

# **Návrh softwaru pro ovládání laserového scanneru a mobilního robota**

Control Software for a Laser Scanner connected to a Mobile  
Robot

Bc. Tomáš Sokol

---

Diplomová práce  
2013



Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky

---

Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky  
akademický rok: 2012/2013

## ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Tomáš Sokol**  
Osobní číslo: **A11446**  
Studijní program: **N3902 Inženýrská informatika**  
Studijní obor: **Počítačové a komunikační systémy**  
Forma studia: **prezenční**

Téma práce: **Návrh softwaru pro ovládání laserového scanneru a mobilního robota**

Zásady pro vypracování:

1. Vypracujte literární rešerši na dané téma.
2. Analyzujte problém konfigurace a sběru dat laserového skeneru a řízení mobilní robotické platformy.
3. Ověřte možnosti vývojových kitů platformy .NET Micro Framework pro dané téma.
4. Navrhněte a vytvořte koncepci hardwaru a softwaru.
5. Vytvořte prototyp navrženého řešení a ověřte jeho funkci.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. WATSON, Ben. C 4.0: řešení praktických programátorských úloh. Vyd. 1. Brno: Zoner Press, 2010, 656 s. Encyklopedie Zoner Press. ISBN 978-80-7413-094-6.
2. NAGEL, Christian. Professional C 4 and .Net 4: řešení praktických programátorských úloh. Vyd. 1. Indianapolis, IN: Wiley Pub., c2010, 1474 p. Encyklopedie Zoner Press. ISBN 04-705-0225-8.
3. NASH, Trey. C 2010: rychlý průvodce novinkami a nejlepšími postupy. Vyd. 1. Brno: Computer Press, 2010, 624 s. Encyklopedie Zoner Press. ISBN 978-80-251-3034-6.
4. GHI ELECTRONICS. Beginners Guide to C and the .NET Micro Framework. United States, Troy, 2012. Dostupné z:  
<http://www.ghielectronics.com/downloads/FEZ/Beginners%20guide%20to%20NETMF.pdf>
5. GHI ELECTRONICS. .NET & Internet of Things: The fun and easy way, the way... United States, Troy, 2011. Dostupné z:  
[http://www.ghielectronics.com/downloads/FEZ/FEZ\\_Internet\\_of\\_Things\\_Book.pdf](http://www.ghielectronics.com/downloads/FEZ/FEZ_Internet_of_Things_Book.pdf)
6. GHI ELECTRONICS. Beginners Guide to Porting NETMF. United States, Troy, 2012. Dostupné z:  
<http://www.ghielectronics.com/downloads/FEZ/Beginners%20Guide%20to%20Porting%20>

Vedoucí diplomové práce:

**Ing. Erik Král**

Ústav počítačových a komunikačních systémů

Datum zadání diplomové práce:

**26. února 2013**

Termín odevzdání diplomové práce:

**31. května 2013**

Ve Zlíně dne 26. února 2013

prof. Ing. Vladimír Vašek, CSc.

*děkan*



prof. Ing. Karel Vlček, CSc.

*ředitel ústavu*

## ABSTRAKT

Cieľom tejto práce bolo navrhnuť a vytvoriť riadiaci systém mobilnej robotickej platformy a k nemu podporný software z využitím technológie .NET Micro Framework. Riadiaci systém pozostáva z dvoch vrstiev, z ktorých jedna riadi motory a ultrazvukové senzory a druhá zbiera dáta z laserového skenera a kamery. Hardvérová časť obsahuje popis robota, z akých komponentov sa skladá. Softvérová časť obsahuje riadiace algoritmy potrebné pre komunikáciu použitých zberníc, k ovládaniu pohybov robotickej platformy a k získavaniu dát zo senzorov.

Kľúčové slová: C#, .NET Micro Framework, .NET Gadgeteer, diferenciálny podvozok, ultrazvukové senzory, laserový skener

## ABSTRACT

The aim of this work is to design and develop the control system for mobile robotic platform with supporting software based on .NET Micro Framework technology. The control system consists of two layers, one of them controls the motors and ultrasonic sensors and the second layer collects data from laser scanner and camera. The hardware part contains describes each components of the robot describes. The software part contains control algorithms used to communication bus, to movement control and retrieving data from sensors.

Keywords: C#, .NET Micro Framework, .NET Gadgeteer, differential chassis, ultrasonic sensors, laser scanner

Týmto by som sa chcel poďakovať svojim rodičom za ich podporu v priebehu štúdia, ďalej vedúcemu diplomovej práce pánu Ing. Eriku Královi za cenné rady a pripomienky pri písaní tejto práce a pánu Ing. Michalovi Brázdovi za umožnenie zúčastniť sa na tomto projekte, vďaka ktorému som získal veľa ďalších skúseností.

**Prohlašuji, že**

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

**Prohlašuji,**

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....  
podpis diplomanta

**OBSAH**

|   |           |
|---|-----------|
| <b>ÚVOD</b> .....   | <b>9</b>  |
| <b>I TEORETICKÁ ČASŤ</b> .....                              | <b>10</b> |
| <b>1 MOBILNÝ ROBOT</b> .....                                | <b>11</b> |
| 1.1    PODVOZKY MOBILNÝCH ROBOTOV .....                     | 11        |
| 1.1.1    Diferenciálny podvozok .....                       | 12        |
| 1.1.2    Ackermanov podvozok .....                          | 13        |
| 1.1.3    Všesmerový podvozok .....                          | 14        |
| 1.2    SENZORICKÝ SUBSYSTÉM .....                           | 14        |
| 1.2.1    Vnútorne senzory .....                             | 15        |
| 1.2.2    Vonkajšie senzory .....                            | 15        |
| <b>2 KOMPONENTY ROBOTICKEJ PLATFORMY</b> .....              | <b>17</b> |
| 2.1    SERVOMOTOR TGN2-0054 .....                           | 17        |
| 2.2    ULTRAZVUKOVÝ SENZOR SRF02 .....                      | 19        |
| 2.3    LASEROVÝ DIALKOMER SICK LMS 400 .....                | 19        |
| 2.3.1    Princíp laserového diaľkomeru .....                | 20        |
| 2.3.2    Formát výstupných dát .....                        | 22        |
| <b>3 POUŽITÉ TECHNOLOGIE</b> .....                          | <b>24</b> |
| 3.1    MICROSOFT .NET MICRO FRAMEWORK.....                  | 24        |
| 3.1.1    Nevýhody .NET Micro Frameworku .....               | 24        |
| 3.1.2    Nástroje pre vývoj .NET Micro Framework .....      | 25        |
| 3.1.3    Prístup k hardwaru .....                           | 25        |
| 3.1.4    Riadený verzus natívny kód .....                   | 26        |
| 3.1.5    Automatická správa pamäti (Garbage collector)..... | 26        |
| 3.1.6    Architektúra .NET Micro Frameworku.....            | 27        |
| 3.1.7    Kompilácia v .NET Micro Framework .....            | 28        |
| 3.1.8    Portovanie verzus používanie .....                 | 29        |
| 3.2    FÁZY BOOTOVANIA .NET MICRO FRAMEWORKU.....           | 30        |
| 3.3    .NET GADGETEER.....                                  | 31        |
| 3.4    GHI, PREMIUM A OSHW NAMESPACE .....                  | 32        |
| <b>II PRAKTICKÁ ČASŤ</b> .....                              | <b>33</b> |
| <b>4 NÁVRH RIADIACEHO SYSTÉMU</b> .....                     | <b>34</b> |
| 4.1    FEZ SPIDER.....                                      | 35        |
| <b>5 ANALÝZA RÝCHLOSTI LASEROVÉHO SKENERA</b> .....         | <b>36</b> |
| <b>6 SOFTVÉROVÁ IMPLEMENTÁCIA</b> .....                     | <b>38</b> |
| 6.1    SOFTVÉROVÉ RIEŠENIE PRE FEZ SPIDER .....             | 38        |
| 6.1.1    Popis tried.....                                   | 40        |
| 6.1.1.1    Trieda UVSensor .....                            | 41        |
| 6.1.1.2    Trieda Ethernet .....                            | 42        |
| 6.1.1.3    Trieda MotorController.....                      | 43        |
| 6.1.1.4    Trieda Convert .....                             | 45        |
| 6.2    SOFTVÉROVÁ RIEŠENIE PRE SICK LMS 400 .....           | 46        |
| <b>7 OVLÁDANIE ROBOTICKEJ PLATFORMY</b> .....               | <b>49</b> |

---

|     |   |           |
|-----|---|-----------|
| 7.1 | MIESTNE OVLÁDANIE .....                           | 49        |
| 7.2 | VZDIALENÉ OVLÁDANIE POMOCOU PC.....               | 50        |
| 7.3 | VZDIALENÉ OVLÁDANIE MOBILNÝM TELEFÓNOM.....       | 51        |
|     | <b>ZÁVER .....</b>                                | <b>52</b> |
|     | <b>CONCLUSION .....</b>                           | <b>53</b> |
|     | <b>ZOZNAM POUŽITEJ LITERATÚRY .....</b>           | <b>54</b> |
|     | <b>ZOZNAM POUŽITÝCH SYMBOLOV A ZKRATIEK .....</b> | <b>56</b> |
|     | <b>ZOZNAM OBRÁZKOV .....</b>                      | <b>57</b> |
|     | <b>ZOZNAM TABULIEK .....</b>                      | <b>58</b> |

## ÚVOD

V dnešnej dobe sú robotické systémy veľmi populárne, pretože človeku uľahčujú prácu. Môžu to byť kuchynské roboty, manipulatory v priemyselných halách, záchranárske prieskumníky či banské zariadenia, alebo inteligentné vesmírne vozidlá. Najviac sú robotické systémy nasadzované v automatizovanom priemysle.

Za mobilného robota považujeme elektromechanické zariadenie schopné sa pohybovať po určitej trase v priestore a reagovať na zmeny okolia. Svoje okolie vníma pomocou rôznych typov senzorov od jednoduchých ultrazvukov až po zložité kamerové systémy. Robotické systémy sa delia podľa rôznych kritérií, ale hlavne podľa oblasti využitia.

Každý robot musí mať vlastný riadiaci systém, ktorý mu dáva príkazy. Väčšina riadiacich systémov pozostáva z viacerých vrstiev, kde každá vrstva zastáva inú funkciu a dokopy tvoria celé riadenie. Napríklad najnižšia vrstva obsluhuje základné senzory na detekciu prekážky. Nadradená vrstva môže zasa obsahovať obslužné programy, ako obísť prípadnú prekážku. Až po najvyššie vrstvy, ktoré môžu obsahovať kamerové systémy, či rôzne navigačné systémy.

Táto práca sa zaoberá návrhom riadiaceho systému pre mobilnú robotickú platformu. Hlavnou úlohou je vytvorenie softwarových knižníc, ktoré budú tvoriť základ riadiaceho systému. V danej fáze robot nie je schopný plného autonómneho riadenia. Je možné ho ovládať pomocou dispečerského počítača. Hlavným rozpoznávacím systémom pre robota v danej fáze vývoja je laserový skener. V teoretickej časti bude predstavená platforma .NET Micro Framework, na ktorej je riadiaci systém postavený. Budú popísané komponenty, z ktorých robot pozostáva, komunikačné protokoly a princíp laserového skenera. V praktickej časti sú popísané jednotlivé vrstvy, z ktorých riadiaci systém pozostáva a akú úlohu bude konkrétna vrstva plniť. Ďalej bude vysvetlené navrhované algoritmy pre spracovávanie dát zo senzorov a laserového diaľkomeru.

## I. TEORETICKÁ ČASŤ

## 1 MOBILNÝ ROBOT

Mobilný robot sa od iných typov robotov odlišuje tým, že má schopnosť pohybovať sa v priestore. Táto mobilita prináša užívateľom veľa výhod a uplatnenia. Schopnosť pohybu mobilných robotov zabezpečuje podvozok.

Robotmi sa zaoberá vedná disciplína robotika, ktorá tiež zahŕňa problematiku návrhu, výroby, designu a cieľu nasadenia robotického systému. Robotika sa zaoberá aj problematikou jednotlivých subsystémov, ktoré tvoria celý robotický systém ako mechanický, elektronický, riadiaci alebo pohonný subsystém.[1]

Mobilných robotov je možno triediť podľa viacerých kritérií. Ich základné rozdelenie je na:

- Autonómne mobilné roboty
- Diaľkovo riadené mobilné roboty

U autonómneho riadenia ide o robotov, ktorý sa spoliehajú len na svoj senzorický subsystém a následného rozhodovania sa. Prekonávajú všetky úlohy samostatne bez zásahu človeka. Napríklad robot sledujúci čiernu čiaru na zemi. K tomu by mohol mať aj schopnosť reagovať na prekážku, vyhnúť sa jej a pokračovať ďalej v ceste. Alebo môže ísť o robota, ktorý sa vie pohybovať v cudzom prostredí. Dokáže zmapovať svoje okolie a následne identifikuje svoju pozíciu. [1]

Diaľkovo riadené robotické systémy sú riadené dispečerom, ktorý má v každom okamžiku informáciu o okolí robota, napr.: obraz z kamery na robotovi. Ovládanie prebieha bezdrôtovou formou wifi, alebo v inom voľnom rádiovom pásme. Roboti v tejto skupine zvyčajne majú určitý stupeň autonómneho chovania, napr.: zastavenie pred prekážkou. [1]

### 1.1 Podvozky mobilných robotov

V dnešnej dobe je veľa rôznych typov podvozkov a je ťažké povedať, ktorý je lepší. Každý podvozok je stavaný pre určité prostredie a terén. Táto kapitola predstavuje len základné typy podvozkov. Nezaoberá sa podvozkami pre robotov do vodného prostredia, či na lietanie, ani chodiacimi robotami. Pri výbere podvozku je dôležité si zvoliť v ktorom prostredí bude robotický systém manévrovať.

Prostredie delíme na:

- vnútorné
- vonkajšie

Ako vnútorné prostredie sú myslené vnútra budov, potrubí a podobné prostredia. Napríklad robot pre diagnostiku plynového potrubia či robot strážiaci sklad. Tento typ robotov nie je určený pre členitý terén.

Vonkajšie prostredie je opakom vnútorného, čiže robot sa bude pohybovať po členitom teréne. Medzi budovami, v lese, či v zamorenom prostredí musí byť robot schopný riadeného pohybu.

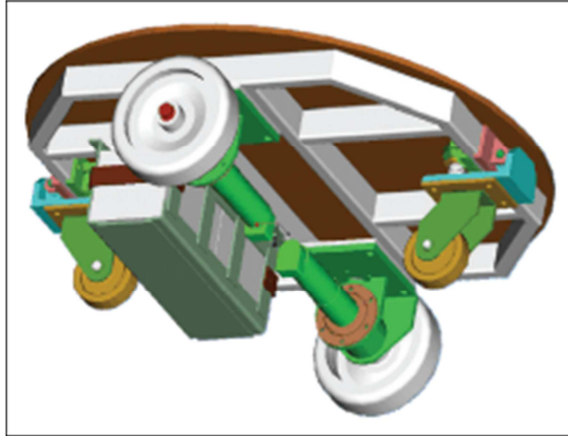
Ďalším dôležitým faktorom podvozku je aj jeho konštrukcia a počet stupňov voľnosti kôl. Každý stupeň voľnosti kolu pridáva ďalší rozmer pohybu. Kolo z jedným stupňom voľnosti sa môže pohybovať len dopredu a dozadu. Zvyčajne sa používajú kolá z jedným či dvomi stupňami voľnosti. U automobilov majú kolesá na zadnej náprave jeden stupeň a na prednej náprave dva stupne voľnosti. Je to z toho dôvodu, že prednými kolesami sa autu určuje smer jazdy. [1]

### 1.1.1 Diferenciálny podvozok

Do tejto kategórie patria všetky robotické systémy, ktoré sú schopné otáčať sa na mieste okolo svojej osy, ale pohybujú sa len dopredu či dozadu. Názov naznačuje, že v tomto podvozku je dôležitý rozdiel rýchlostí ľavého a pravého kola.

Základom tohto typu podvozku sú dve nezávisle poháňané kola a jedno či dve pasívne kolá pre udržanie stability, ak sa jedná o kolový podvozok. V prípade, že ide o pásový podvozok, tak podvozok je tvorený jedným pásom na každej strane. Takýto typ podvozkov je dnes často používaný v záchranárskych robotoch u hasičov, alebo u vojakov ako rôzne pomocné zariadenia typu hľadača mín.

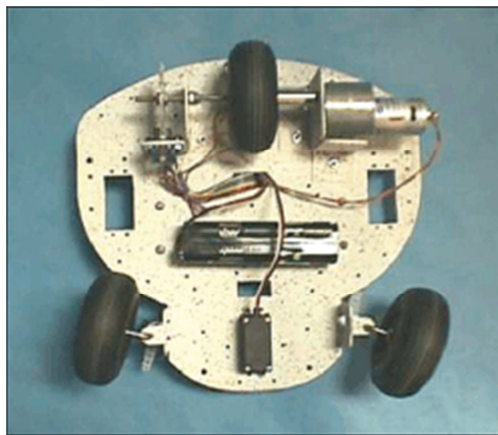
Ak sa kolesá na oboch stranách podvozku pohybujú rovnakou rýchlosťou proti sebe, robot sa pohybuje dopredu, či dozadu. Ak sú rýchlosti rovnaké robot sa otáča na mieste okolo svojej osi, ktorá je medzi kolesami. Posledný prípad nastáva, keď sú rýchlosti rozdielne a zároveň opačné tak sa robot pohybuje po kružnici. [2][3]



Obrázok 1: Model diferenciálneho podvozku [4]

### 1.1.2 Ackermanov podvozok

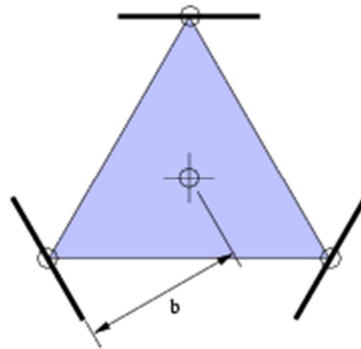
Tento typ podvozku zahŕňa robotov, ktorý sa nedokážu otáčať na mieste, pretože zmenu smeru dosahujú natočením kôl (nápravy) a následným pohybom vpred či vzad celého podvozku. Tento typ podvozkov nájdeme v dnešných automobiloch. Najlepšie sa tento typ podvozku znázorňuje na trojkolke. Trojkolka má jedno riaditeľné kolo vpredu a dve pevné vzadu. Tento model má dve možnosti pohybu. Prvá možnosť po priamke smerom dopredu či dozadu a druhá možnosť pohybu po kružnici. Obvod kružnice, ktorú podvozok opisuje je závislý od uhla natočenia predného riaditeľného kolesa, či celej nápravy. Na obrázku je zobrazený model, ktorý má kolesá rozmiestnené naopak. Pevné koleso má vzadu a dve riaditeľné kolesá vpredu. [2][3]



Obrázok 2: Model Ackermanovho podvozku [4]

### 1.1.3 Všesmerový podvozok

Ako vyplýva z názvu, táto kategória podvozkov dovoľuje robotom sa pohybovať všetkými smermi bez ohľadu aktuálneho smeru. S týmto typom podvozkov sa nestretáme často, ale poslednou dobou začína byť tento podvozok populárny v robotickom futbale. Výhodou tohto podvozku je pohyb okrem dopredu a dozadu i pohyb do strán, bez zmeny orientácie. Každé koleso je tvorené ďalšími nezávislými kolieskami po obvode. Je to z toho dôvodu, aby malo koleso viac stupňov voľnosti a tým pádom sa mohlo pohybovať aj iným smerom. [2][3]



Obrázok 3: Model všesmerového podvozku [2]

## 1.2 Senzorický subsystém

Aby sa robot mohol v určitom stupni autonómnosti pohybovať v priestore, potrebuje vedieť čo sa okolo neho deje. Sensory sú pre robotov oči a uši. Ako každý človek vnímajúci svoje okolie pomocou zmyslových orgánov, takisto robot vníma svoje okolie pomocou senzorov. Existuje veľa druhov senzorov. Nie všetky senzory sú multifunkčné a dokážu vnímať všetko naraz. Každú informáciu, ktorú by mal robot vedieť o svojom okolí získava práve z určitého senzora. Získanú informáciu zo senzorov prevedie do obrazu, podľa ktorého si vytvorí model okolia. Riadiaca jednotka podľa vytvoreného modelu dokáže koordinovať pohyby robota, či inak zareagovať. Sensory sa delia do dvoch veľkých skupín, podľa vzťahu prostredia k robotovi. [1]

Z pohľadu robota sú najdôležitejšie senzory pre navigáciu a vlastnú diagnostiku. Takéto senzory robota informujú o vybočení z dráhy, či o rôznych prekážkach v smere pohybu, alebo o nejakej nečakanej situácii. Podľa prostredia nasadenia, môže robot

obsahovať aj senzory pre meranie teplôt, obsahu CO<sub>2</sub>, radiácie, ale tieto senzory nepatria medzi bežne používané senzory.

### 1.2.1 Vnútorne senzory

Vnútorne senzory robota informujú o vnútorných stavoch subsystémov. V rámci diagnostiky je to napr.: kapacita batérií, monitorovanie komunikácie či teplota riadiacich členov. Pre navigačné systémy je to poloha a rýchlosť. Na základe týchto parametrov je riadiaci systém schopný koordinovať pohyb zo znalosťou kinematického modelu robota. Medzi ďalšie známe vnútorné senzory patria inkrementálne senzory pre meranie relatívnej polohy a natočenia osí motorov. Takéto senzory pracujú na dvoch princípoch.

V prvom prípade ide o magnetické senzory polohy, ktoré využívajú Hallov jav. Tieto senzory zabezpečujú spoľahlivú spätnú väzbu v riadení pohonov. Hallov jav zapríčiňuje vznik Hallovho napätia na bokoch vodivého materiálu, cez ktorý prechádza prúd a zároveň naň kolmo pôsobí magnetická indukcia. Tento typ senzora je najjednoduchší a najrozšírenejší. [1]

V druhom prípade ide o optický senzor. Funguje na princípe prerušovania emitovaného svetla LED diódy. Na jednej strane je LED dióda na druhej strane je napríklad fototranzistor, a medzi nimi sa otáča priesvitný kotúč s rovnomerne rozloženými priesvitnými a čiernymi pruhmi. Svetlo je prerušené čiernym pruhom a priesvitným pruhom je prepustené. Po úprave výstupného analógového signálu dostávame obdĺžnikový signál. Takto stačí počítať jednotlivé impulzy a získame počet otáčok, alebo náklon osi.[1]

### 1.2.2 Vonkajšie senzory

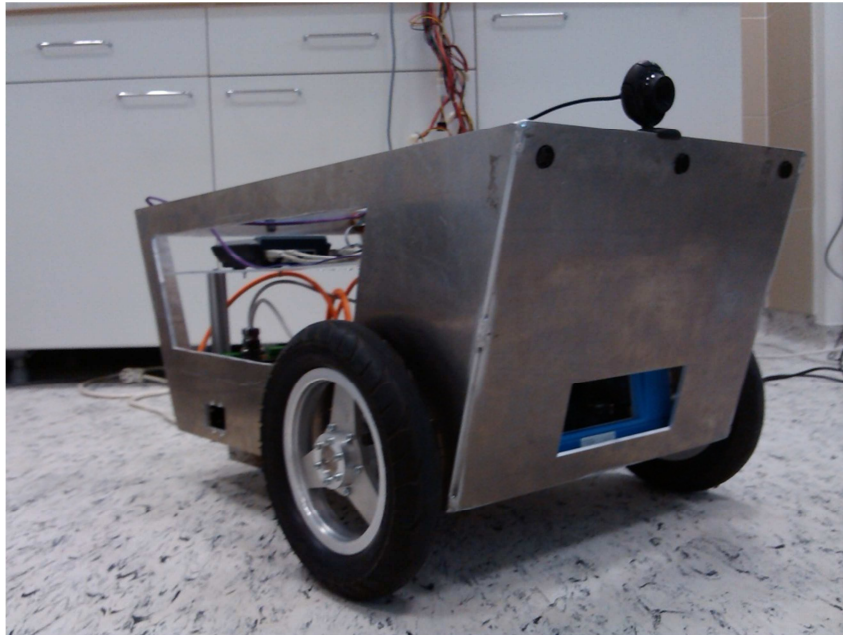
Tento typ senzorov robota informuje o jeho okolí. Napríklad teplota prostredia, rozmiestnenie prekážok, vzdialenosť prekážky od robota. Tieto senzory uľahčujú robotovi orientovať sa v priestore a vyhýbať sa prekážkam.

Medzi najzákladnejšie vonkajšie senzory patrí taktilný senzor. Ide vlastne o kontaktný spínač, ktorý pri kontakte s prekážkou zopne či rozopne elektrický obvod. Táto zmena logickej úrovni napätia v riadiacej jednotke vytvorí externé prerušenie a je volaná následná obsluha. U mobilného robota je tento senzor „poslednou záchranou“ pre zastavenie pred prekážkou.

Ďalším typom vonkajšieho senzoru je infračervený detektor prekážok. Zariadenie funguje na princípe odrazu infračerveného lúča od prekážky. Zdrojom svetla je infračervená dióda a prijímač je fototranzistor citlivý v infračervenej oblasti. Tieto fototranzistory sú citlivé len v úzkom pásme infračerveného žiarenia s vlnovými dĺžkami pod viditeľným svetlom, najčastejšie okolo 880 nm. Nevýhodou tohto senzoru je, že množstvo odrazeného svetla je závislé na farbe prekážky a druhu povrchu. Dosah senzora je okolo 20 cm. Kvôli eliminácii infračerveného žiarenia do prostredia sa signál moduluje. Najčastejšie frekvencie modulovania sú 36, 38, 48 a 56 kHz. [1]

Na podobnom princípe funguje aj ultrazvukový diaľkomer. Ako z názvu vyplýva, toto zariadenie slúži pre meranie vzdialeností. Funguje na princípe merania doby medzi vyslaním akustického signálu a prijatím akustického signálu (echo). Bežne používané frekvencie sú nad 40 kHz, čiže v pásme ktoré ľudské ucho nedokáže zachytiť. Nevýhodou je relatívne nízka rýchlosť šírenia zvuku. Oproti infračerveným sensorom, či radarom dlhšie čakáme na príjem akustického signálu. Ďalšou podstatnou nevýhodou je, že nedokážeme presne detekovať prekážku z pohľadu uhlovej pozície voči senzoru. [1]

## 2 KOMPONENTY ROBOTICKEJ PLATFORMY



*Obrázok 4: Mobilná robotická platforma*

Na obrázku 4 je zobrazená mobilná robotická platforma, pre ktorú je treba vytvoriť riadiaci systém. Konštrukčné práce spojené z výrobou a montážou nie je súčasťou tejto diplomovej práce. Práca sa zaoberá len vytvorením riadiaceho systému, vytvorením softvérových knižníc a oživením jednotlivých častí.

Z obrázku 4 je vidieť, že podvozok je stavaný len do rovného terénu. Vďaka diferenciálnemu podvozku je robot schopný otáčať sa na mieste ako tank, princíp riadenia je v kapitole 1.1.1. Zdrojom napätia sú dva 12V akumulátory o kapacite 12Ah uložené uprostred podvozku, ktoré napájajú celú elektroniku a riadiaci systém.

### 2.1 Servomotor TGN2-0054



*Obrázok 5: Servomotor TGN2-0054 [5]*

O pohyb robota sa starajú dva silné servomotory z permanentnými magnetmi na rotore a trojfázovým vinutím na statore. Každý motor má vnútornú prevodovku z pomerom 10:1. Medzi nábojmi kolies a motormi je remeňový prevod 2:1, čiže výsledný pomer motora ku kolesám je 20:1.

*Tabuľka 1: Technické dáta pre TGN2 [5]*

| Parameter         | Hodnota | Jednotka          |
|-------------------|---------|-------------------|
| Menovitý moment   | 0,51    | Nm                |
| Menovité otáčky   | 3000    | min <sup>-1</sup> |
| Menovitý výkon    | 160     | W                 |
| Menovitý prúd     | 7,7     | A                 |
| Maximálny prúd    | 33      | A                 |
| Napájacie napätie | 24      | VDC               |
| Hmotnosť          | 1,37    | kg                |

O riadenie motorov sa starajú digitálne servozosilovače TGA-24. Ide o menič určený k riadeniu striedavých synchronných servomotorov do výkonu 150 W. Obsahuje výkonný procesor, ktorý zabezpečuje presné riadenie, meranie prúdov a polohy z resolveru. [6]



*Obrázok 6: Servozosilovač TGA-24 [6]*

Digitálny servozosilovač TGA-24 je možné riadiť týmito spôsobmi:

- Analógovým napätím – moment, alebo smer
- Signálmi pre krokové motory - krok a smer
- Digitálnymi vstupmi – spúšťanie predefinovaných profilov
- Digitálnym riadením po zbernici RS232 a CAN BUS – moment, otáčky, polohové profily [6]

## 2.2 Ultrazvukový senzor SRF02

Robot je vybavený ultrazvukovými diaľkometermi SRF02. Tri senzory sú v prednej stene a dva v zadnej stene. Princíp merania je rozobratý v kapitole 1.2.2. Senzor obsahuje elektro-akustický menič a vyhodnocovaciu jednotku (mikrokontrolér). Senzor SRF02 podporuje komunikáciu po zbernici RS232 a I<sup>2</sup>C, ale je využívaná len I<sup>2</sup>C. Každý senzor má svoju jedinečnú adresu, počiatočná je 0xE0. Na jednej zbernici môže byť maximálne 16 senzorov.



Obrázok 7: Ultrazvukový diaľkometer SRF02 [7]

Užívateľ má možnosť adresu zmeniť na ktorúkoľvek zo 16 adries. Senzor má vlastný register 0, do ktorého je možno zapisovať riadiace príkazy. Okrem tohto registru obsahuje ešte ďalších 5, ale z nich sa dá len čítať. Pre meranie je nutné zapísať do registru príslušný príkaz, v akých jednotkách chceme výsledok merania. Počká sa 67ms. Potom sa výsledok merania dá prečítať z registrov 2 a 3 ako jedna 16 bitová hodnota. Registre 2 a 3 stále obsahujú nameranú hodnotu až dovtedy, kým senzor zmeria novú a prepíše ju. Jednotlivé riadiace príkazy a registre nájdeme v dokumentácii [7].

## 2.3 Laserový diaľkometer SICK LMS 400

Laserové skenery rady LMS dodáva nemecká firma SICK na trh už dlhú dobu. Firma patrí medzi celosvetových výrobcov inteligentných senzorov. Predovšetkým sa zameriava na optické senzory v automatizácii a bezpečnostných systémoch.

Meranie laserovým systémom preniklo aj do oblasti meracích metód využívaných v robotike. Laserové diaľkomery sa dajú použiť aj pre 3D digitalizáciu okolitého prostredia. Hlavne v robotike sa používajú laserové diaľkomery ako orientačné senzory pri pohyboch.

Firma SICK vyrába veľa rôznych typov laserových diaľkometerov, ktoré sa odlišujú funkciami, rýchlosťou či rozmermi, ale práca sa zaoberá typom SICK LMS400.

SICK LM400 je bezkontaktný laserový merací systém, ktorý skenuje svoje okolie v dvoch dimenziách. Ide o takzvaný laserový radar „LIDAR“. K presnému meraniu nepotrebuje žiadne iné zariadenia, ako reflexne značky, či nejaké majáky.[8]



*Obrázok 8: SICK LM 400*

Dáta z laserového diaľkomeru môžu byť spracovávané dodávaným softwarom SOPAS, ktorý sa dá získať na oficiálnych stránkach firmy SICK. Ďalšia možnosť je vytvoriť si vlastný software pre komunikáciu a zber dát. Rýchlosť komunikačného rozhrania RS232 nie je dostačujúca pre aplikácie bežiacie v reálnom čase, preto sa viac používa Ethernetové rozhranie.

*Tabuľka 2: Špecifikácia SICK LMS 400 [8]*

|                       |                                     |
|-----------------------|-------------------------------------|
| Maximálny dosah       | 3 m                                 |
| Minimálny dosah       | 0,7 m                               |
| Zorné pole            | 70°                                 |
| Uhlové rozlíšenie     | 0,134° - 1,0°                       |
| Frekvencia skenovania | 360 -500 Hz                         |
| Komunikačné rozhrania | RS232, RS422,<br>Ethernet(10Base-T) |
| Napájacie napätie     | 24 VDC ± 15 %                       |
| Vlnová dĺžka laseru   | 650 nm                              |
| Výkon laseru          | 4,0 mW                              |
| Objektová remisia     | 10 – 100 %                          |

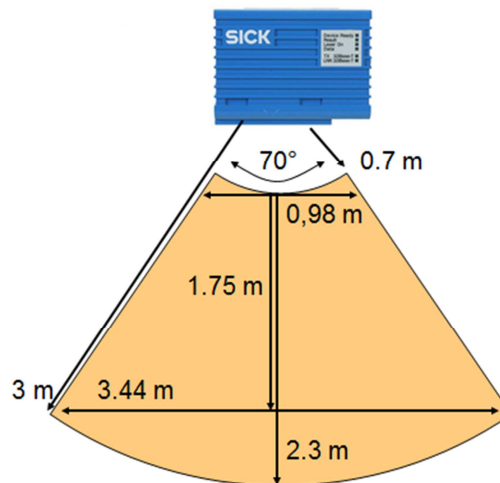
### 2.3.1 Princíp laserového diaľkomeru

K určeniu vzdialenosti u bezdotykových meracích systémov sa používa vyslanie upraveného signálu na povrch meraného objektu a následné spracovanie odrazeného

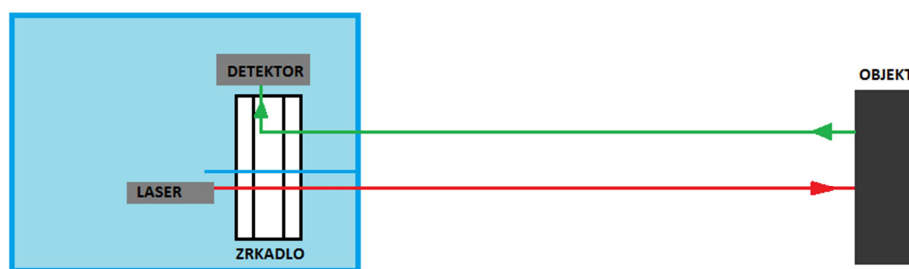
signálu. Zdrojom signálu môže byť elektro-akustický menič, LED dióda, laserová dióda, či rádiová anténa.

Zdrojom u laserových diaľkometerov je laserová dióda, ktorá vyžaruje elektromagnetické žiarenie v úzkom spektre, čiže ide o koherentný svetelný lúč. Laserové diaľkomery sú rýchlejšie a majú menší rozptyl ako ultrazvukové. Laserová dióda prevádza elektrický signál na optický. Vyslaný lúč je namodulovaný na harmonický nosný signál. Odrazený signál od skenovaného objektu dopadá na fotodiódu, ktorá prevedie prijatý signál späť na elektrický signál. Následne je elektrický signál zosilnený, prefiltrovaný na ďalšie spracovávanie. Zistí sa fázový rozdiel medzi vyslaným a prijatým impulzom. Fázový rozdiel je prevedený na frekvenciu a systém následne určí vzdialenosť objektu od nulového bodu, na základe tejto frekvencie. [9][10]

Vďaka zrkadlovému hranolu otáčajúceho sa okolo svojej vlastnej osi dokáže laser pokryť  $70^\circ$  priestoru pred sebou. Na obrázku 10 je znázornené jeho operačné pole, v ktorom dokáže merať vzdialenosti. Jeden merací cyklus trvá práve jednu otočku zrkadla. Princíp je znázornený na obrázku 9. Rýchlosť skenovania a dátová náročnosť je popísaná v kapitole 5.



Obrázok 9: Operačné pole pre SICK LMS 400



Obrázok 10: Všeobecný princíp laserového diaľkomeru

Na kvalitu merania má veľký vplyv reflektivita skenovaných objektov a veľkosť energie vyslaného lúča. Reflektivita je schopnosť skenovaného objektu odraziť svetlo. Povrch každého skenovaného objektu časť dopadajúceho svetelného lúča odrazí späť a zvyšok energie pohltí. Vplyv na reflektivitu má aj farba objektu. Čierna farba má najnižšiu reflektivitu a väčšinu energie lúča pohltí. SICK LMS 400 môže detekovať objekt, ktorý má reflektivitu väčšiu ako 10%.

### 2.3.2 Formát výstupných dát

Na každom komunikačnom rozhraní majú výstupné dáta inú štruktúru. Ku komunikácii je využité ethernetové rozhranie kvôli dostatočnej rýchlosti komunikácie. Ethernetové rozhranie je nastavené na polo duplexnú komunikáciu pri rýchlosti 10Mbit/s (10Base-T).

Tabuľka 3: Štruktúra komunikačného telegramu [8]

|            | STX                       | STX | STX | STX | Dĺžka správy                  | Správa  | Kontrolný súčet                        |
|------------|---------------------------|-----|-----|-----|-------------------------------|---|--|
| Dĺžka (B)  | 1                         | 1   | 1   | 1   | 4                             | <2495   | 1                                      |
| Popis      | Začiatok textových znakov |     |     |     | počet bytov tvoriacich správu | Príkaz v binárnom tvare   | Operácia XOR medzi každým bytom správy |
| Hex formát | 02                        | 02  | 02  | 02  | 00 00 00 32                   | 73 4D 4E 20 6D 53 43 63<br>6F 6E 66 69 67 62 79 66<br>72 65 71 20 33 20 42 31<br>38 32 34 34 42 36 20 2B<br>33 30 30 20 2B 30 20 2B<br>35 35 2E 30 20 2B 37 30<br>2E 30 | 06                                     |

Samostatná správa obsahujúca príkaz nastavenia skenera, či odpoveď je zapuzdrená v telegrame (vis tabuľka 3), kde celý telegram je ďalej zapuzdrený do TCP paketu. Typ správy identifikujú prvé tri znaky, kde môže ísť o žiadosť, alebo o odpoveď. Ďalej

nasleduje príkaz nastaveniam a za ním príslušný parameter. Obrázok 11 zobrazuje príkaz pre začiatok merania, kde príkaz je zobrazený v binárnom aj v ASCII formáte.

(STX)(STX)(STX)(STX)(DĽŽKA)sMN\_mLRreqdata\_20(KS)

02 02 02 02 00 00 00 11 73 4d 4e 20 6d 4c 52 72 65 71 64 61 74 61 20 32 30 77

*Obrázok 11: Príkaz pre začiatok merania*

Po prijatí tohto príkazu skener začne cyklicky merať a posielat' namerané hodnoty pripojenému počítaču. Pre ukončenie merania musí skener prijať podobnú správu, dovedy bude stále merať a posielat'. Výnimku tvorí, keď sa zaplní buffer na sieťovej karte počítača, tak skener prestane posielat' dáta, ale laser stále meria. Ďalšie príkazy pre nastavenie frekvencie merania, rozlišovacej schopnosti, rozsahu a komunikačných rozhraní sa dajú nájsť v dokumentácii [8].

Dobrou výhodou je, že skener má digitálne filtre pre predspracovanie a optimalizovanie nameraných dát. Filtre sa dajú rôzne kombinovať.

Typy filtrov:

- Edge filter – nastavuje krajné body, ktoré sú potláčané
- Median filter – hľadá mediány z určitého počtu skenov
- Range filter – nastavuje minimálny a maximálny dosah
- Mean filter – robí aritmetický priemer z určitého počtu skenov

### 3 POUŽITÉ TECHNOLOGIE

Táto kapitola popisuje softwarové technológie a platformy použité pri vývoji riadiaceho systému.

#### 3.1 Microsoft .NET Micro Framework

Microsoft .NET Micro Framework (.NETMF) je malé a efektívne .NET behové prostredie. Prevažne je určené k spúšťaniu riadeného (managed) kódu na zariadeniach, ktoré sú príliš malé na chod .NET Compact Framework, alebo použitia Windows CE. [11]

NETMF dovoľuje užívateľom vytvárať embedded aplikácie pre malé zariadenia z využitím vývojového prostredia Microsoft Visual Studio a programovacieho jazyka C#. Tým vzniká možnosť využitia rovnakých vývojárskych nástrojov pre tvorbu desktopových aplikácií či chytrých zariadení (smartphone), ale aj pre vývoj aplikácií pre mikrokontroléry. Pre vývoj embedded aplikácií je možnosť použiť vstavaný hardvérový emulátor pre rýchle ladenie. [11]

NETMF nie je operačný systém, ale nie je ani aplikácia. Ide o systém, ktorý je umiestnený medzi vyššou vrstvou C# kódu a nízkou vrstvou C/C++ kódu. Vďaka tomu, že behové prostredie *Common Language Runtime* (TinyCLR) leží priamo na hardvéru, je NETMF prezývané *bootable runtime*. Toto prostredie má malú stopu, ktorá v pamäti RAM zaberá okolo 100 kB. Celú správu pamäti rieši sám a nepotrebuje aby v procesore bola implementovaná jednotka správy pamäti (MMU). Preto môže NETMF bežať na lacných 32-bit mikrokontroléroch. [11][12][13]

##### 3.1.1 Nevýhody .NET Micro Frameworku

.NETMF nemôže byť použitý na aplikácie reálneho času. Aj keď je vytvorený pre rýchly beh veľa aplikácií, nemôžeme od neho čakať real-time deterministické chovanie. Prerušenia od nastavených časovačov nemusia byť vyvolané presne v zadanom časovom intervale. Prerušenie je najprv spracovávané v behovom prostredí a až po nejakom čase (niekoľko milisekúnd) sú volané metódy určené pre spracovávanie týchto prerušení. [11]

Musíme počítať z tým, že o uvoľnenie pamäte sa stará garbage collector, ktorý má právo prerušiť chod vlákna pri nedostatku zdrojov. Tiež sa musí počítať z tým, že vykonávanie riadeného kódu je troška pomalšie vzhľadom na natívny kód. Celý riadený kód je interpretovaný behovým prostredím. Nie je tu žiadny just-in-time preklad časti

riadeného kódu do natívneho pri prvom volaní, ako to funguje v plnom .NET Framework a .NET Compact Framework. [11]

### 3.1.2 Nástroje pre vývoj .NET Micro Framework

Pre vývoj embedded aplikácií na platforme .NETMF je treba .NET Micro Framework SDK (Software Development Kit) a vývojové prostredie Visual Studio (VS). VS je dostupné v platenej verzii, alebo vo voľne šíriteľnej verzii (VS Express). .NETMF je úplne od základu vydaný pod voľne šíriteľnou licenciou Apache 2.0, a je open source. Zaručuje to bezplatné využívanie celého .NETMF aj z SDK.

Druhým nástrojom pre vývoj embedded aplikácií je .NETMF Porting Kit (PK). Ide o nástroj ktorý slúži pre úpravy .NETMF na nižšej úrovni pre rôzne typy mikrokontrolérov. Pomocou PK sa dá skompilovať celý Framework na rôzny mikrokontrolér. Treba len stiahnuť zdrojové kódy .NETMF a hlavne mať znalosť hardvéru, na ktorý ideme portovať embedded aplikáciu. Najťažšia práca je portovať celý Micro Framework, lebo je treba upraviť prístupy k jednotlivým portom, registrom, pamäti a iným komponentom, ktoré hardvér obsahuje. [14][12]

### 3.1.3 Prístup k hardwaru

.NETMF už od základu vývoja ponúka iný prístup k hardwaru. Tento prístup obecné zjednodušuje vývoj embedded aplikácií a odstraňuje problémy z prístupom k jednotlivým hardvérovým komponentom. Hardvér určený pre .NETMF je kompletný mikrokontrolér pripravený k okamžitému použitiu. Obsahuje klasické hardvérové komponenty ako sú vstupné/výstupné porty (GPIO), sériové porty a pamäť. .NETMF abstrahuje hardwarový prístup za použitia vlastných základných knižníc a následne je možné pristupovať k hardvérovým komponentom ako ku objektom. Je to základ objektového programovania embedded aplikácií. Vývojárom embedded aplikácií na aplikačnej vrstve to uľahčuje prácu. Namiesto toho aby v C/C++ či assembly nastavovali bitové masky pre periféria či prerušenia, v .NETMF stačí nastaviť len vlastnosti objektu. Práve vďaka tomuto prístupu prezývaného *managed drivers* môžeme vytvárať embedded aplikácie bez hardwarovej závislosti. [11][13]

### 3.1.4 Riadený verzus natívny kód

Rozdiel medzi natívnym a riadeným kódom je dosť veľký. Natívny strojový kód je strojový kód vlastného procesoru, pre ktorý bol program preložený. Medzi natívne prekladané zdrojové kódy patria jazyky C a C++. Nevýhodou natívneho kódu je, že sa musí preložiť zvlášť pre každý iný procesor, či platformu. Preto sa natívny kód radí medzi neprenositel'né kódy. [15]

Opakom je práve riadený (managed) kód. Ako bolo spomenuté skorej, pre beh riadeného kódu na .NETMF platforme je potrebné TinyCLR. TinyCLR je zmenšená verzia CLR známeho z plného .NET Frameworku. Jazyk C# patrí medzi vyššie programovacie jazyky a je prekladaný do riadeného kódu. Riadený kód je „medzijazykom“ (IL – intermediate language), pretože je napol cesty medzi jazykom C# a strojovým jazykom. Tento jazyk má výhodu, že je prenositeľný medzi rôznymi zariadeniami, ktoré obsahujú TinyCLR. TinyCLR tiež bráni spúšťaniu nebezpečného a nesystémového natívneho kódu, ktorý môže byť zdrojom chýb, napr.: pointre. Dôležitou vlastnosťou riadeného kódu je možnosť vycytiť zle prevedené operácie, napr.: delenie nulou. Táto technika dovoľuje vývojárom vytvoriť alternatívnu cestu pre operáciu, ktorá môže mať rôzne výsledky. [15]  
[11]

Výhody riadeného kódu:

- Správa pamäte pomocou garbage collectoru
- Správa vlákien a ich vzájomná synchronizácia
- Vychytávanie výnimiek
- Typová bezpečnosť
- Ladiace služby

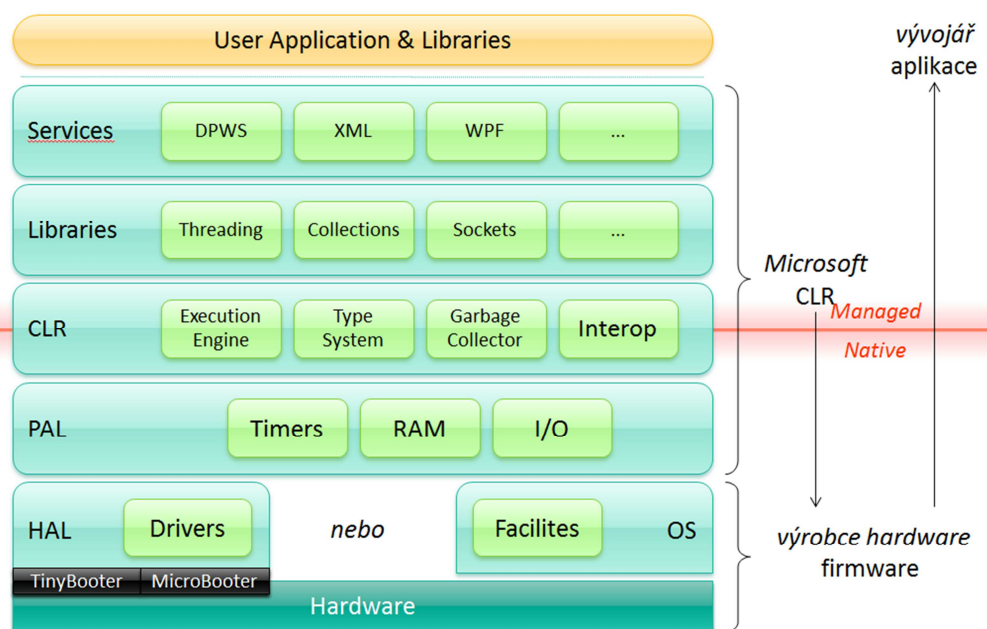
### 3.1.5 Automatická správa pamäti (Garbage collector)

Pre správu pamäte sa používa garbage collector (GC) umiestnený v TinyCLR, ako v plnom .NET Frameworku, tak aj v .NETMF. Je to nástroj, ktorý automaticky uvoľňuje nepoužívané pamäťové bloky a tým riadi beh vlákien, že poskytuje výpočtový čas jednotlivým vláknam a metódam za účelom synchronizácie prístupu k spoločným zdrojom. Oslobodzuje vývojára od manuálnej alokácie a dealokácie pamäti, ktorá môže byť zdrojom chýb. Hlavne v jazyku C/C++ často vznikajú chyby pri zabudnutom uvoľnení pamäti. V CLR existujú hodnotové typy, ktoré sú uložené priamo v zásobníku, alebo sú položkou

nejakého objektu v halde. Druhý typ je referenčný, na ktorý CLR ukazuje nepriamo, ako pointer v C/C++, ale bez príslušnej syntaxe. Pri vytvorení deklarácie referenčného typu sa v skutočnosti rezervuje miesto v halde, alebo zásobníku. Tomuto miestu je priradený nejaký typ a na tomto mieste je uložený odkaz na objekt. CLR uvoľňuje len objekty, na ktoré nie je žiadna referencia z programu. [15]

### 3.1.6 Architektúra .NET Micro Frameworku

Táto časť popisuje architektúru .NETMF od najnižšej vrstvy čiže od hardvéru. Ako bolo spomenuté na začiatku, .NETMF nepotrebuje pod sebou žiadny operačný systém a dokáže bežať priamo na hardvéry.



Obrázok 12: Architektúra .NETMF[16]

Najvyššia vrstva je aplikačná vrstva. Ide o vrstvu, ktorá obsahuje užívateľom vytvorenú aplikáciu. Táto aplikácia pomocou nižších vrstiev pristupuje k hardvérovým komponentom, ako je čítanie zo sériovej linky, či zmena logickej úrovne na digitálnom porte.

Druhá vrstva ponúka služby, ktoré môže využiť vývojár pri komunikácii z celou platformou. Napríklad TCP server, či webový server používa službu DPWS, alebo môže použiť aj zabezpečené spojenie (SSL). Pre prácu z grafickým rozhraním je možnosť

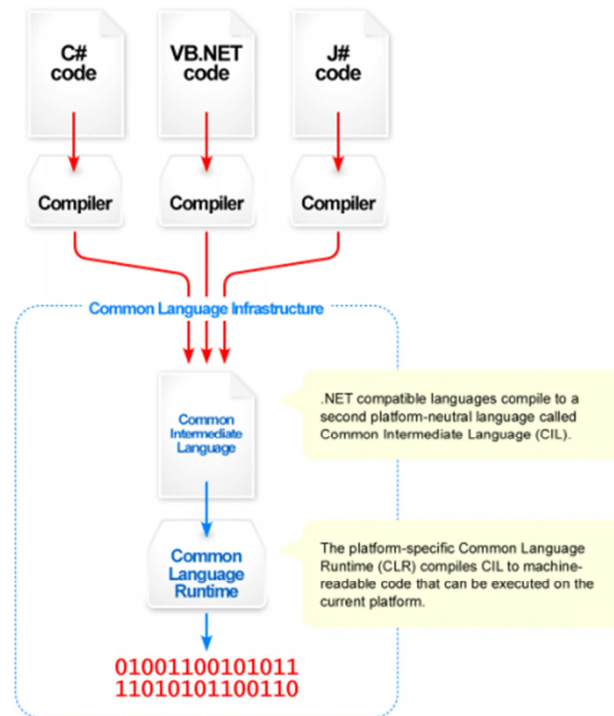
využitia WPF, ktorý slúži pre zobrazovanie grafických elementov na monitor či displej. Známy z plného .NET Frameworku Windows Presentation Foundation. [11][16]

Tretiu vrstvu tvorí .NET Micro Base Class Library (BCL) tzv. základné knižnice. BCL poskytuje knižnice pre užívateľské rozhranie a je základom pre služby poskytované vo vyšších vrstvách. BCL v .NETMF je podmnožinou BCL v plnom .NET Frameworku, ktorá obsahuje menší počet funkcií. [11][16]

Ďalej nasleduje vrstva CLR. .NETMF CLR je podmnožinou plného .NET Framework CLR. Ide o behové prostredie, ktoré sa stará o správu pamäte (GC), správu vlákien, ich vzájomnú synchronizáciu a iné systémové služby. Zabezpečuje volanie metód v natívnom kóde. Vrstva Hardware Abstract Layer (HAL) je tzv. abstraktná hardvérová vrstva, ktorá je najnižšou dostupnou vrstvou v .NETMF. Táto vrstva „leží“ na surovom hardware, čiže je na ňom závislá. Vrstva HAL abstrahuje prístup k jednotlivým hardvérovým perifériám na volanie funkcií. HAL vzniká portovaním celého .NETMF, čiže úpravami zdrojových kódov tejto vrstvy. Nad vrstvou HAL je vrstva Platform Abstraction Layer (PAL), ktorá abstrahuje vrstvu HAL z využitím časovačov a pamäťových blokov. Vrstvy HAL, PAL a CLR spoločne tvoria celé rozhranie TinyCLR. TinyCLR je teda celá softvérová nadstavba na hardvér potrebná ku spúšťaniu riadeného kódu z podpornými knižnicami. Existuje možnosť aby .NETMF bežal na operačnom systéme. Využíva sa toho hlavne pri použití real-time aplikáciách. [11][16]

### 3.1.7 Kompilácia v .NET Micro Framework

Kompilácia v platforme .NETMF je proces, pri ktorom sa vývojárom napísaný zdrojový kód v jazyku C# prekladá do „medzijazyku“ (IL intermediate language). Výsledkom je buď spustiteľný (\*.exe) súbor, alebo dynamická knižnica (\*.dll súbor) obsahujúci IL z vlastnými metadátmi pre popis zdrojového kódu. IL je potom spracovávaný CLR. .NETMF používa špeciálny IL, ktorý je minimalizovaný. CLR nie je interpret a neprekladá IL kód pri každom jeho volaní. CLR v skutočnosti preloží IL kód do strojového kódu predtým, než sa spustí. Celý postup prekladu pre platformu .NET Framework je na obrázku 13. Pre .NETMF sú použiteľné len jazyky C# a VB. [11][15]



Obrázok 13: Preklad riadeného kódu [17]

### 3.1.8 Portovanie verzus používanie

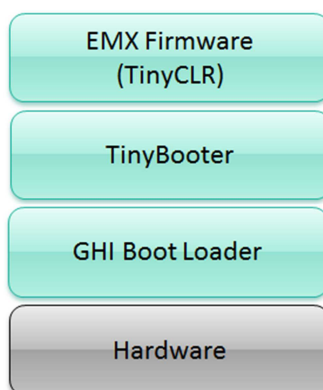
Existujú dve možnosti práce z .NETMF, portovanie a používanie. Predstavme si napríklad, čo je jednoduchšie? Napísať JAVA hru pre mobilný telefón je o mnoho ľahšie, ako portovať JAVA Virtual Machine (JVM) na mobil. Výrobcovia mobilov robia túto prácu, že portujú JAVU na ich mobilné telefóny aby mohli vývojárom hier uľahčiť prácu. Pri tvorbe hier .NETMF funguje tak isto, preto portovanie nie je tak ľahké ako jeho používanie. [14]

.NETMF spája dve hlavné časti, jadro CLR a HAL/PAL. Knížnice sú vytvorené ako hardwarovo nezávislé a pre portovanie aplikácií sú zmeny v knížniciach minimálne. Portovanie .NETMF na hardvérovú platformu si vyžaduje úpravy vrstiev HAL a PAL pre zabezpečenie ovládania hardvéru vyššími vrstvami. Microsoft poskytuje .NETMF PK, ktorý obsahuje zdrojové kódy a referencie HAL a PAL implementácie. Jedným z najznámejších dodávateľov vývojových kitov z podporou .NETMF aj portovaním je firma GHI Electronics z ich EMX modulom. GHI Electronics vytvára aj vlastné softwarové obsluhy pre ich hardvérové komponenty, ako prístup k súborovému systému, či podpora pre klávesnice a myšky. [12][14]

### 3.2 Fázy bootovania .NET Micro Frameworku

Ako už bolo spomenuté, jedna zo známych firiem pre vývoj .NETMF embedded aplikácií je GHI Electronics s ich vlastnou doskou EMX Module. Tento modul ako každý embedded kit obsahuj procesor, FLASH, RAM, vstupné/výstupné porty a rôzne komunikačné zbernice. Jeho hlavnou výhodou je podpora .NETMF z veľkou podporou PAL/HAL ovládačmi. [14]

Aby sa aplikácia, ktorú vytvoríme „rozbehla“ potrebuje .NETMF. Druhou vecou, ktorú potrebujeme je zaviesť .NETMF, respektíve naboťovať behové prostredie. Bootovanie je proces známy z klasického počítača ako hardvérová diagnostika a samotné predanie celej obsluhy nadradenému operačnému systému. Samozrejme bootovanie operačného systému v počítači je zložitý proces, ktorý práca nerozoberá a preto je len stručne spomenutý. Celý proces bootovania .NETMF tvoria tri fázy. Každá fáza je tvorená samostatným softvérom. [14]



Obrázok 14: Vrstvy bootovania

Pri pripojení na zdroj napätia sa ako prvé spustí GHI Boot Loader, ktorý inicializuje FLASH a RAM pamäť. GHI Boot Loader je miniatúrny software vyvinutý firmou GHI Electronics a je súčasťou každej dosky EMX. Nachádza sa v externej pamäti FLASH a zaberá cca 1MB spolu so systémovými knižnicami. Celková veľkosť FLASH pamäti je 4MB. Zvyšné 3MB sú rezervované pre uloženie preloženého riadeného kódu. Z toho 128KB je rezervovaných pre CLR. GHI Boot Loader môže byť použitý na aktualizáciu TinyBooter, avšak samotný GHI Boot Loader je „vypálený“, takže sa nedá zmeniť. Nakoniec po celej inicializácii sa hľadá platný TinyBooter. [14]

TinyBooter je spustený z pamäte RAM, následne preberá hardware a pripravuje prostriedky, ktoré majú byť spracovávané EMX Firmwarom. TinyBooter je uložený ako GHI Boot Loader v externej FLASH, len v inej časti. TinyBooter pochádza z dielne

Microsoft. Okrem spomenutých vecí tento softvér slúži pre aktualizáciu vyššej vrstvy EMX Firmware (TinyCLR). [14]

Nakoniec je spustený EMX Firmware, čo je hlavnou časťou embedded softvéru, ktorý obsahuje .NETMC jadro (CLR) z HAL a PAL drivermi umožňujúce prístup k hardwaru pomocou riadeného kódu. Vývojár následne môže nahráť a ladiť vlastný riadený kód priamo na doske EMX s využitím Visual Studia. [14]

Každú vrstvu je možné upravovať samostatne. Na vývojovej doske sa nachádza prepínač, ktorým sa dajú prepínať módy EMX dosky. Po pripojení dosky k počítaču pomocou USB kábla a nastavenie prepínačov do požadovanej polohy, je doska pripravená a očakáva riadiace príkazy. Pre editáciu jednotlivých softvérových vrstiev slúži nástroj MFDeploy, ktorý je súčasťou .NETMF SDK. Pomocou tohto nástroja je možné nahradiť aktuálny softvér novým, či vývojárom upraveným. Užívateľ za normálnych okolností nikdy nemusí meniť žiadny zo spomínaných softvérov. Len v prípade, keď Microsoft či GHI vydajú vyššiu radu nových knižníc. Napríklad teraz je aktuálny .NETMF 4.3 a keď vydajú .NETMF 4.4 a bude obsahovať väčšie zmeny, bude treba aktualizovať TinyBooter aj z EMX Firmwarom.

### 3.3 .NET Gadgeteer

Okrem .NETMF, Microsoft spolu z GHI Electronics vytvorili novú platformu postavenú na doske EMX a platforme .NETMF. .NET Gadgeteer je Open-source hardvér a softvér pre jednoduchý návrh a vytvorenie rôznych zariadení, na ktorých beží .NET Micro Framework. Ide o modulárny systém, z ktorého si môžeme postaviť čo len chceme. Napríklad robota na súťaž, či jednoduchú meteostanicu. Celá .NET Gadgeteer platforma berie .NETMF flexibilitu a posúva do vyšších úrovní, čo má za následok zjednodušenie práce. [13]

Celý vývoj tejto platformy prináša nové zariadenia, ako sú základné dosky (Mainboard) a k nim veľké množstvo modulov. Moduly sa pripájajú na základné dosky 10pinovou zbernicou. Prvým oficiálnym .NET Gadgeteer zariadením je základná doska FEZ Spider (viď kapitola 4.1). Dodávané moduly sa rozdeľujú do rôznych kategórií, podľa toho čo obsahujú. Napr.: moduly pre spracovanie zvuku, obrazu, rôzne typy zberníc, či spínanie relátiek.

.NET Gadgeteer sa dá programovať vo Visual Studiu ako .NET Gadgeteer aplikácia, čo prináša veľa výhod, alebo klasická .NETMF aplikácia. Obidva typy sú kompatibilné.

### 3.4 GHI, Premium a OSHW Namespace

Ako bolo spomenuté v predošlých kapitolách, GHI Electronics je jedným z dodávateľom hardvéru z podporou .NETMF. Medzi ich prvé výrobky patrí doska EMX Module. Táto doska obsahuje všetko potrebné pre beh .NETMF. Štandardný .NETMF obsahuje veľa rôznych podporných knižníc pre správu s HAL/PAL vrstvou. Celý menný priestor knižníc .NETMF je *Microsoft.SPOT*.

GHI ako výrobca tejto dosky si dodal na trh vlastné podporné knižnice (Premium Library) z dôvodu, že doska obsahuje aj komunikačné rozhrania, ktoré základný .NETMF nepokrýva. Premium knižnice tvoria menný priestor *GHI.Premium*. Tieto knižnice sú dostupné len na Premium doskách, čiže dosky z rovnakou architektúrou ako doska EMX. Na iný typ dosiek nie sú naportované. Premium knižnice prinášajú podporu pre zbernice CAN a USB, priamy prístup k CPU registrom a spúšťanie natívneho kódu (RLP).

GHI Electronics sa snaží rozšíriť svoje pole pôsobnosti a tak vznikajú snahy portovať celý embedded softvér .NETMF na iné architektúry než len EMX. Preto GHI v spolupráci z Microsoftom portoval .NETM pre GHI Open Source Hardware (OSHW). Obsahuje jadro (CLR) a GHI OSHW knižnice. OSHW je dostupný len pre určité architektúry. Jedna z OSHW dosiek je napr.: FEZ Hydra, ktorá obsahuje skoro 3x silnejší CPU (240Mhz) ako doska EMX (72Mhz). Nevýhodou oproti doske EMX je, že FEZ Hydra nemá niektoré typy zberníc. Open Source Hardware tvoria menný priestor *GHI.OSHW*.

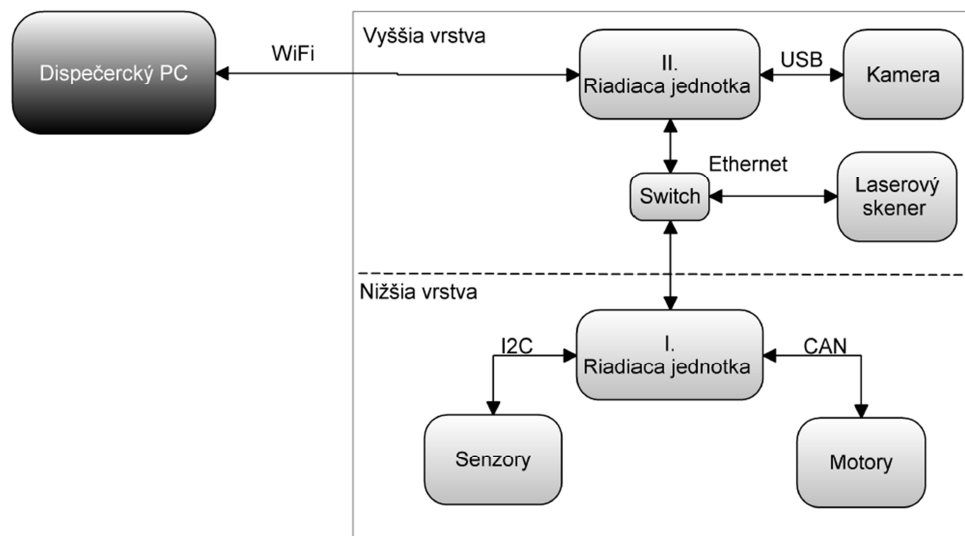
Pre .NET Gadgeteer sú tiež špeciálne knižnice, ktoré uľahčujú prácu a komunikáciu z jednotlivými modulmi. Tvoria menný priestor:

- *GHIElectronics.Gadgeteer*
- *GTM.GHIElectronics*

## **II. PRAKTICKÁ ČASŤ**

## 4 NÁVRH RIADIACEHO SYSTÉMU

Táto kapitola popisuje vývoj a návrh riadiaceho systému. Komponenty, z ktorých riadiaci systém pozostáva, sú zobrazené na obrázku 12.



Obrázok 15: Architektúra riadiaceho systému

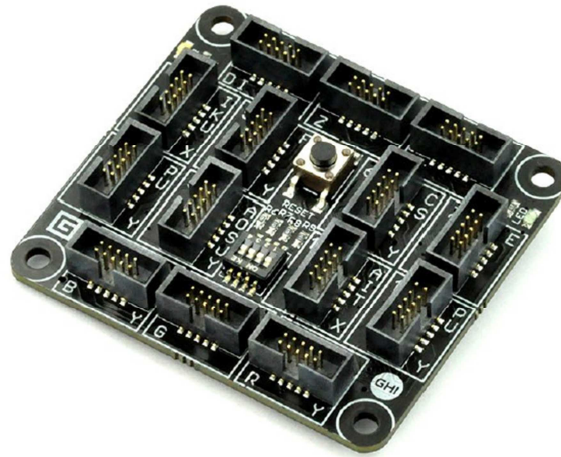
Celý riadiaci systém tvoria dve vrstvy, ktoré medzi sebou komunikujú. Každá vrstva sa stará o niečo iné. V každej vrstve je riadiaca jednotka, ktorá disponuje určitým výkonom pre spracovávanie dát v danej vrstve. V danej fáze vývoja je možné robota ovládať vzdialeným dispečerským počítačom.

Vyššia vrstva obstaráva zber dát z laserového diaľkomeru, kamery a informácie z riadiacej jednotky nižšej vrstvy. Získané dáta, spracováva, vyhodnocuje a posielajú na dispečerský počítač. Ďalšou úlohou je, že nižšej vrstve posielajú príkazy na pohyb, o akú vzdialenosť a ktorým smerom sa má robot pohybovať. Riadiacou jednotkou vo vyššej vrstve tvorí notebook, neskôr bude nahradený panel PC.

Medzi úlohy nižšej vrstvy patrí komunikácia so servozosilovacími jednotkami motorov. Obe servozosilovacie jednotky sú pripojené paralelne ku zbernici CAN a komunikácia prebieha pomocou protokolu CANopen. Riadenie motorov prebieha formou správ, ktoré nastavujú hodnoty v registroch servozosilovacích jednotiek. Ďalšou úlohou tejto vrstvy je zber dát z ultrazvukových diaľkomerov. Komunikácia prebieha po sériovej zbernici I<sup>2</sup>C. Poslednou úlohou je komunikácia s nadradenou riadiacou jednotkou. Výkonnou riadiacou jednotkou nižšej vrstvy je doska FEZ Spider, viz kapitola 4.1.

## 4.1 FEZ Spider

FEZ Spider je vývojový kit založený na EMX module od firmy GHI Electronics. Ide o prvý vývojový kit, ktorý podporuje špecifikáciu .NET Gadgeteer. Kit obsahuje všetky potrebné funkcie pre beh technológie .NET Micro Framework. Obsahuje tiež veľa portov a komunikačných rozhraní, preto bol použitý v tejto práci.



Obrázok 16: FEZ Spider Mainboard

### Základná doska obsahuje:

- 72Mhz 32-bit ARM7 procesor
- 4,5 MB FLASH a 16 MB RAM
- 76 GPIO
- 2x SPI
- I2C
- 4x UART
- 2x CAN
- 7x 10-bit analógový vstup
- 6 PWM
- USB Host/Device
- Možnosť volania natívneho kódu
- Viacej informácii je tu [14]

## 5 ANALÝZA RÝCHLOSTI LASEROVÉHO SKENERA

Ako bolo spomenuté v teoretickej časti, laserový skener SICK LMS 400 patrí medzi veľmi rýchly merací systém, ktorý využíva rýchlosti šírenia svetla. Vďaka rýchlemu procesoru, ktorý skener obsahuje dokáže spracovávať namerané dáta a rýchlo ich posielat' nadradenému systému. Tento merací systém môže byť nasadený v aplikáciách, ktoré vyžadujú spracovávanie v reálnom čase.

Na robotovi bude skener slúžiť ako hlavný rozpoznávací senzor a preto je nutné aby skener bol dostatočne rýchly. Na druhú stranu, keď bude veľmi rýchlo posielat' dáta môže zabrat' väčšiu kapacitu komunikačného kanálu. Hlavnou požiadavkou je, aby nadradená jednotka mala v každom okamžiku aktuálne dáta zo senzoru.

Rýchlosť skenovania sa nastavuje pomocou skenovacej frekvencie, uhlového rozlíšenia a plochy, ktorú má skener pokryť. Uhlové rozlíšenie ovplyvňuje hlavne aký veľký telegram bude skener posielat'. Telegram obsahuje celý jeden sken. Čiže pri uhlovom rozlíšení  $0,25^\circ$  a ploche  $70^\circ$ , ktorú skener pokrýva bude jeden sken obsahovať 280 hodnôt. Na veľkosť telegramu (skenu) je závislá aj plocha, ktorú skener pokrýva. Maximálna plocha je však  $70^\circ$ . Na obrázku 17 je zobrazený záznam komunikácie skeneru z počítačom. Pri uhlovom rozlíšení  $0,25^\circ$ , ploche pokrytia  $70^\circ$  a frekvencii skenovania 360 Hz. Nie sú ani nastavené žiadne filtre pre korekciu nameraných dát.

| No.  | Time         | Source      | Destination | Protocol | Length |
|------|--------------|-------------|-------------|----------|--------|
| 1999 | 13:22:09.908 | 192.168.0.1 | 192.168.0.5 | TCP      | 941    |
| 2000 | 13:22:09.909 | 192.168.0.5 | 192.168.0.1 | TCP      | 54     |
| 2001 | 13:22:09.911 | 192.168.0.1 | 192.168.0.5 | TCP      | 941    |
| 2002 | 13:22:09.914 | 192.168.0.1 | 192.168.0.5 | TCP      | 941    |
| 2003 | 13:22:09.914 | 192.168.0.5 | 192.168.0.1 | TCP      | 54     |
| 2004 | 13:22:09.917 | 192.168.0.1 | 192.168.0.5 | TCP      | 941    |
| 2005 | 13:22:09.920 | 192.168.0.1 | 192.168.0.5 | TCP      | 941    |
| 2006 | 13:22:09.920 | 192.168.0.5 | 192.168.0.1 | TCP      | 54     |
| 2007 | 13:22:09.923 | 192.168.0.1 | 192.168.0.5 | TCP      | 941    |
| 2008 | 13:22:09.926 | 192.168.0.1 | 192.168.0.5 | TCP      | 941    |
| 2009 | 13:22:09.926 | 192.168.0.5 | 192.168.0.1 | TCP      | 54     |
| 2010 | 13:22:09.928 | 192.168.0.5 | 192.168.0.1 | TCP      | 79     |
| 2011 | 13:22:09.928 | 192.168.0.1 | 192.168.0.5 | TCP      | 941    |
| 2012 | 13:22:09.931 | 192.168.0.1 | 192.168.0.5 | TCP      | 941    |
| 2013 | 13:22:09.931 | 192.168.0.5 | 192.168.0.1 | TCP      | 54     |
| 2014 | 13:22:09.931 | 192.168.0.1 | 192.168.0.5 | TCP      | 78     |
| 2015 | 13:22:09.934 | 192.168.0.1 | 192.168.0.5 | TCP      | 941    |
| 2016 | 13:22:09.934 | 192.168.0.5 | 192.168.0.1 | TCP      | 54     |
| 2017 | 13:22:09.944 | 192.168.0.1 | 192.168.0.5 | TCP      | 87     |
| 2018 | 13:22:09.959 | 192.168.0.1 | 192.168.0.5 | TCP      | 87     |

Obrázok 17: Záznam komunikácie z programu Wireshark

Zo záznamu (obrázok 17) je vidieť časový rozdiel príchodu jednotlivých telegramov zo skenera (IP: 192.168.0.1). Prijaté dáta majú periódu 3ms. Z toho sa dá ľahko vyjadriť frekvencia.

$$f = \frac{1}{T[s]} = \frac{1}{0,003} = 333\text{Hz}$$

Z výpočtu je zrejme že skener posiela priemerne 333 hodnôt za 1 sekundu pri danom nastavení. Frekvencia skenovania je pritom nastavená na 360 Hz. Vyjadríme periódu požadovaného skenovania:

$$T = \frac{1}{f[\text{Hz}]} = \frac{1}{360} = 0,0028\text{s}$$

Výsledkom je že skener skenuje s periódou 2,8ms a dáta posiela s periódou 3ms. Ďalej si vyjadríme veľkosť dátového toku. Veľkosť celého prijatého rámca je 941 bytov (vis obrázok 17). Dátová časť má pritom veľkosť 887 bytov. Z toho 47 bytov je hlavička telegramu, ktorá obsahuje rôzne nastavenia skenera, zvyšných 840 bytov sú namerané hodnoty. Keď skener pošle 941 bytov 333krát za jednu sekundu dostávame výsledný dátový tok 313,4 kB/s. Rýchlosť skenovania 333 ot/s je zbytočne veľká a preto použijeme filter, ktorý má za následok určité spomalenie dátového prenosu. Parametre skenovania budú tie isté len bude nastavený filter „mean“ z parametrom 10. Ide o filter ktorý robí aritmetický priemer 10 hodnôt (skenov).

| No. | Time         | Source      | Destination | Protocol | Length |
|-----|--------------|-------------|-------------|----------|--------|
| 723 | 13:38:25.484 | 192.168.0.1 | 192.168.0.5 | TCP      | 941    |
| 724 | 13:38:25.484 | 192.168.0.5 | 192.168.0.1 | TCP      | 54     |
| 725 | 13:38:25.498 | 192.168.0.1 | 192.168.0.5 | TCP      | 941    |
| 726 | 13:38:25.525 | 192.168.0.1 | 192.168.0.5 | TCP      | 941    |
| 727 | 13:38:25.526 | 192.168.0.5 | 192.168.0.1 | TCP      | 54     |
| 728 | 13:38:25.553 | 192.168.0.1 | 192.168.0.5 | TCP      | 941    |
| 729 | 13:38:25.568 | 192.168.0.1 | 192.168.0.5 | TCP      | 941    |
| 730 | 13:38:25.568 | 192.168.0.5 | 192.168.0.1 | TCP      | 54     |
| 731 | 13:38:25.581 | 192.168.0.1 | 192.168.0.5 | TCP      | 941    |
| 732 | 13:38:25.609 | 192.168.0.1 | 192.168.0.5 | TCP      | 941    |
| 733 | 13:38:25.609 | 192.168.0.5 | 192.168.0.1 | TCP      | 54     |
| 734 | 13:38:25.637 | 192.168.0.1 | 192.168.0.5 | TCP      | 941    |
| 735 | 13:38:25.651 | 192.168.0.1 | 192.168.0.5 | TCP      | 941    |
| 736 | 13:38:25.652 | 192.168.0.5 | 192.168.0.1 | TCP      | 54     |
| 737 | 13:38:25.661 | 192.168.0.5 | 192.168.0.1 | TCP      | 79     |
| 738 | 13:38:25.664 | 192.168.0.1 | 192.168.0.5 | TCP      | 941    |
| 739 | 13:38:25.664 | 192.168.0.1 | 192.168.0.5 | TCP      | 78     |
| 740 | 13:38:25.664 | 192.168.0.5 | 192.168.0.1 | TCP      | 54     |

Obrázok 18: Záznam komunikácie z programu Wireshark

Zo záznamu komunikácie (obrázok 18) je vidieť, že rozdiel príchodov jednotlivých nameraných telegramov je priemerne 28 ms. Celé oneskorenie je zapríčinené zmeraným 10 skenov a následného priemerovania. Z toho vychádza, že za 1 sekundu pošle priemerne 35 telegramov a to tvorí dátový tok 32,93 kB/s.

## 6 SOFTVÉROVÁ IMPLEMENTÁCIA

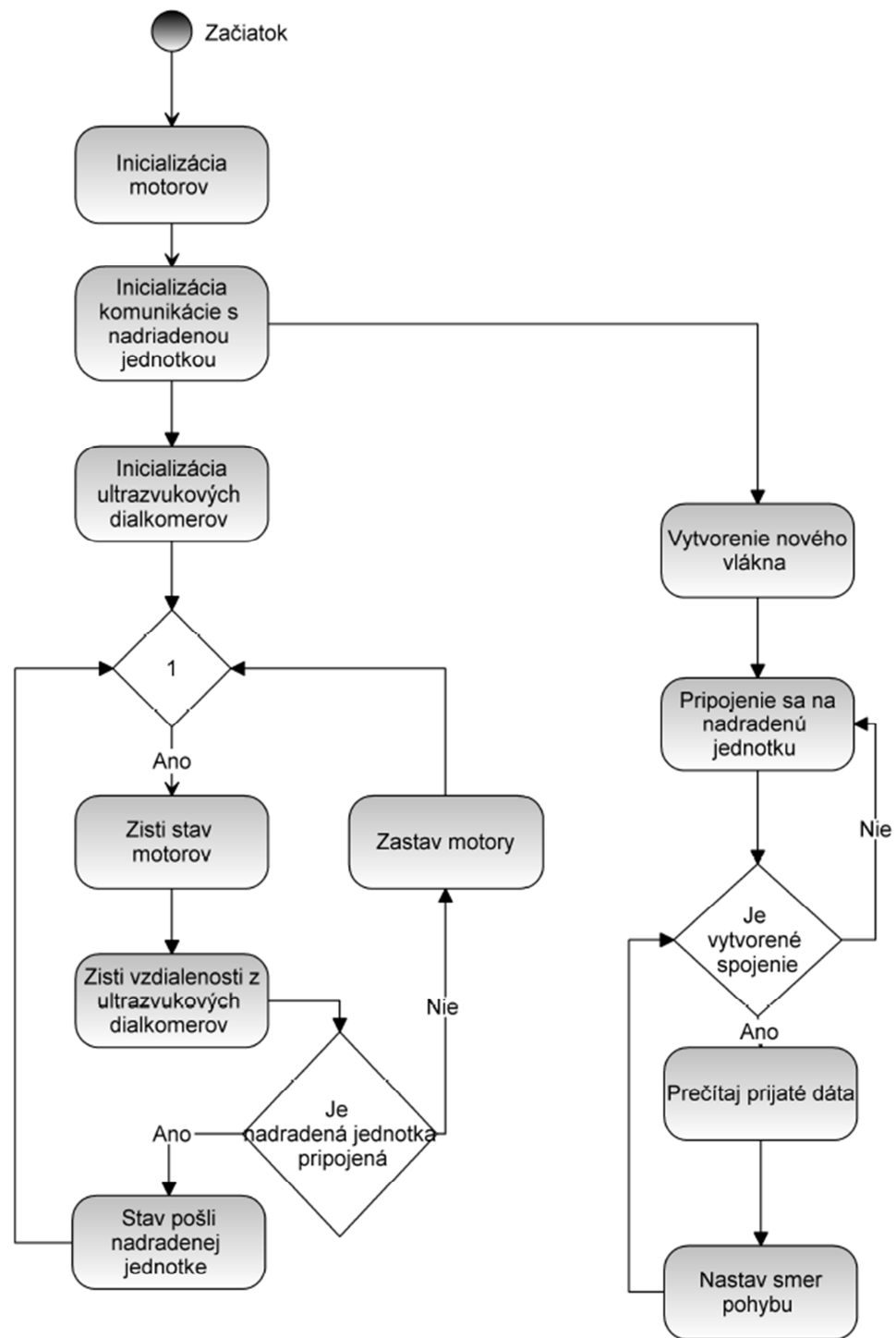
Pre tvorbu celého softvéru bolo použité vývojové prostredie (IDE) Microsoft Visual Studio 2010 Professional. Toto prostredie integruje veľké množstvo nástrojov pre jednotlivé vývojové technológie od spoločnosti Microsoft. Jednou z hlavných výhod vývojového prostredia je zvýrazňovanie syntaktických chýb a inteligentný našepkávač. .NET aplikácie je možné programovať vo viacerých jazykoch, avšak vždy ide o objektový jazyk. Ja som si vybral C# z toho dôvodu, že programovanie .NET Micro Framework aplikácií vo verzii 4.0 sa dá len v jazyku C# a knižnica pre laserový skener ja tiež vytvorená v C#.

Celá implementácia riadiaceho softvéru je navrhnutá tak, aby ku každému zariadeniu (modul) bolo pristupované cez vlastné aplikačné rozhranie (API). Myslené je to tak, že ku komunikácii s každým zariadením v riadiacom systéme bude vytvorená samostatná trieda s požadovanými metódami. Každá trieda, ktorá tvorí tvoriaca aplikačné rozhranie bude preložená ako dynamická knižnica (\*.dll).

### 6.1 Softvérové riešenie pre FEZ Spider

Celá aplikácia využíva dve vlákna, kde každé vlákno spravuje vlastnú komunikáciu z iným zariadením. Na hlavnom vlákne sa nachádza celá správa všetkých tried (zariadení), spracovávanie dát zo senzorov a komunikácia z nadradenou jednotkou. V prvom rade sa na začiatku inicializujú triedy, ktoré predstavujú jednotlivé komponenty. Následne sa inicializujú komunikačné zbernice, ktoré využívajú komponenty.

Ďalej je vytvorené nové vlákno, ktoré sa stará o komunikáciu z nadradenou riadiacou jednotkou. Vlákno obstaráva samotné pripojenie a následný príjem dát. Ako náhle sa jednotka odpojí, alebo v komunikácii príde k chybám, čo bude mať za následok výpadok celej komunikácie, príde znova k novému naviazaniu komunikácie. Prijaté správy nastavujú požadovaný smer pohybu robota. Nakoniec v hlavnom vlákne sa zisťujú údaje z ultrazvukových senzorov a rýchlosť zo servozosilovacích jednotiek a potom sa posielajú nadradenej riadiacej jednotke. Pre znázornenie celého behu embedded aplikácie je na obrázku 19 zobrazený vývojový diagram.



Obrázok 19: Vývojový diagram aplikácie

Na obrázku 19 je znázornený stručný vývojový diagram, ktorý zobrazuje architektúru embedded aplikácie. V diagrame však nie sú zobrazené presné postupy algoritmov, synchronizácia vlákien ani spracovávanie nameraných dát. V nasledujúcej časti budú popísané jednotlivé postupy a algoritmy.

### 6.1.1 Popis tried

Táto časť popisuje jednotlivé triedy tvoriace aplikačné rozhranie, ktoré sú základom riadiaceho softvéru nižšej vrstvy. Vďaka tomu, že .NETMF využíva objektový prístup k hardwaru, prácu to trochu uľahčilo, lebo sa nemuseli vytvárať žiadne HAL drivery, ani priamo zasahovať do registrov.

Jadrom aplikácie je hlavná trieda Program. V tejto triede sa ako prvá volá metóda *Main*, podobne ako v každom programovacom jazyku. Metóda následne volá funkcie pre inicializovanie jednotlivých komponentov. Pri inicializácii sa vytvárajú inštancie objektov z následnou konfiguráciou. Po inicializáciách komponentov je vytvorené nekonečná slučka, v ktorej sa posielajú stavy zo senzorov a motor nadradenej jednotke :

```
mvControl.SendBroadcastTimer();
MotorInfo info = mvControl.MotorInfo;
sendBuffer = new byte[14]; // 2B + 2B + 5*2=10
sendBuffer[0] = (byte)(sendBuffer.Length - 1);
sendBuffer[1] = info.motorL;
sendBuffer[2] = info.motorR;
byte[] tmpBuffer = Convert.GetBytes(info.speed);
sendBuffer[3] = tmpBuffer[0];
sendBuffer[4] = tmpBuffer[1];

int index = 4;
lock(Distances){
    for (int i = 0; i < Distances.Length; i++)
    {
        tmpBuffer = Convert.GetBytes((short)Distances[i]);
        sendBuffer[index++] = tmpBuffer[0];
        sendBuffer[index++] = tmpBuffer[1];
    }
}
if (ethernetConnection)
{
    ethernet.Write(sendBuffer);
}
else
{
    if(info.movement != RobotMovement.STOP)
        MotorManagement.MoveStop();
    Thread.Sleep(200);
}
```

Najprv sa zistí stav jednotlivých motorov. Štruktúra *MotorInfo* obsahuje informácie o motoroch ako sú rýchlosť a stav (run, stop, error, ready). Pomocou časovača sú zisťované údaje s ultrazvukových senzorov a následne sú všetky zistené dáta prevedené

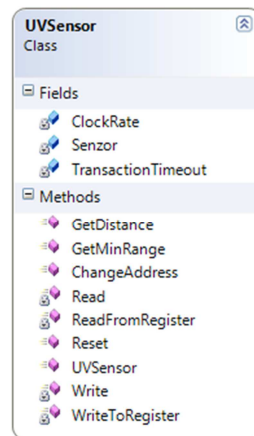
do správy, ktorá sa posiela nadradenej riadiacej jednotke. Ako náhle sa riadiaca jednotka odpojí, dáta sa prestanú posielať a robot zastaví.

Aplikačné rozhrania sú tvorené nasledujúcimi triedami:

- UVSensor
- Ethernet
- MotorController
- Convert

### 6.1.1.1 Trieda UVSensor

Prvá trieda, ktorá bude popísaná slúži pre prácu z ultrazvukovými senzormi a jej štruktúra je zobrazená na obrázku 20.



Obrázok 20: Trieda UVSensor

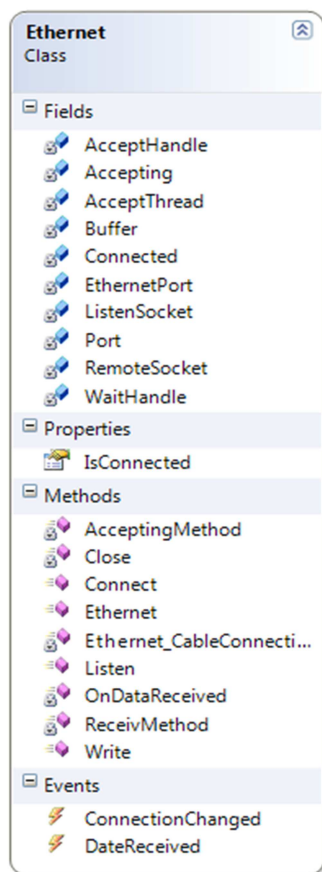
Členské premenné triedy obsahujú nastavenú komunikačnú frekvenciu, periódu merania ultrazvukového senzoru a inštanciu triedy, ktorá abstrahuje I2C zbernicu. Konštruktor nastavuje komunikačnú frekvenciu a inicializuje prístup ku I2C zbernici. Jedna z hlavných verejných metód je *getDistance*, ktorá vracia nameranú vzdialenosť z ultrazvukového senzoru podľa adresy, ktorú prína metóda ako parameter.

```
public int getDistance(byte address, RangingMode mode)
{
    int result = 0;
    byte[] buffer = new byte[2];
    i2cDevice.Config = new I2CDevice.Configuration((byte)(address>>1),
clockRate);
    WriteToRegister(0x00, (byte)mode);
    Thread.Sleep(70);
    ReadFromRegister(0x02, buffer);
    result = (buffer[0] << 8) + buffer[1];
    return result;
}
```

Pre komunikáciu s ultrazvukovým senzorom po I2C zbernici je treba vytvoriť konfiguráciu, v ktorej sa nastavuje adresa a frekvencia. Zvyčajne ide o 7-bitovú adresu takže 8-bitové číslo sa musí previesť tak, že sa spraví bitový posun do ľavej strany o 1 bit. Následne je vytvorená transakcia zápisu, ktorá obsahuje adresu registra a samotné dáta, ktoré majú byť zapísané do uvedeného registru ultrazvukového senzora. Trieda ešte obsahuje metódy pre zmenu adresy senzora, reštart senzora a zistenie aký je minimálny merací rozsah daného senzora.

### 6.1.1.2 Trieda Ethernet

Druhá trieda obstaráva komunikáciu z nadradenou riadiacou jednotkou pomocou ethernetu.

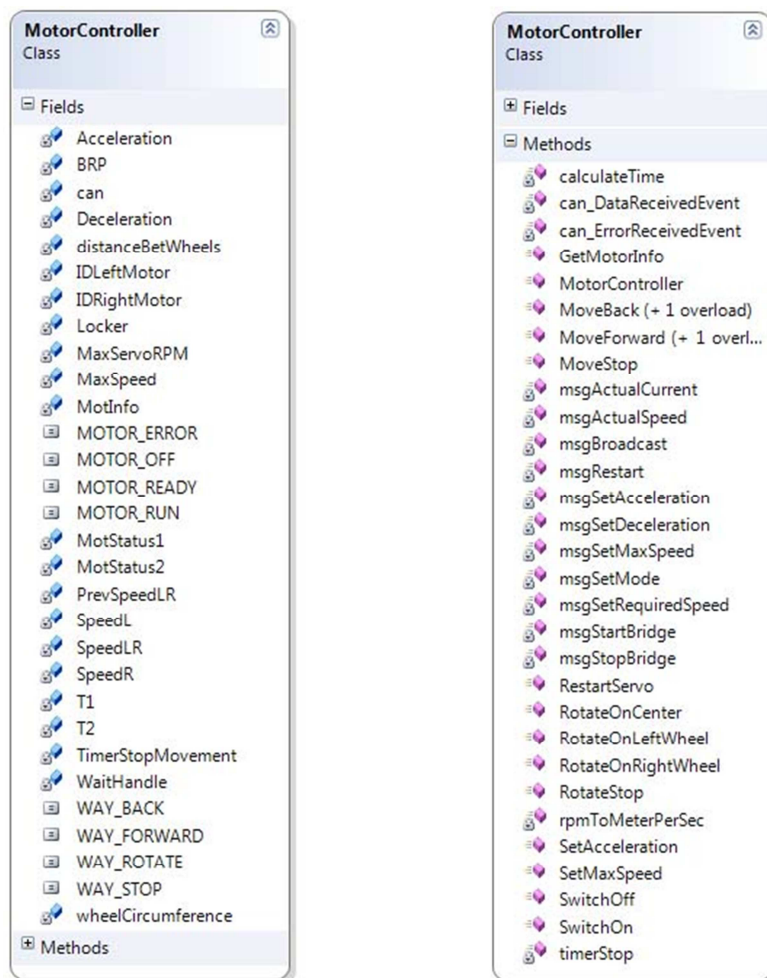


Obrázok 21: Trieda Ethernet

Na obrázku 21 je zobrazená štruktúra triedy *Ethernet*. Táto trieda obsahuje metódy pre posielanie a čítanie dát nadradenej jednotke. V konštruktore sa inicializuje fyzické ethernetové rozhranie, nastaví sa IP adresa, maska siete a sieťová brána. Ďalej sa vytvorí socket, ktorý na novovytvorenom vlákne čaká na prichádzajúcu požiadavku od nadradenej jednotky. Po vytvorení spojenia s nadradenou jednotkou informuje hlavnú triedu pomocou

udalosti a príjem dát z nadradenej jednotky môže začať. Keď prídu dáta z nadradenej jednotky, zapíšu sa do buffera a potom sa prijaté dáta pomocou obsluhy udalosti predajú hlavnej triede, v ktorej sa spracujú. Ďalšou dôležitou metódou je metóda *Write*. Táto metóda príma ako parameter pole bytov, ktoré je treba poslať nadradenej jednotke. Dáta sú skopírované do väčšieho poľa, kde prvý byte obsahuje veľkosť odosielaného rámca a za ním nasleduje celý rámec (dáta). Je to z dôvodu zabezpečenia aby strana ktorá prijíma dáta vedela, koľko dát ma prečítať. Celá komunikácia nepoužíva stream, ale pracuje priamo zo socketami. Keď dôjde k výpadku komunikácie, tak pri odosielaní dát sa chytí výnimka a spustí sa znova proces čakania na prichádzajúcu požiadavku a hlavnej triede bude informované pomocou udalosti, že je nadradená jednotka odpojená.

### 6.1.1.3 Trieda *MotorController*



Obrázok 22: Trieda *MotorController*

Na obrázku 22 je zobrazená štruktúra triedy *MotorController*, ktorá obstaráva obsluhu servozosilovacích jednotiek pomocou zbernice CAN. Čiže nastavuje rýchlosť otáčania jednotlivých kolies, akceleráciu, maximálnu rýchlosť a smer jazdy. Konštruktor

nastavuje časové konštanty a inicializuje zbernicu CAN. Následne sa po sebe volajú metódy na nastavenie módu, zapnutie silového mostu, akcelerácie a maximálnej rýchlosti servozosilovacích jednotiek. Potom sa môžu volať jednotlivé metódy pre nastavovanie rýchlostí a smeru pohybu. Aby sa robot pohyboval rovno vpred či vzad je pre každý smer jedna metóda, ktorá prijíma ako parameter požadovanú rýchlosť. Preťaženou metódou sa môže nastaviť aj vzdialenosť, o ktorú sa má robot posunúť. Druhým typom pohybu je rotácia. Robot sa dokáže otáčať na mieste okolo svojej osi požadovanou rýchlosťou, alebo sa môže otočiť doprava či doľava o určitý počet stupňov. Robot dokáže zabáčať kontinuálne, čiže počas jazdy dokáže zmeniť smer. Celé riadenie prebieha volaním verejných metód, ktoré majú vlastnú logiku na výpočet dráhy a času. Po následnom spracovaní sú volané vnútorné metódy, ktoré posielajú priamo servozosilovacím jednotkám požadovanú rýchlosť pohybu. Princíp nastavovania servozosilovacích jednotiek prebieha pomocou správ. Generovanie správy na nastavenie požadovanej rýchlosti je zobrazené v nasledujúcej časti zdrojového kódu:

```
private CAN.Message[] msgSetRequiredSpeed(uint id, int revolutionsPerMin)
{
    float revPerOneSecond = (float)65536 * (float)262.144;
    int revolutions = ((revolutionsPerMin) / 60);
    int value = (int)System.Math.Ceiling((revolutions * revPerOneSecond));
    byte[] speedArray = Convert.GetBytes(value);
    CAN.Message[] list = new CAN.Message[1];
    list[0] = new CAN.Message();
    list[0].ArbID = 1536 + id; // address of device
    list[0].Data[0] = 0x23; // write or read
    list[0].Data[1] = 0x94; // register lowest byte
    list[0].Data[2] = 0x01; // register highest byte
    list[0].Data[3] = 0x02; // data length
    list[0].Data[4] = speedArray[0]; // data lowest byte
    list[0].Data[5] = speedArray[1]; // data
    list[0].Data[6] = speedArray[2]; // data
    list[0].Data[7] = speedArray[3]; // data highest byte
    list[0].DLC = 8;
    return list;
}
```

Správa obsahuje adresu serovozilovacej jednotky(ArbID) a za ňou nasleduje dátová časť správy, ktorá obsahuje adresu registra do ktorého zapisuje a hodnotu, ktorú zapisuje. Dôležitou časťou je nastavenie dĺžky dátovej časti. Odoslanie správy je zobrazené na nasledujúcej časti kódu:

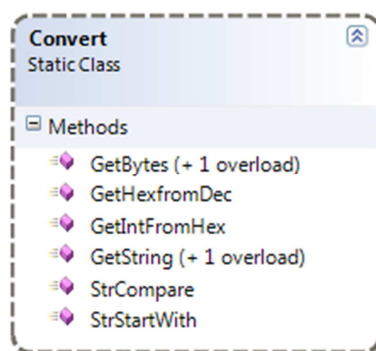
```

public void moveForward(short rpm, double distanceInMeters){
    if (rpm > 0 && rpm < maxServoRPM)
    {
        if (distanceInMeters > 0)
        {
            if (timer1 != null)
            {
                timer1.Dispose();
                timer1 = null;
            }
            prevSpeedLR = rpm;
            double accTime = calculateTime(rpm, distanceInMeters);
            int timeInMilis = (int)(accTime * 1000);
            Debug.Print(accTime.ToString());
            can.PostMessages(msgSetRequiredSpeed(IDLeftMotor, -rpm), 0,
msgSetRequiredSpeed(IDLeftMotor, -rpm).Length);
            can.PostMessages(msgSetRequiredSpeed(IDRightMotor, rpm), 0,
msgSetRequiredSpeed(IDRightMotor, rpm).Length);
            timer1 = new Timer(new TimerCallback(timerStop), null, timeInMilis,
0);
        }
    }
}

```

Touto metódou sa nastavuje vzdialenosť, ktorú musí robot uraziť zadanou rýchlosťou. Dôležité je spočítať čas, za ktorý danú vzdialenosť dosiahne. Do výpočtu je nutné zahrnúť aj zrýchlenie a spomalenie. Motorom sa nastaví požadovaná rýchlosť a keď ubehne vypočítaný čas motory sa stopnú.

#### 6.1.1.4 Trieda Convert

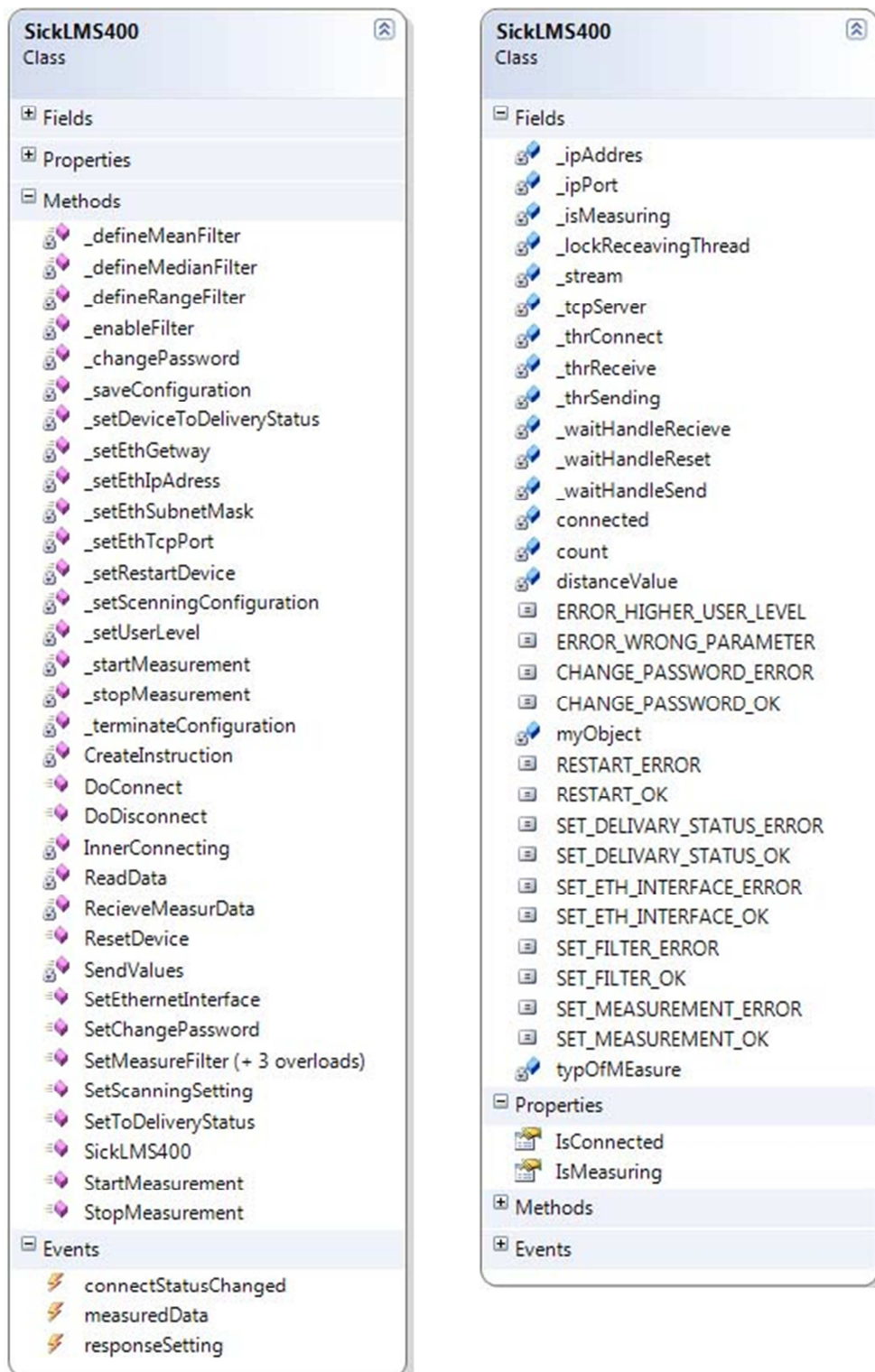


Obrázok 23: Trieda Convert

Trieda Convert je statická trieda, ktorá obsahuje metódy na prevod číselných sústav z hexadecimálnej do dekadickéj sústavy a zase naopak. Obsahuje metódy na prevod typov, napríklad z pola bytov na integer a naopak, pola bytov na reťazec, alebo porovnávanie reťazcov. Je to z toho dôvodu, že .NETMF 4.1 neobsahuje menný priestor Convert, ako klasický plný .NET Framework.

## 6.2 Softvérová riešenie pre SICK LMS 400

Táto časť sa zaoberá návrhom a vývojom softvérovej knižnice pre komunikáciu z laserovým skenerom SICK LMS 400.



Obrázok 24: Trieda SickLMS400

Na obrázku 24 je zobrazená štruktúra triedy *SickLMS400* z menného priestoru *Sick*. Táto trieda tvorí aplikačné rozhranie k laserovému skeneru SICK LMS 400. Trieda implementuje obsluhu pre všetky funkcie, ktoré skener ponúka. Vo väčšine funkcií ide o asynchrónne metódy, lebo funkcie, ktoré trieda obsahuje sú paralelizované. Vďaka využitiu vlákien sa užívateľ nemusí starať o „vytuhnutie“ pri vytváraní komunikácii, alebo meraní.

Konštruktor prijíma IP adresu skenera a port, na ktorom prijíma požiadavky. Následne je možné zavolať metódu *DoConnect*, ktorá vytvorí nové vlákno z následným nastavením a pripojením sa na IP adresu skenera. O úspešnosti či neúspešnosti sa užívateľ dozvie udalosťou, ktorá je volaná. Po korektnom vytvorení spojenia je vytvorené nové pracovné vlákno, na ktorom beží celá komunikácia. Trieda využíva toho, že skener sám od seba nevysiela, ale len odpovedá na požiadavky to znamená, že keď skener nemá odpovedať pracovné vlákno je uspaté. Princíp pracovného vlákna je, že je aktívne len v prípade očakávania odpovede od skenera. V inom prípade je vlákno uspaté, aby nevyťažovalo systémové prostriedky. Pri odoslaní požiadavky skeneru, je vlákno prebudené (aktívne) a následne je prijatá odpoveď. Po prijatí odpovedi a jej spracovaní je pracovné vlákno zasa uspaté.

```
public void DoConnect() {
    if (!connected)
    {
        try
        {
            _thrConnect = new Thread(new ThreadStart(InnerConnecting));
            _thrConnect.Name = "Connect_hread";
            _thrConnect.Start();
        }
        catch (OutOfMemoryException)
        {
            Console.WriteLine("DoConnect OutOFMEemoryException");
        }
        catch (ThreadStateException)
        {
            Console.WriteLine("DoConnect ThreadStateException");
            _thrConnect = new Thread(new ThreadStart(InnerConnecting));
            _thrConnect.Name = "Connect_hread";
            _thrConnect.Start();
        }
    }
}
```

Najdôležitejšou metódou je spustenie a stopnutie merania. Na spustenie merania slúži jeden príkaz, ktorý sa pošle skeneru a skener po jeho prijatí začne posiela namerané dáta. Skener dáta posiela, až pokiaľ neprijme príkaz na stopnutie merania.

Ďalšie metódy, ktoré trieda implementuje slúžia pre nastavenie parametrov skenovania, ethernetového rozhrania a reštartovania do výrobného nastavenia. Medzi parametre skenovania patrí skenovacia frekvencia, uhlové rozlíšenie, veľkosť meracej plochy (v °) a rôzne typy filtrov. Pre niektoré typy nastavenia je nutné zadať heslo podľa bezpečnostnej úrovne. Pre uloženie nastavenia je nutné poslať ďalší príkaz a po nastavení ethernetového rozhrania je nutné reštartovať skener, stačí použiť softwarový reštart. Podrobnejší popis filtrov a nastania sa dá nájsť tu [8]. Nasledujúca časť zdrojového kódu zobrazuje metódu na nastavenie skenovacích parametrov.

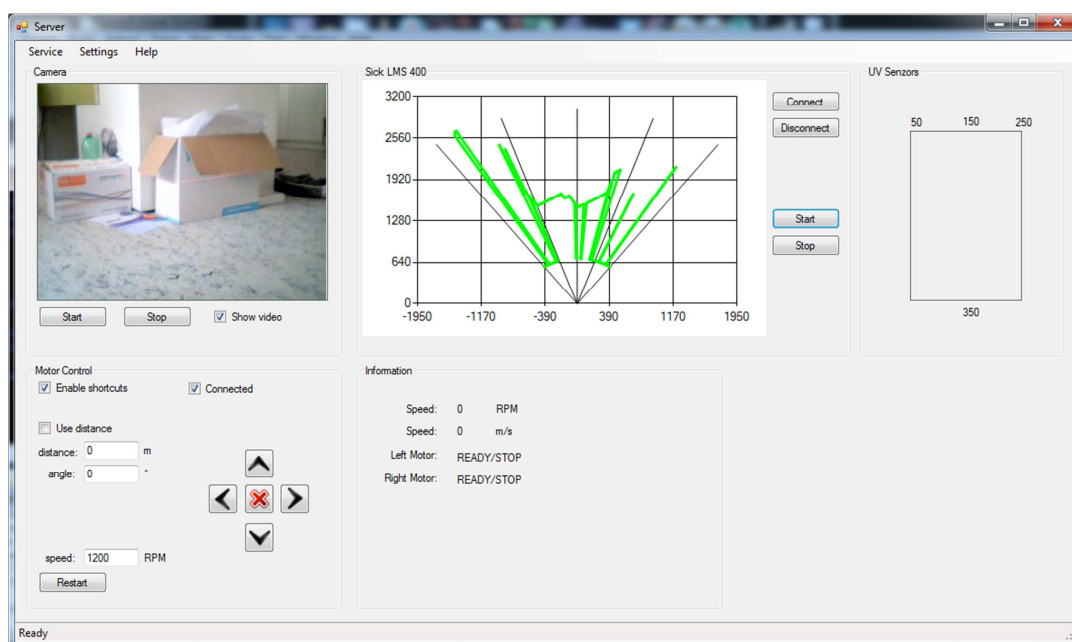
```
public bool SetScanningSetting(string passwordAuthorisedClient, UInt32
frequency, float angularResolution, float startAngle, float lengthArea)
{
    bool result = false;
    if (connected && (angularResolution >= +0.1 && angularResolution <= +1.0) &&
!_isMeasuring)
    {
        if ((startAngle >= +55.0 && startAngle <= +124.0) && (lengthArea >= 0.0
&& lengthArea <= +70.0) && (startAngle + lengthArea <= 125))
        {
            _waitHandleSend.Reset();
            _thrSending = new Thread(delegate()
            {
                _setUserLevel(UserLevel.authorisedClient,
passwordAuthorisedClient);
                _waitHandleSend.WaitOne();
                _setScenningConfiguration(frequency, angularResolution, startAngle,
lengthArea);
                _waitHandleSend.WaitOne();
                _setUserLevel(UserLevel.authorisedClient,
passwordAuthorisedClient);
                _waitHandleSend.WaitOne();
                _saveConfiguration();
                _waitHandleSend.WaitOne();
                _terminateConfiguration();
                _waitHandleSend.WaitOne();
                if (responseSetting != null)
                {
                    responseSetting(this, SET_MEASUREMENT_OK);
                }
            });
            _thrSending.Name = "send_thread";
            _thrSending.Start();
            result = true;
        }
    }
    return result;
}
```

## 7 OVLÁDANIE ROBOTICKEJ PLATFORMY

Ako bolo spomenuté, táto práca sa zaoberá návrhom softvérových knižníc pre definovanú robotickú platformu. Samotné knižnice bez riadiacich povelov, nedokážu ovládať nič. Riadiace povelov dáva nadradená riadiaca jednotka. Softvérové vybavenie nadradenej riadiacej jednotky nie je zadaním diplomovej práce, avšak pre prezentovanie dosiahnutých výsledkov musela byť vytvorená. Vytvorená aplikácia pre ovládanie mobilnej robotickej platformy, ktorá tvorí softvérové vybavenie nadradenej riadiacej jednotky podporuje len základné ovládacie prvky. Aplikácia využíva práve vytvorené aplikačné rozhrania ku zariadeniam ako je laserový skener a riadiaca jednotka nižšej vrstvy. Nadradená jednotka komunikuje s nižšou riadiacou jednotkou formou príkazov. Taktiež nadradená riadiaca jednotka získava z nižšej riadiacej jednotky aktuálne informácie o motoroch rýchlosti pohybu, ale aj prijíma dáta zo senzorov. Nadradená jednotka nedisponuje žiadnou vnútornou logikou ani žiadnym autonómnym systémom pre ovládanie robotickej platformy, čiže nedokáže sa vyhýbať prekážkam či obchádzať ich. V danej fáze vývoja sa dá robotická platforma ovládať viacerými spôsobmi.

### 7.1 Miestne ovládanie

Jedným z možných spôsobov ovládanie je miestne ovládanie pomocou grafického rozhrania, ktoré softvérové vybavenie nadradenej jednotky ponúka. Využíva sa priameho fyzického kontaktu s nadradenou riadiacou jednotkou.

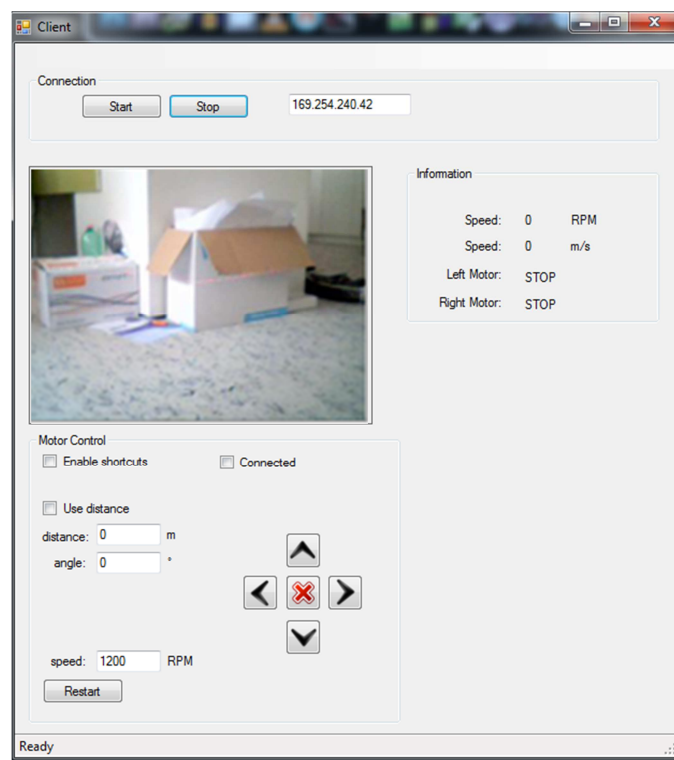


Obrázok 25: Hlavné okno aplikácie

Toto ovládanie je možné len na určitú vzdialenosť, resp. vzdialenosť závisí na dĺžke použitého ethernetového kábla. Tento typ ovládania bol využitý pri vývoji a testovaní softwarového vybavenia celého robotického systému. Užívateľ ovláda smer pohybu pomocou klasických smerových šípok na klávesnici (Up, Down, Left, Right). Grafické rozhranie slúži pre zobrazenie informácií, ktoré má nadradená riadiaca jednotka k dispozícii. V budúcnosti so získaných informácií bude nadradená jednotka spracovávať smer pohybu, prekážky, rozpoznávanie priestoru, čiže pôjde o autonómny systém.

## 7.2 Vzdialené ovládanie pomocou PC

Ďalšou možnosťou ovládania je pomocou vzdialeného dispečerského počítača. Do softwarového vybavenia nadradenej riadiacej jednotky implementovaná služba vzdialeného ovládania. Dispečerský počítač sa pripojí na nadradenú jednotku a následne spolu môžu komunikovať, zasielať riadiace príkazy a prijímať dáta zo senzorov. Pre tieto účely bola vytvorená aplikácia typu klient, ktorá komunikuje z nadradenou jednotkou pomocou WiFi technológie. Softvér nadradenej jednotky v tomto prípade vystupuje ako server.

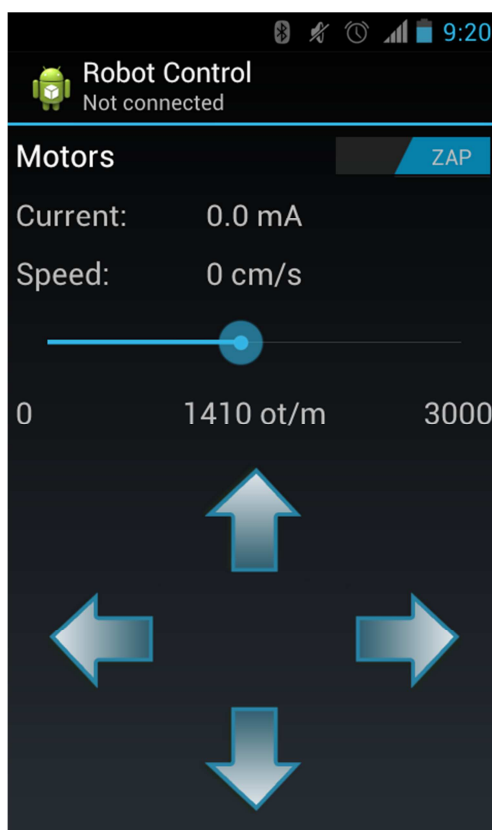


Obrázok 26: Hlavné okno klientskej aplikácie

Táto aplikácia má podobné grafické rozhranie aj ovládanie ako aplikácia v nadradenej riadiacej jednotke mobilného robota.

### 7.3 Vzdialené ovládanie mobilným telefónom

Posledným typom ovládania robotickej platformy je ovládanie pomocou mobilného telefónu s operačným systémom Android. V komunikácii sa využíva Bluetooth technológia, kde k nižšej riadiacej jednotke je pripojený Bluetooth modul pomocou sériovej zbernice UART. Bluetooth je niečo ako bezdrôtová sériová zbernica. Pre tieto účely bola vytvorená aplikácia, pomocou ktorej je možné sa pripojiť na Bluetooth modul robota a následne ovládať pohyby robotickej platformy. Bluetooth modul splňuje špecifikácie tretej triedy a preto má dosah 100 m. Na dosah má veľký vplyv výkon Bluetooth adaptéru v mobilnom telefóne. Väčšina mobilných telefónov majú dosah asi 10m. Pomocou mobilnej aplikácie sa dá nastavovať smer pohybu, rýchlosť pohybu a zapínať, alebo vypínať motory. Mobilná aplikácia je zobrazená na obrázku 27.



Obrázok 27: Mobilná aplikácia

## ZÁVER

Cieľom tejto diplomovej práce bolo navrhnúť a vytvoriť softwarové knižnice pre riadiacu jednotku z využitím platformy .NET Micro Framework a knižnicu pre ovládanie laserového skenera SICK LMS 400. Platforma .NET Micro Framework je veľmi mladé softvérové riešenie embedded aplikácií, ktoré začína zasahovať aj do robotiky. Vďaka veľkej podpore rôznych typov zberníc a komunikačných protokolov z objektovým prístupom, je programovanie v platforme .NET Micro Frameworku omnoho jednoduchšie, ako v klasickom jazyku C pre mikrokontroléry.

Vytvorené knižnice tvoria aplikačné rozhrania, pomocou ktorých pristupuje riadiaci systém k jednotlivým komponentom a tým tvoria základ softvérového vybavenia riadiacich jednotiek. V danej fáze riadiaci softvér nie je schopný plnohodnotného autonómneho riadenia a preto boli vytvorené aplikácie pre vzdialené ovládanie. Jednou z možností, ktorá vznikla ako prvá, dovoľuje ovládať mobilného robota pomocou mobilného telefónu. Aplikácia je prenositeľná na každý mobilný telefón, avšak podmienkou je Android 4.0 a Bluetooth adaptér. Ďalšou možnosťou je ovládanie zo vzdialeného počítača, pomocou wifi technológie. Užívateľ u oboch možností môže nastavovať smer a rýchlosť pohybu robotической platformy.

Mobilná robotická platforma je pôvodne určená do bezpečnostného priemyslu, ako strážca budov, ale môže byť použitá pre výukové účely a na testovanie či vývoj rôznych senzorických systémov. V najbližšej dobe sa plánuje vývoj rozpoznávania priestoru a objektov okolo robota pomocou stereo zobrazenia, laseru a ultrazvuku.

## CONCLUSION

The main aim of this thesis was development of software libraries for control unit based on .NET Micro Framework and laser scanner SICK LMS 400. The platform .NET Micro Framework is one of the newest software solutions for embedded application using in robotic system. Because .NET Microsoft Framework has the large support different types of buses and communication protocols with the object access, programming in .NET Micro Framework platform is much easier then programming in classic C language for microcontroller.

Libraries provide application interface, which using by control system for accessing to every component. It is basic foundation software for controller units. Now the control software does not supports fully autonomous management and therefor was created more software application for remote control. One way of movements control is from mobile phone. The application is portable on every smartphones, which have operating system Android 4.0 and Bluetooth adapter. Next possibility is remote control from remote PC by Wi-Fi. The user can set direction and speed of movement of the robotic platform.

Mobile robotic platform is originally designed for the secure industry like a guardian of buildings interior, but now can be used for education purposes and for testing or creating different sensor systems. In the near future will be developed space and object image recognition around robot using stereo display, laser and ultrasonic systems.

**ZOZNAM POUŽITEJ LITERATÚRY**

- [1] NOVÁK, Petr. *Mobilní roboty: pohony, senzory, řízení*. Vyd. 1. Praha: BEN - technická literatura, 2004, 247 s. ISBN 80-730-0141-1.
- [2] Odometrie. *Robotika.cz* [online]. 2005 [cit. 2013-05-04]. Dostupné z: <http://robotika.cz/guide/odometry/cs>
- [3] GREGORY DUDEK, Michael Jenkin. *Computational principles of mobile robotics*. Reprint. Cambridge, Mass: Cambridge University Press, 2000. ISBN 05-215-6876-5.
- [4] ODBORNECASOPISY. *Lokomoční ústrojí mobilních robotů pro nestrojírenské aplikace*. 2002. Dostupné z: [http://www.odbornecasopisy.cz/index.php?id\\_document=28494](http://www.odbornecasopisy.cz/index.php?id_document=28494)
- [5] TG DRIVES. *Servopohony*. 2010. Dostupné z: [http://www.tgdrives.cz/fileadmin/user\\_upload/Servopohony/TG\\_katalog\\_servomotoru\\_tgn\\_CZ\\_web.pdf](http://www.tgdrives.cz/fileadmin/user_upload/Servopohony/TG_katalog_servomotoru_tgn_CZ_web.pdf)
- [6] TGA-24. *TG Drives* [online]. 2010 [cit. 2013-05-04]. Dostupné z: <http://www.tgdrives.cz/digitalni-servozesilovace/tga-24/>
- [7] SRF02 Ultrasonic range finder. *Robot-electronics* [online]. 2005 [cit. 2013-05-05]. Dostupné z: <http://www.robot-electronics.co.uk/htm/srf02tech.htm>
- [8] SICK. *LMS400 Laser Measurement System* [online]. 2005 [cit. 2013-05-05]. Dostupné z: <http://www.robotsinsearch.com/sites/default/files/products/literature/LMS4xx/Operating%20Instructions.pdf>
- [9] LIDAR. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2013-05-06]. Dostupné z: <https://en.wikipedia.org/wiki/LIDAR>
- [10] *Software and hardware specification for area segmentation with laser scanner SICK LMS 400* [online]. 2011 [cit. 2013-05-05]. Dostupné z: <http://www.universitypress.org.uk/journals/saed/20-840.pdf>
- [11] KÜHNER, Jens. *Expert .NET Micro Framework*. New York: Distributed to the book trade worldwide by Springer-Verlag New York, c2008, xxi, 424 p. ISBN 15-905-9973-X.

- [12] GHI ELECTRONICS. *Beginners Guide to Porting NETMF* [online]. 2012 [cit. 2013-05-09]. Dostupné z: <http://www.ghielectronics.com/downloads/FEZ/Beginners%20Guide%20to%20Porting%20NETMF.pdf>
- [13] GHI ELECTRONICS. *Beginners Guide to C# and the .NET Micro Framework* [online]. 2012 [cit. 2013-05-09]. Dostupné z: <http://www.ghielectronics.com/downloads/FEZ/Beginners%20guide%20to%20NETMF.pdf>
- [14] GHIELECTRONICS. *EMX User Manual* [online]. 2011 [cit. 2013]. Dostupné z: [http://www.ghielectronics.com/downloads/EMX/EMX\\_User\\_Manual.pdf](http://www.ghielectronics.com/downloads/EMX/EMX_User_Manual.pdf)
- [15] NASH, Trey. *C# 2010: rychlý průvodce novinkami a nejlepšími postupy*. Vyd. 1. Brno: Computer Press, 2010, 624 s. ISBN 978-80-251-3034-6.
- [16] *Microsoft .NET Micro Framework* [online]. 2012 [cit. 2013-05-11]. Dostupné z: [robodoupe.cz/wp-content/uploads/2012/06/MFP2012.pptx](http://robodoupe.cz/wp-content/uploads/2012/06/MFP2012.pptx)
- [17] *Moduly pro Ethernet komunikaci s .NET Micro Framework. Mespraha* [online]. 2007 [cit. 2013-05-12]. Dostupné z: <http://www.mespraha.cz/index.php/vechny-lanky/48-moduly-pro-ethernet-komunikaci-s-net-micro-framework>

**ZOZNAM POUŽITÝCH SYMBOLOV A ZKRATIEK**

|                  |   |
|------------------|---|
| LMS              | Leasure Measurement System – laserový merací systém   |
| CAN              | Controller Area Network – je zbernica pre vnútornú komunikačnú sieť senzorov a riadiacich jednotiek   |
| I <sup>2</sup> C | Integer-Integrated Circuit – počítačová sieťová zbernica pre pripojovanie periférií na základnú dosku |
| CLR              | Common language runtime – behové prostredie .NET Framework  |
| VS               | Microsoft Visual Studio – vývojové štúdio pre Microsoft .NET Framework                                |
| SDK              | Software Development Kit – sada nástrojov pre vývoj softvéru  |
| PK               | Porting Kit – nástroje pre portovanie embedded aplikácií .NET Micro Framework                         |
| GPIO             | General Purpose Input/Output – vstupné/výstupné porty   |
| SSL              | Secure Socket Layer – zabezpečuje komunikáciu medzi transportnou a aplikačnou vrstvou                 |
| HAL              | Hardware Abstract Layer – softvérová vrstva, ktorá beží priamo na hardvéru                            |
| RAM              | Random Access Memeory – Pamäť z priamym prístupom   |
| OSHW             | Open Source Hardware – vývoj hardvéru z otvorenou licenciou   |

**ZOZNAM OBRÁZKOV**

|   |    |
|---|----|
| Obrázok 1: Model diferenciálneho podvozku [4].....        | 13 |
| Obrázok 2: Model Ackermanovho podvozku [4].....           | 13 |
| Obrázok 3: Model všesmerového podvozku [2].....           | 14 |
| Obrázok 4: Mobilná robotická platforma .....              | 17 |
| Obrázok 5: Servomotor TGN2-0054 [5] .....                 | 17 |
| Obrázok 6: Servozosilovač TGA-24 [6].....                 | 18 |
| Obrázok 7: Ultrazvukový diaľkomer SRF02 [7] .....         | 19 |
| Obrázok 8: SICK LM 400 .....                              | 20 |
| Obrázok 9: Operačné pole pre SICK LMS 400.....            | 21 |
| Obrázok 10: Všeobecný princíp laserového diaľkomeru ..... | 22 |
| Obrázok 11: Príkaz pre začiatok merania .....             | 23 |
| Obrázok 12: Architektúra .NETMF[16] .....                 | 27 |
| Obrázok 13: Preklad riadeného kódu [17].....              | 29 |
| Obrázok 14: Vrstvy bootovania.....                        | 30 |
| Obrázok 15: Architektúra riadiaceho systému.....          | 34 |
| Obrázok 16: FEZ Spider Mainboard .....                    | 35 |
| Obrázok 17: Záznam komunikácie z programu Wireshark ..... | 36 |
| Obrázok 18: Záznam komunikácie z programu Wireshark ..... | 37 |
| Obrázok 19: Vývojový diagram aplikácie .....              | 39 |
| Obrázok 20: Trieda UVSensor .....                         | 41 |
| Obrázok 21: Trieda Ethernet.....                          | 42 |
| Obrázok 22: Trieda MotorController.....                   | 43 |
| Obrázok 23: Trieda Convert .....                          | 45 |
| Obrázok 24: Trieda SickLMS400.....                        | 46 |
| Obrázok 25: Hlavné okno aplikácie.....                    | 49 |
| Obrázok 26: Hlavné okno klientskej aplikácie .....        | 50 |
| Obrázok 27: Mobilná aplikácia.....                        | 51 |

**ZOZNAM TABULIEK**

|   |    |
|---|----|
| Tabuľka 1: Technické dáta pre TGN2 [5] .....          | 18 |
| Tabuľka 2: Špecifikácia SICK LMS 400 [8].....         | 20 |
| Tabuľka 3: Štruktúra komunikačného telegramu [8]..... | 22 |

## ZOZNAM PRÍLOH

### PRÍLOHA P I: CD-ROM