za vteřinu (ve Spojených státech), typicky 20krát.

Technika přeskoků kmitočtů rozděluje data pro přenos přes dostupné frekvenční pásmo za použití pomocných nosných vln. Datová zpráva je tak vysílána pomocí mnoha nosných frekvencí tzv. hops. Vysoké spolehlivosti je dosaženo díky tomu, že nepotvrzené tj. chybně přenesené rámce jsou znovu přenášeny s jinou nosnou frekvencí. Umístění více systémů v jednom místě je umožněno použitím různých sekvencí v každém systému. Ovšem žádná z aktuálních implementací založených na 802.11 FHSS nepoužívá.

2.2.4.5 Shrnutí a porovnání FHSS a DSSS

Oba popisované systémy mají definovaný vlastní inicializační sekvenci bitů (hlavičku), aby přijímač byl schopen rozpoznat použitý modulační formát a očekávanou délku datového řetězce. Tyto hlavičky jsou vždy přenášeny na rychlosti 1,6Mb/s a obsahují pole, na základě kterého následná rychlost přenosu dat může být zvýšena na 3,2Mb/s.

DSSS dokáže přenést větší šířku pásma než FHSS. DSSS ale vysílá na jednom frekvenčním kanále, přičemž data vysílá víckrát, čímž je zajištěna robustnost přenosu dat a znemožněno snadné rušení úzkopásmovým vysílačem. DSSS rozděluje pásmo 2,4GHz prakticky na 3 nezávislé kanály (kmitočty 2412, 2437 a 2462 MHz). Ostatní kanály se navzájem překrývají.
Přenos dat na FHSS je pomalejší, robustnost přenosu dat je dána nepoužíváním kanálů, které již používá jiná technologie.

2.2.5 Výhody WiFi

- Velmi rychlá instalace bezdrátového přístupového modulu
- Trvalé připojení k Internetu velmi vysokou rychlostí bez datového limitu
- Výše měsíčního paušálu je fixní, není tedy závislá na délce připojení ani objemu přenesených dat
- Zrychlení elektronické pošty, odpadá zdlouhavé vytáčení a vaše pošta ihned odchází vašemu příjemci, přijatá pošta je doručována ihned až na váš počítač
- Kompletní bezdrátový přístupový okruh – úspora nákladů za nákup nebo pronájem drahého koncového zařízení

2.3 Bluetooth

2.3.1 Historie

V roce 1994 švédskou společnost Ericsson napadlo vyrobit bezdrátové sluchátko. Dva zaměstnanci firmy, Holanďan Jaap Haartsen a Švéd Sven Mattisson, začali pracovat na projektu, jehož výsledkem měla být technologie, která bude levná, velmi nenáročná na energii a dokáže nahradit kabely spojující mobilní telefony s jejich příslušenstvím. Tento projekt byl nazván MC-Link. Bluetooth je také znám jako norma IEEE 802.15.1.

Ericsson oslovil čtyři velké firmy: Nokii, IBM a Toshibu a Intel. Bezdrátový MC-Link se jim zjevně líbil, proto v únoru 1998 založili společné sdružení a hledali název pro svůj projekt. Zaměstnanec Intelu zrovna dostal knihu o dánském králi Haraldovi a jméno bylo na světě. Bluetooth. Tak jako Harald sjednotil severské národy, tak chce Bluetooth, coby technologický standard, sjednotit různá, většinou mobilní zařízení.

2.3.2 Technologie Bluetooth

Bluetooth je podobně jako wifi, všeobecně dostupná rádiová frekvence 2,4GHz, pomocí které se mohou vzájemně propojit zařízení vybavená rozhraním Bluetooth na vzdálenost do deseti až sta metrů. Bluetooth protokol dělí pásmo na 79 kanálů, každé o šířce

pásma 1MHz a přepíná kanály 1600krát za vteřinu. Pomocí tohoto rozhraní můžete připojit svůj notebook nebo handheld k jiným notebookům, mobilním telefonům, fotoaparátům, tiskárnám, klávesnicím, reproduktorům a dokonce k myši.

Existuje několik verzí Bluetooth. Verze 1.0 a 1.0B měla nespočet chyb. Různé továrny měly velké problémy s tím aby jejich produkty byly schopny spolupracovat.

Mnoho chyb z verze 1.0 a 1.0B se vyskytovalo i ve verzi 1.1, byla však přidána podpora nešifrovaných kanálů.

Verze 1.2 je zpětně kompatibilní s verzí 1.1 dosahuje v praxi vyšší přenosové rychlosti. Třikrát vyšší přenosové rychlosti (v některých případech i 10x), nižší nároky na spotřebu díky omezení cyklů, přinesla až verze 2.0. Také spojování je jednoduší díky větší šířce pásma.

Verze 1.1 a 1.2 dosahují rychlosti 723.1kb/s, verze 2.0 2.1Mb/s. Technicky by Bluetooth zařízení verze 2.0 měli mít vetší spotřebu energie, ale 3krát vyšší rychlost snižuje dobu přenosu, efektivně redukuje spotřebu na polovinu než mají Bluetooth zařízení verze 2.0.

Díky rychlému a snadnému propojení zařízení vybavených rozhraním Bluetooth lze vytvořit osobní síť (PAN), která umožňuje sloučit všechny důležité pracovní nástroje do plně funkčního celku. Rovnocenné připojení pomocí rozhraní Bluetooth umožňuje snadnou výměnu souborů na schůzkách nebo tisk dokumentů bez nutnosti připojení k pevné nebo bezdrátové síti.

Bluetooth však není totéž co WiFi. Ač se tak zpočátku může zdát. Bluetooth není tak rychlý jako WiFi, také má menší nároky na napájení a v neposlední řadě má WiFi mnohem dražší zařízení. Na druhou stranu má větší dosah. Bluetooth stejně tak jako WiFi jsou radiové technologie, rozdíl je pouze a hlavně v používaných frekvencích a protokolech. Používají stejné frekvenční pásmo, ale jsou vybaveny různými systémy multiplexování. Zatímco Bluetooth nahrazuje kabel u různých aplikací, WiFi nahrazuje kabel pouze u LAN sítí. V podstatě se dá říct, že Bluetooth je bezdrátový USB, kdežto WiFi je bezdrátový Ethernet.

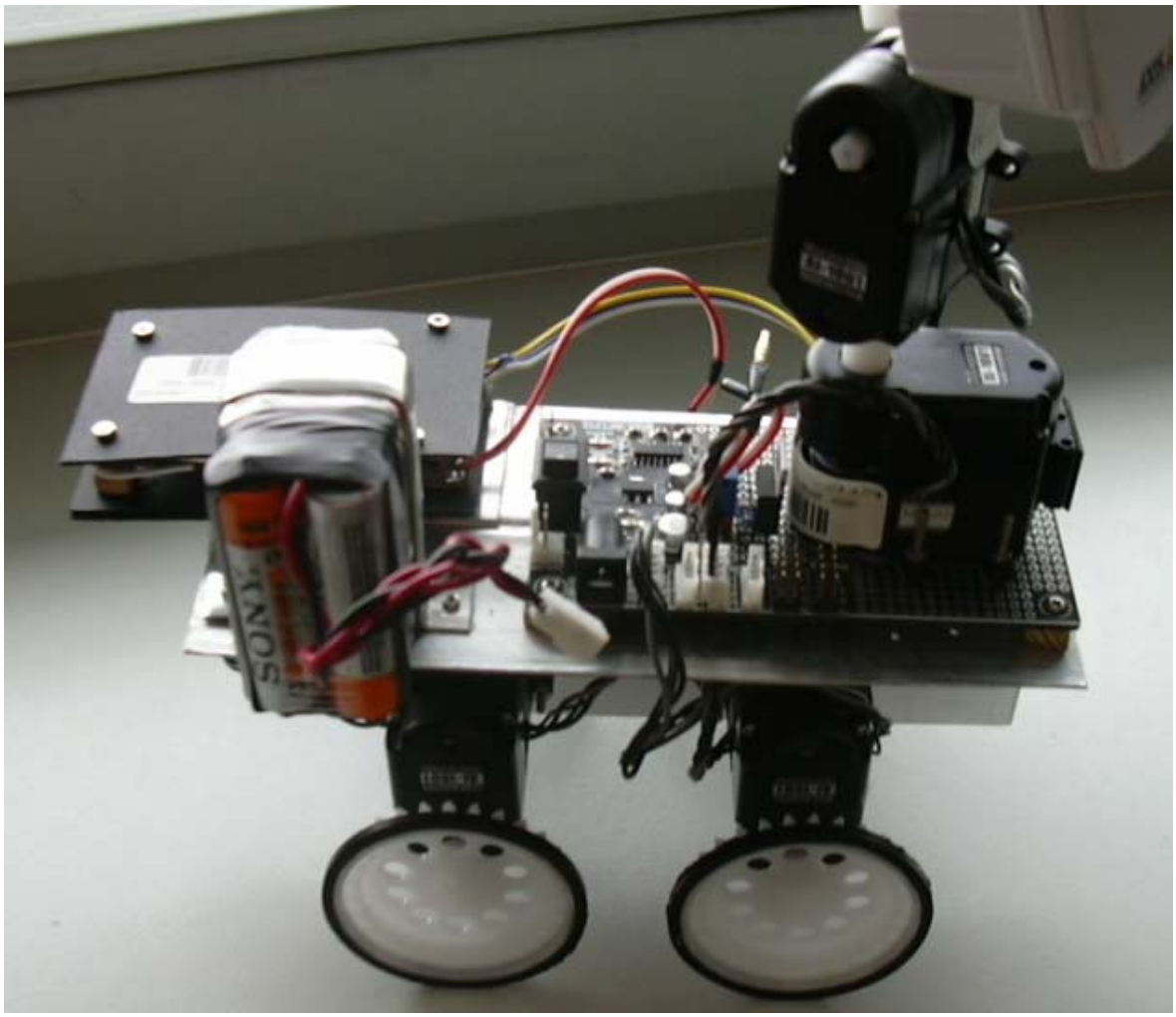
2.4 Bezpečnost v bezdrátových sítích

Z podstaty bezdrátové komunikace vyplývá, že kdokoli s bezdrátovým zařízením, kdo přijde do oblasti pokryté bezdrátovou sítí, bude moci síť využívat a sdílet její služby – odtud potřeba bezpečnosti. Nejběžnějším standardem je WEP (Wireless Equivalent Privacy), který šifruje komunikaci, takže uživatelé bez správného klíče nemohou přistupovat do sítě. Jenže klíč samotný není zašifrován, takže je možné tuto ochranu prolomit a měla by být považována za pouhý základní stupeň bezpečnosti. WEP klíč je 40, 64 nebo 128 bitový. V poslední době jsou používány nové standardy jako WPA (WiFi Protected Access), které odstraňují některé nedostatky WEP.

II. PRAKTICKÁ ČÁST

3 SESTAVENÍ A OŽIVENÍ MOBILNÍHO ROBOTA

Pro sestavení robota bylo použito šest AI-MOTORů 1001 od firmy Megarobotics. Čtyři pro ovládání kol.a další dva pro případnou montáž koncového manipulátoru či senzoru (např. kamery). Jako trup byly použity dva hliníkové plechy (viz obr.), na které byla shora připevněna základní deska a z každé strany dva AI motory. Toto tvoří základní část robota. Na zadní části je místo na uchycení baterií a modulu pro bezdrátovou komunikaci. Všechny jednotlivé součásti včetně jejich zapojení a propojení se základní deskou jsou popsány níže.



Obrázek 12 - Mobilní čtyřkolový robot

3.1 AI-MOTOR 1001

AI-MOTOR 1001 je akčním členem k řízení robotů. Servomotorky jsou použitelné pro všechny druhy pohybu. Nicméně nejsou určeny pro běžného uživatele, ale spíše do labora-

torních podmínek a to nejen kvůli své ceně, ale je potřeba mít i odbornější znalosti v oblasti elektroniky a programování.

Obrázek 13 – AI-MOTOR 1001

AI-MOTOR 1001 tvoří komplex servomotorku, pevných částí a řídících obvodů tak, aby jejich spojení v jeden celek bylo jednoduché a praktické. Díky své konstrukci, mohou být jednoduše navrhovány spoje pohyblivých zařízení, je jednoduché zařízení dodatečně rozložit a vyrovnat problémy. Je možné spojovat servomotorky do série v jednoduchou síť. V tomto případě tvoří jednu větev čtyři motorky spojené do série ovládající kola a druhou větev tvoří dva motorky pro uchopení kamery. Základní deska poskytuje místo pro čtyři větve. Ovládací instrukce a data mohou být dodávány pomocí jednoduchého použití RS232 sériové komunikace přes TTL logiku.

3.1.1 Vnitřní funkce a propojení AI-MOTOR 1001

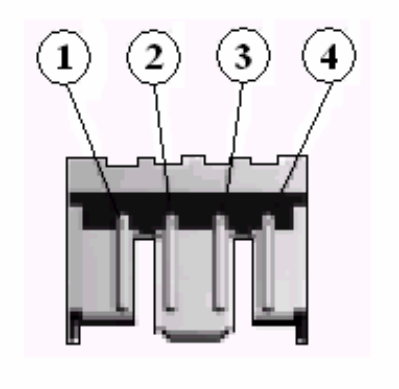
Servomotorky mají svoji instrukční sadu. Vnitřní parametry servomotorků, jako jsou ID, přenosová rychlost sériové komunikace, regulace polohy, prahové přepětí, mohou být měněny programově skrz sériový port.(Tab.2)

Tab. 2: Vlastnosti AI-MOTOR 1001

Vlastnost	Rozsah
Přenosová rychlost	2400 až 460800 bit/s
Proudová ochrana	400mA(5V) až 1000mA(10V)

Rozsah pohybu	0 až 254(360°)
Točivý moment	7Kg/cm při 9,5V
Rychlost otáček	84 otáček za minutu

AI MOTOR 1001 má 2 konektory (viz Obr. 19).



Obrázek 14 - Konektor AI-MOTOR

Tab. 3: Funkce konektorů

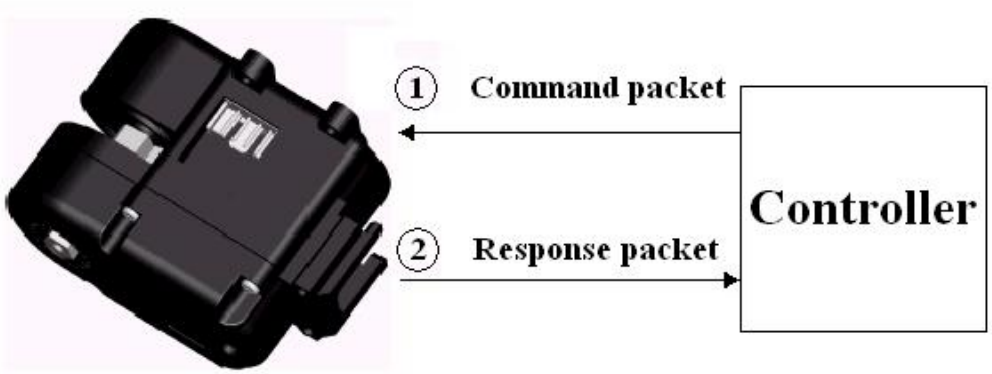
Číslo	Funkce
1	Vcc
2	TXD
3	RXD
4	GND

Po připojení konektoru k základní desce je servomotorek plně funkční, pouze po dobu 64ms po sepnutí, nebo restartu nejsou přijímány žádné instrukce. Druhý konektor slouží k připojení dalšího servomotorku do série. Takto sériově spojených bloků může být až 30. Některé zdroje udávají až 31. AI-MOTOR 1001 automaticky kontroluje vnitřní řídící okruh tak, že vnitřní odezva je stále zajištěna, i přes různé napájecí napětí. To chrání motorek proti jeho zničení, ten je automaticky odpojen pokud je protékající proud příliš vysoký. [4]

Protože AI-MOTOR přijímá instrukce po RS-232 lince, nemohou všechny servomotorky přijímat instrukce v tomtéž čase. Ačkoliv zpoždění je zanedbatelné, v závislosti na přenosové rychlosti pouze několik desetin milisekundy, je třeba s ním počítat.

3.1.2 Komunikační tok

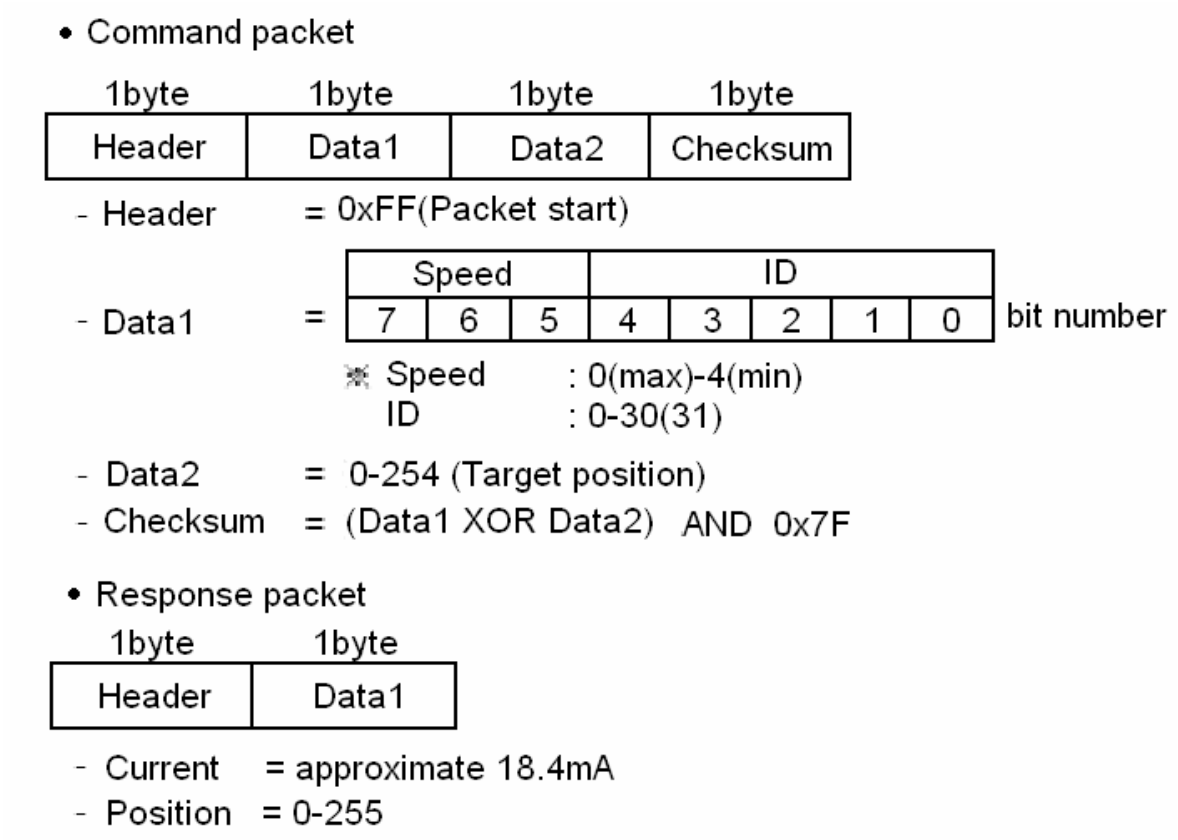
Kontroler zasílá balíček příkazů (command packet) k AI-MOTOR, ten posílá odezvu (response packet) o obdržení příkazů.



Obrázek 15 - Komunikace mezi AI-MOTOR

U AI-MOTOR se jedná o dva druhy příkazů (command packets), a to o řídicí a nastavovací příkazy.

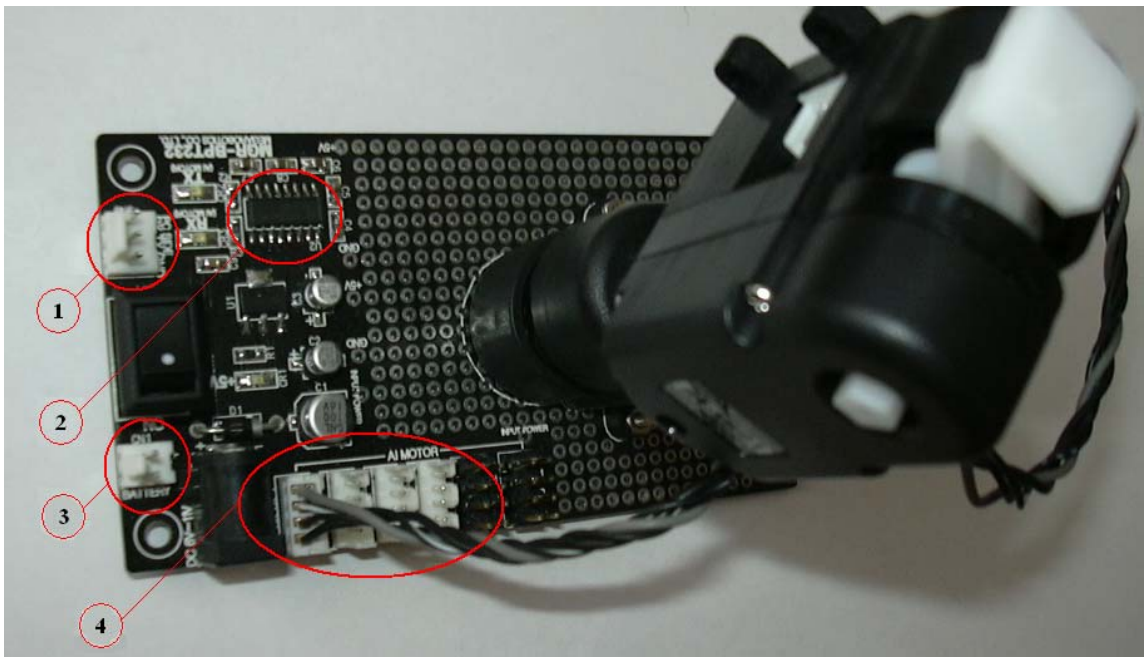
Když uživatel pošle svou požadovanou absolutní pozici natočení od 0 do 254, je hřídel AI-MOTOR natočena do požadované polohy mezi 0 a 332°. Z uvedeného vyplývá, ze jedna pozice natočení odpovídá 1.307 úhlovému stupni. Regulaci, přesněji řečeno nastavení polohy provádí funkce „Position Send Command“. Comand packet a response packet jsou pak popsány na Obr. 16.



Obrázek 16 - Command a response packet k funkci „Position Send Command“

Mohlo by se zdát, že servomotorky nelze otočit o 360° ovšem není tomu tak. K otočení o 360° slouží funkce „360 degrees Rotation Command“ v podstatě se jedná o ekvivalentní funkci k funkci „Position Send Command“. Servomotorky mají pochopitelně více funkcí, ale nemělo by smysl je zde všechny rozvádět.

3.2 Základní deska MGR-BPT232



Obrázek 17 - Základní deska MGR-BPT232

Tab. 4: Popis základní desky MGR-BPT232

Číslo	Funkce
1	Konektor sériového portu CD3pin(M)
2	Integrovaný obvod MAX232
3	Konektor baterií
4	4 konektory k připojení AI MOTOR

Základní deska MGR-BPT232 je určena pro řízení a ovládání skupiny AI-MOTOR 1001. Napájecí napětí, které může být připojeno pomocí trafa, nebo v podobě šesti AA-baterií, se pohybuje v rozmezí 6 až 11V. Na základní desce jsou implementovány 4 konektory k připojení AI-MOTOR 701 a AI-MOTOR 1001, dále 4 konektory k připojení AI-MOTOR 601. Rozdíly mezi jednotlivými verzemi servomotorků jsou v použitém materiálu poháněcího mechanizmu.[5]

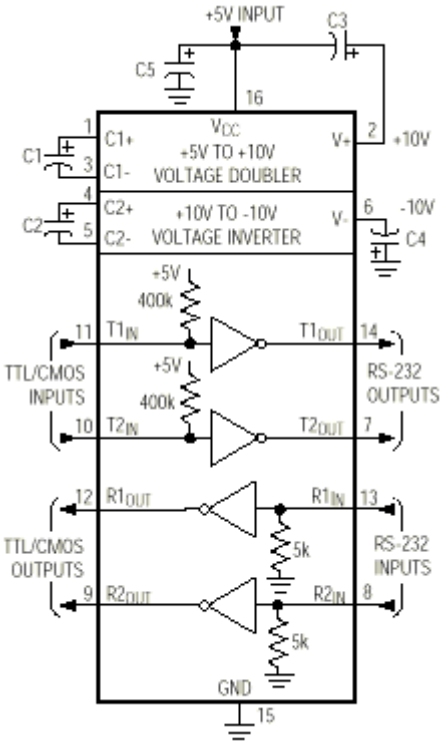
Verze 1001 má poháněcí mechanizmus z ušlechtilého kovu. Jak je z Obr.17 patrné, základní deska využívá nejpoužívanější zapojení RS232, kdy kabel má pouze tři žíly.

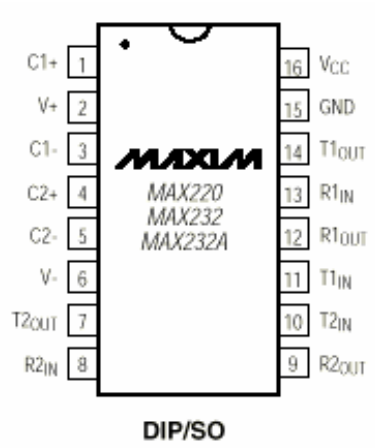
Velkou výhodou je umístění LED diod (RX, TX) na samotné desce. Tyto diody jsou aktivovány vždy při vysílání resp. přijetí signálu, což usnadňuje hledání případné hardwarové či softwarové závady. Jsou vlastně jakýmsi indikátorem správnosti zasílání instrukcí či otevření portu. Dalším plusem je možnost návrhu a samotná realizace uživatelských myšlenek přímo na základní desce.

3.2.1 MAX232

Integrovaný obvod MAX232 je převodníkem TTL na RS232. MAX232 má integrované dvě nábojové pumpy. Výstupní napětí proto značně závisí na použitých kondenzátorech, je tedy nutné použít kondenzátory kvalitní, bohužel u elektrolytických kondenzátorů tato kvalita časem klesá, proto je výhodnější použít tantalové. Z první nábojové pumpy se získává napětí po RS232. Napětí, které je možné získat z druhé pumpy na pinech 2 a 6, a které se pohybuje v rozmezí ±10V, lze použít pro napájení dalších obvodů.

Obvod je pro svou jednoduchost a univerzálnost velmi oblíbený mezi uživateli. Lze jej nebo jeho různé modifikace od firmy MAXIM nalézt téměř ve všech zařízeních připojovaných na RS232.





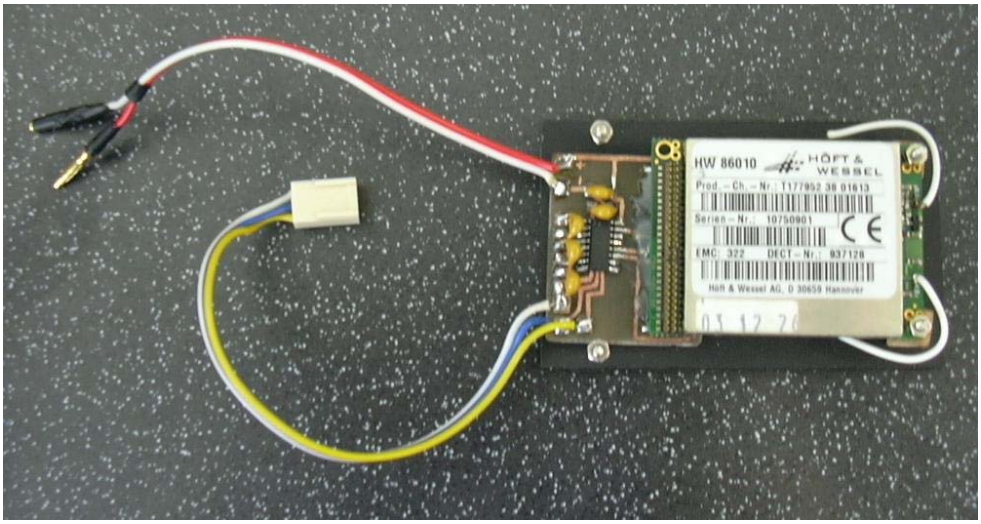
CAPACITANCE (µF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	4.7	4.7	10	10	4.7
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1

3.3 Radiomodul HW86010 a radiomodem HW8612

Pro bezdrátovou komunikaci mezi základní deskou robotického systému MGR-BPT232 a osobním počítačem byly vybrány rádiové modemy HW8612 a radiový modul HW86010, které slouží k přenosu informace vzduchem.

3.3.1 Radiomodul HW86010

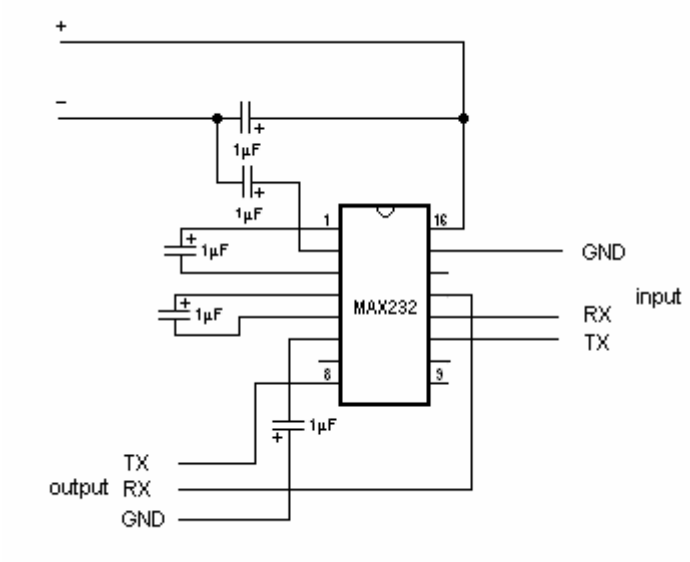
HW86010 je rádiový modul od firmy Hőft & Wessel pracující v DECT pásmu 1880,064 - 1898.208MHz, což je bezlicenční pásmo. Modul obsahuje rozhraní RS232 pro obousměrný přenos dat (přenosová rychlost až 115200 bit/s), PCM rozhraní k připojení standardních ISDN a PBX systémů, I^2C k pomocným funkcím a analogové vstupy a výstupy k přenosu hlasu. Automaticky vyhledává volnou nosnou a volný time slot. Radiofrekvenční výkon je 250mW, možnost využití dvou interních nebo jedné externí antény. Dosah je cca 300m ve volném prostoru, 60m v zástavbě, za jistých okolností až 5km. Napájecí napětí se pohybuje v rozmezí 3,3V až 4,7V (5.5V).



Obrázek 18-Radiomodul HW86010

Rozhraní RS232 používá signály uvedené v Tab.3. Jedná se ovšem o signály pracující na úrovni 3,3V CMOS. Z tohoto důvodu bylo nutné stabilizovat signály přicházející z připojených externích zařízení(MGRT-BPT232, modem, PC atd.), protože se běžně pohybují mezi ±12V, jinak by došlo ke zničení modulu.

Nejjednodušším řešením se ukázalo využití stejného integrovaného obvodu, který je implementován na základní desce robotického systému a popsán v kapitole 4.2.1 tedy MAX232, který tyto vlastnosti beze zbytku splňuje a převádí signály RS232 na úroveň TTL/CMOS.

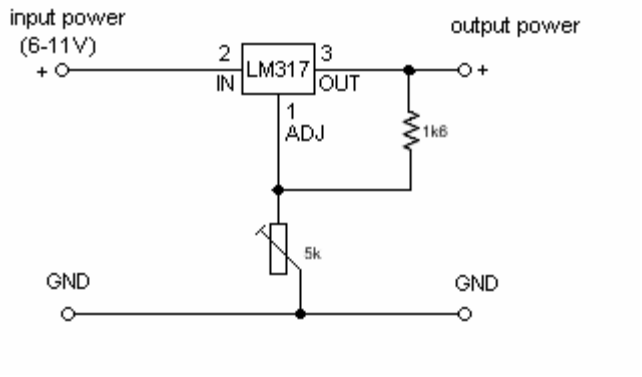


Obrázek 19-Schéma převodníku

Převodník byl přiveden na piny 22(GND), 25(RX), 26(TX) a 28(3,3V) radiového modulu HW86010 a na výstup byl opatřen konektorem CD3pin(F) ke snadnému připojení k základní desce MGR-BPT232.

Vlastnosti základní desky MGR-BPT232 pomohly vyřešit i napájení modulu. Deska má výstupy GND, +5V a input power(je rovno napájecímu napětí přivedenému k základní desce). Jelikož se napájení pohybuje v rozmezí 6-11V (v našem případě 9V), radiový modul však pracuje na 3,3V, bylo nutné toto napětí stabilizovat a přivést k radiomodulu. Proto byl navržen malý stabilizátor napětí, který byl připojen přímo k základní desce.

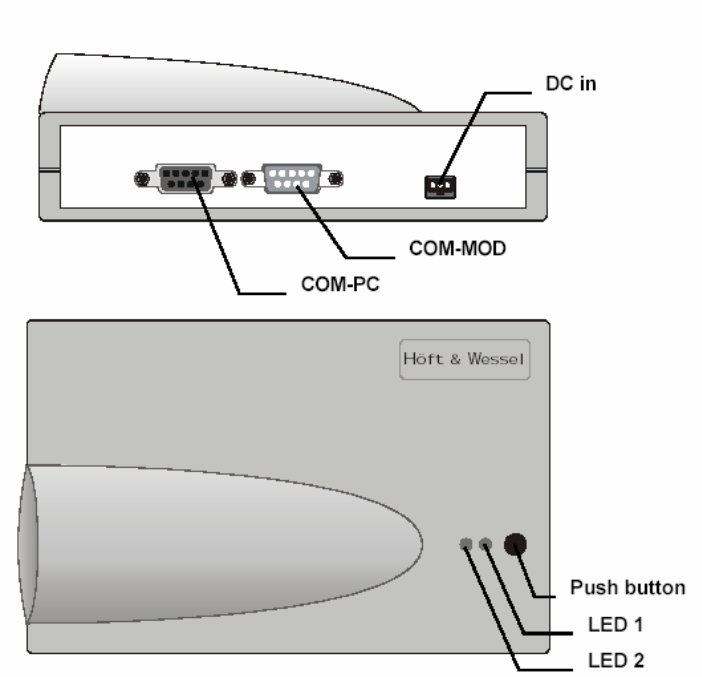
LM317t je integrovaný obvod vhodný pro malé regulovatelné zdroje a stabilizátory napětí. Je velmi vhodný pro stabilizaci pevných napětí. Stabilizátor je opatřen odporovým trimrem 5kΩ pro snadné nastavení výstupního napětí. Schéma zapojení je na Obr. 20.



Obrázek 20-Zapojení stabilizátoru napětí s integrovaným obvodem LM317

3.3.2 Radiomodem HW8612

HW8612 je rádiový modem s rozhraním RS232(V.24), jeho základem je modul HW86010. Tedy i všechny vlastnosti jsou společné. Modem je zakrytován v lehkém plastovém krytu, má vstupy k napájení, vstup ke spojení s PC a výstup k připojení externího zařízení. Jeho výhodou je, že výstupní signály není potřeba převádět na úroveň 3,3V. Proto je možné modem spojit přímo se základní deskou MGR-BPT232. Samotný radiomodem lze samozřejmě použít i v dalších aplikacích například ve spojení s HW8621 jako přístupový bod k Internetu.



Obrázek 21-Radiový modem HW8612

4 NÁVRH A REALIZACE PROGRAMOVÉHO VYBAVENÍ

Na základě uživatelských manuálů AI-MOTOR 1001 a MGR-BPT232 bylo vytvořeno aplikační programové vybavení pro řízení mobilního robota v programovém prostředí Borland C++ Builder.

4.1 Bázový instrukční soubor

4.1.1 Otevření, uzavření a nastavení sériového portu pro komunikaci

Otevření portu v systému Windows či WinCE pro malé přenosné PDA. OpenPort je funkce, která zajistí otevření specifikovaného portu. Funkce otevření je typu public, což znamená, že se jedná o veřejnou položku, která je přístupná z jakékoliv části programu, ve které je viditelná její třída. Parametr PortName je název portu a standardně je to například „COM1“ nebo „COM2“.

```
DWORD TBase::OpenPort(const char *PortName)
{
   DWORD Result = 0;
   ComFileHandle = CreateFile(PortName, GENERIC_READ | GENERIC_WRITE,
                              0, 0, OPEN_EXISTING, 0, 0);

   if(ComFileHandle == INVALID_HANDLE_VALUE)
      {
      Result = GetLastError();
      CloseHandle(ComFileHandle);
      return Result;
      }
   if(!SetCom())
      {
      return 0;
      }
   return Result;
```

Po ukončení komunikace je třeba proměnnou ComFileHandle znovu uvolnit pomocí příkazu CloseHandle.

```
DWORD TBase::ClosePort(const char *PortName)
{
   DWORD Result = 0;
   CloseHandle(ComFileHandle);
      Result = GetLastError();
      return Result;
}
```

Pokud požadujeme jiné než standardní nastavení ať už přenosové rychlosti či velikosti bytu je potřebné použít funkce DCB. Na rozdíl od funkce OpenPort je funkce SetCom, která zajistí nastavení parametrů portu (přenosová rychlost, atd.), funkce typu private, což jsou soukromé atributy a metody, které nejsou přístupné vně třídy.

```
DWORD TBase::SetCom()
{
    DCB ControlBlock;
    DWORD Result = 0;

    if(GetCommState(ComFileHandle, &ControlBlock)){
       Result = GetLastError();
    }
    ControlBlock.BaudRate = CBR_57600 ;
    ControlBlock.ByteSize = 8;
    ControlBlock.Parity = NOPARITY;
    ControlBlock.StopBits = ONESTOPBIT;

    ControlBlock.fOutxCtsFlow = false;
    ControlBlock.fOutxDsrFlow = false;
    ControlBlock.fDsrSensitivity = false;
    ControlBlock.fAbortOnError = false;
    ControlBlock.fDtrControl = DTR_CONTROL_DISABLE;
    ControlBlock.fRtsControl = RTS_CONTROL_DISABLE;
    ControlBlock.fOutX = false;
    ControlBlock.fInX = false;

    if(SetCommState(ComFileHandle, &ControlBlock)){
       Result = GetLastError();
       }
    Result = SetTimeOut(true);
    return Result;
}
```

Dále je potřebné nastavit parametr TimeOut, což je maximální čas, kdy program čeká na odpověď při zaslání instrukce na základní desku.

//DefTimeOut == 1 ---> TIMEOUT = 35ms

//DefTimeOut == 0 ---> TIMEOUT = 155ms

Je zřejmé, že jsou dva druhy konstant TimeOutu. Jeden se používá při zasílání řídicích instrukcí, které ovládají činnost motorů (Command Operations), zatímco druhý při zasílání tzv. nastavovacích instrukcí (Set Operations), které nastavují vnitřní parametry motorů, jako je jejich identifikační číslo, přenosová rychlost, prahové přepětí apod.

4.1.2 Funkce pro nastavení a ovládání AI-MOTORů

Bázový soubor obsahuje deklarace funkcí pro zasílání dat na port a čtení dat z portu, které jsou reprezentovány funkcemi *ComOperation* a *SetOperation*. Tyto funkce zašlou instrukci na základní desku AI-MORORů a přečtou odpověď. Rozdíl mezi nimi je pouze ve velikosti zaslaných dat (u funkce Command Operations jsou to 4 byty, u funkce Set Operations 6

bytů) a velikosti parametru TimeOut. Zbytek bázového souboru tvoří deklarace funkcí pro řízení a nastavení motorů např.

```
DWORD TBase::PosRead(unsigned char ServoID, unsigned char *Response)
{
   unsigned char Data[4];
   DWORD Result = 0;

   Data[0]=HEADER;
   Data[1]=0xa0|ServoID;
   Data[2]=NULL;
   Data[3]=(Data[1]^Data[2])&0x7f;

   Result = ComOperation(&Data[0], Response);
   return Result;
}
```

Tato funkce slouží ke zjištění aktuální pozice motoru. Má pouze jeden určující parametr a to ServoID neboli identifikační číslo motoru, které se pohybuje v rozmezí 0 – 30, neboť to je maximální počet motorů, které je možno zároveň připojit na základní desku. Do parametru Response se uloží odpověď z motoru (v tomto případě jeho aktuální pozice). Základem je nastavení dat (Data[0] – Data[3], respektive Data[0] – Data[5]), kde HEADER je číslo sloužící desce jako informace, že přijde instrukce. Data[1] obsahují vždy pořadové číslo instrukce (0xa0). Data[2] jsou pomocné parametry tam, kde je to nutné (např. rychlost otáčení), a poslední byte je určen pro kontrolní součet.

Uvedené funkce se liší pouze pořadovým číslem instrukce a použitými parametry, takže není zapotřebí je zde všechny vypisovat.

4.2 Aplikace AI_Manager

Aplikace AI_Manager slouží především pro zadávání trajektorií, podle nichž se bude robot pohybovat. Zadání úlohy spočívá v navigaci robota po předepsané trajektorii. Základní funkcí programu je tedy proložení křivky trajektorie několika body. Uživatel tedy zadá souřadnice počátečního a koncového bodu, určí počet prokládaných bodů a též zadá jejich

souřadnice. Program pak proloží tyto body křivkou dle výběru (polynom,…), podle níž se bude robot pohybovat. Další možností je zadávat souřadnice bodů zlomu trajektorie, podle kterých se bude robot pohybovat přímo, tj. po přímkách. Výhodou systému, kde má každé kolo vlastní ovládání je, že robot zvládá i 180ti stupňové obraty na místě. Další a poslední možností zadání je možnost nechat robota vykonávat pohyb po uzavřené křivce, například kružnici o rovnicí $ax^2+by^2=r^2$ s volitelnými parametry a,b,r.

4.2.1 Proložení bodů křívkou

Proložení bodů určitou křivkou je klasickou úlohou interpolační aproximace funkce. Mějme funkci f(x) a aproximační funkci φ(x). Interpolační aproximace je definována, tak že funkce f(x) a φ(x) se na konečném intervalu musí shodovat v konečném počtu uzlových bodů $x_{i,}$ i=1,2,…,n

Aproximace pomocí funkcionálních řad

$\varphi(x)=a_0\mu_0(x)+ a_1\mu_1(x)+ a_2\mu_2(x)+\ldots+ a_n\mu_n(x)$

Funkci μ(x) se pak říká bazická funkce a tvoří základ (bázi) aproximace.

Nejčastějšími bazickými funkcemi jsou např. polynomická –

$1,x,x^2, x^3, x^4,\ldots$

nebo s použitím goniometrických funkcí –

1,sin(x),cos(x),sin(2x),cos(2x),sin(3x),cos(3x),…

a platí, že $x \epsilon <a,b>$, kde $x_0=a$, kde $x_n=b$

Uzly na tomto intervalu nemusí být ekvidistantně rozloženy.

$a_0\mu_0(x_0)+ a_1\mu_1(x_0)+ a_2\mu_2(x_0)+\ldots+ a_n\mu_n(x_0)=f(x_0)$

$a_0\mu_0(x_1)+ a_1\mu_1(x_1)+ a_2\mu_2(x_1)+\ldots+ a_n\mu_n(x_1)=f(x_1)$

…

$\mu_i(x_j)=\mu_{ij}$, $f(x_j)=f_j$

Vytvoříme matici U

$$U = \begin{pmatrix} u_{00} & u_{10} & \dots & u_{n0} \\ u_{01} & u_{11} & \dots & u_{n1} \\ \dots & & & \\ u_{0n} & u_{1n} & \dots & u_{nn} \end{pmatrix},$$

a dále vektor funkčních hodnot $F^T = (f_0 \quad f_1 \quad \dots \quad f_n)$.

Hledáme vektor koeficientů $A^T = (a_0 \quad a_1 \quad \dots \quad a_n)$,

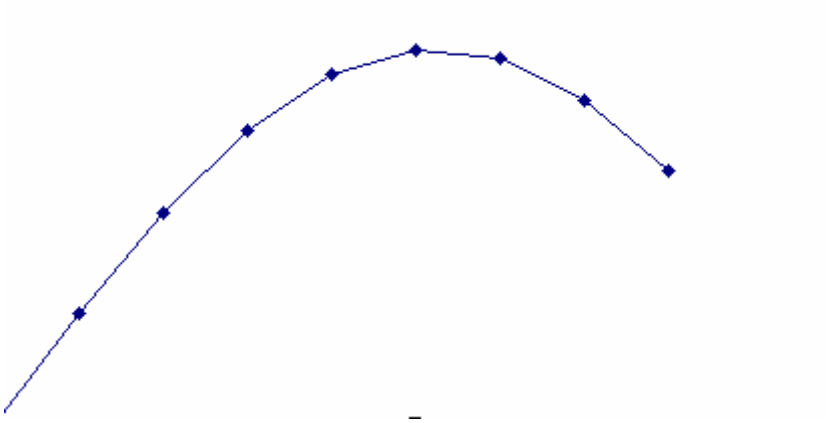
Platí, že $UA = F \Rightarrow A = U^{-1}F$

Samozřejmě existují také jiné metody pro výpočet koeficientů jako jsou například různé druhy numerických iteračních metod pro řešení soustavy lineárních rovnic.

4.2.2 Algoritmus pro výpočet změny směru

Zde bude uveden algoritmus pro výpočet změny směru robota v závislosti na jeho aktuální pozici.

Po zadání bodů a výpočtu křivky program rozdělí křivku na určitý počet částí (vzdálenost mezi jednotlivými body lze nastavit, jako postačující se ukázalo 10cm, jako základní nastavení). Plochu, na níž se robot pohybuje si je možno představit jako pravoúhlý dvourozměrný souřadnicový systém, takže existuje vektor bodů P[x,y]. Souřadnicový systém je pevný, mění se pozice a natočení robota, který jezdí tak, aby co nejpřesněji projel všemi body.



Program si uchovává informaci o úhlu aktuálního natočení. Změna úhlu je dána rozdílem mezi novým úhlem a aktuálním natočením. Všechny úhly jsou absolutní vzhledem k souřadnému systému, přičemž 1. a 2. kvadrant je v rozmezí 0 - 180°a 3. a 4. kvadrant v rozmezí 0 - -180°. Každý nový úhel je vypočten jednoduše jako α=ArcTan(y/x) a přiřazen do příslušného kvadrantu. Z porovnání takto vypočtených nových a starých úhlů lze také zjistit směr zatočení (doleva, doprava).

4.2.3 Princip pohybu robota

Robot je určen do laboratorních podmínek, za předpokladu, že se bude pohybovat v daném prostředí po stabilním povrchu. Kolečka. Jako součást vybavení AI motorů dodaných firmou Megarobotics jsou z umělé hmoty. To mělo za následek jejich možné prokluzování a tím značnou závislost na povrchu. Minimalizaci této závislosti byla dosažena osazením všech kol s gumovou „pneumatikou“. Přesto zde závislost zůstává a je nutno před každou změnou povrchu robota „kalibrovat“ pro daný povrch.

Řízení pohybu vpřed je to jednoduché. Program obsahuje proměnnou FORWARD, což je vlastně koeficient udávající, za jak dlouho robot při určité rychlosti urazí jeden metr v milisekundách. Toto je jeden z parametrů, které, jak bylo uvedeno, je nutno před každou změnou povrchu změřit a nastavit. Změny by neměly být nijak výrazné při použití klasických hladkých povrchů (lino, parkety,…), přesto však doporučuji toto udělat. Tento koeficient se tedy vynásobí vzdáleností vypočtenou jako rozdíl dvou sousedních bodů křivky.

Pohyb zpět se u pohybu po trajektorii neuvažuje, přesto však aplikace obsahuje možnost ručního ovládání (něco jako auto na dálkové ovládání), kde je zařazen i pohyb zpět.

Zatáčení je poněkud složitější. Protože po rozčlenění křivky na krátké diskrétní úseky se úhly, o které se má robot otočit, pohybují nejvíce v rozmezí 1-10°, bylo nutné se zaměřit především na tyto malé úhly. Přesto je obtížné bez jakékoliv zpětné vazby říci, že se robot po ujetí dané vzdálenosti otočil opravdu jen o 1° nebo o 2°.

Algoritmus obsahuje dva druhy funkcí pro zatáčení, nazvané Left, EasyLeft (resp. Right, EasyRight). Jak je z názvu patrné, jedná se o způsob zatáčení pro velké úhly a malé úhly. Protože plynulé zatočení o velký úhel na velmi krátkém úseku už někdy není možné, je řešena možnost otočení robota na místě. To funguje tak, že se jedna strana otáčí dopředu a

druhá stejnou rychlostí opačně. Takto je robot schopen otočit se o 360° a zůstat přitom na místě. Takto také funguje funkce Left (Right). Stejně jako je FORWARD, existuje i parametr LEFT (RIGHT), který udává, za jak dlouho se robot otočí na tu kterou stranu o 90°. Tento parametr je také vhodné před změnou povrchu změřit a nastavit. Stejně jako v předchozím případě, tento parametr je vynásoben skutečným úhlem natočení poděleným 90. Robot se tedy otočí na místě a k dalšímu bodu jede po přímce.

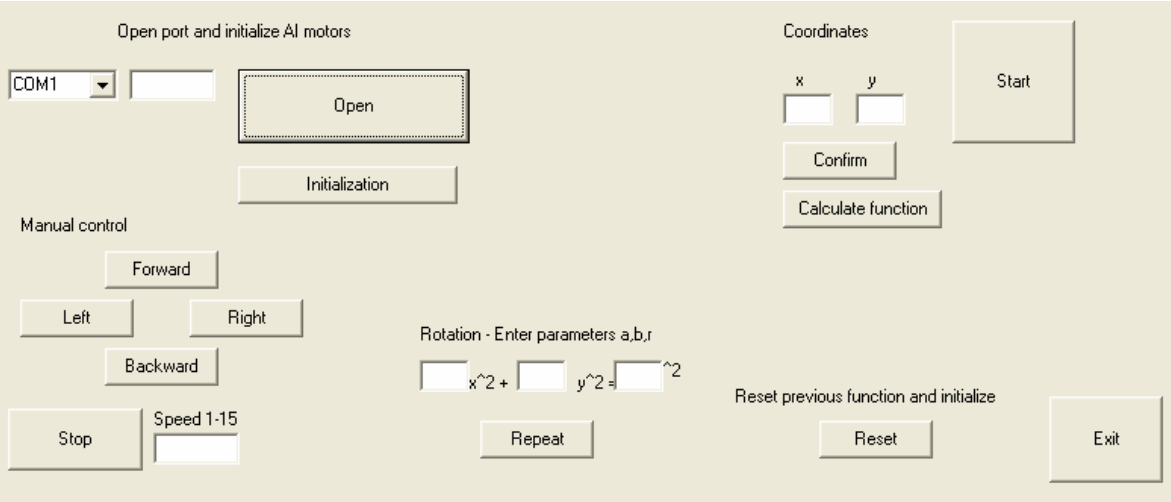
Funkce EasyLeft (EasyRight) je pro velmi malé úhly <10°. Veškeré konstanty a parametry v ní jsou změřeny a nastaveny experimentálně. Motory AI umožňují rotaci kola v 15-ti stupních rychlosti. Narozdíl od předchozího způsobu se obě strany neotáčejí různým, ale stejným směrem, pouze je u jedné strany snížen rychlostní stupeň rotace. To ale není postačující, protože počet stupňů rychlosti není dostatečný pro pokrytí všech úhlů. Proto je zde také jako v předchozím případě zaveden časový parametr vycházející ze stejného parametru LEFT (RIGHT), který opět udává dobu otáčení. Zbytek vzdálenosti k novému bodu je překonám po přímkové trajektorii. Vzhledem k tomu, že úhly otáčení jsou velmi malé a vzdálenosti mezi jednotlivými body trajektorie jsou poměrně krátké, vytváří se představa, že se jedná o plynulý pohyb.

4.2.4 Běh programu

V programu jsou použity tři časovače. Jeden pro časovou segmentaci přímého pohybu, druhý pro časovou segmentaci zatáčení a třetí pro časovou segmentaci jemného zatáčení.

Po stisknutí tlačítka Start se spustí funkce, která vyhodnotí pozici, natočení, nový úhel a zavolá příslušnou funkci pro pohyb (Forward, Left, Right, EasyLeft, EasyRight). Během provádění těchto funkcí je spuštěn příslušný časovač. Po jejím ukončení byl uzavřen první cyklus programu, z vektoru úhlů se přečte další a opět je zavolána funkce pro vyhodnocení natočení a nového směru. Po dosažení posledního bodu robot zastaví (kromě případu krouživého pohybu, kdy je jako další bod přiřazen opět první) a jakoby se dostal zase na první souřadnici, tzn. že po opětovném stisku „Start“ se bude pohybovat znovu po stejné trajektorii.

4.2.5 Popis aplikace AI_Manager



Obrázek 22-Základní okno aplikace řízení trajektorie mobilního robota

Na obrázku je základní okno aplikace AI_Manager, které se objeví po spuštění. Ovládání je následné:

Nejprve se otevře sériový port pro komunikaci. K tomu slouží tlačítko „Open“ po vybrání příslušného portu (COM1,COM2,…). Tím se samočinná nastaví příslušné parametry pro správnou komunikaci. Opětovným kliknutím na toto tlačítko se sériový port uzavře a uvolní pro jiné programy.

Dále je potřebné motory inicializovat. Po stisknutí tlačítka „Initialization“ se všechny připojené AI motory resetují a připraví pro příjem instrukcí.

Robot lze ovládat třemi způsoby. První způsob s názvem Manual control slouží pro ruční ovládání , robot se bude chovat jako auto na dálkové ovládání. Stačí zadat stupeň rychlosti.

Druhý oddíl s názvem Rotation je pro zadání krouživého pohybu po elipse (kružnici). Stačí zadat parametry a,b,r v rovnici $ax^2 + by^2 = r^2$, stisknout „Repeat“ pro výpočet trajektorie a následně „Start“.

Poslední oddíl je určen pro zadání bodů. Do oken s názvy x,y se zadávají souřadnice bodů trajektorie robota od prvního do posledního. Zadání každého bodu se potvrdí stiskem tlačítka „Confirm“. Vložené souřadnice se objeví v tabulce ve spodní části okna aplikace. Stiskem tlačítka „Calculate function“ po vložení posledního bodu program proloží zadanými body polynomiální křivkou. Stiskem „Start“ se následně robot uvede do chodu.

Robot se umí pohybovat po přímkách a též se dokáže otáčet o libovolně velké úhly na místě. V případě zadání pohybu robota např. po obvodu čtverce, zadají se souřadnice vrcholů jako v předchozím případě, ale po zadání posledního bodu se stiskne rovnou tlačítko „Start“.

„Reset“ – po stisknutí tohoto tlačítka se vymažou všechny zadané body, vymaže se každá zadaná trajektorie a program je připraven pro zadání nových bodů.

ZÁVĚR

V diplomové práci byl uskutečněn návrh, sestavení, vývoj programového vybavení a oživení mobilního robotického systému s časovým inkrementálním řízením pro každé kolo zvlášť. Základními moduly tohoto robotického systému jsou základní deska MGR-BPT232 a servomotory AI-MOTOR 1001.

Na přenos řídících instrukcí z osobního počítače k základní desce systému bylo využito třížilového sériového kabelu s konektorem Cannon9, který je též připojitelný na sériový port osobního počítače. K tomu, aby byl celý systém mobilní a použitelný i v místech, která nejsou vybavena síťovým napájením, byla implementována bezdrátová komunikace se základní deskou. Tento přístup umožňuje použití konfigurací radiového modemu HW8612, komunikujících v nelicencovaném pásmu 1880-1900MHz. Je zřejmé, že připojení jak sériového kabelu nebo velkého radiového modemu by omezovalo mobilitu robotického systému, proto byl implementován i radiový modul HW86010, jehož malé rozměry umožňují montáž sestavy na malý mobilní robot. Modul stejně jako modemy potřebuje vlastní napájení, a signály přicházející ze základní desky jsou vyšší napěťové úrovně, než je modul schopen bez poškození snést, proto bylo nutné i tyto signály upravit. Převod řídících signálů na požadovanou úroveň TTL/CMOS zajišťuje připojený integrovaný obvod MAX232. Napájení radiového modulu bylo vyřešeno pomocí desky MGR-BPT232, která má připojení GND, input power (hodnota výstupního napětí je rovna napájecímu). Bylo provedeno přizpůsobení připojení pomocí napěťového stabilizátoru, tvořeného integrovaným obvodem LM317 a odporovým trimrem pro nastavení hodnoty napájecího napětí.

Programové vybavení tvořené souborem instrukcí pro komunikaci mezi PC a robotem a vlastním algoritmem pro řízení robota je vytvořeno v programovém prostředí Borland C++ Builder. Diplomová práce je vypracována tak, aby ji bylo možno použít jako učební pomůcku k laboratorním pracím z předmětů Základy robotiky a Elektronické zabezpečovací systémy II.

SEZNAM POUŽITÉ LITERATURY

[1] KADLEC, Václav: Učíme se programovat v Borland C++ Builder a jazyce C++ 2.vyd. Brno: Computer Press, 2004. 387 s. ISBN 80-7226-550-4.

[2] DUMEK, V., ROUPEC, J.: Programování v jazyku C 1.vyd. Brno: VUT v Brně, 1992

[3] LIBERTY, Jesse: Naučte se C++ za 21 dní 2.vyd. Praha: Nakladatelství Computer Press, a.s., 2002. 792 s. ISBN

[4] MEGAROBOT: AI MOTOR-1001 MANUAL [online]. [cit.:neuvedeno] dostupný z WWW <http://www.megarobot.net/manual_download/AIMotor701_manual.pdf>

[5] MEGAROBOT: MGR-BPT232 MANUAL [online]. [cit.:neuvedeno] dostupný z WWW http://megarobot.net/manual_download/MGR-BPT232_manual.pdf

[6] HAVEL, Ivan M.: Robotika-úvod do teorie kognitivních robotů, 1.vyd. Praha 1980, 280s.

[7] OHAREK, Petr: Ovládání internetové komunikační kamery se dvěma stupni volnosti, 2006, 77s

[8] KVASNICA, M: Assistive Technologies for Man-Machine Interface and Applications in Education and Robotics. International Journal of Human-Friendly Welfare Robotic Systems, October 2005, Volume 6, Number 3, KAIST Press, Daejeon, Korea. ISSN 1598-3150

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

UTB	Univerzita Tomáše Bati ve Zlíně
Bd	Baud – jednotka pro měření rychlosti přenesených dat
SW	Software
TTL	Transistor-Transistor Logic – jeden z typů integrovaných obvodů
CSMA/CA	Carrier-sense, Multiple-Access, Collision Avoidance
PAN	Personal Area Network
DECT	Digital Enhanced Cordless Telecommunications
PBX	Private Branch Exchange
CMOS	Complementary Metal-Oxide-Semiconductor
LED	Light Emitting Diode
Bd	Baud – jednotka pro měření rychlosti přenesených dat
SW	Software
TTL	Transistor-Transistor Logic – jeden z typů integrovaných obvodů
CSMA/CA	Carrier-sense, Multiple-Access, Collision Avoidance

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: KNIHOVNA ŘÍDICÍCH INSTRUKCÍ

```
TBase::TBase()
{
      ctr=0;pom=0,Angle=90;
}
DWORD TBase::OpenPort(const char *PortName)
{
  DWORD Result = 0;
  ComFileHandle = CreateFile(PortName, GENERIC_READ | GENERIC_WRITE,
                        0, 0, OPEN_EXISTING, 0, 0);
  if(ComFileHandle == INVALID_HANDLE_VALUE)
    {
    Result = GetLastError();
    CloseHandle(ComFileHandle);
    return Result;
    }
  if(!SetCom())
    {
    return 0;
    }
  return Result;
}
DWORD TBase::ClosePort(const char *PortName)
{
  DWORD Result = 0;
  CloseHandle(ComFileHandle);
    Result = GetLastError();
    return Result;
}
//bool TBase::SetCom(DWORD BaudRate, DWORD Parity, THandshake Handshake)
DWORD TBase::SetCom()
{
   DCB ControlBlock;
   DWORD Result = 0;
   if(GetCommState(ComFileHandle, &ControlBlock)){
     Result = GetLastError();
   }
   ControlBlock.BaudRate = CBR_57600 ;
   ControlBlock.ByteSize = 8;
   ControlBlock.Parity = NOPARITY;
   ControlBlock.StopBits = ONESTOPBIT;
   ControlBlock.fOutxCtsFlow = false;
   ControlBlock.fOutxDsrFlow = false;
   ControlBlock.fDsrSensitivity = false;
   ControlBlock.fAbortOnError = false;
   ControlBlock.fDtrControl = DTR_CONTROL_DISABLE;
   ControlBlock.fRtsControl = RTS_CONTROL_DISABLE;
   ControlBlock.fOutX = false;
   ControlBlock.fInX = false;
   if(SetCommState(ComFileHandle, &ControlBlock)){
     Result = GetLastError();
   }
   Result = SetTimeOut(true);
   return Result;
}
//DefTimeOut == 1  ---> TIMEOUT = 35ms
//DefTimeOut == 0  ---> TIMEOUT = 155ms
```

```
DWORD TBase::SetTimeOut(bool TimeOut)
{
   COMMTIMEOUTS CommTimeouts;
   DWORD Result = 0;
   if(TimeOut){
     CommTimeouts.ReadIntervalTimeout = 0;   //MAXDWORD
     CommTimeouts.ReadTotalTimeoutMultiplier = 0;
     CommTimeouts.ReadTotalTimeoutConstant = 35;
     CommTimeouts.WriteTotalTimeoutMultiplier = 0;
     CommTimeouts.WriteTotalTimeoutConstant =35;
     if (SetCommTimeouts(ComFileHandle, &CommTimeouts)){
       Result = GetLastError();
       return Result;
       }
   }else{
     CommTimeouts.ReadIntervalTimeout = 0;
     CommTimeouts.ReadTotalTimeoutMultiplier = 0;
     CommTimeouts.ReadTotalTimeoutConstant = 155;
     CommTimeouts.WriteTotalTimeoutMultiplier = 0;
     CommTimeouts.WriteTotalTimeoutConstant = 155;
     if (SetCommTimeouts(ComFileHandle, &CommTimeouts)){
       Result = GetLastError();
       return Result;
       }
   }
   return Result;
}
/*
bool TBase::WriteData()
{
  DWORD Result;
  unsigned char Response[2];
  Result = PowerDown(&Response[0]);
  Form1->RichEdit1->Lines->Add("Result: " + IntToStr(Result));
  return 1;
} */
/*! \fn DWORD TSerialPort::ComOperation(unsigned char Data[4], unsigned char Response)
 *  \brief Vykoná požadovanou operaci.
 *
 *  Odešle 4 byty specifikující požadovanou operaci na seriový port a
 *  očekává odpověď, kterou uloží na místo odkzované v \a Response.
 *  \param Data[4] Požadovaná operace, viz. dokumentace "AI MOTOR-1001".
 *  \param *Response Ukazatel na odpověď z AI motoru.
 *  \return Vrací chybový kód. Tyto chyby jsou definované na adrse:
 *  http://msdn.microsoft.com/library/default.asp?url=/library/en-us/debug/base/system_error_codes.asp
 */
DWORD TBase::ComOperation(unsigned char Data[4], unsigned char *Response)
{
  DWORD BytesSend, BytesRead, Result = 0;
  PurgeComm(ComFileHandle, PURGE_TXCLEAR);
  if(!WriteFile(ComFileHandle, Data, 4, &BytesSend, 0)){
    Result = GetLastError();
    return Result;
  }
  PurgeComm(ComFileHandle, PURGE_RXCLEAR);
  if(!ReadFile(ComFileHandle, Response, 2, &BytesRead, 0)){
    Result = GetLastError();
    return Result;
  }
```

```
  return Result;
}
/*! \fn DWORD TSerialPort::SetOperation(unsigned char Data[6], unsigned char *Response)
 *  \brief Vykoná požadovanou operaci.
 *
 *  Odešle 6 bytů specifikujících požadovanou operaci na seriový port a
 *  očekává odpověď, kterou uloží na místo odkzované v \a Response.
 *  \param Data[4] Požadovaná operace, viz. dokumentace "AI MOTOR-1001".
 *  \param *Response Ukazatel na odpověď z AI motoru.
 *  \return Vrací chybový kód. Tyto chyby jsou definované na adrse:
 *  http://msdn.microsoft.com/library/default.asp?url=/library/en-us/debug/base/system_error_codes.asp
 */
DWORD TBase::SetOperation(unsigned char Data[6], unsigned char *Response)
{
  DWORD BytesSend, BytesRead, Result = 0;
  SetTimeOut(false);
  PurgeComm(ComFileHandle, PURGE_TXCLEAR);
  if(!WriteFile(ComFileHandle, Data, 6, &BytesSend, 0)){
    Result = GetLastError();
    SetTimeOut(true);
    return Result;
  }
  PurgeComm(ComFileHandle, PURGE_RXCLEAR);
  if(!ReadFile(ComFileHandle, Response, 2, &BytesRead, 0)){
    Result = GetLastError();
    SetTimeOut(true);
    return Result;
  }
  SetTimeOut(true);

  return Result;
}
/////////Function declaration start/////////////////////////////////////////////
/*! \fn DWORD TSerialPort::PosSend(char ServoID, char SpeedLevel, char Position, unsigned char
*Response)
 *  \brief
 *  \param *Response Ukazatel na odpověď z AI motoru.
 *  \return Vrací chybový kód.
 */
DWORD TBase::PosSend(unsigned char ServoID, unsigned char SpeedLevel,
                     unsigned char Position, unsigned char *Response)
{
  unsigned char Data[4];
  DWORD Result = 0;
  Data[0] = HEADER;
  Data[1] = SpeedLevel<<5|ServoID;
  Data[2] = Position;
  Data[3] = (Data[1]^Data[2])&0x7f;
  Result = ComOperation(&Data[0], Response);
  return Result;
}
/*! \fn DWORD TSerialPort::PosSend(char ServoID, char SpeedLevel, char Position, unsigned char
*Response)
 *  \brief
 *  \param *Response Ukazatel na odpověď z AI motoru.
 *  \return Vrací chybový kód.
 */
DWORD TBase::PosRead(unsigned char ServoID, unsigned char *Response)
{
```

```
  unsigned char Data[4];
  DWORD Result = 0;
  Data[0]=HEADER;
  Data[1]=0xa0|ServoID;
  Data[2]=NULL;
  Data[3]=(Data[1]^Data[2])&0x7f;
  Result = ComOperation(&Data[0], Response);
  return Result;
}
/*! \fn DWORD TSerialPort::ActDown(unsigned char ServoID, unsigned char *Response)
 *  \brief
 *  \param *Response Ukazatel na odpověď z AI motoru.
 *  \return Vrací chybový kód.
 */
DWORD TBase::ActDown(unsigned char ServoID, unsigned char *Response)
{
  unsigned char Data[4];
  DWORD Result = 0;
  Data[0]=HEADER;
  Data[1]=0xc0|ServoID;
  Data[2]=0x10;
  Data[3]=(Data[1]^Data[2])&0x7f;
  Result = ComOperation(&Data[0], Response);
  return Result;
}
/*! \fn DWORD TSerialPort::PowerDown(unsigned char *Response)
 *  \brief
 *  \param *Response Ukazatel na odpověď z AI motoru.
 *  \return Vrací chybový kód.
 */
DWORD TBase::PowerDown(unsigned char *Response)
{
  unsigned char Data[4];
  DWORD Result = 0;
  Data[0]=HEADER;
  Data[1]=0xdf;
  Data[2]=0x20;
  Data[3]=(Data[1]^Data[2])&0x7f;
  Result = ComOperation(&Data[0], Response);
  return Result;
}
/*! \fn DWORD TSerialPort::Rotation360(unsigned char ServoID, unsigned char SpeedLevel,
                                       unsigned char RotationDir, unsigned char *Response)
 *  \brief
 *  \param *Response Ukazatel na odpověď z AI motoru.
 *  \return Vrací chybový kód.
 */
DWORD TBase::Rotation360(unsigned char ServoID, unsigned char SpeedLevel,
                         unsigned char RotationDir, unsigned char *Response)
{
  unsigned char Data[4];
  DWORD Result = 0;
  Data[0]=HEADER;
  Data[1]=6<<5|ServoID;
  if(RotationDir==ROTATE_CCW)
    { Data[2]=ROTATE_CCW<<4|SpeedLevel; }
  else
    { Data[2]=ROTATE_CW<<4|SpeedLevel; }
  Data[3]=(Data[1]^Data[2])&0x7f;
```

```
  Result = ComOperation(&Data[0], Response);
  return Result;
}
/*! \fn DWORD TSerialPort::BaudrateSet(unsigned char ServoID, unsigned char NewBaud,
                                       unsigned char *Response)
 *  \brief
 *  \param *Response Ukazatel na odpověď z AI motoru.
 *  \return Vrací chybový kód.
 */
DWORD TBase::BaudrateSet(unsigned char ServoID, unsigned char NewBaud,
                         unsigned char *Response)
{
  unsigned char Data[6];
  DWORD Result = 0;
  Data[0]=HEADER;
  Data[1]=7<<5|ServoID;
  Data[2]=0x08;
  Data[3]=NewBaud;
  Data[4]=NewBaud;
  Data[5]=(Data[1]^Data[2]^Data[3]^Data[4])&0x7f;
  Result = SetOperation(&Data[0], Response);
  return Result;
}
/*! \fn DWORD TSerialPort::GainSet(unsigned char ServoID, unsigned char *NewPGain,
                                   unsigned char *NewDGain, unsigned char *Response)
 *  \brief
 *  \param *Response Ukazatel na odpověď z AI motoru.
 *  \return Vrací chybový kód.
 */
DWORD TBase::GainSet(unsigned char ServoID, unsigned char *NewPGain,
                     unsigned char *NewDGain, unsigned char *Response)
{
  unsigned char Data[6];
  DWORD Result = 0;
  Data[0]=HEADER;
  Data[1]=7<<5|ServoID;
  Data[2]=0x09;
  Data[3]=*NewPGain;
  Data[4]=*NewDGain;
  Data[5]=(Data[1]^Data[2]^Data[3]^Data[4])&0x7f;
  Result = SetOperation(&Data[0], Response);
  return Result;
}
/*! \fn DWORD TSerialPort::IDSet(unsigned char ServoID, unsigned char NewID, unsigned char
*Response)
 *  \brief
 *  \param *Response Ukazatel na odpověď z AI motoru.
 *  \return Vrací chybový kód.
 */
DWORD TBase::IDSet(unsigned char ServoID, unsigned char NewID, unsigned char *Response)
{
  unsigned char Data[6];
  DWORD Result = 0;
  Data[0]=HEADER;
  Data[1]=7<<5|ServoID;
  Data[2]=0x0a;
  Data[3]=NewID;
  Data[4]=NewID;
  Data[5]=(Data[1]^Data[2]^Data[3]^Data[4])&0x7f;
```

```
  Result = SetOperation(&Data[0], Response);
  return Result;
}
/*! \fn DWORD TSerialPort::GainRead(unsigned char ServoID, unsigned char *PGain,
                                    unsigned char *DGain, unsigned char *Response)
 *  \brief
 *  \param *Response Ukazatel na odpověď z AI motoru.
 *  \return Vrací chybový kód.
 */
DWORD TBase::GainRead(unsigned char ServoID, unsigned char *PGain,
                      unsigned char *DGain, unsigned char *Response)
{
  unsigned char Data[6];
  DWORD Result = 0;
  Data[0]=HEADER;
  Data[1]=7<<5|ServoID;
  Data[2]=0x0c;
  Data[3]=NULL;
  Data[4]=NULL;
  Data[5]=(Data[1]^Data[2]^Data[3]^Data[4])&0x7f;
  Result = SetOperation(&Data[0], Response);
  return Result;
}
/*! \fn DWORD TSerialPort::ResolSet(unsigned char ServoID, unsigned char NewResol,
                                    unsigned char *Response)
 *  \brief
 *  \param *Response Ukazatel na odpověď z AI motoru.
 *  \return Vrací chybový kód.
 */
DWORD TBase::ResolSet(unsigned char ServoID, unsigned char NewResol,
                      unsigned char *Response)
{
  unsigned char Data[6];
  DWORD Result = 0;
  Data[0]=HEADER;
  Data[1]=7<<5|ServoID;
  Data[2]=0x0d;
  Data[3]=NewResol;
  Data[4]=NewResol;
  Data[5]=(Data[1]^Data[2]^Data[3]^Data[4])&0x7f;
  Result = SetOperation(&Data[0], Response);
  return Result;
}
DWORD TBase::BoundSet(unsigned char ServoID, unsigned char *NewLBound,
                      unsigned char *NewUBound, unsigned char *Response)
{
  unsigned char Data[6];
  DWORD Result = 0;
  Data[0]=HEADER;
  Data[1]=7<<5|ServoID;
  Data[2]=0x11;
  Data[3]=*NewLBound;
  Data[4]=*NewUBound;
  Data[5]=(Data[1]^Data[2]^Data[3]^Data[4])&0x7f;
  Result = SetOperation(&Data[0], Response);
  return Result;
}
/*! \fn DWORD TSerialPort::BoundRead(unsigned char servoID, unsigned char *LBound,
                                     unsigned char *UBound, unsigned char *Response)
```

```
 *  \brief
 *  \param *Response Ukazatel na odpověď z AI motoru.
 *  \return Vrací chybový kód.
 */
DWORD TBase::BoundRead(unsigned char ServoID, unsigned char *LBound,
                       unsigned char *UBound, unsigned char *Response)
{
  unsigned char Data[6];
  DWORD Result = 0;
  Data[0]=HEADER;
  Data[1]=7<<5|ServoID;
  Data[2]=0x12;
  Data[3]=NULL;
  Data[4]=NULL;
  Data[5]=(Data[1]^Data[2]^Data[3]^Data[4])&0x7f;
  Result = SetOperation(&Data[0], Response);
  return Result;
}
DWORD TBase::OverCurRead(unsigned char ServoID, unsigned char *Response)
{
  unsigned char Data[6];
  DWORD Result = 0;
  Data[0]=HEADER;
  Data[1]=7<<5|ServoID;
  Data[2]=0x10;
  Data[3]=NULL;
  Data[4]=NULL;
  Data[5]=(Data[1]^Data[2]^Data[3]^Data[4])&0x7f;
  Result = SetOperation(&Data[0], Response);
  return Result;
}
DWORD TBase::OverCurSet(unsigned char ServoID, unsigned char NewOverCur,
                        unsigned char *Response)
{
  unsigned char Data[6];
  DWORD Result = 0;
  Data[0]=HEADER;
  Data[1]=7<<5|ServoID;
  Data[2]=0x0f;
  Data[3]=NewOverCur;
  Data[4]=Data[3];
  Data[5]=(Data[1]^Data[2]^Data[3]^Data[4])&0x7f;
  Result = SetOperation(&Data[0], Response);
  return Result;
}
```

PŘÍLOHA P II: FUNKCE PRO VÝPOČET TRAJEKTORIE A ŘÍZENÍ ROBOTA

```
void __fastcall TForm1::CalculateClick(TObject *Sender)
{
	int x,y,newx,newy,quadr,Angle,Newangle;
	char side;
	bool dir;
	for(int i=0;i<Base.ctr;i++)
	{
	newx=Base.x[i+1]-Base.x[i];
	newy=Base.y[i+1]-Base.y[i];
	Base.Len[i]=sqrt(pow(newx,2)+pow(newy,2));
	if(newy<0) quadr=4; else quadr=1;
	switch(quadr){
	case 1:{Angle=RoundTo(abs(180/3.14159*ArcTan2(newy,newx)),0);};break;
	case 4:{Angle=360-RoundTo(abs(180/3.14159*ArcTan2(newy,newx)),0);};break;
	};
	if(Angle>180) Angle-=360;
	if(Base.Angle>180) Base.Angle-=360;
	Newangle=abs(Angle-Base.Angle);
	if(Newangle>180) Newangle=360-Newangle;
	Base.NewAngle[i]=Newangle;
	if(Angle>=0&&Base.Angle>=0)
		{if(Angle-Base.Angle>0){dir=true;}
			else{dir=false;}};
	if(Angle<=0&&Base.Angle<=0)
		{if(Angle-Base.Angle>0){dir=true;}
			else{dir=false;}};
	if(Angle>=0&&Base.Angle<=0)
		{if(Angle-Base.Angle>=180){dir=false;}
			else{dir=true;}};
	if(Angle<=0&&Base.Angle>=0)
		{if(Angle-Base.Angle<=-180){dir=true;}
			else{dir=false;}};
	Base.olda[i]=Base.Angle;
	Base.newa[i]=Angle;
	Base.Dir[i]=dir;
	Base.Angle=Angle;
	}
	Rich1->Clear();
	Rich1->Lines->Add(IntToStr(Base.ctr));
	for(int i=0;i<Base.ctr;i++)
	Rich1->Lines->Add(IntToStr(Base.NewAngle[i])+" "+Base.newa[i]+" "+int(Base.Dir[i]));
}
//EasyLeft
void __fastcall TForm1::EasyLeft(TObject *Sender)
{
	int Speed,s,sl;
	unsigned char Response[2];
	Speed=4;
	s=2;
	if(Base.NewAngle[Base.pom]>0&&Base.NewAngle[Base.pom]<=3)
	Speed=s;
	if(Base.NewAngle[Base.pom]>3&&Base.NewAngle[Base.pom]<=6)
	Speed=s-1;
	if(Base.NewAngle[Base.pom]>6&&Base.NewAngle[Base.pom]<=9)
```

```
        Speed=s-2;
        Timer3->Interval=RoundTo(9*Base.Len[Base.pom]+9*Base.NewAngle[Base.pom],0);
        Timer1->Enabled=false;
        Timer2->Enabled=false;
        Timer3->Enabled=true;
        Base.Rotation360(2,10,4,&Response[0]);
        Base.Rotation360(3,10,4,&Response[0]);
        Base.Rotation360(1,Speed,3,&Response[0]);
        Base.Rotation360(4,Speed,3,&Response[0]);
        switch(Base.NewAngle[Base.pom]){
        case 1:sl=50;break;
        case 2:sl=100;break;
        case 3:sl=150;break;
        case 4:sl=250;break;
        case 5:sl=350;break;
        default :sl=(RoundTo(5*Base.Len[Base.pom],0));}
        Sleep(sl);
        Base.Rotation360(1,10,3,&Response[0]);
        Base.Rotation360(4,10,3,&Response[0]);
        Base.Rotation360(2,10,4,&Response[0]);
        Base.Rotation360(3,10,4,&Response[0]);
        Rich1->Lines->Add(IntToStr(Base.NewAngle[Base.pom])+" "+Base.Len[Base.pom]);
        if(Base.pom>=Base.ctr) StopClick(Sender);
        Base.pom++;
}
//EasyRight
void __fastcall TForm1::EasyRight(TObject *Sender)
{
        int Speed,s,sl;
        unsigned char Response[2];
        Speed=4;
        s=2;
        if(Base.NewAngle[Base.pom]>0&&Base.NewAngle[Base.pom]<=3)
        Speed=s;
        if(Base.NewAngle[Base.pom]>3&&Base.NewAngle[Base.pom]<=6)
        Speed=s-1;
        if(Base.NewAngle[Base.pom]>6&&Base.NewAngle[Base.pom]<=9)
        Speed=s-2;
        Timer3->Interval=RoundTo(9*Base.Len[Base.pom]+9*Base.NewAngle[Base.pom],0);
        Timer1->Enabled=false;
        Timer2->Enabled=false;
        Timer3->Enabled=true;
        Base.Rotation360(1,10,3,&Response[0]);
        Base.Rotation360(4,10,3,&Response[0]);
        Base.Rotation360(2,Speed,4,&Response[0]);
        Base.Rotation360(3,Speed,4,&Response[0]);
        switch(Base.NewAngle[Base.pom]){
        case 1:sl=50;break;
        case 2:sl=100;break;
        case 3:sl=150;break;
        case 4:sl=250;break;
        case 5:sl=350;break;
        default :sl=(RoundTo(5*Base.Len[Base.pom],0));}
        Sleep(sl);
        Base.Rotation360(1,10,3,&Response[0]);
        Base.Rotation360(4,10,3,&Response[0]);
        Base.Rotation360(2,10,4,&Response[0]);
        Base.Rotation360(3,10,4,&Response[0]);
        Rich1->Lines->Add(IntToStr(Base.NewAngle[Base.pom])+" "+Base.Len[Base.pom]);
```

```
        if(Base.pom>=Base.ctr) StopClick(Sender);
        Base.pom++;
}
//Left
void __fastcall TForm1::LeftClick(TObject *Sender)
{
        Timer2->Interval=RoundTo(LEFT*Base.NewAngle[Base.pom]/90,0);
        Timer1->Enabled=false;
        Timer3->Enabled=false;
        Timer2->Enabled=true;
        unsigned char Response[2];
        //zmenit 1,4
        Base.Rotation360(1,10,4,&Response[0]);
        Base.Rotation360(4,10,4,&Response[0]);
        Base.Rotation360(2,10,4,&Response[0]);
        Base.Rotation360(3,10,4,&Response[0]);
}
//Right
void __fastcall TForm1::RightClick(TObject *Sender)
{
        Timer2->Interval=RoundTo(RIGHT*Base.NewAngle[Base.pom]/90,0);
        Timer1->Enabled=false;
        Timer3->Enabled=false;
        Timer2->Enabled=true;
        unsigned char Response[2];
        //zmenit 2,3
        Base.Rotation360(1,10,3,&Response[0]);
        Base.Rotation360(4,10,3,&Response[0]);
        Base.Rotation360(2,10,3,&Response[0]);
        Base.Rotation360(3,10,3,&Response[0]);
}
//Forward
void __fastcall TForm1::ForwardClick(TObject *Sender)
{
        Timer1->Interval=RoundTo(FOR*Base.Len[Base.pom],0);
        Timer1->Enabled=true;
        Timer2->Enabled=false;
        Timer3->Enabled=false;
        unsigned char Speed,Response[2];
        Speed=10;
        Base.Rotation360(1,Speed,3,&Response[0]);
        Base.Rotation360(2,Speed,4,&Response[0]);
        Base.Rotation360(3,Speed,4,&Response[0]);
        Base.Rotation360(4,Speed,3,&Response[0]);
        Rich1->Lines->Add(IntToStr(Base.NewAngle[Base.pom])+" "+Base.Len[Base.pom]);
        if(Base.pom>=Base.ctr) StopClick(Sender);
        Base.pom++;
}
//Stop
void __fastcall TForm1::StopClick(TObject *Sender)
{
        unsigned char Response[2];
        Base.Rotation360(1,0,4,&Response[0]);
        Base.Rotation360(2,0,3,&Response[0]);
        Base.Rotation360(3,0,3,&Response[0]);
        Base.Rotation360(4,0,4,&Response[0]);
        Timer1->Enabled=false;
        Timer2->Enabled=false;
        Timer3->Enabled=false;
```

```
}
```
8.Funkce pro výpočet koeficientů interpolační aproximace polynomiální řadou a vzorkování vzniklé funkce

```
void __fastcall TForm1::Interpolation(TObject *Sender)
{
	double pole[100][100],f[100],a[100],del;
	int i,j,n;
	n=Base.ctr-1;
	for(i=0;i<=n;i++) a[i]=1;
	for(i=0;i<=n;i++) f[i]=Base.y[i]*10;
	for(i=0;i<=n;i++) pole[i][0]=10;
	for(i=0;i<=n;i++)
	for(j=1;j<=n;j++)
	pole[i][j]=pow(Base.x[i],j)*10;
	for(i=0;i<=n;i++)
	{	del=pole[i][i]-1;
		f[i]=f[i]/del;
		for(j=0;j<=n;j++)
		if(i==j) pole[i][i]=(del-pole[i][i])/del;
		else pole[i][j]=(0-pole[i][j])/del;
	}
	for(int coun=0;coun<=1000;coun++)
	for(i=0;i<=n;i++)
	{	del=0;
		for(j=0;j<=n;j++)
		del+=a[j]*pole[i][j];
		a[i]=del+f[i];
	}
}
//Sampling the function
	ResetClick(Sender);
	double x,y,nx,ny,Lo,Hi;
	int x1,x2;
	Rich1->Clear();
	Lo=StrToFloat(LoInt->Text);
	Hi=StrToFloat(HiInt->Text);
	x=Lo;
	y=0;
	for(i=0;i<=n;i++)
	y+=a[i]*pow(x,i);
	Base.x[Base.ctr]=RoundTo(x,0);
	Base.y[Base.ctr]=RoundTo(y,0);
	while(x<Hi)
	{
		nx=x+50;
		if(nx<=Hi)
		{
		Base.ctr++;
		ny=0;
		for(i=0;i<=n;i++)
		ny+=a[i]*pow(nx,i);
		if(abs(ny-y)>100){nx-=20;ny=0;for(i=0;i<=n;i++)
		ny+=a[i]*pow(nx,i);};
		x=nx;y=ny;
		Base.x[Base.ctr]=RoundTo(x,0);
		Base.y[Base.ctr]=RoundTo(y,0);
		}else x=Hi;
	}
	x=Base.x[0];y=Base.y[0];
	for(int i=0;i<=Base.ctr;i++)
```

```
        {
                Base.x[i]=Base.x[i]-x;
                Base.y[i]=Base.y[i]-y;
        }
        for(int i=0;i<=Base.ctr;i++)
        Rich1->Lines->Add(IntToStr(Base.x[i])+" "+Base.y[i]);
}
```

PŘÍLOHA P III: VOLÁNÍ FUNKCE PRO OTEVŘENÍ PORTU

```
void __fastcall TForm1::btComOpenClick(TObject *Sender)
{
  if(btComOpen->Caption == "Open"){
    Base.OpenPort(ComboBox1->Text.c_str());
    btComOpen->Caption = "Close";
    ComboBox1->Enabled = false;
  }else{
    Base.ClosePort(ComboBox1->Text.c_str());
    btComOpen->Caption = "Open";
    ComboBox1->Enabled = true;
  }
}
```