Fakulta elektrotechnická
Katedra aplikované elektroniky a telekomunikací

Diplomová práce

Využití ovladače Kinect k řízení mobilní platformy

Autor práce: Bc. Ondřej Lufinka
Vedoucí práce: Ing. Petr Weissar, Ph.D.

Plzeň 2015

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok: **2014/2015**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Ondřej LUFINKA**
Osobní číslo:	**E12N0051P**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Elektronika a aplikovaná informatika**
Název tématu:	**Využití ovladače Kinect k řízení mobilní platformy**
Zadávající katedra:	**Katedra aplikované elektroniky a telekomunikací**

Zásady pro vypracování:

Ke stávající mobilní platformě doplňte ovládání pomocí ovladače MS Kinect.

1. Uvažujte stávající HW konfiguraci platformy.
2. Navrhněte možnosti interakce prostřednictvím ovladače Kinect.
3. Využijte další senzorovou výbavu mobilní plaftormy.

Rozsah grafických prací:	**podle doporučení vedoucího**
Rozsah pracovní zprávy:	**30 - 40 stran**
Forma zpracování diplomové práce:	**tištěná/elektronická**
Seznam odborné literatury:	

Student si vhodnou literaturu vyhledá v dostupných pramenech podle doporučení vedoucího práce.

Vedoucí diplomové práce:	**Ing. Petr Weissar, Ph.D.** Katedra aplikované elektroniky a telekomunikací
Datum zadání diplomové práce:	**15. října 2014**
Termín odevzdání diplomové práce:	**11. května 2015**

Doc. Ing. Jiří Hammerbauer, Ph.D.
děkan

L.S.

Doc. Dr. Ing. Vjačeslav Georgiev
vedoucí katedry

V Plzni dne 15. října 2014

Abstrakt

Tato diplomová práce se zabývá projektem mobilní platformy běžícím na Katedře aplikované elektroniky a telekomunikací. Jejím úkolem je uvažovat stávající hardwarovou konfiguraci a senzorovou výbavu modelu a navrhnout propracovanější řídicí systém s využitím GPS navigace a ovladače MS Kinect. Nejprve je tedy nutné navázat na předcházející projekty a diplomové práce, které jsou uvedeny v seznamu literatury. Práce pro přehlednost popisuje počáteční hardwarový stav projektu prostřednictvím blokového schématu a odkazů na předešlé práce. Zároveň se podrobněji věnuje částem, které byly dodány v rámci stávající diplomové práce. Dále se zabývá jednotlivými moduly po softwarové stránce. Pro požadované možnosti řízení je nezbytné pozměnit předešlé neúplné nebo ne zcela funkční části kódu v mikroprocesorech a dodělat další chybějící funkce do jednotlivých senzorových jednotek a do jednotky řízení motorů. Poté se teoreticky věnuje možnostem navigačních algoritmů a postupům jednoduchého zpracování obrazu a jeho vyhodnocování pro schopnost sledování vzoru nebo postavy s využitím dostupných knihoven. Nakonec popisuje samotnou řídicí aplikaci napsanou programovacím jazykem C#. Ukazuje knihovny a funkce pro komunikaci s jednotlivými periferiemi, návrh uživatelského rozhraní a použité postupy pro řízení mobilní platformy v různých módech, kterými jsou například sledování vzoru nebo lidské postavy a autonomní navigace.

Klíčová slova

mobilní platforma, C# řídicí aplikace, autonomní navigace, sledování vzoru, sledování lidské postavy, GPS, MS Kinect, kompas, infračervený senzor, ultrazvukový senzor, CAN

Abstract

Lufinka, Ondřej. *Using Kinect for control of mobile platform* [*Využití ovladače Kinect k řízení mobilní platformy*]. Pilsen, 2015. Master thesis (in Czech). University of West Bohemia. Faculty of Electrical Engineering. Department of Applied Electronics and Telecommunications. Supervisor: Petr Weissar

---

This paper deals with the project of mobile platform from the Department of Applied Electronics and Telecommunications. Its goal is to consider the current hardware configuration and sensors equipment of the model and to design a new sophisticated control system with the usage of GPS and MS Kinect controller. In the beginning it is therefore necessary to study previous projects that are stated in the list of information sources. The paper describes the initial hardware situation through the block diagram and links to the previous projects. Simultaneously it talks in more detail about the hardware parts that are attached during the current project. Then the paper deals with the firmware of the modules. It is necessary to upgrade non functional parts of the code and add new functions to the microcontrollers in sensors modules and engines control unit to meet the requested possibilities of the navigation. After that it goes theoretically through the possibilities of the navigation algorithms and basic image processing with support of the existing libraries that can be used for the glyph or human figure following. Finally it describes the control application written in the C# language. It shows libraries and functions for the communication with the peripherals, design of the user interface and the algorithms used for the control of the mobile platform in different modes such as following of the glyph or human figure and autonomous navigation.

Keywords

mobile platform, C# control application, autonomous navigation, glyph following, human figure following, GPS, MS Kinect, compass, infrared sensor, ultrasound sensor, CAN

Prohlášení

Předkládám tímto k posouzení a obhajobě diplomovou práci, zpracovanou na závěr studia na Fakultě elektrotechnické Západočeské univerzity v Plzni.

Prohlašuji, že jsem svou závěrečnou práci vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 270 trestního zákona č. 40/2009 Sb.

Také prohlašuji, že veškerý software, použitý při řešení této diplomové práce, je legální.

V Plzni dne 4. května 2015

Bc. Ondřej Lufinka

.....

Podpis

Obsah

Seznam obrázků

Seznam symbolů a zkratek

A/D	Analog to Digital Converter. Analogově-digitální převodník.
CAN	Controller Area Network. Sběrnice pro komunikační síť v automobilu.
CD	Compact Disc. Optický disk pro ukládání digitálních dat.
CRC	Cyclic Redundancy Check. Cyklický redundantní součet.
FPS	Frames per Second. Počet snímků za vteřinu.
GPS	Global Positioning System. Globální polohovací systém.
HW	Hardware. Fyzicky existující elektronické vybavení.
I^2C	Inter-Integrated Circuit. Sériová sběrnice pro komunikaci mezi integrovanými obvody.
ID	Identifier. Jednoznačný výraz – identifikátor.
IR	Infrared Radiation. Infračervené záření.
LCD	Liquid Crystal Display. Displej z tekutých krystalů.
LED	Light Emitting Diode. Dioda emitující světlo.
Li-ion	Lithium-ion. Lithium-iontová technologie pro akumulátory.
LSB	Least Significant Bit. Bit s nejnižší hodnotou.
MCU	Microcontroller unit. Jednočipový mikropočítač.
MS	Microsoft. Společnost Microsoft.
MSB	Most Significant Bit. Bit s nejvyšší hodnotou.
MSI	Micro-Star Int'l Co., Ltd. Společnost MSI.
NRZ	Non Return to Zero. Kódování bez návratu k nule.
PC	Personal Computer. Osobní počítač.
PWM	Pulse Width Modulation. Pulzně šířková modulace.
RAM	Random Access Memory. Paměť s libovolným přístupem.
SPI	Serial Peripheral Interface. Sériové periferní rozhraní pro komunikaci mezi mikroprocesorem a periferiemi.
SW	Software. Programové vybavení.
USB	Universal Serial Bus. Univerzální sériová sběrnice.
VGA	Video Graphics Array. Standard firmy IBM pro počítačovou zobrazovací techniku.
Wi-Fi	Standard pro bezdrátovou komunikaci v počítačových sítích.

1

Úvod

Tématem diplomové práce je projekt mobilní platformy na Katedře aplikované elektroniky a telekomunikací. Mobilní platformou se obecně rozumí systém schopný pohybu, sběru dat a orientace v terénu. V tomto konkrétním případě je platforma postavena na modelu pásového vozidla – tanku (o hmotnosti přibližně 25 kg a rozměrech 75x35 cm). Platforma slouží převážně pro výukové účely a dále pro prezentaci fakulty na akcích pro veřejnost. Na projektu se již podílelo v rámci svých prací několik studentů a odkazy jsou uvedeny v seznamu literatury. Celý systém je řešen jako modulární, což umožňuje jeho jednoduchou modifikaci během na sebe navazujících projektů.

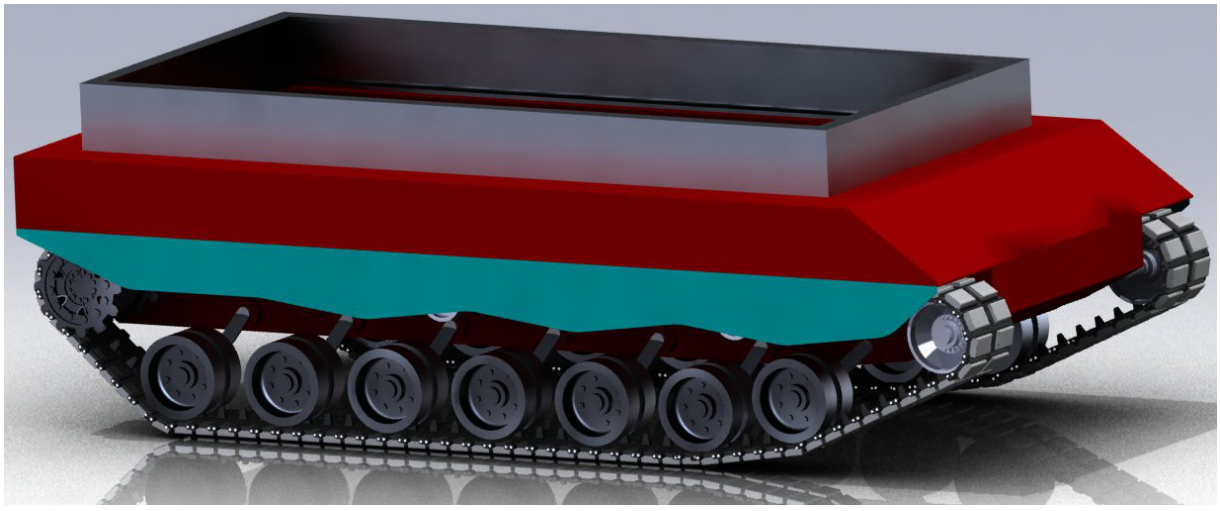
Úvodní kapitola této práce je zaměřena na popis hardwaru. Práce má uvažovat stávající koncepci a senzorovou výbavu, a proto se v ní pro získání přehledu věnuji i blokům již hotovým. Vše je vysvětleno na blokovém schématu. Model před touto prací obsahoval řídicí počítač s USB periferiemi (např. joystick) a dále prvky komunikující přes sběrnici CAN (jednotka motorů a tři moduly pro senzory). V neposlední řadě měla platforma také jednotku pro správu napájení. V rámci této práce je pak model vybaven modulem GPS, rozšiřujícími USB konektory, reproduktorem a kompasem. Dále také chybějící kabeláží a senzory, které jsou uvedeny v předchozích projektech, ale ne již namontovány. Další kapitola přímo navazuje na úvodní a řeší se v ní modifikace softwaru a kompletace chybějících funkcí v jednotkách senzorů a motorů. Zejména v modulech senzorů většina funkcí pro sběr dat zcela chyběla a bylo nutné se věnovat jejich zprovoznění. Příslušné katalogové listy a zdrojové kódy jsou obsaženy také na přiloženém CD.

V následující kapitole práce popisuje tvorbu samotné propracovanější řídicí aplikace podle požadavků zadání na ovládání mobilní platformy. U příslušných částí se také stručně věnuje teorii algoritmů pro navigaci a pro rozpoznávání obrazu prostřednictvím ovladače Microsoft Kinect a zmiňuje využité knihovny. Aplikace je tvořena v programovacím jazyce C#. Jsou vysvětleny knihovny pro komunikaci s periferiemi, funkce pro sběr dat a ovládání bloků a uživatelské rozhraní. Dále jsou na vývojových diagramech ukázány implementace konkrétních algoritmů na platformě pro sledování vzoru nebo lidské postavy a pro autonomní navigaci. V přílohách je možné prohlédnout ukázky z módů aplikace a na přiloženém CD jsou opět k nahlédnutí zdrojové kódy.

2

Hardware mobilní platformy

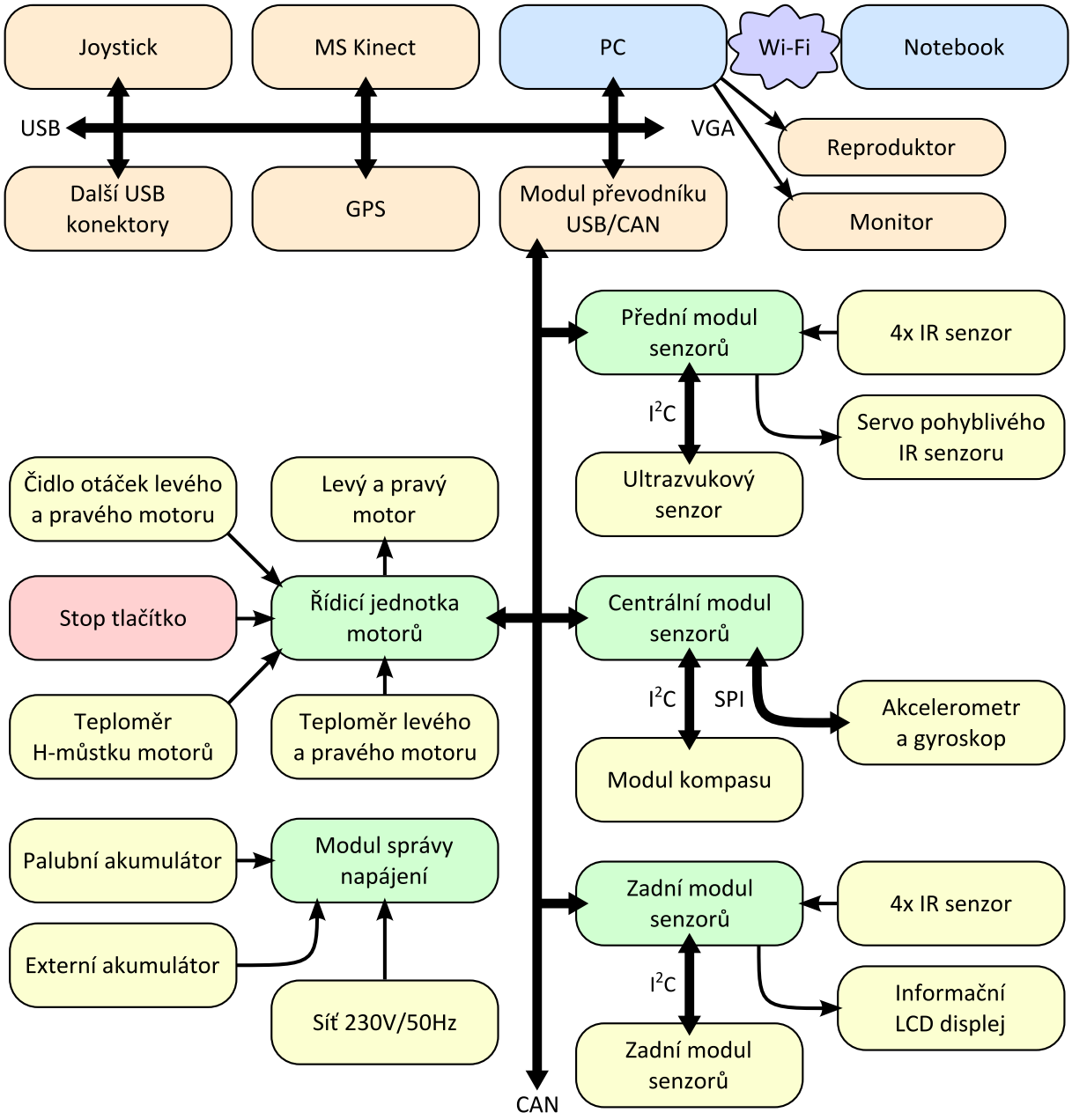
Tato kapitola popisuje koncepci hardwaru mobilní platformy. Cílem je podat kompletní přehled o hardwaru, aby čtenář nemusel dále hledat v předešlých pracích a katalogových listech pro pochopení konceptu návrhu. Celý návrh je řešen jako modulární. To znamená, že jednotlivé bloky jsou propojeny přes sběrnici (v tomto případě CAN a USB), což umožňuje jednoduché přidávání dalších částí a jejich upravování bez zásahu do celkového návrhu. Na blokovém schématu (obr. 2.2), které se výborně hodí pro popis systému složeného z jednotlivých celků, ukazuje kapitola části, kterými je platforma vybavena. Na obr. 2.3 je poté vidět reálné rozmístění komponentů v modelu. Hardware je kompletně namontován na 75 cm dlouhý pásový podvozek pro modelové tanky bez zbytečných ozdobných prvků, který je možno vidět na počítačovém modelu z obr. 2.1. Fotografii reálného modelu je možné shlédnout v příloze F. Pro přehled se dále kapitola detailněji zabývá jednotlivými bloky, kterými již byl model vybaven a které byly popsány v předešlých pracích, a zároveň k nim přidává reference, aby je případný zájemce mohl nastudovat více do hloubky. Nadále zmiňuje části, jež bylo nutné doplnit v rámci této diplomové práce pro splnění požadavků na propracovanější řídicí aplikaci, a k nim opět přidává příslušné reference.



Obr. 2.1: Počítačový model podvozku mobilní platformy |Převzato z [2]|

2.1 Blokové schéma

Na následujícím obrázku (obr. 2.2) se nachází zmíněné blokové schéma. Detailnější popis jednotlivých bloků je dále v této kapitole. Hlavním mozkem celé platformy je počítač MSI MS-7677 (blok PC), který již byl součástí platformy (umístěn v přední části modelu). Běží na něm řídicí aplikace a přes sběrnice je schopen komunikovat se všemi periferiemi. Externí (USB) Wi-Fi adaptér umožňuje připojení k internetu nebo ovládání tanku z jiného PC přes vzdálenou plochu. Na VGA konektor můžeme připojit monitor a na klasický audio výstup je v rámci této práce přidán malý reproduktor, umožňující například pomocí zvukové signalizace sledovat stav tanku v terénu bez připojeného monitoru.



Obr. 2.2: Blokové schéma hardwaru mobilní platformy

Přímo na počítač je napojena sběrnice USB sloužící pro obsluhu USB periférií. Z dosavadních částí můžeme vidět joystick (Wireless Gamepad F710) pro snadné ovládání platformy bez dalšího připojeného PC nebo monitoru, klávesnice a myši. Ovladač MS Kinect (vpředu platformy) zajišťuje strojové vidění prostřednictvím klasické a hloubkové kamery. Dále je zde převodník USB/CAN (umístěný pod počítačem), který tvoří datovou cestu mezi PC a dalšími moduly vybavenými rozhraním CAN. Během svého projektu jsem dále přidal modul GPS navigace (Navilock NL-303P) a panel s rozšiřujícími USB konektory pro připojení například klávesnice nebo myši.

Přes zmíněný převodník USB/CAN se dostáváme po sběrnici CAN k dalším modulům. Platforma již obsahovala tři senzorové jednotky a jednu pro řízení motorů. Senzorové moduly (přední, centrální a zadní) slouží ke čtení dat z připojených prvků a jejich posílání zpět do počítače, k zobrazování informací na LCD a k ovládání servomechanismů. K dispozici byly na platformě čtyři přední infračervené senzory, přední servomechanismus, přední ultrazvukový senzor, akcelerometr s gyroskopem a zadní LCD. Během tohoto projektu se přidaly čtyři zadní infračervené senzory, zadní ultrazvukový senzor, modul kompasu a přední otočný infračervený senzor na přední servomechanismus. Podle katalogových listů a předchozích projektů, odkud bylo zjištěno zapojení jednotlivých konektorů, byla také namontována veškerá kabeláž nezbytná pro nové senzory.

Modul pro řízení motorů (umístěn v zadní části) slouží k nastavení a regulaci otáček a k monitorování stavu motorů a řídicích H-můstků. Dále sleduje a posílá po sběrnici CAN do počítače údaje o aktuálních otáčkách, teplotách motorů a H-můstku a elektrickém proudu tekoucím do obou motorů. Navíc je během tohoto projektu naplánováno doplnění nouzového červeného stop tlačítka, které může podnětem obsluhy v případě poruchy odpojit jednotku motorů a zastavit tak pohyb mobilní platformy.

Dále se ještě na platformě nachází modul správy napájení (také již osazen během dřívějších projektů, umístěn pod počítačem), který má za úkol vytvářet požadované napěťové úrovně pro PC a další jednotky. Zajistí také standardní ukončení provozu počítače při vypnutí modelu bočním vypínačem. Aby nedošlo ke ztrátě dat, které hrozí běžícímu počítači při náhlém výpadku napájení, modul nejprve pošle do PC pokyn k ukončení jeho činnosti a až poté ho odpojí. Energie pro celou platformu je dodávána z palubního akumulátoru technologie Li-ion, který je usazen ve středu mobilní platformy. Akumulátor se skládá ze tří sériově spojených hlavních článků (každý o nominální hodnotě napětí 3,7 V), z nichž každý je sestaven z několika paralelních základních článků. Celková kapacita akumulátoru je 108 Ah. V mobilní platformě tvoří hlavní část spotřeby palubní počítač (přibližně 2 A) a motory (při maximální zátěži každý 18 A podle publikace [1], odhadem přibližně do 10 A oba při rovnoměrném pohybu). Model tak vydrží v provozu při neustálé jízdě kolem 10 h. Z boku je poté na mobilní platformě vyveden konektor, kam bude možné připojit síťové napájení nebo externí akumulátor. Pro tuto funkci je ovšem nutné přidat další modul, který zajistí dobíjení palubního akumulátoru právě z těchto dvou možných zdrojů. Takový modul může být předmětem například jiné diplomové práce.

2.2 Rozmístění komponentů

V této sekci je vložen diagram rozložení jednotlivých prvků popsaných u předchozího blokového schématu pro získání názorné představy o hardwaru mobilní platformy (obr. 2.3). Zkratka IR v diagramu značí infračervený senzor. Barevně odpovídají prvky těm v blokovém schématu (modrý počítač, oranžové USB periférie, zelené moduly, žluté senzory a další koncové prvky, červené stop tlačítko). V porovnání s blokovým schématem je zde navíc zobrazen boční vypínač, který vypíná nebo zapíná modul správy napájení a tím i počítač a všechny ostatní periférie. Jak bylo zmíněno dříve, při vypnutí se nejprve pošle signál počítači a ten je odpojen skrze modul správy napájení až poté, co je bezpečně vypnut, aby nedošlo k možné ztrátě dat nesprávným ukončením provozu. Naopak na diagramu chybí joystick a monitor, které jsou ze zřejmých důvodů mimo samotnou platformu. Joystick komunikuje bezdrátově se svým adaptérem v jednom z USB konektorů a monitor lze připojit do VGA konektoru, který se také nachází na panelu s rozšiřujícími USB konektory.

1. Servo s otočným IR senzorem
2. Přední modul senzorů
3. Převodník USB/CAN
4. GPS modul
5. Stop tlačítko
6. Přední ultrazvukový senzor
7. MS Kinect
8. Řídicí jednotka motorů
9. Zadní ultrazvukový senzor
10. Modul správy napájení
11. Centrální modul senzorů
12. Akcelerometr a gyroskop
13. Kompas
14. Reproduktor
15. Wi-Fi adaptér
16. Zadní modul senzorů
17. Další USB konektory
18. Vypínač
19. Konektor pro síťové napájení
20. LCD displej

Obr. 2.3: Diagram rozmístění bloků v mobilní platformě

2.3 Centrální počítač

Jak již bylo řečeno, centrální počítač je hlavním řídicím centrem celé mobilní platformy. Přes sběrnice USB a CAN sbírá data ze senzorů a vydává pokyny akčním členům (motory, servomechanismus atd.). Probíhá v něm vyhodnocování dat, zpracování obrazu a výpočet řídicího algoritmu. V mobilní platformě je namontován PC od firmy MSI, konkrétně typ MSI MS-7677. Disponuje dvoujádrovým procesorem Intel Pentium G630T o maximální frekvenci 2,3 GHz, který zajišťuje dostatečný výkon pro probíhající výpočty. K dispozici jsou dále 4 GB RAM a integrovaná grafická karta Intel HD Graphics 2000. Nainstalován je operační systém Microsoft Windows 7, který byl dodán během tohoto projektu (dříve MS Windows XP). Spolu s instalací nového systému bylo nutné vyřešit ovladače pro správnou funkci periférií, které je možné najít na přiloženém CD nebo pod odkazy v seznamu literatury pro případné pozdější potřeby. Kompletní specifikaci PC lze prohlédnout v příloze A, kde je vidět tabulka z programu, který detekuje hardware počítače.

2.4 Periférie připojené po USB sběrnici

Následující část uvádí přehled periférií připojených po USB. Těmi jsou ovladač MS Kinect, USB joystick, GPS modul, modul převodníku USB/CAN a panel s rozšiřujícími konektory.

2.4.1 MS Kinect

Senzor Kinect od firmy Microsoft (verze 1) je použit na platformě ke strojovému vidění. Kompletní dokumentaci lze najít ve zdroji [3] a ovladače pod číslem [24] ze seznamu literatury. Obsahuje barevnou kameru s rozlišením 1280x960 bodů při 30 FPS a hloubkovou kameru o rozlišení 640x480 bodů také při 30 FPS. Výstupem z obou kamer je zrcadlový obraz. Hloubková kamera je řešena pomocí zdroje infračerveného rozmítaného laseru a IR senzoru. Systém funguje tak, že zdroj postupně vysílá pulsy do bodů prostoru a senzor měří dobu, za kterou se vrátí zpět. Ze znalosti rychlosti světla lze poté zjistit vzdálenost jednotlivých bodů. Kinect má dále akcelerometr ke zjišťování jeho naklonění a mikrofon umožňující záznam zvuku. Originální senzor ještě obsahuje podstavec s motory pro jeho náklon, ale v této mobilní platformě je Kinect rozebrán a bez svého pouzdra napevno zabudován do přední části. Pohybuje se tedy spolu s ní.

Obr. 2.4: Senzor MS Kinect |Převzato z [3]|

2.4.2 Joystick

Joystick funguje jako základní jednoduché bezdrátové ovládání mobilní platformy. Jedná se o klasický typ známý z ovládání herních konzolí. Konkrétně je zde využit model USB Wireless Gamepad F710 (dokumentace je ve zdroji [4] v seznamu literatury, výrobce Logitech International, S.A.). Joystick umožňuje například přepínat módy aplikace nebo manuálně ovládat pohyb. Připojen je k rozhraní USB prostřednictvím adaptéru, který je dodáván spolu s ním. O samotnou bezdrátovou komunikaci se tedy není potřeba starat a po nainstalování příslušných ovladačů, které jsou k nalezení na přiloženém CD diplomové práce, je počítačem rozpoznán jako standardní periferní zařízení USB.

Obr. 2.5: Wireless Gamepad F710 |Převzato z [4]|

2.4.3 GPS modul

V rámci této práce byl k mobilní platformě přidán modul GPS s názvem Navilock NL-303P (doplňující informace opět v seznamu literatury pod zdrojem [5]). Modul poskytuje aktuální GPS souřadnice, které jsou nepostradatelnou součástí navigace mobilní platformy v terénu.

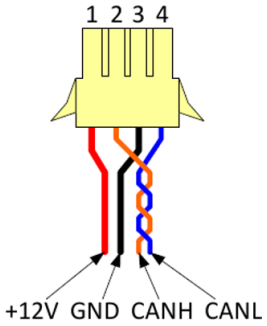
Obr. 2.6: GPS modul SF200 |Převzato z [5]|

Protože tento modul podporuje pouze komunikaci přes sériový port, ke sběrnici USB řídicího počítače je připojen přes převodník mezi sériovým portem a sběrnicí USB. Po in-

stalaci ovladačů převodníku (k dispozici na CD) se vytvoří virtuální sériový port. Přes něj lze přijímat zprávy a získávat tak informace o poloze mobilní platformy.

2.4.4 Modul převodníku USB/CAN a specifikace sběrnice CAN

Tato sekce se věnuje popisu převodníku USB/CAN, který umožňuje počítači navázat komunikaci s dalšími jednotkami, které nepodporují přímo USB rozhraní. Sekce také podává přehled o obou použitých sběrnicích. Převodník byl vytvořen v rámci diplomové práce, jež je uvedena pod odkazem [1] v seznamu literatury. Detailnější popis je podrobněji rozveden právě ve zmíněné práci a zde uvedu jen stručně jeho hlavní funkci. Tou je tedy předávat zprávy obousměrně mezi sběrnicí CAN a USB bez významného zpoždění a zároveň také rozvádět napájení dodávané od modulu pro správu napájení skrze konektor CAN sběrnice pro další jednotky připojené právě na CAN. Konektor sběrnice je znázorněn na obrázku 2.7 a je vidět, že dva příslušné datové vodiče jsou rozvedeny ve svazku společně s napájecími vodiči.



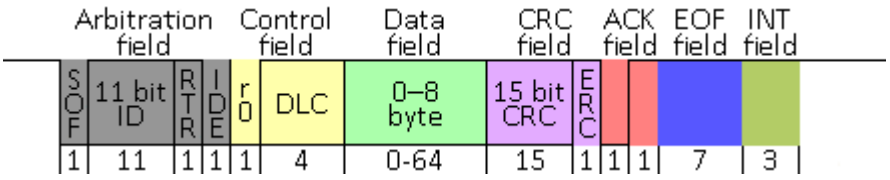
Obr. 2.7: Konektor pro rozvod sběrnice CAN a napájení |Převzato z [1]|

Dále bych zde zmínil teorii sběrnice CAN, proč byla vybrána tvůrci předchozích diplomových prací jako hlavní komunikační prostředek celé mobilní platformy a jaké výhody její výběr přináší do budoucna. Sběrnice vznikla v 80. letech (firma Bosch) pro automobilový průmysl a díky svým vlastnostem se rychle rozšířila a získala mezinárodní standard ISO 11898. Vybrána byla pro svoje vlastnosti vyhovující modulárnímu konceptu mobilní platformy, které jsou následující:

- zaručená doba odezvy.
- podpora režimu „multimaster“ (každý uzel může vysílat bez potřeby řídicího uzlu).
- není pevně daná struktura (na mobilní platformu lze kdykoli přidat další jednotky a pouze rozšířit protokol o nové identifikátory, které zpracuje řídicí aplikace v PC – jednoznačná výhoda pro budoucí rozvoj projektu).
- prioritní rozhodování na základě identifikátoru (například zprávy pro motory mají vyšší prioritu než zprávy od senzorů).

- rámce můžou obsahovat až 8 bytů (dostatečně dlouhé pro zprávy v platformě).
- hardwarová detekce chyb a opakované vysílání chybných rámců.
- autonomní odpojení poškozených jednotek.
- libovolná přenosová rychlost až 1 Mb/s (v mobilní platformě 100 kbit/s, což umožňuje zcela postačující délku sběrnice až 620 m).
- normy CAN2.0A pro 11b identifikátor (použito v mobilní platformě – zajišťuje dostatečný počet 2^{11} různých zpráv) a CAN2.0B pro 29b identifikátor.

Sběrnice zahrnuje ve své specifikaci fyzickou a linkovou vrstvu, což umožňuje vykonávat většinu jejích funkcí přímo v řadiči (např. zmíněná detekce chyb) a uspořit tak výpočetní výkon MCU na připojené jednotce. Naopak nespecifikuje použité médium a napěťové úrovně. Využívá se diferenční vedení schopné přenést dominantní „0“ a recesivní „1“ (nejčastěji kroucená dvojlinka zakončená rezistorem 120 Ω s označením vodičů CANH a CANL doplněné o zemní vodič pro sjednocení potenciálů). Spolu s vyjádřením úrovní pomocí diferenčního napětí, NRZ kódováním, synchronizací pomocí metody tzv. „bit stuffing“, dělením bitu do časových kvant a zabezpečením rámce pomocí CRC (a dalších) je sběrnice velmi odolná vůči rušení, což se hodí například v blízkosti motorů mobilní platformy.



Obr. 2.8: Rámec sběrnice CAN. V dolním řádku počty bitů polí. |Převzato z [6]|

Sběrnice také zaručuje dostatečný datový tok pro všechny zprávy v mobilní platformě. Při rychlosti 100 kbit/s a maximálnímu počtu bitů na zprávu přibližně 124 pro CAN2.0A (rámec 108 bitů – na obr. 2.8, prodleva mezi zprávami 3 bity, možný přidaný počet bitů kvůli funkci „bit stuffing“ až 13) se dostáváme na minimální počet přibližně 80 zpráv na 100 ms (výpočet patrný z rovnice 2.1). To znamená, že pokud by všechny senzory a jednotka motorů komunikovaly s periodou 100 ms a s maximální délkou zpráv, můžeme připojit na sběrnici až 80 těchto prvků. Jak je vidět například na blokovém schématu, tato kapacita zdaleka není dosažena. Perioda také nikdy není kratší než 100 ms (nastavuje se v řídicí aplikaci). Pro další informace, detailní popis rámce a vysvětlení jednotlivých pojmů slouží dokument [6] v seznamu literatury, ze kterého jsem také čerpal informace pro tento popis.

$$n = \frac{100000\ bit/s}{124\ b} \cdot \frac{1}{10} \quad [(100\ ms)^{-1}] \tag{2.1}$$

Na druhé straně převodníku se nachází dobře známá sběrnice USB, přes kterou, jak již bylo řečeno dříve, počítač komunikuje i s ostatními USB zařízeními. Důvodem jejího nasazení je skutečnost, že se jedná o nejpoužívanější standard k propojení PC a jeho periférií. Převodník USB/CAN je tomu přizpůsoben. Nebudu ji zde více rozebírat a pro její specifikace lze nahlédnout do dokumentu [7] v seznamu literatury.

2.4.5 Rozšiřující USB konektory

Během svého projektu jsem dodal na mobilní platformu také postranní panel s dalšími dvěma USB konektory pro snadnější připojování například klávesnice, myši nebo USB disku. Důvodem byl zároveň i klesající počet volných konektorů po přidání periférií.

2.5 Řídicí jednotka motorů

Jednotka pro řízení motorů se stará o pohyb mobilní platformy a je připojena na CAN sběrnici. Podrobně je popsána v práci pod číslem [1] v seznamu literatury. V jednotce se nachází regulátor na principu fuzzy logiky, který zajišťuje udržování požadované hodnoty otáček na každém z pásů. Tyto požadované hodnoty pro každý pás zvlášť poskytne modulu počítač prostřednictvím poslaných zpráv. Zpětnou vazbu o skutečných otáčkách zajišťuje jednotce dvojice inkrementálních čidel otáček a tyto hodnoty posílá jednotka také po sběrnici CAN do počítače. Dále jsou modulu k dispozici čidla odebíraného elektrického proudu a teplotní čidla na obou motorech a H-můstku, který zajistí motorům dostatečný příkon. H-můstek je pro vysoký výkon napájen vlastním vedením mimo sběrnici CAN. Jednotka je schopna motory vypnout při překročení mezních hodnot elektrického proudu či teplot a zabránit tak zničení H-můstku nebo i samotných motorů. Tyto informace jsou také posílány po sběrnici CAN a počítač tak může sledovat jejich hodnoty pro další možné zpracování. Jednotka také zastaví jakýkoli pohyb motorů, pokud nedostává pravidelné informace z PC (tzv. funkce „watchdog“ – rozvedeno dále v sekci 3.2).

Pokud je jednotka uvedena do chybového stavu překročením některé mezní hodnoty nebo zásahem funkce „watchdog“, posílá také zprávu o této chybě a nereaguje na povely požadovaných otáček. Naopak může být zase uvedena zpět do normálního provozu buď znovuspuštěním (vypnutí a zapnutí napájení, uzemnění reset vývodu jejího MCU) nebo posláním konfigurační zprávy z počítače do jednotky. Více o těchto zprávách se píše v částech o programovém vybavení jednotek (3.1 a 3.2).

Modul ovládá dva stejnosměrné motory o výkonu na hřídeli 160 W a tažnou silou na každém z pásů 81,5 N, jak uvádí práce zmíněná na začátku této sekce. Zde bych ještě přidal výpočet maximální rychlosti vycházející z maximálních otáček motorů, které jsou 14600 ot/min. Podle rovnice 2.2, kde za r dosadíme poloměr hnacího kola pásů (0,03 m), za ω maximální otáčky motorů za vteřinu (243,3 ot/s) a za p poměr převodovky od motoru k pásu (1:16), nám vyjde rychlost přibližně 2,86 m/s, což je asi 10,3 km/h. Tuto hodnotu

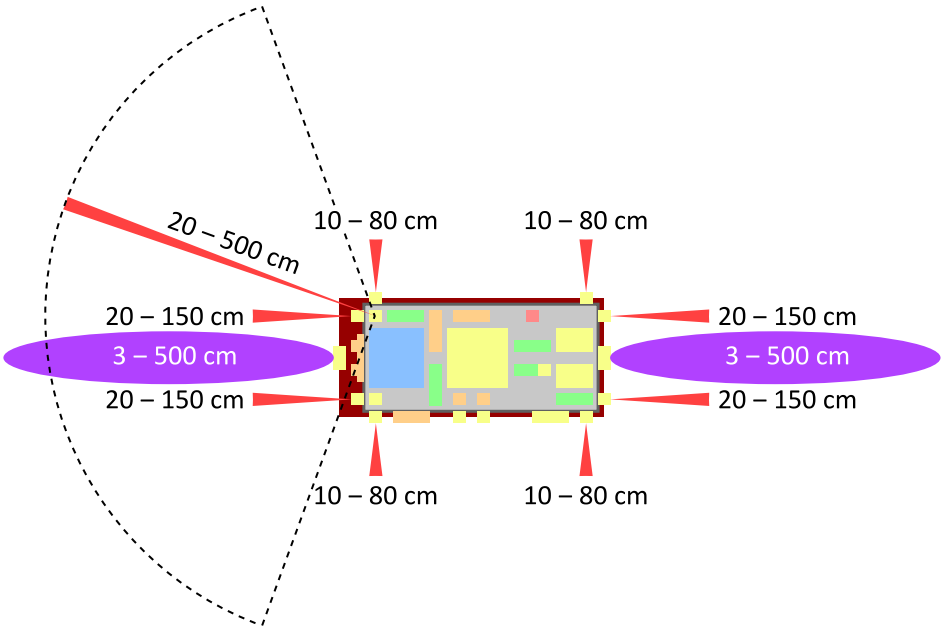
je ovšem nutné brát jako absolutní maximum, které nelze překročit. Reálná maximální rychlost je pak po omezení regulátorem a aplikací kolem 5 km/h. Řídicí aplikace totiž neposílá regulátoru maximální možnou hodnotu kvůli zamezení „splašenému“ pohybu modelu, kdy následkem zpoždění v komunikaci od senzorů k PC, od PC k motorům a v reakci jednotky motorů nestíhá aplikace reagovat na změny polohy v reálném terénu.

$$v = (2 \cdot \pi \cdot r) \cdot \omega \cdot p \quad [m \cdot s^{-1}] \tag{2.2}$$

Nakonec je k jednotce motorů v rámci této práce naplánováno připojení bezpečnostního červeného stop tlačítka, konkrétně na napájení výkonové části (H-můstky pro motory). Při stisku obsluhou odpojí motory a zastaví jakýkoli pohyb, dokud není tlačítko znovu odjištěno. Schéma je k vidění v příloze B.

2.6 Moduly senzorů

Na modelu se nacházejí celkem tři moduly obsluhující soustavu senzorů, jak již bylo řečeno u blokového schématu. Starají se o sběr dat a jejich posílání po sběrnici CAN do počítače. Zpátky naopak přijímají konfigurační zprávy z řídicí aplikace (viz. 3.1). Znázornění dohledu dálkových senzorů je vidět na obrázku 2.9, přičemž každá z jednotek má na starost skupinu ve svém okolí. Přední ultrazvukový (fialově), sousední čtveřici infračervených (červeně) a rotační infračervený senzor (s čárkovanou výsečí) spolu se servomechanismem obsluhuje přední jednotka, zadní ultrazvukový a zadní čtveřici infračervených senzorů zase zadní jednotka. Na tu je navíc připojen i LCD displej. Střední jednotka komunikuje s modulem akcelerometru a gyroskopu a s modulem kompasu.



Obr. 2.9: Znázornění dohledu jednotlivých senzorů

Dále zde uvedu pár přehledových informací k jednotlivým typům využitých senzorů. Pro získání podrobností lze nahlédnout do práce [2] v seznamu literatury, která se touto částí mobilní platformy zabývá. Tam jsem také nastudoval například potřebné znalosti o zapojení konektorů při doplňování chybějící kabeláže a rozšiřujících senzorů (modul kompasu a dálkoměry – čtyři zadní infračervené, otočný infračervený a zadní ultrazvukový). Komunikace se senzory bude dále probrána v rámci kapitoly 3.3.

2.6.1 Infračervený senzor

Infračervené senzory od firmy SHARP Corporation jsou využity jako dálkoměry na principu měření doby odrazu infračerveného záření od překážky před nimi a v obrázku 2.9 mají červenou barvu dohledu. Jsou připojeny na vstupy A/D převodníků MCU své jednotky, která tak zajišťuje periodické čtení jejich hodnot. Na modelu se nacházejí tři typy. Boční (katalogový list [8] v seznamu literatury) s rozsahem 10 – 80 cm, přední a zadní (viz. [9]) s rozsahem 20 – 150 cm a otočný na věži. Ten je kombinací dvou senzorů. Jeden stejný jako přední a druhý s rozsahem 100 – 550 cm (viz. [10]). Tím se dostane celkový dohled 20 – 550 cm. Všechny dálkoměry jsou pro sjednocení maximálního dohledu omezeny řídicí aplikací na 5 m, a proto je v obrázku uveden tento údaj. V této práci byla dodána čtveřice zadních a dvojice na otočné věži. Ostatní již byly namontovány.

(a) S rozsahem 10 – 80 cm |Převzato z [8]|

(b) S rozsahem 100 – 550 cm |Převzato z [10]|

Obr. 2.10: Ukázky použitých infračervených senzorů

2.6.2 Ultrazvukový senzor

Ultrazvukové senzory SRF08 od výrobce ROBOT-ELECTRONICS (viz. [12]) jsou využity jako doplňující přední a zadní dálkoměr. V rámci této práce byl dodán zadní. Fungují na principu měření doby odrazu ultrazvuku od překážky před nimi. V obrázku 2.9 mají fialovou barvu. Na jednotky jsou připojeny pomocí sběrnice I^2C. V porovnání s infračervenými senzory mají širší vyzařovací rozptyl a vyplní tak prostor, kam infračervené „nevidí“. Naopak mají horší přesnost a větší kolísání hodnot. Spolu s infračervenými se vhodně doplňují. V rámci práce bylo nutné vyřešit problém s umístěním předního senzoru příliš nízko nad zemí během předchozích projektů, což způsobovalo nechtěné odrazy ultrazvuku od země, které nastávaly v případě malých nerovností na povrchu. Ty ovšem nejsou pro model překážkou (přejede je) a jejich detekce je proto nežádaná. Přední ultrazvukový senzor byl tedy namontován na 10 cm vysokou věž v přední části modelu.

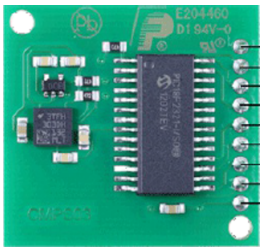
Obr. 2.11: Ultrazvukový senzor SRF08 |Převzato z [11]|

2.6.3 Servomechanismus

Servomechanismus se nachází na mobilní platformě pod rotačním infračerveným senzorem. Umožňuje jeho otáčení v rozsahu necelých 180°. Jedná se o klasický modelářský servomechanismus připojený na PWM výstup MCU přední jednotky senzorů. Více informací lze nalézt v práci zabývající se senzory (viz. [2]).

2.6.4 Modul kompasu

Modul kompasu CMPS03 od ROBOT-ELECTRONICS umožňuje zjistit natočení mobilní platformy vzhledem k magnetickému poli země. Je připojen ke střední jednotce senzorů pomocí sběrnice I^2C a byl dodán v rámci této práce spolu s věží, na které je umístěn. Důvodem osazení senzoru na věž je zabránění jeho rušení kovovou konstrukcí mobilní platformy. Vzdálenost, kde funkce kompasu již není ovlivněna, byla určena experimentálně na přibližně 10 cm nad horním okrajem mobilní platformy tak, že jsem porovnával výstupní hodnotu kompasu v různých vzdálenostech a sledoval její změny. Ty s rostoucí mezerou mezi konstrukcí a kompasem klesaly. Problémem je také náklon mobilní platformy (například většími nerovnostmi povrchu), proti kterému není stávající modul imunní (jeho hodnota se může s náklonem měnit i bez změny natočení). V budoucnu bude nejspíše nutné ho nahradit modulem lepším, jenž funkci kompenzace proti náklonu obsahuje. Katalogový list s více údaji o modulu se nachází pod číslem [13] v seznamu literatury.

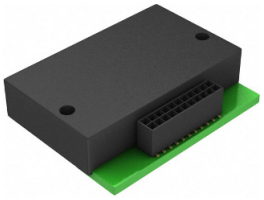


Obr. 2.12: Modul kompasu CMPS03 |Převzato z [13]|

2.6.5 Akcelerometr a gyroskop

Akcelerometr a gyroskop jsou součástí mobilní platformy v rámci jednoho integrovaného senzoru ADIS16300 od výrobce Analog Devices, Inc. (viz. [15]). Komunikuje přes sběrnici

SPI se střední jednotkou senzorů, na které je také umístěn, a sděluje informace o zrychlení působícím ve všech třech osách na mobilní platformu, o jejím náklonu v obou horizontálních osách a o úhlovém zrychlení způsobeném rotací platformy kolem vertikální osy.



Obr. 2.13: Akcelerometr a gyroskop ADIS16300 |Převzato z [14]|

2.6.6 Displej

Na mobilní platformě je umístěn displej sloužící k zobrazování krátkých zpráv o aktuálním stavu řídicí aplikace mobilní platformy v PC. Hodí se pro případy, kdy nemáme k dispozici připojený monitor a podobně. Displej je napojen na zadní jednotku senzorů, která na něm zobrazuje informace obdržené z PC po sběrnici CAN. Jedná se o 2x16 znakový LCD s podsvícením a vlastním řadičem a více informací k jeho připojení na jednotku je v katalogovém listu [17].

Obr. 2.14: LCD displej ATM1602B |Převzato z [16]|

2.7 Modul správy napájení

Jednotka pro správu napájení vytváří potřebné napěťové úrovně pro počítač a pro převodník USB/CAN. Jednotka je vlastně komerční zdroj pro PC, který umožňuje připojit na vstup Li-ion baterii. Počítač poté napájí své USB periférie a z převodníku se napětí paralelně rozvádí s kabeláží sběrnice CAN pro další jednotky připojené na této sběrnici, jak již bylo řečeno u popisu převodníku (viz 2.4.4). Pro koncové prvky (servomechanismus, senzory atd.) je následně napájecí napětí bráno z jednotek, ke kterým jsou připojeny. Potřebnou energii modul čerpá z palubní baterie, která již byla popsána na konci sekce 2.1 společně s možností konstrukce další jednotky, která by umožnila externí dobíjení, sledování stavu baterie a posílání této informace po sběrnici CAN do PC.

3

Modifikace softwaru senzorových jednotek a jednotky řízení motorů

Následující kapitola se věnuje programování jednotek připojených na sběrnici CAN, které jsou sice na mobilní platformě po hardwarové stránce kompletně osazeny, ale neměly dostatečný software nutný pro následný vývoj propracovanější řídicí aplikace na PC. Jedná se o drobné modifikace softwaru řídicí jednotky motorů, která jinak fungovala po minulém projektu téměř bezchybně (viz. [1]), a poté zejména o tvorbu softwaru pro senzorové jednotky, jež měly po předchozí diplomové práci (viz. [2]) hotovou jen hrubou kostru programu a ze senzorů byly zprovozněné jen infračervené dálkoměry, což je pro aplikaci snažící se o autonomní navigaci zcela nedostatečné. Zároveň je v této kapitole dopodrobna popsán protokol zpráv na sběrnici CAN, který je nutný dodržovat jak ze strany jednotek, tak ze strany aplikace v PC, aby byly zprávy doručeny bez chyb na jejich požadované místo určení a došlo ke správnému zpracování dat, které obsahují.

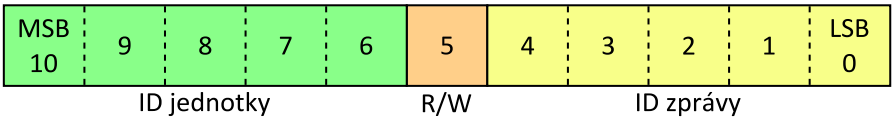
Na jednotkách se nacházejí mikroprocesory od firmy Freescale Semiconductor, Inc., konkrétně typ MC9S08DZ32. Katalogový list se specifikacemi je pod číslem [19] a s podklady k programování pod číslem [20] v seznamu literatury. Ke druhému zmíněnému katalogovému listu lze navíc stáhnout ze stránek výrobce archiv obsahující příklady kódu (viz. [21]). Software pro mikroprocesory se vyvíjí v prostředí CodeWarrior Development Studio od stejné společnosti v jazyce C. K programování se následně používá programátor USBDM s otevřeným zdrojovým kódem. Podporuje mikroprocesory řad RS08, HCS08, HCS12 a Coldfire od Freescale Semiconductor, Inc. Více informací je na internetové stránce (viz. [22]), kde je ke stažení také driver, který je nutný nainstalovat na počítač, z něhož chceme programovat.

3.1 Protokol zpráv po sběrnici CAN

V první sekci této kapitoly bych se tedy nejprve zaobíral protokolem na sběrnici CAN. Protokol vychází z obou předešlých prací zmíněných v úvodu této kapitoly a je dále rozší-

řen. Zde ho uvádím v kompletním přehledu, aby čtenář nemusel dále dohledávat jednotlivé zprávy v předcházejících dílech podle typu jednotky. Zároveň jsou zprávy v minulých projektech z většiny popsány jen teoreticky tak, jak se zamýšlelo jejich nasazení na sběrnici CAN, ale reálně nebyly nikdy zprovozněny, a proto je vysvětlení mnohdy nejasné. Protokol uvedený v této sekci popisuje zprávy podle toho, jak jsem je skutečně implementoval do vzájemné komunikace mezi PC a jednotkami.

Tabulky shrnující jednotlivé zprávy jsou z důvodu velkého rozsahu umístěny v příloze C a tato sekce bodově ve stejném pořadí popisuje a dále rozvádí zprávy v nich uvedené. V tabulce s přehledem zpráv pro jednotky senzorů (obr. C.2) je vidět, že některé zprávy jsou svým významem stejné pro různé jednotky. Kompletní ID pro zprávu (11 bitů) bylo proto sestaveno podle obr. 3.1 tak, že 5 nejvyšších bitů (zleva od MSB) zaujímá ID jednotky. Určuje tedy, pro kterou jednotku je zpráva adresována nebo ze které je vyslána. Další bit určuje, jestli jde zpráva z PC do jednotky nebo opačně (sloupec *R/W* – Read/Write). 5 nejnižších bitů je vyhrazeno pro vlastní ID zprávy. To říká, pro kterou funkci dané jednotky je zpráva určena (například nastavuje IR senzory a podobně). Kompletní ID zpráv pro jednotku motorů (obr. C.1) byly ovšem převzaty z původní práce (viz. [1]) a principem z obr. 3.1 se neřídí. Mají vždy nižší hodnoty než ID jednotek senzorů a tím vyšší prioritu (priorita na sběrnici CAN je udělována podle hodnoty ID).



Obr. 3.1: Struktura ID zpráv pro jednotky senzorů na sběrnici CAN

Význam tabulek v přílohách je následující:

- obrázek C.1:
  - první tabulka – vysvětlení značení jednotek na platformě a jejich ID zabírající 5 bitů (jednotka motorů nemá ID uvedeno, protože ID zpráv pro ní jsou převzata).
  - druhá tabulka – význam sloupce *Směr* použitého následně v tabulkách přehledu zpráv pro jednotku motorů a jednotky senzorů.
  - třetí tabulka – význam sloupce *R/W* (Read/Write) použitého následně v tabulce detailního popisu zpráv.
  - čtvrtá tabulka – přehled zpráv pro jednotku motorů (A) s jejich významem a kompletním ID podle standardu CAN2.0A (tedy ID o rozsahu 11 bitů). ID je převzato z minulé práce.

- obrázek C.2:
  - první tabulka – přehled zpráv pro jednotky senzorů (B, C a D) s jejich významem a kompletním ID podle standardu CAN2.0A (tedy ID o rozsahu 11 bitů). ID je vytvořeno podle principu z obr. 3.1.
- obrázek C.3:
  - první tabulka – detailní popis zpráv s jejich vlastním ID o velikosti 5 bitů (u zpráv jednotky motorů ID opět neuvedeno, protože se nepodílí na tvorbě jejich kompletního ID). Uvádí se také směr zpráv a přehled jejich datových polí.

Zde je dále rozepsán význam datových polí zpráv ve stejném pořadí jako v tabulce s detailním popisem (obr. C.3). Uveden je také počet bitů, které veličina skutečně zabírá (vždy zprava od LSB), a případně vyjádření (např. dvojkový doplněk), násobek (kolikrát je třeba příchozí hodnotu vynásobit nebo kolikrát bude odchozí hodnota vynásobena při zpracování) a jednotka veličiny. Maximální velikost dat zprávy je 64 bitů (8 B). Data jsou rozdělena na pole o velikosti 1 B, jak udává standard komunikace CAN. Některá pole v tabulce zabírají 2 B, což je ovšem zavedeno pro lepší názornost. Ve skutečnosti je nutné přijmout obě části takového pole (vyšší a nižší byte) zvlášť a poté je sečíst s tím, že vyšší byte posuneme o 8 bitů vlevo, aby souhlasil rozměr veličiny. Naopak při vysílání musíme zase hodnotu do těchto dvou částí (vyšší a nižší byte) rozdělit. V následujícím seznamu jsou datová pole zpráv oddělena středníkem:

- TRACKS_INFO_CURRENT – elektrický proud tekoucí do levého motoru (16 b, $0,578\times$, [mA]); elektrický proud tekoucí do pravého motoru (16 b, $0,578\times$, [mA]).
- TRACKS_INFO_ERROR – kód chyby (viz. sekce 3.2), v níž se modul nachází (8 b).
- TRACKS_INFO_PWM – otáčky levého motoru (16 b, [1/min]); otáčky pravého motoru (16 b, [1/min]).
- TRACKS_INFO_TEMPERATURE – teplota levého motoru (16 b, $0,1\times$, [°C]); teplota pravého motoru (16 b, $0,1\times$, [°C]); teplota H-můstku (16 b, $0,1\times$, [°C]).
- TRACKS_SET_CONFIG – kód nastavení (viz. sekce 3.2) jednotky (8 b).
- TRACKS_SET_PWM – otáčky levého motoru (16 b, [1/min]); otáčky pravého motoru (16 b, [1/min]).
- ACM_INFO_BURST1 – napájecí napětí senzoru (12 b, $2,42\times$, [mV]); úhlové zrychlení kolem vertikální osy (14 b, dvojkový doplněk, $0,05\times$, [°/s]); zrychlení v ose y vzhledem k platformě – horizontální osa kolmá ke směru pohybu vpřed a orientovaná doprava (14 b, dvojkový doplněk, $0,6\times$, [mg = 0,001 gravitačního zrychlení]);

zrychlení v ose x vzhledem k platformě – horizontální osa ve směru pohybu vpřed orientovaná dopředu (14 b, dvojkový doplněk, $0,6\times$, [mg]).

- ACM_INFO_BURST2 – zrychlení v ose z vzhledem k platformě – vertikální osa orientovaná nahoru (14 b, dvojkový doplněk, $0,6\times$, [mg]); teplota senzoru (12 b, dvojkový doplněk, $0,14\times$, $+25$, [°C]); náklon ve smyslu do stran – kladné hodnoty při náklonu vpravo (13 b, dvojkový doplněk, $0,044\times$, [°]); náklon ve smyslu vpřed/vzad – kladné hodnoty při náklonu vpřed (13 b, dvojkový doplněk, $0,044\times$, [°]).
- ACM_INFO_REGISTERS – adresa registru na sběrnici SPI (8 b); přečtená data z registru (16 b).
- ACM_SET_CONFIG – perioda měření senzoru (8 b, $10\times$, [ms]); perioda odesílání dat do PC (8 b, $10\times$, [ms]). Hodnoty 0 znamenají vypnutí daného senzoru (platí pro všechny další konfigurační zprávy).
- ACM_SET_REGISTERS – adresa registru na sběrnici SPI (8 b); čtení (0)/zápis (1); data k zapsání – v případě čtení se pole zahodí (16 b).
- COMPASS_INFO_BURST – natočení senzoru vzhledem k severnímu magnetickému pólu – hodnoty rostou po směru hodinových ručiček (16 b, $0,1\times$, [°]).
- COMPASS_SET_CONFIG – perioda měření senzoru (8 b, $10\times$, [ms]); perioda odesílání dat do PC (8 b, $10\times$, [ms]); adresa senzoru na sběrnici I^2C – výchozí pro kompas je 0x0C (8 b).
- DISPLAY_SET_DATA – znaky 1 – 8, kde podle ID zapisujeme na danou pozici displeje – zpráva s ID 0x04 zapisuje na 1. řádek prvních 8 znaků až s ID 0x07 zapisuje na 2. řádek posledních 8 znaků (každý znak 8 b).
- IIC_INFO_REGISTERS – adresa registru na sběrnici I^2C dané jednotky senzorů (16 b – horních 8 b je adresa modulu (př. kompas nebo ultrazvukový senzor) na sběrnici I^2C, dolních 8 b je adresa registru modulu); přečtená data z registru (8 b).
- IIC_SET_REGISTERS – adresa registru na sběrnici I^2C dané jednotky senzorů (16 b – horních 8 b je adresa modulu (př. kompas nebo ultrazvukový senzor) na sběrnici I^2C, dolních 8 b je adresa registru modulu); čtení (0)/zápis (1); data k zapsání – v případě čtení se pole zahodí (16 b).
- INFRA_INFO_BURST – kontrolní byte (LSB kanálů 1 až 7) – LSB kanálu 1 na pozici bitu 0 zprava až kanál 7 na pozici bitu 6 zprava (8 b); naměřená vzdálenost ze senzorů na kanálech 1 až 7 dané jednotky (každý kanál 8 b, nutný přepočet hodnoty podle experimentálně určeného vzorce regresní přímky proložené naměřenými hodnotami z konkrétního infračerveného senzoru – na platformě použito více senzorů s různým dohledem – soubor s výpočtem se nachází na přiloženém CD).

- INFRA_INFO_REQUEST – stejná struktura jako předchozí zpráva.
- INFRA_REQUEST_DATA – bez datových polí.
- INFRA_SET_CONFIG – perioda měření senzorů na dané jednotce (8 b, 10×, [ms]); perioda odesílání dat do PC (8 b, 10×, [ms]); počet kanálů připojených infračervených senzorů – podle tohoto údaje přijde stejný počet kanálů ve zprávách s daty z IR senzorů (8 b).
- LED_SET_CONFIG – automatický (0xFF)/manuální (0x00) režim LED na dané jednotce (8 b). V manuálním režimu reagují LED na data z PC v následující zprávě a v automatickém nikoliv. V něm znázorňují blikáním aktivitu jednotky (konkrétně příjem a odesílání zpráv po sběrnici CAN).
- LED_SET_DATA – zapnutí/vypnutí LED na dané jednotce – každý bit opět nastavuje hodnotu jedné odpovídající LED – „0“ vypnutá/„1“ zapnutá (8 b).
- SENSORS_SET_MODULE_ID – nové ID dané jednotky senzorů (5 b); kontrolní byte – staré ID $\oplus$ nové ID (5 b). Zde je nutno dodat, že pokud změníme ID některé jednotky senzorů, je nutné vypočítat i nová kompletní ID jednotlivých zpráv pro danou jednotku (tabulka na obr. C.2) podle principu z obr. 3.1. Vlastní 5-bitová ID zpráv zůstanou samozřejmě zachována.
- SERVO_INFO_POSITION – pozice servomechanismu 1 až 3 – hodnota 0 až 255, kde 128 odpovídá pozici uprostřed (každé pole 8 b). Zpráva je odeslána z jednotky po požadavku na data ze servomechanismu.
- SERVO_REQUEST_POSITION – bez datových polí.
- SERVO_SET_POSITION – pozice servomechanismu 1 až 3 – hodnota 0 až 255, kde 0 odpovídá pozici vlevo a 128 pozici uprostřed (každé pole 8 b); informace určující, které servomechanismy reagují na novou polohu – bit 0 zprava od LSB v „1“ znamená, že Servo_1 mění polohu, až bit 2 v „1“ znamená, že Servo_3 mění polohu (3 b). Touto zprávou lze také přepínat automatický (hodnota 0x63 v poli pro kanály) a manuální režim (hodnota 0x64) pro Servo_1. V manuálním reaguje na příkazy a v automatickém se pohybuje zleva doprava a zpět („rozhlíží se“).
- ULTRA_INFO_BURST – naměřená vzdálenost z prvního připojeného senzoru (16 b, [cm]); naměřená vzdálenost z druhého připojeného senzoru na dané jednotce (16 b, [cm]).
- ULTRA_SET_CONFIG – perioda měření senzorů na dané jednotce (8 b, 10×, [ms]); perioda odesílání dat do PC (8 b, 10×, [ms]); adresa prvního senzoru na sběrnici I^2C (8 b); adresa druhého senzoru na sběrnici I^2C (8 b).

3.2 Programové vybavení jednotky řízení motorů

V jednotce pro řízení motorů byly třeba pouze menší úpravy, jak již bylo řečeno v úvodu kapitoly, a na nich jsem spolupracoval s kolegou Lukášem Pušmanem (autor práce [1] v seznamu literatury). Jednotlivé funkce z minulého projektu jsou uvedeny v sekci 2.5 a příchozí a odchozí zprávy pro tuto jednotku v protokolu CAN (viz. 3.1).

Nejprve jsme specifikovali možné kódy (datový byte) použitelné v konfigurační zprávě (TRACKS_SET_CONFIG) a ve zprávě o chybách (TRACKS_INFO_ERROR). Ty byly následně zapracovány do softwaru modulu řízení motorů. Jejich přehled je v následující tabulce na obr. 3.2. Konfigurační zpráva umožňuje nastavení jednotky po sběrnici CAN. Smazání jedné ze dvou hardwarových poruch uvede jednotku zpět do provozu, pokud předcházelo její hlášení, které je zároveň zastaveno. Zpráva o chybách hlásí poruchový stav jednotky a je odesílána periodicky v případě výskytu alespoň jedné chyby (více zpráv v případě většího počtu). Poruchy 0x0A, 0x0B a 0x0C omezí funkce jednotky (nereaguje na požadované otáčky) až do jejich smazání. Naopak zbylé případy jsou pouze informativní a chod neovlivní. Jednotka funguje správně, pokud žádnou takovou zprávu nevysílá.

Kódy konfigurační zprávy	Význam
0x0A	Smazání hardwarové chyby H-můstku levého motoru
0x0B	Smazání hardwarové chyby H-můstku pravého motoru

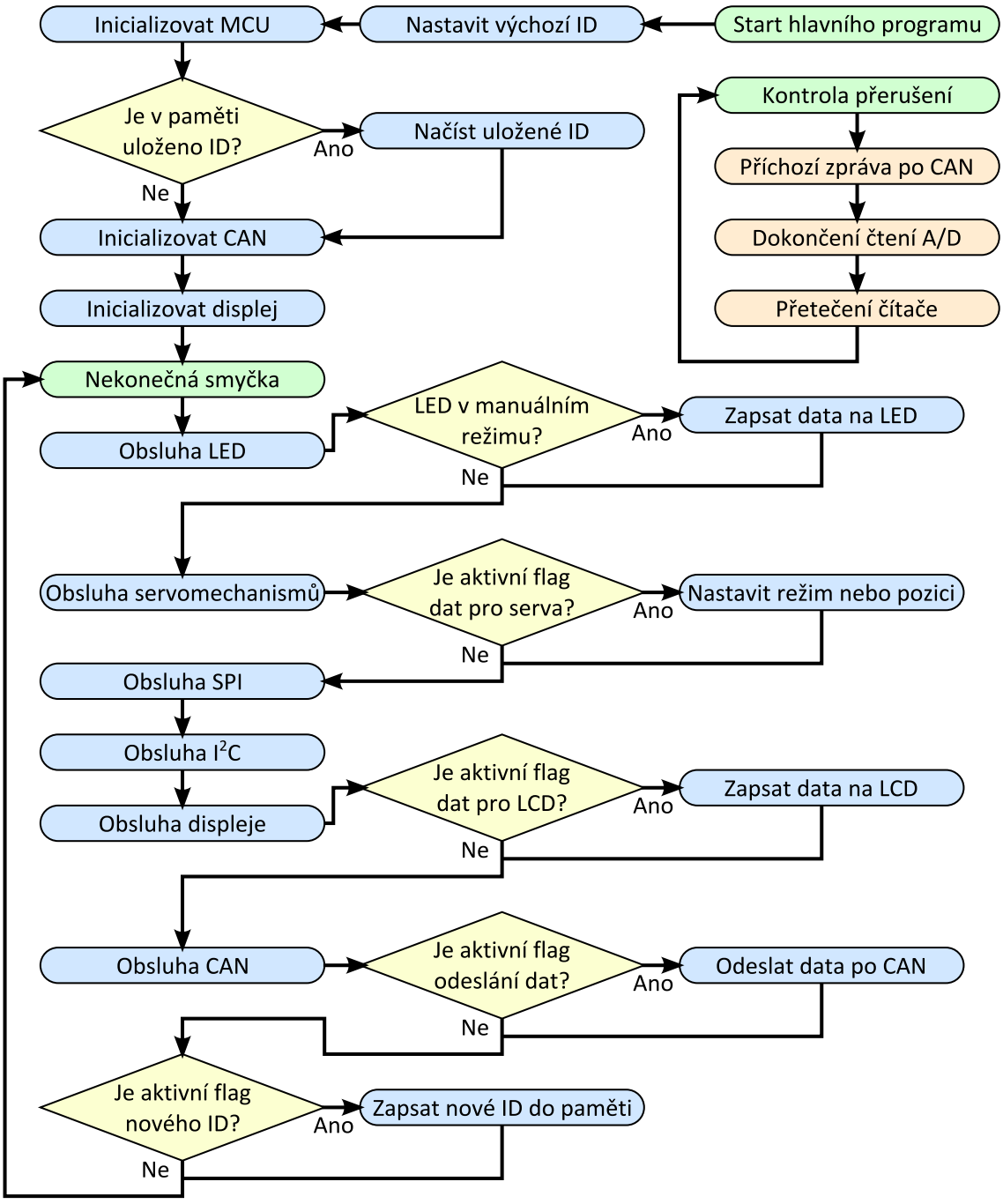
Kódy zprávy o chybách	Význam
0x0A	Chyba H-můstku levého motoru (překročený max. el. proud či teplota)
0x0B	Chyba H-můstku pravého motoru (překročený max. el. proud či teplota)
0x0C	Chyba funkce „watchdog“
0x14	Teploměr H-můstku není připojen (nebo jednotce neodpovídá)
0x15	Teploměr levého motoru není připojen (nebo jednotce neodpovídá)
0x16	Teploměr pravého motoru není připojen (nebo jednotce neodpovídá)

Obr. 3.2: Tabulky kódů zpráv jednotky motorů

Dále jsme se věnovali zprovoznění funkce „watchdog“. Obecně se jedná o funkci, kdy musíme danému zařízení periodicky posílat určitou zprávu, aby toto zařízení fungovalo správně. Pokud zprávu neobdrží do stanoveného časového intervalu, omezí nebo zcela ukončí svoji činnost až do opětovného spuštění. V případě mobilní platformy je tato vlastnost implementována tak, že řídicí aplikace posílá jednotce motorů údaje o požadovaných otáčkách pásů každých 100 ms (nulové hodnoty v případě, že není vyžadován pohyb). Jednotka má stanovený maximální interval čekání na zprávu 2 s. To znamená, že pokud řídicí aplikace z nějakého důvodu selže a jednotka měla poslední pokyn k jízdě, mobilní platforma nejdéle do 2 s zastaví svůj pohyb (jednotka nedostane nový požadavek, zastaví motory a nahodí chybu 0x0C). Jednotka je znovu zprovozněna obdržením nové informace o otáčkách, jakmile aplikace obnoví svůj chod, což opět umožní pohyb modelu (smaže se porucha 0x0C v jednotce a ta o ní zároveň přestane posílat zprávu). Chyba je také indikována zhasnutím LED na jednotce (jinak je rozsvícená).

3.3 Programové vybavení jednotek senzorů

Jednotky senzorů vyžadovaly podstatně větší zásah než jednotka motorů, jak jsem také zmínil v úvodu kapitoly. Nejprve bylo nutné se seznámit s předchozím kódem. Na obr. 3.3 jsem poté vytvořil vývojový diagram kompletního softwaru jednotky senzorů dokončeného v této práci se všemi nezbytnými funkcemi pro správnou činnost modulů. Software je zcela totožný pro každou ze tří jednotek a záleží až na řídicí aplikaci běžící na PC, které funkce na dané jednotce spustí pomocí konfiguračních zpráv.



Obr. 3.3: Vývojový diagram softwaru MCU senzorových jednotek

Pokud se podíváme na vývojový diagram (obr. 3.3), tak z původního návrhu před touto prací byly v hlavním programu funkční části s nastavením výchozího 5-bitového ID jednotky (viz. protokol CAN – sekce 3.1), inicializace MCU a CAN a poté v nekonečné smyčce obsluha LED, servomechanismů a CAN. V části s obsluhou přerušení byla ošetřena částečně všechna zmíněná přerušení – příchozí zprávy po CAN (týkající se LED, servomechanismů a infračervených senzorů), dokončení čtení A/D (zajišťuje čtení dat z IR senzorů) a přetečení čítače (týkající se periodického spouštění čtení z IR senzorů). Nadále jsou zde podrobněji vysvětleny jednotlivé části ze zmíněného diagramu pro pochopení tohoto softwaru. Pro kompletní informace, pokud by se chtěl projektem někdo dále zabývat, slouží podrobné komentáře v samotném zdrojovém kódu. Celý projekt je na přiloženém CD a je rozdělen do 4 následujících souborů:

- *main.c* – obsahuje hlavní program (funkci *main()* – viz. 3.3.2), ve kterém jsou části od položky *Start hlavního programu* ve vývojovém diagramu (obr. 3.3) včetně nekonečné smyčky. V souboru jsou také funkce pro jednotlivé obsluhy.

- *MCUinit.c* – je v něm funkce pro položku *Inicializovat MCU* (funkce *MCU_init()*), která je vytvořena pomocí prvku Processor Expert zahrnutého ve vývojovém prostředí. Jedná se o grafickou nadstavbu, kde si zvolíme potřebné periférie MCU a přerušení, která chceme obsluhovat, a poté se automaticky vygeneruje část kódu pro zmíněnou inicializaci. Tu zde pro velký rozsah uvádět nebudu a lze ji dohledat v samotném projektu (s nastavením jednotlivých periférií apod.). V souboru jsou dále rutiny pro tři povolená přerušení (viz. 3.3.1) pod položkou *Kontrola přerušení* ve vývojovém diagramu.

- *buffers.h* – definuje proměnné (pole) pro ukládání informací o konfiguraci senzorů, přijatých datech po CAN a pro předávání informací mezi funkcemi pomocí tzv. „flagů“, což jsou bity, které po nahození jednou funkcí aktivují určitou část kódu ve funkci jiné. Ta daný „flag“ opět smaže, aby nevykonávala část kódu opakovaně. Soubor také vytváří definice konstant pro 5-bitová ID zpráv, která jsou stejná pro každou jednotku (viz. protokol CAN – 3.1).

- *peripherals.h* – zahrnuje definice dalších pomocných konstant pro zpřehlednění kódu.

3.3.1 Kontrola přerušení

Software podporuje tři různá přerušení, jak je vidět na vývojovém diagramu (obr. 3.3), kterými jsou *Příchozí zpráva po CAN* (rutina *isrVcanrx()*), *Dokončení čtení A/D* (rutina *isrVadc()*) a *Přetečení čítače* (rutina *isrVtpm2ovf()*). Celý program je založen na předávání zmíněných „flagů“. Obecně jsou nahozeny (nastaveny do „1“) v některém z přerušení, což vyvolá určitou činnost v hlavním programu, kde jsou také smazány (nastaveny do „0“). To zamezí neustálému opakování činnosti, než bude znova vyžadována. Tento

systém umožňuje zpracování časově citlivých událostí (komunikace po CAN nebo čítač pro periodická měření) okamžitě v přerušení a vykonání zbytku operací v hlavním programu, když se žádné přerušení neobsluhuje.

První z přerušení kontroluje příchozí komunikaci po připojené sběrnici CAN. Reaguje na zprávy tak, jak jsou vysvětleny v sekci o protokolu CAN (3.1). Konfigurační zprávy zapisují do polí v souboru *buffers.h* nastavení pro jednotlivé senzory a nahodí příslušný „flag“ v případě potřeby dalšího zpracování v hlavním programu. Ostatní příchozí zprávy uloží přijatá data a také nahodí daný „flag“.

```
/* Ukázka kódu zpracování konfigurační zprávy */
case INID_C_GYRO: // Konfigurace akcelerometru a gyroskopu
    if(CANRDLR == 2) { // Pokud délka souhlasí
        period_list[BUF_GYRO] = CANRDSR0; // Nastavit odesílací periodu
        config_list[BUF_GYRO] = CANRDSR1; // Nastavit periodu měření
        RX_flag_list[RXF_GYRO] = GYRO_NEW_CONFIG; // Nahodit flag
    }
    break;

/* Ukázka kódu zpracování jiné příchozí zprávy */
case INID_SPI: // SPI zápis
    if(CANRDLR == sizeof(spi_frame_in)) { // Pokud délka souhlasí
        for (i = 0 ; (i < sizeof(spi_frame_in)) ; i++) // Vyčíst data CAN
            spi_frame_in[i] = *(&CANRDSR0 + i);
        RX_flag_list[RXF_SPI] = BUF_INCOMING; // Nahodit flag
    }
    break;
```

Druhé přerušení (*Dokončení čtení A/D*) zajišťuje data z IR senzorů. Čtení z prvního kanálu je nejprve vyvoláno čítačem nebo zprávou po CAN a následně se v každé iteraci přerušení (dokončení předchozího kanálu) zapíšou data do pole a vyvolá čtení z dalšího kanálu. Po posledním IR senzoru se nahodí „flag“ pro odeslání dat po CAN do PC, což se následně provede v hlavním programu.

```
/* Ukázka kódu přečtení jednoho IR kanálu a vyvolání dalšího */
adc_data[1] = ADCRH<<4 | ADCRL>>4; // Získat hodnotu z infra senzoru
adc_data[0] = (ADCRL>>4 & 0x01);  // Uložit LSB do kontrolního pole
if (adc_channels >= 1) // Pokud jsou následující kanály
    ADCSC1 = (ADCSC1&0xE0)|0x00; // Začít měření dalšího kanálu
else done = 1; // Jinak konec
break;
```

Poslední přerušení (*Přetečení čítače*) je nastaveno ke spuštění každých 100 us a pomocnou proměnnou zpomaleno na 10 ms. V každém průběhu porovnává dosažený počet

průběhů s násobkem měřicí a odesílací periody každého ze senzorů (nastaveno pomocí konfiguračních zpráv z PC) a v případě shody spustí čtení nebo odesílání dat pro daný senzor. To znamená, že pokud například modul kompasu má nastavený svůj násobek periody na 20, bude měřen každých 200 ms ($10ms \times 20$). Toto přerušení spouští měření a posílání dat pro IR senzory, ultrazvukové senzory, gyroskop s akcelerometrem a kompas.

```
/* Ukázka kódu spuštění měření pro určitý senzor */
if (config_list[BUF_CMPS]) // Pokud perioda měření kompasem není nulová
{
    if (!(state % config_list[BUF_CMPS])) // Pokud perioda dosažena
        RX_flag_list[RXF_CMPS] = CMPS_START; // Začít měření
}
```

3.3.2 Hlavní program – funkce *main()*

Průběh hlavního programu (funkce *main()* ve zdrojovém souboru *main.c*) je patrný víceméně z vývojového diagramu na obr. 3.3. Po spuštění jednotky se nejprve nastaví výchozí 5-bitové ID jednotky a spustí inicializace MCU vytvořená pomocí Processor Expert (zmíněno u popisu souborů). Poté se zkontroluje, jestli není v trvalé paměti (EEPROM) uloženo ID dříve nastavené z řídicí aplikace. Pokud ano, načte se tato hodnota. Následně proběhne inicializace CAN pomocí funkce *can_init()* v témže souboru. V ní se nastaví její konfigurační registry (rychlost komunikace apod.) a filtry podle specifikace použitého protokolu CAN (viz. 3.1), které se aplikují na příchozí zprávy, aby se přijímaly jen ty, jež jsou určeny pro jednotky senzorů. Ještě proběhne inicializace displeje, k níž se využívá driver v souborech *lcd.c* a *lcd.h* (viz. [18]). Dále program pokračuje nekonečnou smyčkou, která je nezbytnou součástí každého softwaru pro mikroprocesory. Procesor v ní zůstane až do svého vypnutí.

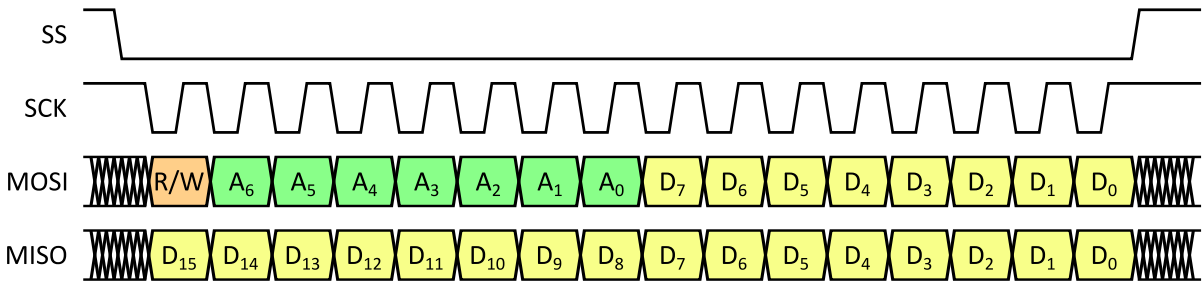
Nekonečná smyčka obsluhuje části kódu vyvolané pomocí „flagů“ z přerušení. Každá položka obsluhy spouští funkci umístěnou ve stejném souboru (*main.c*):

- *Obsluha LED* – po kontrole, zda jsou LED v manuálním režimu, na ně zapíše data přijatá po CAN. V automatickém režimu zobrazují LED aktivitu jednotky (blikání při příchozích a odchozích zprávách po CAN).
- *Obsluha servomechanismů* – pokud je nahozen „flag“ příchozích dat pro servomechanismy, provede příslušnou akci (nastavení požadované polohy nebo režimu). V manuálním režimu se reaguje na požadavky polohy a v automatickém se poloha mění automaticky skokově přes rozsah servomechanismu a zpět, což umožňuje namontovaným IR senzorům periodicky zkoumat vzdálenost v půlkruhu před mobilní platformou.
- *Obsluha SPI* – spustí funkci pro sběrnici SPI, na které se nachází senzor gyroskopu s akcelerometrem. Dále je rozebrána v následující sekci 3.3.3.

- *Obsluha I^2C* – vyvolá funkci pro sběrnici I^2C, kde jsou ultrazvukové senzory a modul kompasu. Podrobněji je vysvětlena v další sekci 3.3.4.
- *Obsluha displeje* – v případě aktivního „flagu“ nových dat pro LCD se na něj daná data zapíšou opět s využitím ovladače pro LCD (viz. [18]).
- *Obsluha CAN* – spustí funkci pro odesílání dat po CAN, ve které se postupně zkontroluje, jestli jsou aktivní „flagy“ pro jednotlivé senzory. Pokud ano, vyšle zprávy po sběrnici CAN. Funkce je napsána tak, aby vyhovovala specifikacím CAN, které již byly probrány dříve (viz. 2.4.4) a dále je zde tedy nerozebírám. Využity byly také příklady kódu pro MCU (viz. [21]).
- kontrola nového ID – pokud přišlo nové ID jednotky po sběrnici CAN, nastaví se do proměnné pro aktuální použití a zároveň zapíše do trvalé paměti (EEPROM), aby bylo zachováno po vypnutí a znovuzapnutí jednotky.

3.3.3 Obsluha sběrnice SPI

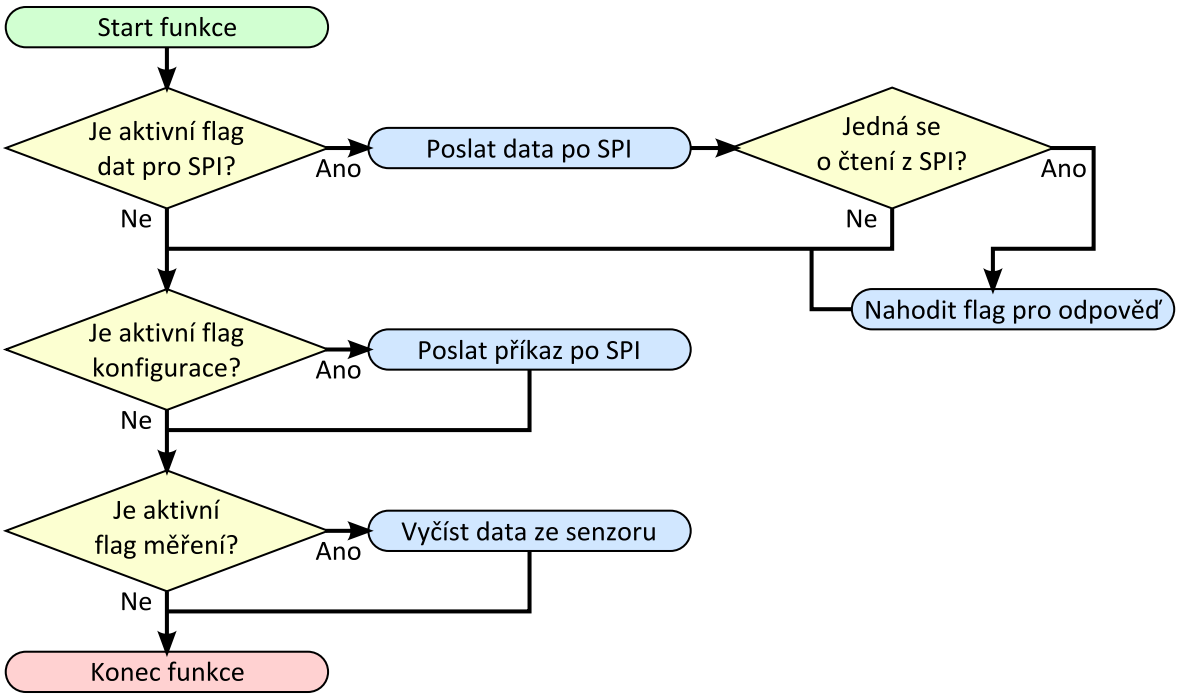
Pro nastavování a čtení dat z modulu gyroskopu s akcelerometrem je nutné porozumět sběrnici SPI. Nejprve bych tedy stručně popsal její specifikace (doplňující informace viz. [23]). Jedná se o synchronní sběrnici schopnou propojit dva nebo více uzlů, kde jeden je vždy Master (řídicí) a ostatní Slave (podřízené). Je velmi jednoduchá na implementaci a má postačující rychlost (běžně 1 MHz). Skládá se ze čtyř vodičů: SCK (Serial Clock – hodinový signál), MOSI (Master Out Slave In – data od uzlu Master ke Slave), MISO (Master In Slave Out – data od uzlu Slave k Master) a SS (Slave Select – výběr aktivního uzlu Slave – těchto vodičů může být více podle počtu uzlů Slave, kde každý vodič vybírá jeden uzel). Komunikace probíhá tak, že Master vybere daný uzel (udělá ho aktivním) pomocí signálu SS do „0“ a následně vysílá hodinové impulsy, pomocí nichž se synchronizují obě datové linky. Během například 8 pulsů se tak zároveň přenese 8 bitů z registru v uzlu Master do registru v uzlu Slave a opačně. Rutina pro SPI v MCU jednotky senzorů (zde uzel Master) se odvíjí od tohoto průběhu komunikace. Průběh je znázorněn na obr. 3.4 (kompletní informace o komunikaci s tímto konkrétním senzorem viz. [15]).



Obr. 3.4: Průběh komunikace SPI mezi MCU a senzorem

Data se čtou nebo zapisují v tomto případě vždy při vzestupné hraně SCK. Pokud chceme do senzoru zapsat, provedeme operaci z obrázku jednou (16 hodinových pulsů) s nastavením bitu R/W (Read/Write) do „1“, 7 adresových bitů podle zvoleného registru a 8 datových bitů podle požadovaného obsahu zápisu. Příchozí bity (16) zahodíme. Naopak v případě čtení ze senzoru musíme provést sekvenci dvakrát. V prvním průběhu nastavíme bit R/W do „0“ a adresové bity (7) opět podle registru (na datových nezáleží). Příchozí zahodíme. V druhém průběhu odešleme 16 nulových bitů a přijde nám 16 datových bitů ze senzoru (MSB je vždy vlevo), které si uložíme.

Funkce pod položkou *Obsluha SPI* tedy za pomocí rutiny popsané v předchozím odstavci, jež se nachází v pomocném souboru *serials_com.c*, nastavuje a čte data ze senzoru gyroskopu s akcelerometrem. Provádí funkce z vývojového diagramu na obr. 3.5. Reaguje na aktivní „flagy“ nastavené v některém z přerušení (příchozí zpráva po CAN s daným požadavkem nebo přetečení čítače). Postupně může zapisovat nebo číst některý registr na SPI (při čtení navíc nahodí „flag“ pro následnou odpověď přečtených dat po CAN, která se provede v obsluze sběrnice CAN v hlavním programu). Poté může posílat konfiguraci pro senzor akcelerometru s gyroskopem nebo z něj data číst. Data jsou již zpracována v samotném modulu senzoru a není nutná jejich další filtrace nebo úprava.

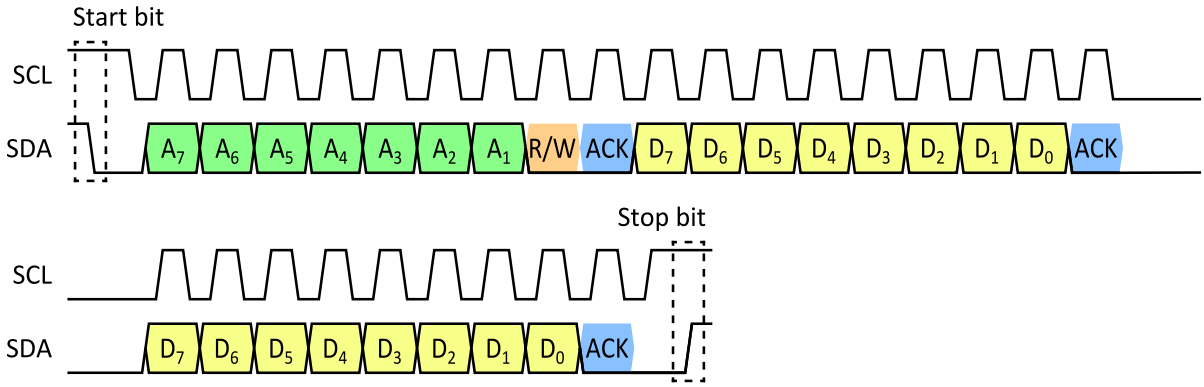


Obr. 3.5: Vývojový diagram obslužné funkce SPI

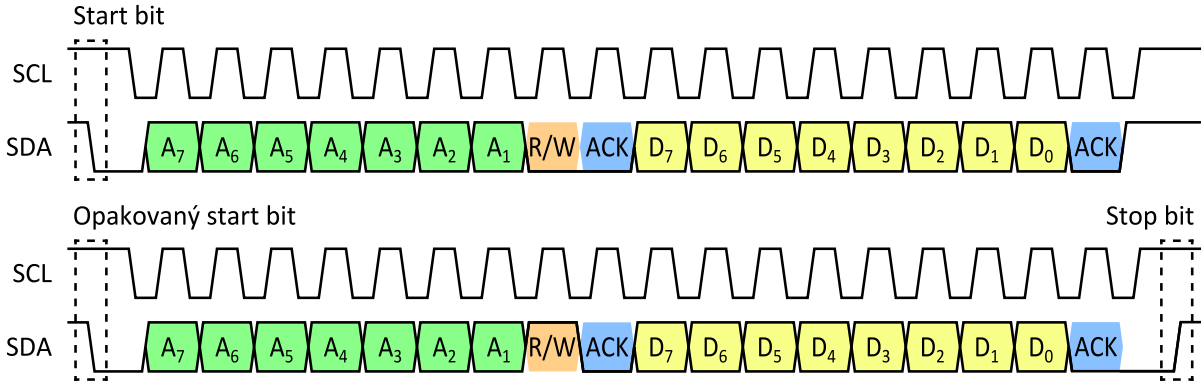
3.3.4 Obsluha sběrnice I²C

Obdobně pro komunikaci s ultrazvukovými senzory a kompasem musíme znát specifikace sběrnice I²C, které jsou zde uvedeny (doplňující informace viz. [23]). Jedná se opět

o synchronní typ komunikace, která může propojit jeden uzel Master (zde MCU) a jeden nebo více uzlů Slave (zde senzory kompasu a ultrazvukových dálkoměrů). Dosahuje typické rychlosti 100 kHz, která je dostatečná pro jednotky senzorů. Hlavní rozdíl oproti SPI je využití pouze dvou vodičů: SCL (Serial Clock – hodinový signál) a SDA (Serial Data – datový signál). Obecně má I^{2}C komplexněji definovaný komunikační protokol a je proto také složitější na obsluhu. Výhodou je méně vodičů, na druhou stranu je ovšem nutné posílat před daty také adresu uzlu Slave, protože nemáme vodič, jenž by ho zvolil jako aktivní. Nevýhodou je také nemožnost současné komunikace oběma směry. Rutiny pro I^{2}C jsou tedy dvě (zápis a čtení) – opět k nalezení v souboru *serials_com.c*. Průběhy jsou znázorněny na obr. 3.6 a kompletní informace o komunikaci s konkrétními senzory lze najít pod čísly [12] a [13] v seznamu literatury.



(a) Sekvence zápisu



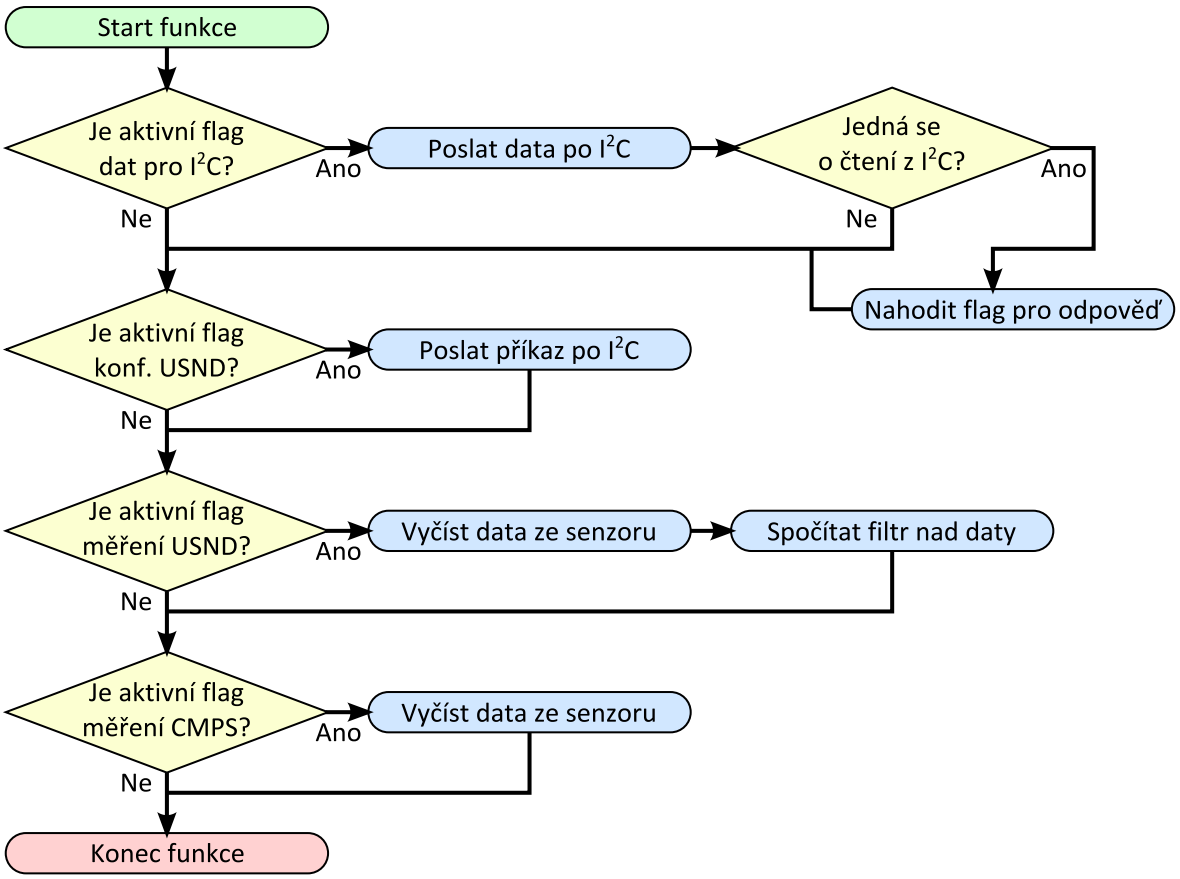
(b) Sekvence čtení

Obr. 3.6: Průběh komunikace I^{2}C mezi MCU a senzory

Při zápisu zahájí MCU (Master) průběh start bitem, následně vyšle 7-bitovou adresu daného modulu a bit R/W (Read/Write) v „0“. V bitu ACK musí modul (Slave) potvrdit příjem udržením SDA v „0“, jinak je komunikace neúspěšná a ukončena ze strany Master. Poté pošleme byte dat určující číslo registru modulu, kam se zapisuje, a opět proběhne bit ACK. Následuje jeden byte dat k zapsání do registru (lze i více) a každý je následován bitem ACK. Komunikace je ukončena stop bitem.

Čtení také začíná start bitem a úvodní sekvence adresy a čísla registru je totožná. Poté ale MCU vyšle bit opakovaného startu, za kterým je stejná adresa, ovšem s bitem R/W v „1“. Po bitu ACK se vysílání ujme Slave a vyšle byte dat s obsahem daného registru do MCU (lze jeden nebo i více – tuto délku ovšem musí MCU znát). Tentokrát MCU potvrdí příjem bitem ACK v „0“ (umístěn za každý přijatý byte dat). Stejně jako při zápisu je pak komunikace ukončena stop bitem.

Funkce pod položkou *Obsluha I^2C* s pomocí rutiny popsané výše nastavuje a čte data ze senzorů kompasu a ultrazvukových dálkoměrů. Probíhá podle vývojového diagramu na obr. 3.7. V případě aktivních „flagů“ (nastavené v některém z přerušení) může zapisovat nebo číst specifikovaný registr na I^{2}C (při čtení navíc nahodí „flag“ pro následnou odpověď přečtených dat po CAN, která se provede v obsluze sběrnice CAN v hlavním programu). Dále může posílat konfiguraci pro ultrazvukové senzory nebo číst data ze zmíněných senzorů (ultrazvukové dálkoměry a kompas). Data z ultrazvukových senzorů jsou ještě v rámci této obsluhy filtrována prostřednictvím klouzavého průměru (jeho funkce je vysvětlena v sekci 4.3.3), protože je jejich průběh zašuměný a může obsahovat odchylky způsobené náhodnými odrazy. Naopak data z kompasu jsou již zpracována samotným modulem a není zde nutná jejich další modifikace.

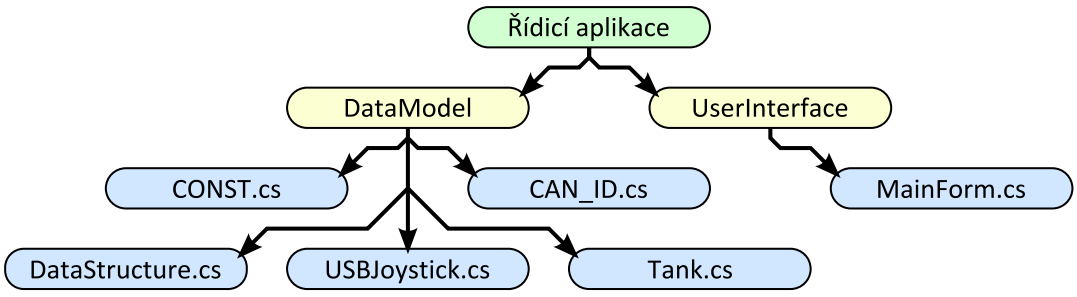


Obr. 3.7: Vývojový diagram obslužné funkce I^{2}C

4

Řídicí aplikace mobilní platformy

Po osazení nezbytného hardwaru a naprogramování funkcí do jednotlivých jednotek se lze v této poslední kapitole zaobírat tvorbou řídicí aplikace, což je nejdůležitější část celé diplomové práce. Aplikace je podobně jako celá platforma řešena modulárně tak, aby opět usnadnila následující rozvoj řídicích algoritmů. Psána je v jazyce C# na principu objektově orientovaného programování a k vývoji je využit software Microsoft Visual Studio 2010 (Verze 10.0. Copyright ©2010 Microsoft Corporation). Řešení aplikace je rozděleno do dvou projektů, jak je vidět z obr. 4.1. První projekt s názvem *DataModel* (dále popsáno v sekcích 4.1 a 4.2) obsahuje datovou strukturu pro zpřehlednění použitých proměnných (soubor *DataStructure.cs*), definice konstant a ID pro zprávy po sběrnici CAN (*CONST.cs* a *CAN_ID.cs*), knihovnu pro komunikaci s ovladačem USB joystick (*USBJoystick.cs*) a třídu s funkcemi pro obsluhu periférií (*Tank.cs*). V této části se řeší veškeré funkce pro sběr dat ze senzorů a posílání dat akčním členům tak, aby následné samotné řízení bylo přehlednější a nemuselo se v něm zaobírat těmito knihovními funkcemi. V druhém projektu s názvem *UserInterface* (sekce 4.4) se nachází soubor s grafickým rozhraním a algoritmy pro jednotlivé módy řízení (*MainForm.cs*). Pro druhý projekt aplikace je ještě v této kapitole stručně vysvětlena teorie jednotlivých algoritmů, jež byly použity pro rozpoznávání obrazu s využitím ovladače MS Kinect, a teorie základních navigačních algoritmů (sekce 4.3). Celá aplikace je umístěna na přiloženém CD a ke kompletnímu pochopení nutnému například pro další rozšiřování také přehledně okomentována.



Obr. 4.1: Struktura řídicí aplikace

Modulární řešení spočívá v tom, že pokud se rozhodneme přidat na mobilní platformu například nový senzor, stačí vytvořit nové knihovní funkce bez zásahu do řídicích algoritmů (do nich můžeme samozřejmě tyto nové funkce dále zapracovat). Pokud chceme vylepšit některý z řídicích módů, nemusíme v tomto případě modifikovat knihovní funkce, ale pouze soubor ve druhém projektu. Stejně tak lze navrhnout zcela nové řídicí rozhraní s využitím stávajících knihoven.

4.1 Datová struktura a definice konstant

Nejprve bych zmínil datovou strukturu. Jedná se o třídu obsahující struktury s proměnnými pro jednotlivé funkční celky mobilní platformy. Proměnné jsou umístěny v jedné třídě mimo soubory s knihovními funkcemi a řízením, aby byly přístupny z obou těchto zdrojových kódů a zároveň jejich název logicky odpovídal jejich významu. Struktury jsou uvedeny v následujícím výpisu s popisem, jaké proměnné obsahují:

- *tracks* – proměnné vázané k jednotce motorů.
- *servoPosition* – proměnná s pozicí servomechanismu.
- *display* – proměnné k zobrazování dat na LCD.
- *infraSensors* – proměnné s daty z infračervených dálkoměrů.
- *ultraSensors* – proměnné s daty z ultrazvukových dálkoměrů.
- *compass* – proměnná s údajem o natočení kompasu.
- *acm* – proměnná s daty ze senzoru gyroskopu s akcelerometrem.
- *iic* – proměnné vázané k datům z registrů na sběrnici I^2C.
- *kinectACM* – proměnné s údaji z akcelerometru na zařízení MS Kinect.
- *gps* – proměnné pro data z GPS modulu.
- *skeleton* – proměnné pro pozice kloubů postavy sledované pomocí MS Kinect.
- *navigation* – proměnné pro navigační algoritmus.

Příklad přístupu k proměnné je poté vidět na následujícím úryvku kódu. Pro získání kompletního přehledu proměnných slouží samotný zdrojový soubor *DataStructure.cs*.

```
data.tracks.pwmLeft // Proměnná pro otáčky za minutu levého pásu
```

V souboru *CONST.cs* se nacházejí konstanty ke zpřehlednění kódu, aby nebylo nutné dohledávat významy použitých číselných hodnot (např. kódy pro datový byte zpráv jednotky motorů apod.). Ve zdrojovém kódu *CAN_ID.cs* se nachází ze stejného důvodu definice ID zpráv po sběrnici CAN. Tyto ID odpovídají použitému protokolu CAN popsanému v sekci 3.1.

4.2 Funkce pro obsluhu periférií

V této části se práce věnuje funkcím pro obsluhu jednotlivých periférií, jejich nastavování a získávání dat. Popisuje již hotové knihovny a jejich zapracování do aplikace (všechny zdrojové soubory se nachází ve složce *References* projektu řídicí aplikace) a zároveň nově vytvořené v rámci této práce, které bylo třeba naprogramovat na míru daným perifériím.

4.2.1 Knihovny pro MS Kinect

Pro komunikaci se zařízením MS Kinect se využívá soubor ovladačů přímo od společnosti Microsoft s názvem Kinect for Windows SDK (ke stažení viz. [24]). Po instalaci do PC v mobilní platformě jsou dostupné v programu Microsoft Visual Studio 2010 potřebné knihovny. Příklad sekvence příkazů (k zavolání v projektu se řízením) pro připojení k zařízení Kinect je zobrazen na následující ukázce zdrojového kódu. Kompletní návody, jak přistupovat k MS Kinect prostřednictvím jazyka C#, se nachází na stránkách společnosti Microsoft (viz. [3] a dále pod odkazem Programming Guide). Pomocí těchto návodů lze přistupovat k datům z barevné i hloubkové kamery, datům o pozici kloubů rozpoznané postavy a údajům z akcelerometru zabudovaného do zařízení. Licence na stránkách z odkazů umožňují využití knihoven a návodů pro nekomerční využití a tedy i v tomto projektu.

```
/* Ukázka kódu vytvoření komunikace s MS Kinect */
using Microsoft.Kinect; // Načtení reference na knihovnu
private KinectSensor sensor = KinectSensor.KinectSensors[0]; // Objekt
sensor.Start(); // Zapnout MS Kinect a povolit barevnou kameru
sensor.ColorStream.Enable(ColorImageFormat.RgbResolution640x480Fps30);
```

4.2.2 Knihovny pro USB joystick

Získávání informací o stisknutých tlačítkách na ovladači USB joystick umožňuje knihovna *USBJoystick* od vydavatele AForge.NET (dokumentace viz. [25], ke stažení jako součást AForge.NET Framework – internetové stránky viz. [26] a dále pod odkazem Downloads). Dostupná je pod licencí LGPL v3, která umožňuje jeho využití v tomto projektu. Nachází se také v souboru *USBJoytick.cs*. Po instalaci ovladačů (dostupné na přiloženém CD) a zahrnutí této knihovny do projektu je připojení zařízení jednoduché a nezbytné příkazy (opět k volání v projektu s řízením) jsou na následující ukázce kódu. Kompletní dokumentace, jak třídu používat, je k dispozici na uvedených stránkách poskytovatele.

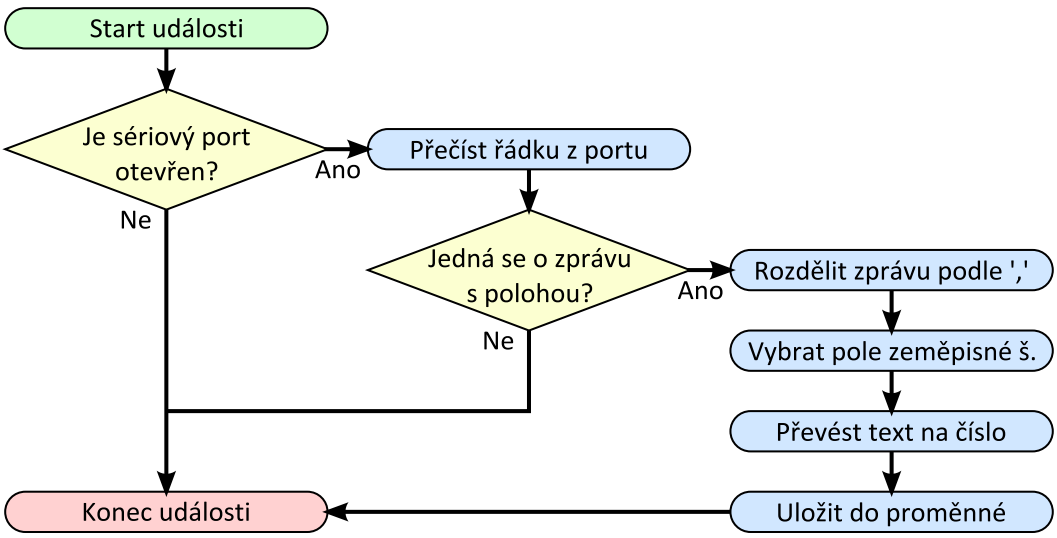
```
/* Ukázka kódu vytvoření komunikace s ovladačem USB joystick */
using AForge.Controls; // Načtení reference na knihovnu
private List<Joystick.DeviceInfo> joystickList; // Nový list zařízení
private USBJoystick joystick = null; // Nový objekt
joystickList = Joystick.GetAvailableDevices(); // Získat zařízení
joystick = new USBJoystick(0); // Připojit se k zařízení
```

4.2.3 Funkce pro modul GPS

Pro GPS modul byly vytvořeny zcela nové funkce (v souboru *Tank.cs*). Jak již bylo řečeno v sekci 2.4.3, po připojení se GPS chová jako sériový port a podle toho se s ní také komunikuje. Pro navázání a zrušení komunikace slouží funkce *GPSInit()* a *GPSDeInit()*, v nichž je v podstatě naprogramováno otevření a zavření sériového portu s daným číslem a komunikační rychlostí (viz. katalogový list o GPS s číslem [5]) a vytvoření nebo smazání událostí, které se vyvolají při nových datech přijatých z GPS. Tyto události umožňují příjem dat do řídicí aplikace. Celý kód je okomentován a k nahlédnutí ve zmíněném souboru. Funkce se poté volají ve druhém projektu aplikace (řízení).

```
/* Ukázka kódu vytvoření komunikace s modulem GPS */
using System.IO.Ports; // Načtení reference na knihovnu sériových portů
serialPortGPS.PortName = "COM3"; // Nastavení čísla portu
serialPortGPS.BaudRate = 4800; // Nastavení rychlosti komunikace
serialPortGPS.Open(); // Otevření portu a navázání události pro příjem
serialPortGPS.DataReceived +=
    new System.IO.Ports.SerialDataReceivedEventHandler(dataReceived);
```

Událost pro příjem dat (*dataReceived()*) zpracovává zprávy, které GPS periodicky posílá do sériového portu zakódované prostřednictvím protokolu NMEA 0183. K vytvoření události je nutné tento protokol nastudovat a následně zprávy dekódovat a získat z nich potřebná data. K pochopení protokolu mi cenné informace podal článek pod číslem [27] v seznamu literatury. Pro více informací je událost opět okomentována ve zdrojovém kódu a zde jen stručně nastíním její princip na příkladu vyčtení zeměpisné šířky ze zprávy od GPS, jenž je zobrazen na vývojovém diagramu z obr. 4.2.



Obr. 4.2: Vývojový diagram čtení informace ze zprávy od GPS

4.2.4 Funkce pro jednotky na sběrnici CAN

Jednotky připojené na sběrnici CAN komunikují skrze převodník USB/CAN a vyžadují obslužné funkce, které byly také vytvořeny v rámci tohoto projektu a nacházejí se opět v souboru *Tank.cs*. Převodník USB/CAN se stejně jako GPS modul chová po připojení a instalaci ovladačů (vytvořené v rámci projektu [1] a dostupné na přiloženém CD) jako sériový port a funkce na inicializaci (*CANCommInit()*), ukončení komunikace (*CANCommDeInit()*) a příjem dat tedy budou podobné. Dále se tyto funkce volají v projektu s řízením. Využívá se knihovna, která vynikla také jako součást zmíněného projektu (*Communication.dll* – dostupná ve složce *References*). Navíc je zde nutné vytvořit funkce pro odesílání dat z PC do jednotek, protože jednotky na rozdíl od GPS vyžadují také nastavování z řídicí aplikace. Na následující ukázce je funkce inicializace, kde se nastaví sériový port a naváže událost pro příjem dat.

```
/* Ukázka kódu vytvoření komunikace s převodníkem USB/CAN */
using Communication; // Načtení reference Communication.dll
CommunicationRoutines.SetConsoleEcho = true; // Povolit výpis do konzole
CommunicationRoutines.SetDefaultPortName = "COM8"; // Nastavení portu
commClass = new CommunicationRoutines(); // Inicializace komunikace
commClass.FrameReceived +=
    new CommunicationFrameReceivedHandler(commClass_FrameReceived);
if (commClass.Connect()) return true; // Otevření portu
else return false;
```

Následují ještě ukázky kódu pro přečtení konkrétní příchozí zprávy v události pro příjem dat (*commClass_FrameReceived()*) a pro odeslání zprávy dané jednotce.

```
/* Ukázka kódu pro příjem zprávy z jednotky motorů */
Byte[] dataCAN = e.frameData; // Vyčíst data z přijaté zprávy
Int32 ID = e.frameID; // Získat ID zprávy
switch (ID) { // Rozhodnutí o typu zprávy podle ID
    case CAN_ID.TRACKS_INFO_PWM: // Případ zprávy o otáčkách pásů
    data.tracks.pwmLeft = (Int16)(dataCAN[0]<<8) + (Int16)(dataCAN[1]);
    data.tracks.pwmRight = (Int16)(dataCAN[2]<<8) + (Int16)(dataCAN[3]);
    break; } // Přepočtení a uložení příchozích dat do proměnné

/* Ukázka kódu pro odeslání zprávy pro rozsvícení LED na jednotce */
Int32 id = CAN_ID.LED_SET_DATA_UNIT_B; // Nastavení ID
Byte[] dataCAN = new Byte[1]; // Vytvoření pole pro data a jeho naplnění
if (unit_B_yellow) dataCAN[0] |= 0x2; if (unit_B_red) dataCAN[0] |= 0x1;
if (commClass.Write(id, dataCAN)) Console.WriteLine("frame transmitted");
else Console.WriteLine("transmission failed"); // Odeslání a výpis
```

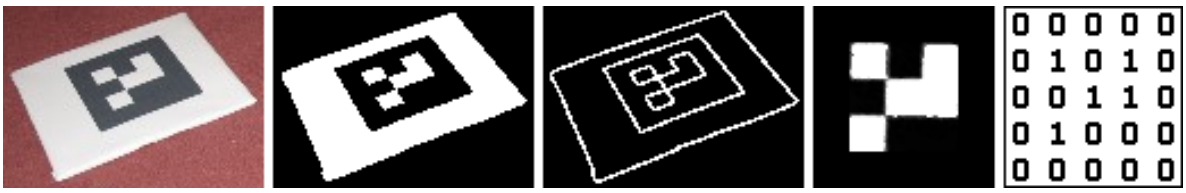
4.3 Teorie použitých algoritmů

V následující části práce stručně zmiňuje teorii algoritmů, jež se poté využívají při programování jednotlivých řídicích módů. Jedná se o rozpoznávání vzoru, ke kterému se používají knihovny vydavatele AForge.NET, rozeznávání lidské postavy skrze knihovny Kinect for Windows SDK v1.8 a navigaci pomocí GPS a kompasu, kde se využívají poznatky ze stránek robotika.cz.

4.3.1 Rozpoznávání vzoru pomocí knihoven AForge.NET

Rozpoznávání vzoru v obraze je komplexní problém, svým rozsahem odpovídající samotné diplomové práci, a v tomto projektu, který se zaobírá konceptem celé mobilní platformy, jsou proto k tomuto účelu využity knihovny od vydavatele AForge.NET (využity také pro komunikaci s ovladačem USB joystick, ke stažení jako součást AForge.NET Framework – internetové stránky viz. [26] a dále pod odkazem Downloads). Dostupné jsou pod licencí LGPL v3, která umožňuje jejich nasazení do tohoto projektu.

Teorie rozpoznávání vzorů s využitím zmíněných knihoven je popsána v článku pod číslem [28] v seznamu literatury, jehož součástí je také vzorový program pro rozpoznávání obrazců. Využívá se sekvence algoritmů pro zpracování obrazu: převedení do černo-bílé s definovaným prahem, rozdílový hranový detektor, separace oblastí připomínajících vzor, získání jejich souřadnic, transformace a rozdělení na mřížku 5x5 bodů a konečné porovnání se zadanými vzory v maticové podobě (bílá barva jako „1“ a černá jako „0“). Příklad detekce vzoru je na obr. 4.3. Jednotlivé kroky jsou detailně vysvětleny v uvedeném článku a v této práci, která není přímo zaměřena na zpracování obrazu, je z důvodu velkého rozsahu dále nerozebírám.



Obr. 4.3: Příklad detekce vzoru |Převzato z [28]|

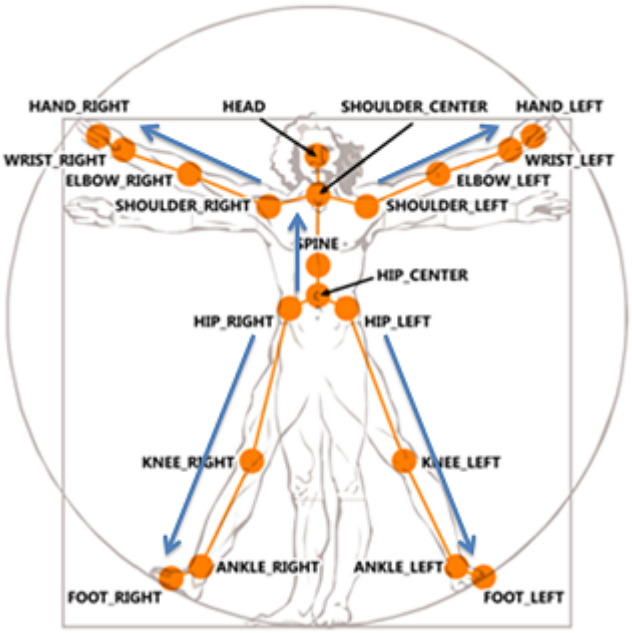
Implementace do samotné řídicí aplikace je inspirována dalším vzorovým programem, který propojuje předchozí algoritmy přímo s obrazem z kamery (viz. [29]). V aplikaci z této práce je dále upraven pro potřeby obrazu ze senzoru MS Kinect a doplněn o algoritmus výpočtu pohybu mobilní platformy podle pozice vzoru. Více o tomto algoritmu je v sekci o řídicím módu – rozpoznávání obrazce a jeho sledování (viz. 4.4.3).

4.3.2 Rozpoznávání lidské postavy s využitím knihoven Kinect for Windows SDK

Rozeznávání lidské postavy je opět velmi složitým problémem a za poměrně jednoduchou implementaci do aplikace využívající senzoru MS Kinect můžeme vděčit firmě Microsoft, která uvolnila ovladače (popsané v sekci 4.2.1) a ukázky zdrojových kódů pro práci s tímto senzorem k nekomerčnímu využití. Kompletní návody jsou k dispozici na stránkách o senzoru MS Kinect (viz. [3] a dále pod odkazem Programming Guide), kde jsou k nalezení i licence na ně vztažené. Zde je stručně shrnut postup získání dat o postavě. Nejprve je nutné povolit data o lidské postavě stejně jako například obraz z barevné kamery.

```
/* Ukázka kódu k zapnutí rozpoznávání postavy */
kinectDevice.SkeletonStream.Enable(); // Povolit stream a navázat událost
kinectDevice.SkeletonFrameReady += skeleton_NewFrameGestureRecognition;
```

Poté je možné přistupovat k informacím o rozpoznané postavě pomocí příkazů ze stránek o senzoru MS Kinect. Data jsou již zpracována v hardwaru senzoru, jsou k dispozici v navázané události a nevyžadují žádné další úpravy. Na obr. 4.4 jsou znázorněny klouby o kterých senzor podává informace a šipkami dostupné orientace kostí. Souřadný systém dat je již transformován do reálného prostoru a o každém kloubu známe souřadnice x, y a z. Jednotkou dat je metr. Bod [0, 0, 0] reprezentuje pozici senzoru, osa z kolmou vzdálenost od senzoru, osa x horizontální vzdálenost od osy z a osa y vertikální vzdálenost od osy z. Data jsou dále použita v módu rozpoznávání lidské postavy (viz. 4.4.4).



Obr. 4.4: Pozice kloubů a orientace kostí rozpoznané postavy |Převzato z [3]|

4.3.3 Algoritmy pro navigaci pomocí GPS a kompasu

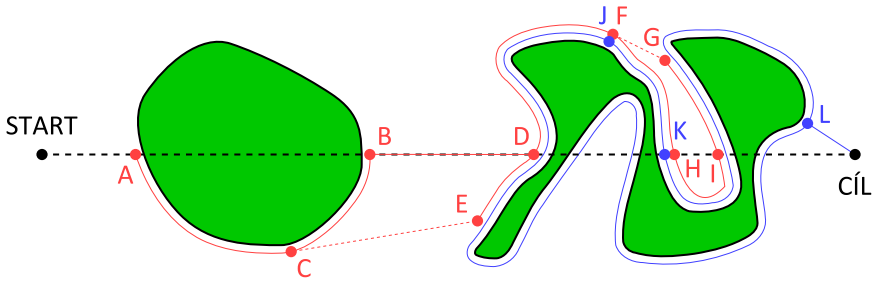
Navigace podle GPS s detekcí a objížděním překážek vyžaduje přesná data ze senzorů a důmyslné řídicí algoritmy. Úvod do problematiky je přehledně zpracován na stránkách robotika.cz (viz. [30] a dále sekce Průvodce), odkud jsem čerpal informace do tohoto projektu. Zde stručně nastíním základní principy a pro více informací poslouží zmíněné stránky a další odkazy, jež jsou na nich uvedeny. Implementace algoritmů přímo do mobilní platformy je dále rozebrána v popisu řídicích módů (sekce 4.4.5).

Prvním problémem je lokalizace. Předpokladem pro úspěšnou navigaci je znalost polohy a směru natočení. Pro jednoduché případy lze využít tzv. odometrii, kdy počítáme z pohybu modelu změnu od původní polohy a tím získáme polohu novou. Jedná se o metodu inkrementální. Pro navigaci v terénu s nerovnostmi a možností prokluzování pásů je ovšem zcela nedostatečná, protože vnáší chybu (více ve zmíněném zdroji informací). Proto musíme využít senzory absolutní, k čemuž slouží GPS a kompas. Ty vrací aktuální informace nezávislé na předcházejících. Jejich data jsou také nepřesná a proto musí být filtrována, což odstraní rychlé drobné změny. V projektu využívám jednoduché filtry klouzavého průměru (počítají průměr z daného počtu n posledních hodnot, m je číslo aktuálního vzorku), které se chovají jako dolní propust (rovnice 4.1). Takto upravená data nejsou dokonale přesná, ale lze je již využít v navigačních algoritmech pro hrubé určení směru a zadání cíle s tolerancí (pro GPS souřadnice jednotky až desítky metrů). Filtry vnášejí do dat zpoždění, které ale není pro pomalou jízdu pomocí navigačního algoritmu překážkou. Přesnost a stabilitu dat lze vylepšit použitím složitějších filtrů (např. Kalmanův), jež mohou být předmětem vylepšení navigace v dalších projektech.

$$x = \frac{x_m + x_{m-1} + x_{m-2} + ... + x_{m-(n-2)} + x_{m-(n-1)}}{n} \tag{4.1}$$

Se znalostí polohy platformy a cíle lze již vypočítat požadovaný směr (úhel vektoru s počátkem v aktuální poloze a koncem v souřadnicích cíle). Porovnáním s hodnotou z kompasu získáme úhel, o který se musí platforma otočit, aby směřovala předkem k cíli. Po provedení natočení již může platforma vyrazit vpřed a její pohyb se může vylepšit například mírným zatáčením s rostoucí odchylkou od požadovaného směru, což zvýší plynulost pohybu. Není pak nutné při ztracení směru o malou hodnotu ihned rotovat (jeden pás vpřed a druhý vzad), ale do určité meze je odchylka kompenzována zpomalováním pohybu jednoho z pásů, což má za následek zatáčení a zároveň pohyb vpřed. Cíl je detekován v případě shody aktuálních a cílových souřadnic, čímž je navigace úspěšně ukončena.

Po rozpohybování modelu je nutné vyřešit druhý zásadní problém, kterým je detekce a objíždění překážek. Prozatím máme mobilní platformu směřující nejkratší cestou k cíli, čímž se pohybuje doslova jako tank a ignoruje veškeré předměty, které jí stojí v cestě. Překážky jsou detekovány s využitím dálkoměrů (infračervené a ultrazvukové). Pro zahrnutí jejich objíždění existují například algoritmy s názvem „bug“, které jsou dále využity v tomto projektu. Jejich princip je zřejmý z následujícího nákresu (obr. 4.5).



Obr. 4.5: Princip algoritmu pro objíždění překážek

Myšlenka je taková, že po detekci překážky se upustí od směřování k cíli podle kompasu a začne se vykonávat algoritmus pro objíždění. Algoritmy pro zcela neznámé prostředí se znalostí polohy, což je případ mobilní platformy v terénu, existují dva. Bug1 je robustnější, ale pomalejší. Bug2 je rychlejší (pohyb se jeví jako více inteligentní), ale může v některých případech zcela selhat. Řešení je tedy takové, že při objevení překážky začneme s algoritmem Bug2 a při jeho selhání přepneme na Bug1. Algoritmus objíždění je ukončen po objetí překážky a platforma se vrátí k jízdě podle kompasu.

Bug2 je na obrázku znázorněn červeně a funguje pro jakoukoli konvexní překážku. Po vydání se od startu k cíli se v bodě A narazí na překážku a kroky jsou následující:

1. začni obcházet náhodným směrem (lze vylepšit zvolením směru podle orientace překážky - vybere se příznivější vzhledem k cíli)
2. při nalezení úsečky start-cíl (bod B) blíže k cíli než bod A se opět vydej k cíli podle kompasu (lze vylepšit opuštěním překážky v místě, kde platforma směřuje předkem k cíli a nemá v cestě překážku - bod C)
3. pokud se po návratu do bodu A nenalezne bod opuštění, navigace je ukončena a je oznámena nemožnost nalezení cesty

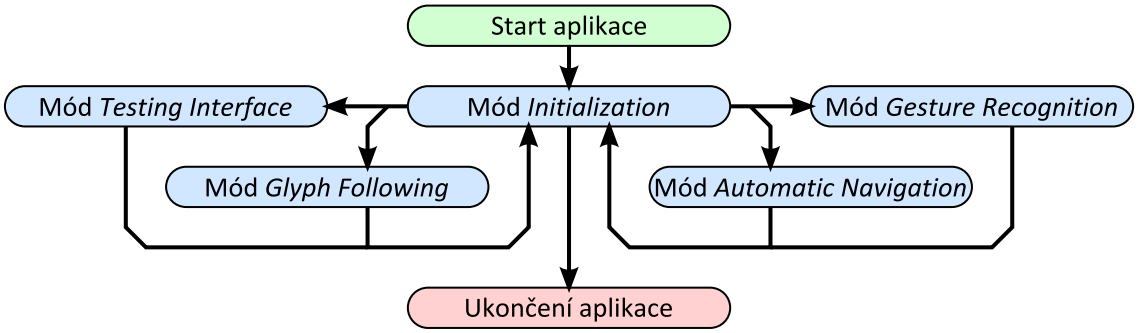
Dále se řeší případ, kdy Bug2 selže. Narazí se na novou překážku (body D nebo E) a začne opět Bug2. Vydá se doleva. Ovšem podle zvolené metody opuštění – v bodě F (při opuštění v místě natočení k cíli) nebo H (při nalezení úsečky) – se Bug2 následně v bodě G nebo I, kdy opět narazí na překážku, začne pohybovat v protisměru („výhodnější“ sklon k cíli). Až by se dostal do prohlubně na druhé straně, otočil by se ze stejného důvodu. Proto je nutné přepnout na Bug1, jakmile nastane situace, že po nárazu na překážku se jejím objížděním Bug2 dostane do stejného bodu, kde ji prve opustil. Bug1 je zdlouhavější, ale funguje vždy. Jeho princip je následující (v obrázku modře):

1. pokračuj v obcházení stejným směrem a zapamatuj výchozí bod (J nebo K)
2. objeď celou překážku kolem dokola a zapamatuj nejbližší bod k cíli (L)
3. dojeď do bodu L kratší cestou a vydej se k cíli podle kompasu (pokud směr k cíli směřuje do překážky, je opět oznámeno nenalezení cesty)

Tímto je implementována metoda objíždění překážek, navigační algoritmus pro cestu k cíli je kompletní a lze ho nasadit do mobilní platformy.

4.4 Módy řízení mobilní platformy

V poslední sekci kapitoly se věnuji druhé části aplikace – projektu samotného řízení (*UserInterface*). Jeho zdrojový kód se nachází převážně v souboru *MainForm.cs* a několika dalších podpůrných souborech. Jedná se o projekt s grafickým rozhraním postaveném na principu Windows Forms (vlastnost vývojového prostředí Microsoft Visual Studio 2010). Na vývojovém diagramu (obr. 4.6) je nejprve vysvětlena struktura řídicí části, kde šipky zobrazují možné přechody, a poté se práce zabývá jednotlivými módy – inicializace mobilní platformy (*Initialization*), pohyb pomocí ovladače joystick (*Testing Interface*), rozpoznávání obrazce s jeho sledováním (*Glyph Following*), rozpoznávání lidské postavy (*Gesture Recognition*) a navigace pomocí GPS s kompasem (*Automatic Navigation*). Ukázky jejich grafického rozhraní jsou umístěny v příloze D. Z důvodu velkého rozsahu projektu se zde vysvětlují principy a celková funkčnost, přičemž pro podrobnější pochopení jednotlivých metod slouží okomentované zdrojové kódy na přiloženém CD.



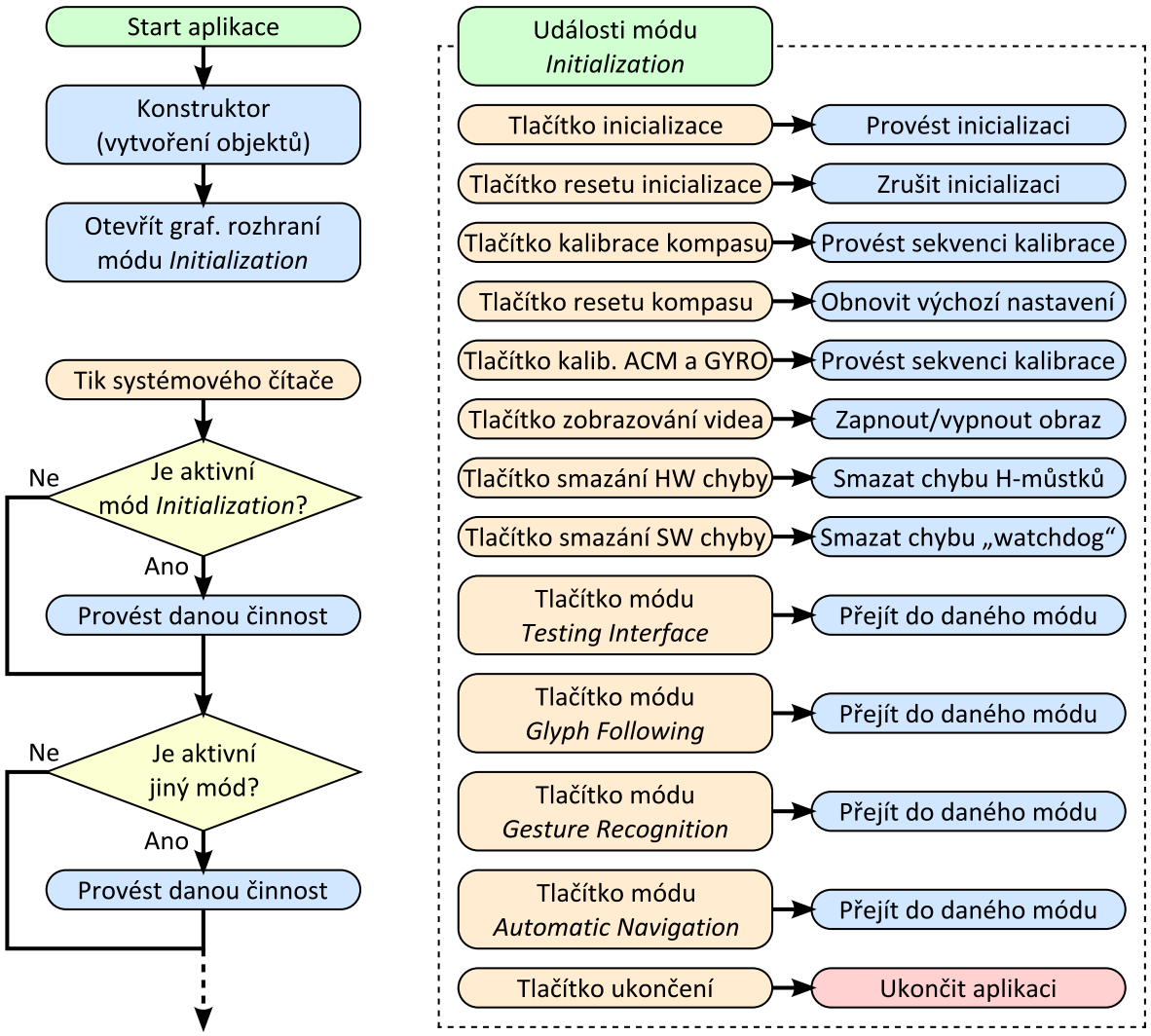
Obr. 4.6: Vývojový diagram přechodů mezi módy aplikace

Vzhledem k tomu, že aplikace je tvořena na principu objektově orientovaného programování v jazyce C#, nenajdeme v dalších vývojových diagramech této kapitoly jeden souvislý sled událostí, kde by příkazy na sebe navazovaly tak, jak jsou fyzicky napsány v kódu. Takový průběh by byl typický pro strukturované programování (např. jazyk C). Objektově orientované programování je založeno na třídách, objektech a metodách (funkce a události). Třída je soubor s popisem daného objektu, jeho vlastností (proměnných) a schopností (funkcí a událostí). Od dané třídy můžeme vytvářet libovolné množství objektů s konkrétním jménem. S nimi se pak dále pracuje v samotném kódu. Můžeme měnit jejich proměnné, vyvolávat jejich metody pro určitou činnost nebo obsluhovat události, jež nastanou v případě nějaké změny v daném objektu. O periodickou kontrolu událostí se stará prostředí vytvořené překladačem a aplikace se tímto dále zaobírat nemusí.

4.4.1 Inicializace mobilní platformy – *Initialization*

Prvním módem je inicializace aplikace a periférií mobilní platformy, která je nutná k dalšímu fungování řídicí aplikace. Podíváme-li se na vývojový diagram z obr. 4.7, po startu

aplikace se nejprve v konstruktoru vytvoří objekty nutné pro její fungování. Například objekt s názvem *Tank* od třídy v souboru *Tank.cs* (viz. 4.2.3 a 4.2.4), který umožní interakci s periferiemi na platformě. Stejně tak se vytvoří objekty od dalších knihoven (MS Kinect a USB joystick). Následně se spustí grafické rozhraní začínající právě v módu *Initialization* (jeho rozvržení lze vidět v příloze na obr. D.1). Grafické rozhraní obsahuje prvky (tlačítka atd.), což jsou také objekty od svých tříd, a mají tedy svoje proměnné i události, se kterými lze dále pracovat. Například při stisku tlačítka se vyvolá událost, ve které je napsán kód určující další činnost. Celou aplikaci včetně přepínání módů je umožněno ovládat pomocí zařízení joystick z důvodu, že ne vždy máme k dispozici monitor a grafické rozhraní. Tabulka významu tlačítek je uvedena v příloze E a možnost řízení pomocí ovladače joystick je zmiňována i v následujícím popisu u příslušných příkazů.



Obr. 4.7: Vývojový diagram módu inicializace mobilní platformy

Na vývojovém diagramu jsou dále vidět události zpracovávané v tomto módu. Jedná se o tlačítka sloužící k nastavení mobilní platformy, přechodu do dalších módů vykonávajících určitý druh řízení a o tlačítko ukončení aplikace:

- inicializace spustí příkazy, kde se zapne komunikace s periferiemi mobilní platformy (MS Kinect, USB joystick, jednotky na sběrnici CAN, GPS), jež začnou periodicky posílat data, a spustí se systémový čítač. Reset inicializace opět funkce vypne. Během obou operací se zobrazují zprávy o úspěchu či neúspěchu do výpisu v grafickém rozhraní aplikace, který slouží k rychlému odhalení chyb. Sekvenci reset a spuštění inicializace je také možné vyvolat tlačítkem *Start* na ovladači USB joystick.

- příkazy nastavení provedou požadované úkony:

  - kalibrace kompasu spočívá v postupném manuálním natočení modelu do směrů čtyř světových stran, což umožní senzoru uložit si tyto hodnoty a zpřesnit tak svoje měření v dané oblasti zemského povrchu. Požadované kroky se zobrazují ve zprávě v grafickém rozhraní. Více informací o kalibraci viz. [13]. Reset kompasu pošle senzoru příkaz k návratu do továrního nastavení.
  - kalibrace senzoru akcelerometru a gyroskopu spustí jeho vnitřní funkci, která trvá 30 s a během níž musí být model ve zcela vodorovné poloze. Cílem je zpřesnit výstupní hodnoty tohoto senzoru. Více informací viz. [15].
  - přepnutí videa umožňuje povolení nebo zakázání zobrazování přenosu z kamer senzoru MS Kinect v grafickém rozhraní. Například při připojení k PC mobilní platformy pomocí vzdálené plochy přenos videa velmi zpomaluje běh celé aplikace a je proto nutné ho zakázat.

- smazání HW chyby odstraní z jednotky řízení motorů chybu H-můstků v případě jejího výskytu a uvede modul zpět do provozu. Smazání SW chyby povolí v řídicí aplikaci opětovné odesílání zprávy o otáčkách pro jednotku motorů, pokud ta dříve nahlásila chybu „watchdog“. Tato vlastnost je přítomna z důvodu, aby uživatel musel zasáhnout po výskytu této poruchy. Jednotka by se jinak po obdržení nových otáček pro motory znovu rozběhla a uživatel by nemusel výpadky aplikace zaznamenat a dále řešit jejich příčinu. Chyby lze také vymazat pomocí ovládacích páček ze zadní strany zařízení joystick. Více informací, jak nastávají jednotlivé poruchy, bylo probráno v sekci 3.2.

- tlačítka pro přepnutí módů spustí vybraný mód a zobrazí jeho grafické rozhraní v novém panelu. Přepínání je také umožněno pomocí ovladače joystick (tlačítka *A*, *B*, *X*, *Y*). Návrat do módu inicializace lze provést tlačítkem v jeho grafickém rozhraní nebo opět ovladačem joystick (tlačítko *Back*).

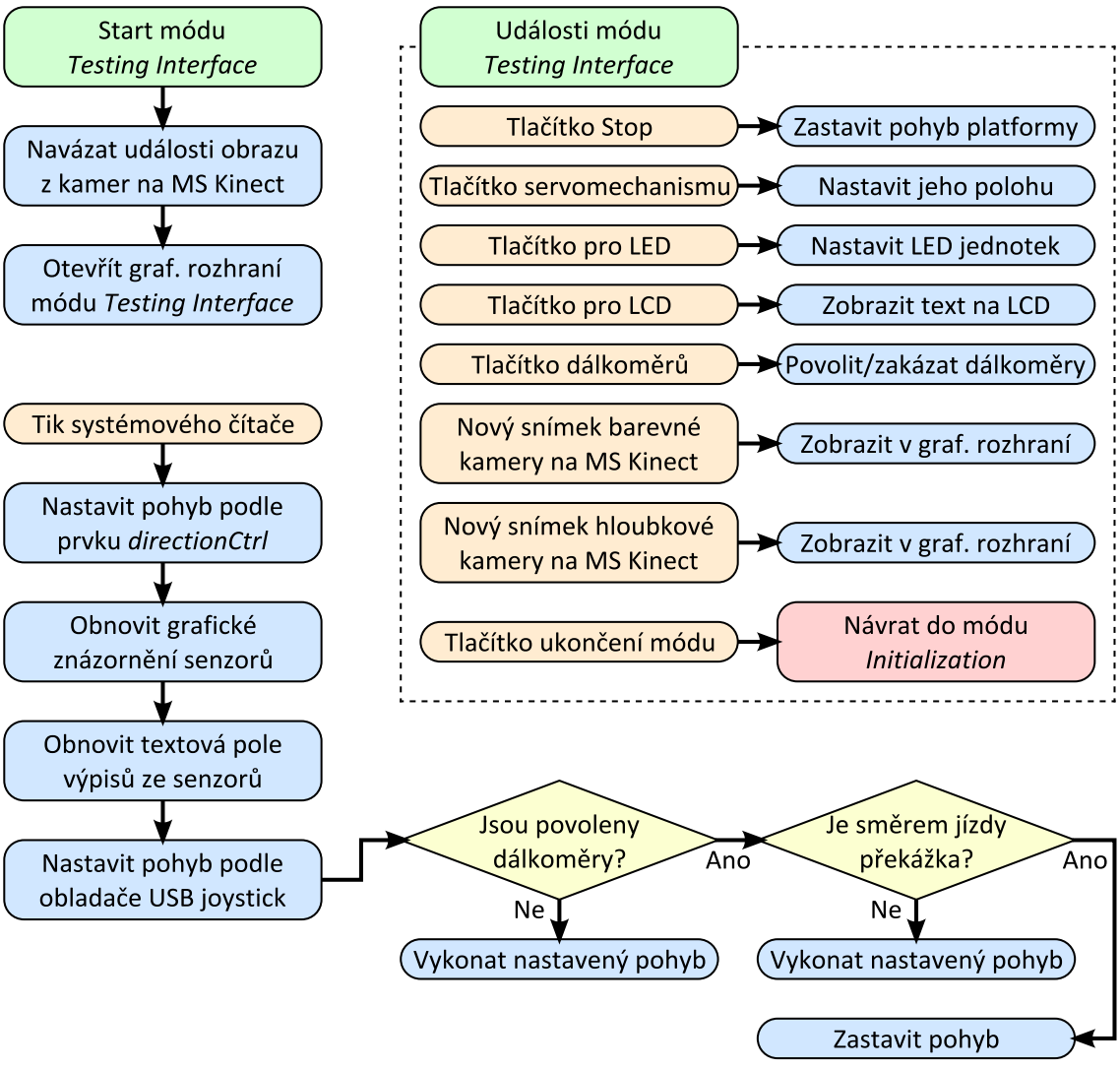
- ukončení provede zavření řídicí aplikace s předchozím vyvoláním resetu inicializace, aby se nejdříve vypnula činnost periférií mobilní platformy.

V diagramu je ještě znázorněn průběh systémového čítače, který obsluhuje činnost vyžadující periodické opakování. Nastaven je na periodu 100 ms. Pokud nastane jeho událost (tik), provede se kód podle aktuálního módu. Při inicializaci se pouze posílají zprávy

o otáčkách s nulovými hodnotami jednotce motorů pro zabránění chybě funkce „watchdog“ (viz. 3.2) a zároveň se zobrazují chybové zprávy přicházející z jednotky. Ve zbylých módech se pak čítač stará o procesy řízení.

4.4.2 Řízení pomocí ovladače joystick – *Testing Interface*

Další mód umožňuje řízení mobilní platformy pomocí ovladače joystick. Hlavní funkcí je kontrola jízdy mobilní platformy pomocí tohoto ovladače a využití dálkoměrů (infračervených a ultrazvukových) k automatickému zastavování před překážkami. V aplikaci je nazván *Testing Interface* (testovací rozhraní), protože zároveň slouží pro přehledné zobrazení dat ze všech senzorů a k ovládání akčních členů mobilní platformy pomocí prvků grafického rozhraní. To umožňuje rychlé vyhodnocení funkčnosti periférií modelu. Jeho metody jsou shrnuty na vývojovém diagramu z obr. 4.8 a ukázka grafického rozhraní je k nalezení v příloze na obr. D.2.



Obr. 4.8: Vývojový diagram módu řízení pomocí ovladače joystick

Po startu módu se nejprve vytvoří reference na události pro zpracování obrazu ze senzoru MS Kinect a následně se otevře panel grafického rozhraní pro tento režim aplikace. Na něm jsou znázorněny pohledy obou kamer senzoru MS Kinect, panel pro vykreslování údajů z ostatních senzorů, znázornění otáček pásů a několik textových polí pro výpis údajů, které nelze jednoduše graficky znázornit (př. GPS). Dále je k dispozici ovládací prvek umožňující kontrolu pohybu modelu pomocí myši a několik tlačítek, jejichž události vidíme na vývojovém diagramu.

Mezi událostmi tohoto módu jsou tedy zmíněná tlačítka a obsluhy obrazu z kamer senzoru MS Kinect:

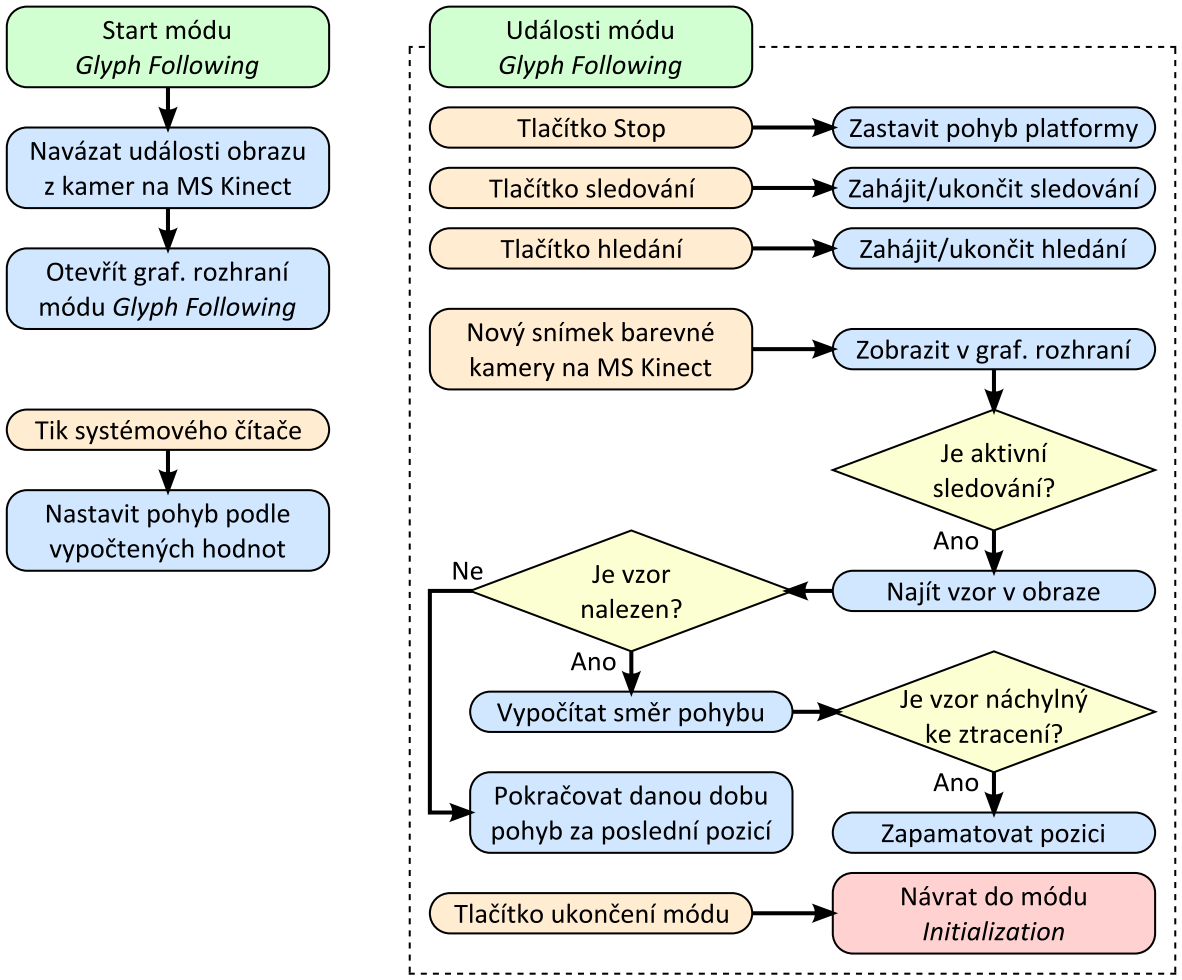
- příkaz stop zastaví pohyb mobilní platformy a nepovolí ho po dobu dalších 10 s.
- nastavení servomechanismů, LED a LCD otevře příslušné dialogové okno, odkud lze odeslat příkazy pro dané periférie. Tlačítkem *OK* se výběr potvrdí a prvek je zprávou po sběrnici CAN přepnut do manuálního režimu, kde vykoná požadovanou akci a dále nereaguje na automatické příkazy z aplikace (poslouchá pouze příkazy poslané uživatelem). Tlačítkem *Cancel* se přepne zpět do automatického režimu. V něm servomechanismus pravidelně rotuje tam a zpět, aby jeho infračervené senzory zkoumali půlkruh před sebou, LED na jednotkách senzorů indikují aktivitu sběrnice CAN (příchozí a odchozí zprávy) a displej zobrazuje stavové informace o chodu aplikace. Automatický režim je pro tyto prvky aktivní také v ostatních módech aplikace.
- přepnutí dálkoměrů povolí nebo zakáže jejich kontrolu nad možným pohybem platformy.
- události nového snímku z kamer senzoru MS Kinect předávají tato data na grafické rozhraní aplikace, pokud je obraz povolen během inicializace.
- ukončení módu přepne grafické rozhraní zpět do inicializačního panelu.

Při tiku systémového čítače se v rámci tohoto režimu provedou následující akce. Do proměnných pro otáčky pro levý a pravý pás se uloží hodnoty z prvku pro ovládání pohybu platformy pomocí myši, který má přednost před ovládáním pomocí zařízení USB joystick. Na grafické rozhraní se vykreslí znázornění senzorů do panelů a textových polí (viz. obr. D.2). Proměnné pro otáčky se aktualizují podle polohy ovládacích prvků na zařízení USB joystick. Levá a pravá ovládací páčka řídí každá příslušný pás (pozor na nastavení ovladače USB joystick, který musí být v analogovém režimu, jinak ovládací páčky nefungují – zadní přepínač v poloze *X* a tlačítko *Mode* stisknuté tak, aby zelená LED vedle něj byla zhasnuta). Na závěr se v čítači vykoná samotný pohyb odesláním zprávy jednotce motorů a při povolených dálkoměrech se vyhodnotí vzdálenost překážek v požadovaném směru. V případě, že jsou blíže než 1 m, maximální rychlost modelu se omezí na polovinu, aby platforma stihla zastavit před překážkou i v plné rychlosti, kdy

pohyb pokračuje déle kvůli odezvě jednotky motorů a setrvačnosti celého modelu. Pokud se nacházejí ve vzdálenosti do 50 cm, pohyb je zcela zastaven a zvukovým signálem pomocí reproduktoru je oznámena překážka ve směru pohybu.

4.4.3 Rozpoznávání obrazce a jeho sledování – *Glyph Following*

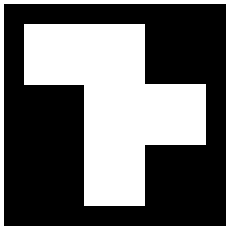
Následující mód podporuje sledování předem známého vzoru pomocí senzoru MS Kinect. V aplikaci má název *Glyph Following*. Pro rozpoznávání jsou využity knihovny zpracování obrazu od vydavatele AForge.NET. Teorie jejich použití je popsána v sekci 4.3.1. Po nalezení obrazce jsou z jeho pozice vypočteny směr a rychlost pohybu mobilní platformy a tyto informace jsou předány jednotce motorů, čímž je docílena jízda za obrazcem. V následujícím vývojovém diagramu (obr. 4.9) je vysvětlena funkce módu a ukázka z jeho grafického rozhraní je v příloze na obr. D.3.



Obr. 4.9: Vývojový diagram módu rozpoznávání obrazce a jeho sledování

Se startem módu se vytvoří událost pro zpracování obrazu z barevné kamery senzoru MS Kinect, která je využita ke sledování obrazce. Poté se otevře panel s grafickým rozhraním pro tento režim. Na diagramu jsou vidět události, které jsou zde vykonávané:

- příkazem stop se opět zastaví pohyb mobilní platformy a nepovolí po dobu dalších 10 s.
- tlačítko pro sledování obrazce zapne nebo vypne jízdu za obrazcem.
- tlačítko pro hledání obrazce je aktivní pouze pokud je povoleno sledování obrazce. Při stisku uvede mobilní platformu do stavu, kdy se otáčí kolem dokola po směru hodinových ručiček. V případě nalezení vzoru ho začne sledovat. Pokud se během jedné otočky kolem modelu obrazec nevyskytuje, je dokončena bez další akce.
- událost nového snímku z barevné kamery senzoru MS Kinect zobrazí její data na grafické rozhraní aplikace, pokud je obraz povolen během inicializace. Dále se pracuje s předem známým vzorem (požadovaný tvar pro tuto konkrétní aplikaci je na obr. 4.10), když je zapnuto jeho sledování. Nejprve se provede pokus o nalezení obrazce pomocí knihoven AForge.NET. Pokud je nalezen, knihovny vrátí jeho pozici v obraze a jeho velikost. Z těchto údajů se vypočítá rychlost a směr pohybu a tyto hodnoty jsou převedeny na požadované otáčky pro každý z pásů. Rychlost se zvětšuje s rostoucí vzdáleností obrazce od modelu a poloměr zatáčení klesá se vzdáleností vzoru od středu obrazu kamery. Pokud je příliš blízko, model couvá. V případě že je vzor náchylný ke ztracení (tzn. že se nachází příliš u kraje obrazu nebo příliš blízko), zapamatuje se tato poslední pozice. Ztratí-li se vzor a je-li zapamatována nějaká z pozic, model vykonává dále daný pohyb. Byl-li vzor u kraje, rotuje se půl otočky daným směrem a při jeho znovunalezení se sleduje dále. Pokud se nenalezne, pohyb je zastaven a servomechanismus se otočí tam a zpět, čímž se dá najevo definitivní ztráta obrazce. Byl-li příliš blízko, pokračuje se v couvání se stejným vyhodnocením (pokračování sledování nebo konec pohybu) jako u rotace. Jakmile se vzor znovu umístí do zorného pole platformy, pohyb je obnoven a celý algoritmus je opakován s každým novým snímkem.
- ukončením módu se přepne grafické rozhraní zpět do inicializačního panelu.

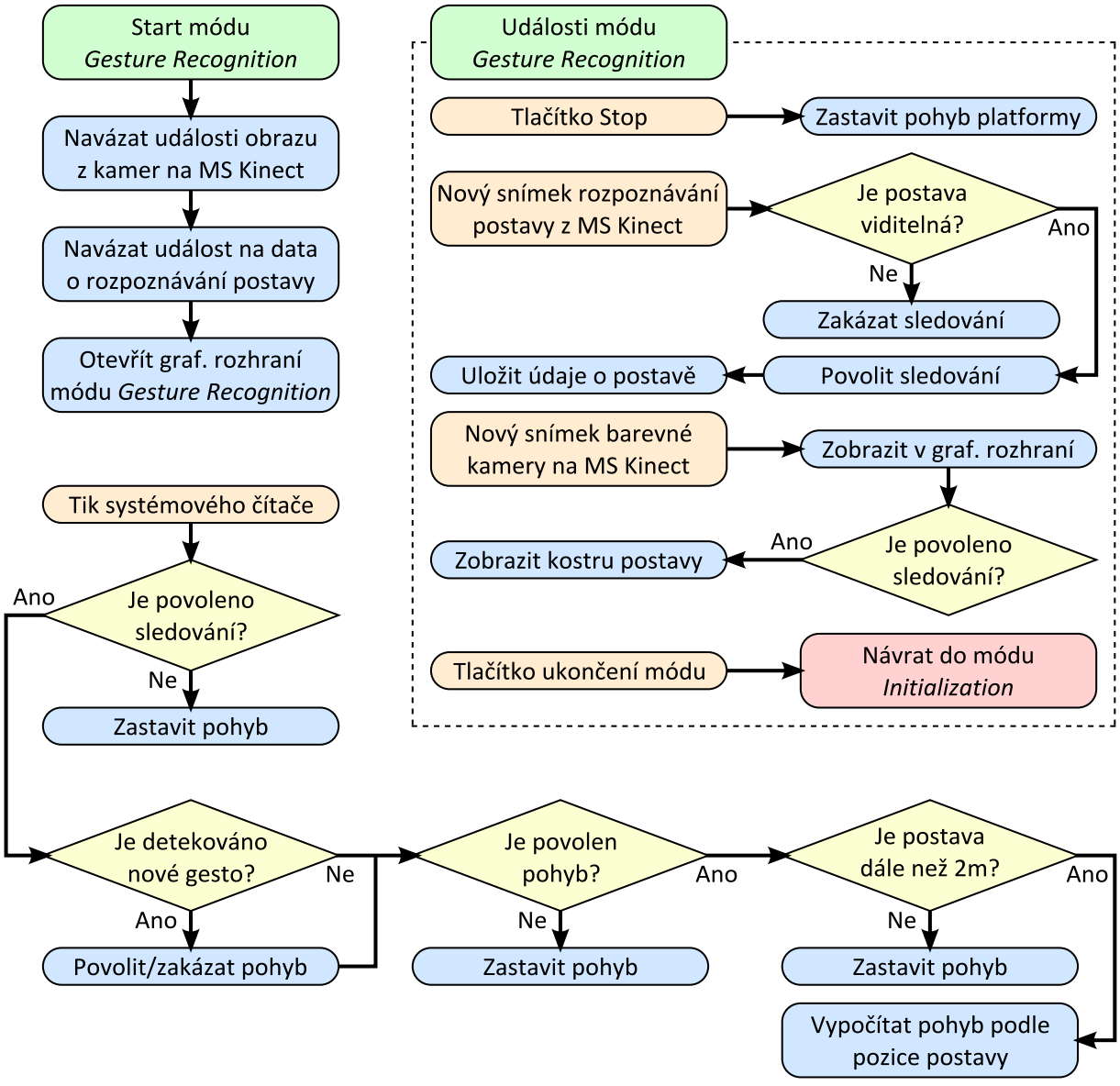


Obr. 4.10: Vzor pro sledování mobilní platformou

V události systémového čítače se v tomto módu provádí pouze odesílání požadovaných otáček pásů, které byly vypočteny při novém snímku z barevné kamery, jednotce motorů. Kompletní výpočet pohybu je rozsáhlý a k nahlédnutí je proto ve zdrojovém kódu na CD s komentáři u jednotlivých kroků.

4.4.4 Rozpoznávání lidské postavy – *Gesture Recognition*

V tomto módu je umožněno mobilní platformě rozpoznávat lidskou postavu a sledovat její pohyb. V aplikaci nese název *Gesture Recognition*. K rozpoznávání aplikace využívá knihovny od společnosti Microsoft, jejichž implementace je vysvětlena v sekci 4.3.2. Pokud senzor MS Kinect nalezne postavu, začne ji sledovat. Zvednutím pravé ruky začne platforma pohyb za postavou a stejným gestem ho také zastaví. Algoritmus je podrobněji vysvětlen na vývojovém diagramu z obr. 4.11. Grafické rozhraní panelu s tímto módem je k nahlédnutí v příloze na obr. D.4.

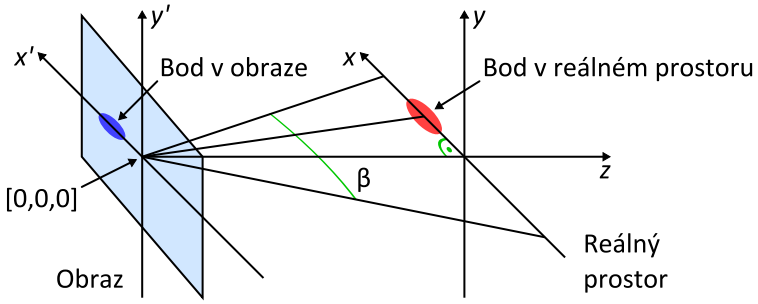


Obr. 4.11: Vývojový diagram módu rozpoznávání lidské postavy

Po startu módu se vytvoří událost pro zpracování obrazu z barevné kamery senzoru MS Kinect k zobrazování videa v grafickém rozhraní. Zároveň se naváže událost pro zpracování dat o rozpoznané postavě, jež posílá senzor MS Kinect do PC po jejich povolení

danou sekvencí příkazů (viz. 4.3.2). Následně se otevře panel s příslušným grafickým rozhraním a v diagramu jsou vidět události vykonávané v tomto režimu, jejichž popis pokračuje zde:

- příkazem stop se zastaví pohyb a nepovolí po dobu dalších 10 s.
- při přijetí nových dat o rozpoznané postavě se povolí nebo zakáže její další sledování podle toho, jestli je detekována, nebo ne. V kladném případě se tyto údaje uloží do proměnných.
- s událostí nového snímku z barevné kamery senzoru MS Kinect se zobrazí její data na grafické rozhraní aplikace, pokud je obraz povolen během inicializace. Dále je při povoleném sledování (tzn. postava je rozpoznána) vykreslena její kostra do obrazu z barevné kamery. Zde je nutné povést transformaci měřítka z reálného prostoru do obrazu kamery. Výpočet je totožný pro osu x i y a je patrný z obr. 4.12 a rovnice 4.2. Pro osu x vychází z nutnosti stejného poměru vzdálenosti bodu v reálném prostoru od středu (x) ku polovině rozsahu vidění v dané vzdálenosti od senzoru ($tg(\beta/2) \cdot z$) a vzdálenosti bodu v obraze od středu (x') ku polovině šířky obrazu ($X/2$). Úhel horizontálního rozsahu senzoru MS Kinect (β) je 57° a vertikálního 43°.



Obr. 4.12: Transformace měřítka z reálného prostoru do obrazu kamery

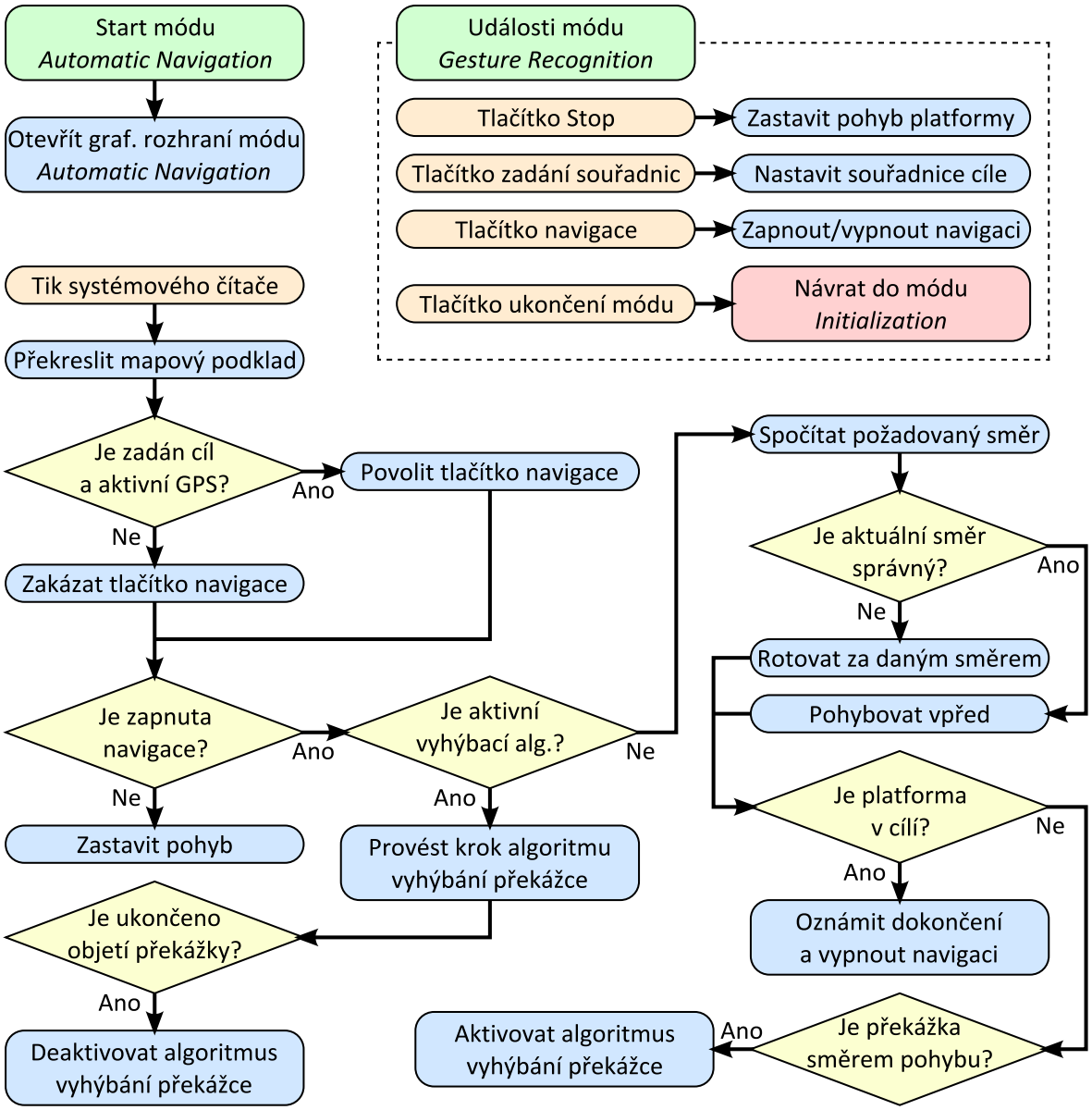
$$x' = \frac{x}{tg(\beta/2) \cdot z} \cdot X/2 \tag{4.2}$$

- ukončením módu se přepne grafické rozhraní zpět do panelu inicializace.

V události systémového čítače v tomto módu probíhá výpočet pohybu za lidskou postavou. Pokud je povoleno sledování, zkontroluje se gesto ovládající pohyb. Jinak se pohyb zruší, což zastaví platformu v případě, že jela za postavou a ta byla poté ztracena. Gestem se zde rozumí zvednutí pravé ruky (porovnávají se souřadnice pravé ruky a hlavy a gesto je zaznamenáno, když je ruka o 20 cm výše než hlava). Při jeho detekci je povolen pohyb, pokud byl do té doby zakázán, a naopak. Dále se v případě povoleného pohybu počítá směr a rychlost modelu, jež jsou nastaveny zprávou po sběrnici CAN do jednotky motorů. Základní podmínkou je vzdálenost postavy od platformy větší než dva metry. Po jejím splnění se rychlost zvyšuje se vzdáleností a otáčení s rostoucím úhlem od středu obrazu. Pokud je postava blíže než dva metry, pohyb je opět zastaven.

4.4.5 Navigace pomocí GPS a kompasu – *Automatic Navigation*

Poslední mód řízení představuje navigaci v terénu pomocí GPS, kompasu a dalších senzorů. V aplikaci se jmenuje *Automatic Navigation*. Umožňuje pohyb směrem k zadanému cíli, jenž se specifikuje pomocí GPS souřadnic. Platforma se zároveň vyhýbá překážkám, k čemuž využívá algoritmy popsané v sekci 4.3.3. Algoritmus je dále vysvětlen na vývojovém diagramu na obr. 4.13. Grafické rozhraní je poté možno vidět v příloze (viz. obr. D.5).



Obr. 4.13: Vývojový diagram módu navigace pomocí GPS a kompasu

Grafické rozhraní obsahuje mapový podklad oblasti, na kterém je zobrazena aktuální poloha. Kolečkem myši lze mapu přibližovat a táhnutím přesouvat. Mapa funguje offline a zobrazuje předem vytvořený soubor, jenž lze exportovat ze stránek openstreetmap.org. Mapy jsou pod licencí ODbL umožňující nekomerční využití (více na stránkách v sekci

Copyright). Pro jejich zobrazení je využit framework s názvem CartoType (viz. [31], licence opět umožňuje nekomerční využití – více na stránkách v sekci Downloads).

Na vývojovém diagramu je popsán navigační algoritmus. Startem módu se aplikace přepne do příslušného panelu grafického rozhraní, kde se obsluhují tyto události:

- příkaz stop zastaví pohyb platformy a nepovolí ho po dobu dalších 10 s.
- zadáním souřadnic se nastaví v dialogu GPS souřadnice cíle pro navigační algoritmus. Lze je také zadat stisknutím levého tlačítka myši na zobrazené mapě.
- tlačítko navigace zapíná nebo vypíná algoritmus vykonávající pohyb platformy k cíli. Výpočet probíhá v události systémového čítače, která nastane každých 100 ms.
- ukončení módu přepne grafické rozhraní zpět do panelu inicializace.

V tiku systémového čítače poté probíhá samotný navigační algoritmus. Nejprve se zkontroluje, jestli je funkční modul GPS (posílá data do PC) a zadán cíl. Pokud ano, povolí se tlačítko, kterým lze zapnout výpočet pohybu. V opačném případě se zakáže (nereaguje na stisknutí myši). Dále se provede výpočet a odeslání informací o požadovaném pohybu jednotce motorů, jestliže je algoritmus zapnut zmíněným tlačítkem.

Navigační algoritmus se následně dělí na dvě větve (podle podmínky: „Je aktivní vyhýbací algoritmus?“). První je jízda k cíli podle souřadnic GPS s kompasem v případě, že není algoritmus vyhýbání aktivován. Nejprve se vypočte požadovaný směr pohybu (úhel ve stupních) z aktuální pozice mobilní platformy a pozice cíle. Porovná se s momentálním natočením platformy (údaj z kompasu – opět úhel ve stupních). Pokud je v toleranci $\pm$ 20°, odešle se jednotce motorů požadavek na pohyb vpřed. V jiném případě se platforma otáčí kolem své osy za požadovaným směrem. Dále se zkontroluje, jestli již cíl nebyl dosažen, což se v kladném vyhodnocení oznámí zvukovým signálem a algoritmus je ukončen. Jestliže platforma v cíli není, zkontrolují se překážky ve směru pohybu nebo rotace pomocí senzorů měřících vzdálenosti a při objevení překážky se aktivuje algoritmus vyhýbání. Celá sekvence pokračuje znovu od začátku s dalším tikem čítače.

Do druhé větve se dostaneme kladným vyhodnocením podmínky (vyhýbací algoritmus je aktivní). V ní se provede daný krok objíždění překážky. Celý algoritmus je postaven tak, aby co nejlépe fungoval na základě teorie algoritmů popsané v sekci 4.3.3. Pokud je objíždění dokončeno (senzory dále překážku nedetekují), deaktivuje se algoritmus vyhýbání a sekvence opět pokračuje od začátku s dalším tikem čítače. V případě dokončení algoritmu, kdy se platforma nachází ve stejné pozici jako při detekci překážky, se konstatuje nemožnost nalezení cesty (zvukový signál) a navigace je ukončena.

Venkovní terén je často složitý a nepřehledný a dosavadní algoritmy mohou leckdy selhávat. Nabízí se zde velký prostor pro budoucí vylepšování (např. přidávání senzorů, přesnější určení polohy, rozpoznávání okolí pomocí MS Kinect nebo nový princip orientace – exaktní, mřížkové a pravděpodobnostní plánování). Více informací viz. [30], kde jsou také odkazy na další odbornou literaturu zabývající se tímto problémem.

5

Závěr

V diplomové práci jsem se zabýval projektem mobilní platformy na Katedře aplikované elektroniky a telekomunikací. Úkolem bylo uvažovat současný hardware vytvořený během minulých projektů a vylepšit možnosti ovládání implementací různých módů s využitím ovladače MS Kinect a dalších senzorů. Na cestě k vyřešení tohoto zadání jsem se postupně soustředil na tři hlavní problémy, jimiž jsou kompletace a podání přehledu o hardwaru, úpravy a doplnění softwaru do modulů obsluhujících periférie a tvorba samotné řídicí aplikace v PC, jež umožní propracovanější možnosti ovládání a interakci s okolím.

Nejprve bylo v úvodní kapitole nutné vyřešit hardwarovou stránku modelu. Model byl dáván dohromady v rámci několika minulých projektů, a proto jsem se rozhodl věnovat úvodní kapitolu jeho přehledovému popisu, aby pro pochopení konceptu návrhu stačila tato práce. K detailnějšímu studování bloků je samozřejmě možné nahlédnout do dřívějších prací nebo katalogových listů, jejichž odkazy jsou uvedeny u příslušných sekcí a následně v seznamu literatury. Zároveň jsou v kapitole zahrnuty části, jež byly doplněny v rámci tohoto projektu pro vytvoření základu složitější řídicí aplikaci (osazení chybějících senzorů, přidání nových bloků – např. moduly GPS, kompasu a reproduktoru). Hardware naskýtá možnosti pro další rozšíření do budoucích prací a zde bych nastínil některé z nich. Platformu lze například doplnit o modul pro nabíjení palubního akumulátoru a pro podávání informace o jeho stavu do PC (zbývající kapacita), jež by usnadnil práci s akumulátorem (dobití ze sítě bez vyjímání a připojování na externí zdroj, informace o nutnosti dobití). Dále je vhodné nahradit stávající modul kompasu jiným, který je kompenzován proti náklonu a nebude tak citlivý na větší nerovnosti terénu. Zajímavou modifikací může být vybavení senzoru MS Kinect pohyblivým podstavcem (pro natáčení a náklon), který umožní lepší „rozhled“. V neposlední řadě lze z důvodu modulárního řešení platformy (veškeré jednotky jsou připojeny na sběrnice USB nebo CAN) vybavovat model jakýmikoli dalšími senzory nebo akčními členy.

Následně jsem se věnoval modifikacím softwaru jednotek na platformě, které popisuje další kapitola, z důvodu splnění požadavků řídicí aplikace na data ze senzorů a obsluhu motorů. Jednotka motorů vyžadovala drobné změny například v definici konfiguračních zpráv a zavedení funkce „watchdog“. Naopak v jednotkách senzorů fungoval pouze zlomek

senzorů a jejich firmware bylo nutné doplnit o celé části pro stávající i nové senzory. Přidal jsem např. funkce pro sběrnice I^2C a SPI, na něž jsou dále připojeny kompas, ultrazvukové dálkoměry a akcelerometr s gyroskopem. Během této části jsem také vytvořil tabulky protokolu sběrnice CAN (částečně převzaté z minulých projektů a dále doplněné), který přehledně vysvětluje použité zprávy, jejich ID a data. Důvodem je shrnutí pro usnadnění dalších modifikací části aplikace v PC ke komunikaci po sběrnici CAN a zamezení použití stejného ID při přidávání nových modulů.

Posledním a hlavním bodem celé práce byla po vyřešení hardwaru platformy a softwaru jednotek tvorba řídicí aplikace, která je probrána v závěrečné kapitole. Vychází ze zadání požadujícího doplnit ovládání a navrhnout možnosti interakce prostřednictvím ovladače MS Kinect a využít další dostupné senzory. Aplikaci jsem tvořil v jazyce C#. Rozdělena je opět modulárně a to na dvě části. V první jsou obsaženy knihovny pro obsluhu veškerých periférií, což usnadní vytváření řídicích algoritmů. Lze je také kdykoli využít při doplňování algoritmů nových. Ve druhé části vznikla komplexní řídicí aplikace s grafickým rozhraním umožňující inicializaci periférií s chybovým výpisem pro rychlou diagnostiku nefunkčních částí a následně demonstrující schopnosti mobilní platformy ve čtyřech různých módech. Umožněno je ovládání pohybu pomocí zařízení USB joystick, kdy platforma zastavuje před překážkami detekovanými senzory dálkoměrů. Na grafickém rozhraní je také zobrazena vizualizace dat ze senzorů a lze manuálně ovládat jednotlivé periférie, což slouží pro ukázku funkcí a pro odhalení nefunkčních prvků. Dále je navrženo ovládání prostřednictvím senzoru MS Kinect a to ve dvou módech. Aplikace podporuje rozpoznávání definovaného vzoru a jeho sledování a rozpoznávání lidské postavy s jízdou za jejím pohybem s využitím dostupných knihoven. Posledním módem je automatická navigace, kterou se rozumí jízda k cíli zadanému souřadnicemi GPS. Využívají se senzory GPS a kompasu pro určení směru a dálkoměry pro detekci překážek. Implementovány jsou jednoduché algoritmy pro jejich objíždění. V tomto směru se opět nabízejí nové možnosti rozšíření. Lze vytvářet knihovny pro nově dodané periférie nebo vylepšovat řídicí módy. Největší prostor je dle mého názoru v upravování navigačních algoritmů. Vzhledem ke složitosti venkovního terénu mohou dosavadní jednoduché principy selhávat a je vždy možné je zdokonalovat o nové metody. Může se také přidat plánování na mapě se sledováním cest pomocí kamer senzoru MS Kinect a podobně.

Ve shrnutí jsem tedy v rámci práce postupně vyřešil problém hardwaru tak, aby splňoval požadavky řídicí aplikace, a nastínil další možné směry rozvoje po této stránce. Do stávajících senzorových jednotek jsem dodal software obsluhující dostupné periférie a ve spolupráci s kolegou Lukášem Pušmanem vylepšil jednotku motorů. Dále vznikla výše popsaná řídicí aplikace, která je schopna předvést model v různých činnostech, a je postavena tak, že ji lze dále rozvíjet (např. novými módy s využitím knihoven), čímž podporuje další projekty. Zachovává se tak princip mobilní platformy, která je především projektem pro realizaci studentských prací a pro prezentaci fakulty na různých akcích pro veřejnost.

Literatura

[1] PUŠMAN, Lukáš. *Řídicí systém systému mobilní platformy.* Plzeň, 2011. Diplomová práce. ZČU.

[2] ŠTĚTKA, Petr. *Senzorová výbava mobilní platformy.* Plzeň, 2011. Diplomová práce. ZČU.

[3] MICROSOFT. *Kinect for Windows Sensor* [online]. Copyright ©2014, Microsoft. [cit. 14.12.2014]. Dostupné z:
`msdn.microsoft.com/en-us/library/hh855355.aspx`

[4] LOGITECH INTERNATIONAL S.A. *Wireless Gamepad F710* [online]. Copyright ©2015, Logitech. [cit. 8.2.2015]. Dostupné z:
`support.logitech.com/en_us/product/wireless-gamepad-f710`

[5] NAVILOCK. *Navilock NL-303P serial MD6 GPS Receiver* [online]. Copyright ©2015, Navilock. [cit. 18.3.2015]. Dostupné z:
`www.navilock.de/produkt/61423/pdf.html?sprache=en`

[6] KOSTURIK, Kamil. *Controller Area Network.* Plzeň, 2014. Přednáška KAE/RIS. ZČU.

[7] KOSTURIK, Kamil. *Universal Serial Bus.* Plzeň, 2014. Přednáška KAE/RIS. ZČU.

[8] SHARP CORPORATION. *GP2Y0A21YK0F* [online]. Copyright ©2006, SHARP Corporation. Poslední změna prosinec 2006 [cit. 9.2.2015]. Dostupné z:
`www.sharpsma.com/webfm_send/1489`

[9] SHARP CORPORATION. *GP2Y0A02YK* [online]. Copyright ©2006, SHARP Corporation. Poslední změna prosinec 2006 [cit. 9.2.2015]. Dostupné z:
`www.sharpsme.com/download/GP2Y0A02YK-DATA-SHEETPDF`

[10] SHARP CORPORATION. *GP2Y0A710K0F* [online]. Copyright ©2006, SHARP Corporation. Poslední změna prosinec 2006 [cit. 9.2.2015]. Dostupné z:
`www.sharpsme.com/download/gp2y0a710k-epdf`

[11] SOLARBOTICS LTD. *Devantech SRF08 High Performance Ultrasonic Ranger* [online]. Copyright ©2015, Solarbotics Ltd. [cit. 14.2.2015]. Dostupné z: `solarbotics.com/product/40320/`

[12] ROBOT-ELECTRONICS. *SRF08 Ultra sonic range finder* [online]. Poslední změna duben 2012 [cit. 9.2.2015]. Dostupné z: `www.robot-electronics.co.uk/htm/srf08tech.html`

[13] ROBOT-ELECTRONICS. *CMPS03 – Compass Module* [online]. Poslední změna leden 2015 [cit. 10.2.2015]. Dostupné z: `www.robot-electronics.co.uk/htm/cmps3tech.htm`

[14] DIGI-KEY CORPORATION. *ADIS16300AMLZ-ND* [online]. Copyright ©2015, Digi-Key Corporation. [cit. 14.2.2015]. Dostupné z: `www.digikey.com/product-detail/en/ADIS16300AMLZ/ADIS16300AMLZ-ND/1979408`

[15] ANALOG DEVICES. *ADIS16300* [online]. Copyright ©2009, Analog Devices, Inc. Poslední změna 2009 [cit. 10.2.2015]. Dostupné z: `www.analog.com/static/imported-files/data_sheets/ADIS16300.pdf`

[16] ARROWTECH. *ATM1602B* [online]. Copyright ©2009, Xiamen Arrow-tech Electronic Co., Ltd. [cit. 14.2.2015]. Dostupné z: `www.arrowtech.cn/productdetail.asp?id=8&sort=1`

[17] ARROWTECH. *ATM1602B* [online]. Copyright ©2009, Xiamen Arrow-tech Electronic Co., Ltd. Poslední změna 2006 [cit. 10.2.2015]. Dostupné z: `www.arrowtech.cn/upload/ATM1602B.pdf`

[18] FREESCALE SEMICONDUCTOR, INC. *LCD Driver for the HC08/HCS08 Family* [online]. Copyright ©2006, Freescale Semiconductor, Inc. Poslední změna duben 2005 [cit. 10.2.2015]. Dostupné z: `cache.freescale.com/files/microcontrollers/doc/app_note/AN2940.pdf`

[19] FREESCALE SEMICONDUCTOR, INC. *MC9S08DZ60* [online]. Copyright ©2008, Freescale Semiconductor, Inc. Poslední změna červen 2008 [cit. 28.2.2015]. Dostupné z: `cache.freescale.com/files/microcontrollers/doc/data_sheet/MC9S08DZ60.pdf`

[20] FREESCALE SEMICONDUCTOR, INC. *HCS08QRUG* [online]. Copyright ©2006, Freescale Semiconductor, Inc. Poslední změna únor 2006 [cit. 28.2.2015]. Dostupné z: `cache.freescale.com/files/microcontrollers/doc/user_guide/HCS08QRUG.pdf`

[21] FREESCALE SEMICONDUCTOR, INC. *HCS08QRUGSW.zip* [online]. Copyright ©2006, Freescale Semiconductor, Inc. [cit. 28.2.2015]. Dostupné z: `cache.freescale.com/files/microcontrollers/doc/user_guide/HCS08QRUGSW.zip`

[22] USBDM. *USBDM Debugger interface for Freescale RS08, HCS08, HCS12, Coldfire and ARM-Kinetis Devices* [online]. 2015 [cit. 28.2.2015]. Dostupné z:
`usbdm.sourceforge.net/index.html`

[23] Pavel Tišnovský. *Externí sériové sběrnice SPI a I^2C* [online]. Copyright ©2015, Internet Info, s.r.o. Poslední změna prosinec 2008 [cit. 4.3.2015]. Dostupné z:
`www.root.cz/clanky/externi-seriove-sbernice-spi-a-i2c/`

[24] MICROSOFT. *Kinect for Windows SDK v1.8* [online]. Copyright ©2015, Microsoft. [cit. 16.3.2015]. Dostupné z:
`www.microsoft.com/en-us/download/details.aspx?id=40278`

[25] AForge.NET. *Joystick Class* [online]. Copyright ©2013, AForge.NET. [cit. 16.3.2015]. Dostupné z:
`aforgenet.com/framework/docs/html/4c221262-68d8-c613-ced0-e64b380646c6`

[26] AForge.NET. *AForge.NET Framework*. [online]. Copyright ©2012, AForge.NET. [cit. 30.3.2015]. Dostupné z:
`www.aforgenet.com/framework/`

[27] Ondřej Karas. *Práce s GPS moduly* [online]. Copyright ©2015, Programujte.com. Poslední změna únor 2008 [cit. 16.3.2015]. Dostupné z:
`programujte.com/clanek/2008021500-prace-s-gps-moduly/`

[28] Andrew Kirillov. *Glyphs' recognition* [online]. Copyright ©2012, AForge.NET. Poslední změna listopad 2010 [cit. 30.3.2015]. Dostupné z:
`www.aforgenet.com/articles/glyph_recognition/`

[29] Evangelos Georgiou. *Tutorial: WPF C#Kinect SDK AForge GRATF Glyph Tracking* [online]. Copyright ©2015, My Mobile Robots. Poslední změna květen 2012 [cit. 30.3.2015]. Dostupné z:
`mymobilerobots.com/myblog/tag/gratf/`

[30] Martin Dlouhý. *ABC mobilní robotiky* [online]. Poslední změna září 2005 [cit. 2.4.2015]. Dostupné z:
`robotika.cz/guide/cs`

[31] CartoType. *Maps, routes and navigation using off-line data* [online]. Copyright ©2015, CartoType Ltd. [cit. 8.4.2015]. Dostupné z:
`www.cartotype.com/`

[32] Martin Malík. *HWiNFO32* [online]. Copyright ©2015, REALiX™ [cit. 28.4.2015]. Dostupné z:
`www.hwinfo.com`

Příloha A

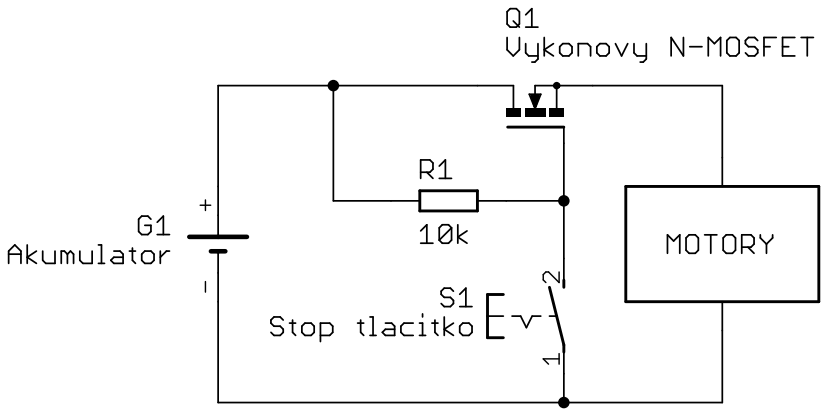
Specifikace počítače mobilní platformy

Computer Name	TANK-PC
Computer Brand Name	MSI MS-7677
Operating System	Microsoft Windows 7 Professional Build 7600
Processor Name	Intel Pentium G630T
Number Of Processors Cores	2
Original Processor Frequency	2300.0 MHz
CPU Code Name	Sandy Bridge-DT
CPU Platform	Socket H2 (LGA 1155)
CPU LFM (Minimum)	1600.0 MHz = 16 x 100.0 MHz
CPU HFM (Maximum)	2300.0 MHz = 23 x 100.0 MHz
L1 Cache	Instruction: 2 x 32 Kbytes, Data: 2 x 32 Kbytes
L2 Cache	Integrated: 2 x 256 KBytes
L3 Cache	3 MBytes
Motherboard Model	MSI H61I-E35 (MS-7677)
Motherboard Chipset	Intel H61 (Cougar Point)
Hard Drive	SATA 3 Gb/s, ADATA SP800 [31.3 GB]
Total Memory Size	4 GBytes
Maximum Memory Clock	533.3 MHz
Memory Devices	2
Device Size	2048 Mbytes
Device Type	DDR3 SDRAM
Device Type Detail	Synchronous
Device Mode	Dual Channel
Integrated Graphics Controller	Intel Sandy Bridge-DT GT1 – Integrated Graphics Controller
GPU Type	Intel HD Graphics 2000
GPU Memory	1588212 KB
High Definition Audio Controller	Intel Cougar Point PCH – High Definition Audio Controller [B3]
Ethernet Adapter	RealTek Semiconductor RTL8168/8111 PCI-E Gigabit Ethernet
Maximum Link Speed	2.5 Gb/s

Obr. A.1: Specifikace počítače – vytvořené podle programu s názvem HWiNFO32 (ke stažení pod odkazem číslo [32] v seznamu literatury)

Příloha B

Schéma připojení stop tlačítka



Obr. B.1: Schéma připojení stop tlačítka

Příloha C

Protokol komunikace po sběrnici CAN

Popisu protokolu sběrnice CAN se věnuje sekce 3.1, kde jsou podrobněji vysvětleny významy polí tabulek z obrázků v této příloze.

Jednotka	Význam	ID
A	Řídicí jednotka motorů	-
B	Přední jednotka senzorů	0x0B
C	Střední jednotka senzorů	0x0C
D	Zadní jednotka senzorů	0x0D

Definice sloupce – Směr	Význam
0	Zpráva do počítače (příjem)
1	Zpráva z počítače (vysílání)

Definice sloupce – R/W	Význam
0	Zpráva čte z cílové jednotky
1	Zpráva zapisuje do cílové jednotky

Zpráva	Význam	Směr	Jednotka	ID
TRACKS_INFO_CURRENT	Aktuální proud motorů	0	A	0x064
TRACKS_INFO_ERROR	Informace o chybě jednotky	0	A	0x00A
TRACKS_INFO_PWM	Aktuální otáčky motorů	0	A	0x020
TRACKS_INFO_TEMPERATURE	Aktuální teploty	0	A	0x065
TRACKS_SET_CONFIG	Konfigurace jednotky	1	A	0x01E
TRACKS_SET_PWM	Nastavení otáček motorů	1	A	0x01F

Obr. C.1: Protokol zpráv po sběrnici CAN (tabulky shora: ID jednotek, definice sloupce *Směr*, definice sloupce *R/W*, přehled zpráv jednotce motorů)

Zpráva	Význam	Směr	Jednotka	ID
ACM_INFO_BURST1	Periodická data z acm+gyro 1	0	C	0x301
ACM_INFO_BURST2	Periodická data z acm+gyro 2	0	C	0x302
ACM_INFO_REGISTERS	Data z registrů acm+gyro	0	C	0x300
ACM_SET_CONFIG	Nastavení acm+gyro	1	C	0x329
ACM_SET_REGISTERS	Data do registrů acm+gyro	1	C	0x320
COMPASS_INFO_BURST	Periodická data z kompasu	0	C	0x307
COMPASS_SET_CONFIG	Nastavení kompasu	1	C	0x32C
DISPLAY_SET_DATA	Data k zobrazení na displej	1	D	0x364 – 0x367
IIC_INFO_REGISTERS	Data z I^2C registru	0	B	0x2C5
		0	C	0x305
		0	D	0x345
IIC_SET_REGISTERS	Data do I^2C registru	1	B	0x2E2
		1	C	0x322
		1	D	0x362
INFRA_INFO_BURST	Periodická data z infračerveného senzoru	0	B	0x2C3
		0	D	0x343
INFRA_INFO_REQUEST	Data z infračerveného senzoru jako odpověď na požadavek	0	B	0x2C4
		0	D	0x344
INFRA_REQUEST_DATA	Požadavek na data z infračerveného senzoru	1	B	0x2E1
		1	D	0x361
INFRA_SET_CONFIG	Nastavení infračerveného senzoru	1	B	0x2EA
		1	D	0x36A
LED_SET_CONFIG	Nastavení LED	1	B	0x2ED
		1	C	0x32D
		1	D	0x36D
LED_SET_DATA	Data k zobrazení na LED	1	B	0x2E8
		1	C	0x328
		1	D	0x368
SENSORS_SET_MODULE_ID	Změna ID jednotky	1	B	0x2EE
		1	C	0x32E
		1	D	0x36E
SERVO_INFO_POSITION	Data o pozici serva po požadavku	0	B	0x2C8
SERVO_REQUEST_POSITION	Požadavek na data o pozici serva	1	B	0x2EF
SERVO_SET_POSITION	Nastavení pozice serva	1	B	0x2E3
ULTRA_INFO_BURST	Periodická data z ultrazvukového senzoru	0	B	0x2C6
		0	D	0x346
ULTRA_SET_CONFIG	Nastavení ultrazvukového senzoru	1	B	0x2EB
		1	D	0x36B

Obr. C.2: Protokol zpráv po sběrnici CAN (tabulka přehledu zpráv pro jednotky senzorů)

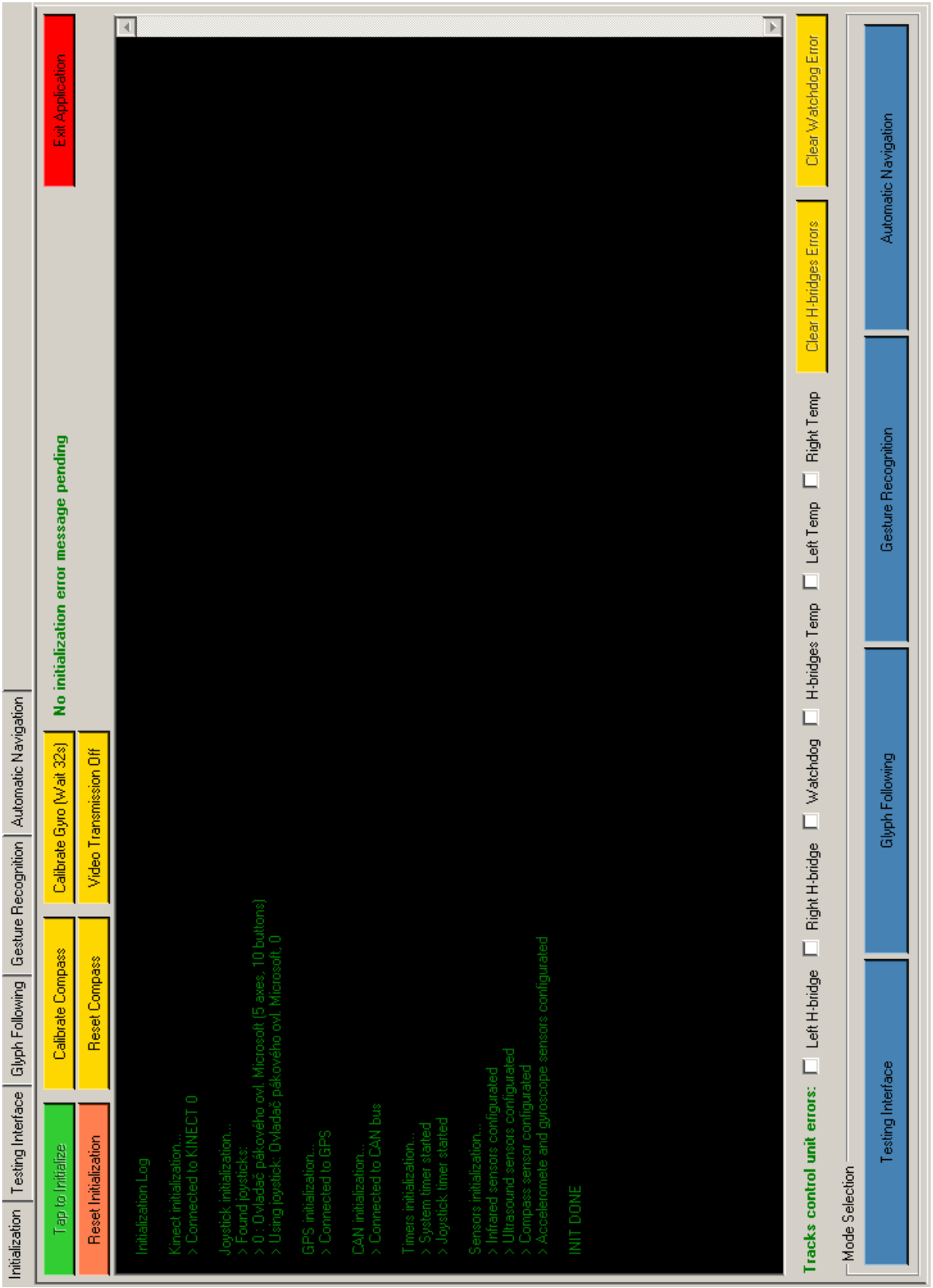
Zpráva	R/W	ID	Data							
TRACKS_INFO_CURRENT	0	-	Proud motor_1	Proud motor_2						
TRACKS_INFO_ERROR	0	-	Kód chyby							
TRACKS_INFO_PWM	0	-	Otáčky motor_1	Otáčky motor_2						
TRACKS_INFO_TEMPERATURE	0	-	Teplota motor_1	Teplota motor_2	Teplota H-můstku					
TRACKS_SET_CONFIG	1	-	Nastavení							
TRACKS_SET_PWM	1	-	Otáčky motor_1	Otáčky motor_2						

Zpráva	R/W	ID	Data							
ACM_INFO_BURST1	0	0x01	Elektrické napětí	Gyroskop	Akcelerometr_X					
ACM_INFO_BURST2	0	0x02	Akcelerometr_Z	Teplota	Akcelerometr_Y	Akcelerometr_X				
ACM_INFO_REGISTERS	0	0x00	Adresa	Obsah registru	Náklon do stran	Náklon vpřed/vzad				
ACM_SET_CONFIG	1	0x09	T_měření	T_posílání						
ACM_SET_REGISTERS	1	0x00	Adresa	Čtení/zápis	Obsah registru					
COMPASS_INFO_BURST	0	0x07	Hodnota							
COMPASS_SET_CONFIG	1	0x0C	T_měření	T_posílání	Adresa					
DISPLAY_SET_DATA	1	0x04 – 0x07	Znak_1	Znak_2	Znak_3	Znak_4	Znak_5	Znak_6	Znak_7	Znak_8
IIC_INFO_REGISTERS	0	0x05	Adresa		Obsah reg.					
IIC_SET_REGISTERS	1	0x02	Adresa		Čtení/zápis	Obsah reg.				
INFRA_INFO_BURST	0	0x03	Kontrola	Kanál_1	Kanál_2	Kanál_3	Kanál_4	Kanál_5	Kanál_6	Kanál_7
INFRA_INFO_REQUEST	0	0x04	Kontrola	Kanál_1	Kanál_2	Kanál_3	Kanál_4	Kanál_5	Kanál_6	Kanál_7
INFRA_REQUEST_DATA	1	0x01								
INFRA_SET_CONFIG	1	0x0A	T_měření	T_posílání	Počet					
LED_SET_CONFIG	1	0x0D	Režim							
LED_SET_DATA	1	0x08	Hodnota							
SENSORS_SET_MODULE_ID	1	0x0E	Nové ID	Kontrola						
SERVO_INFO_POSITION	0	0x08	Servo_1	Servo_2	Servo_3					
SERVO_REQUEST_POSITION	1	0x0F								
SERVO_SET_POSITION	1	0x03	Servo_1	Servo_2	Servo_3	Kanály				
ULTRA_INFO_BURST	0	0x06	Senzor_1		Senzor_2					
ULTRA_SET_CONFIG	1	0x0B	T_měření	T_posílání	Adresa_1	Adresa_2				

Obr. C.3: Protokol zpráv po sběrnici CAN (tabulka detailního popisu zpráv)

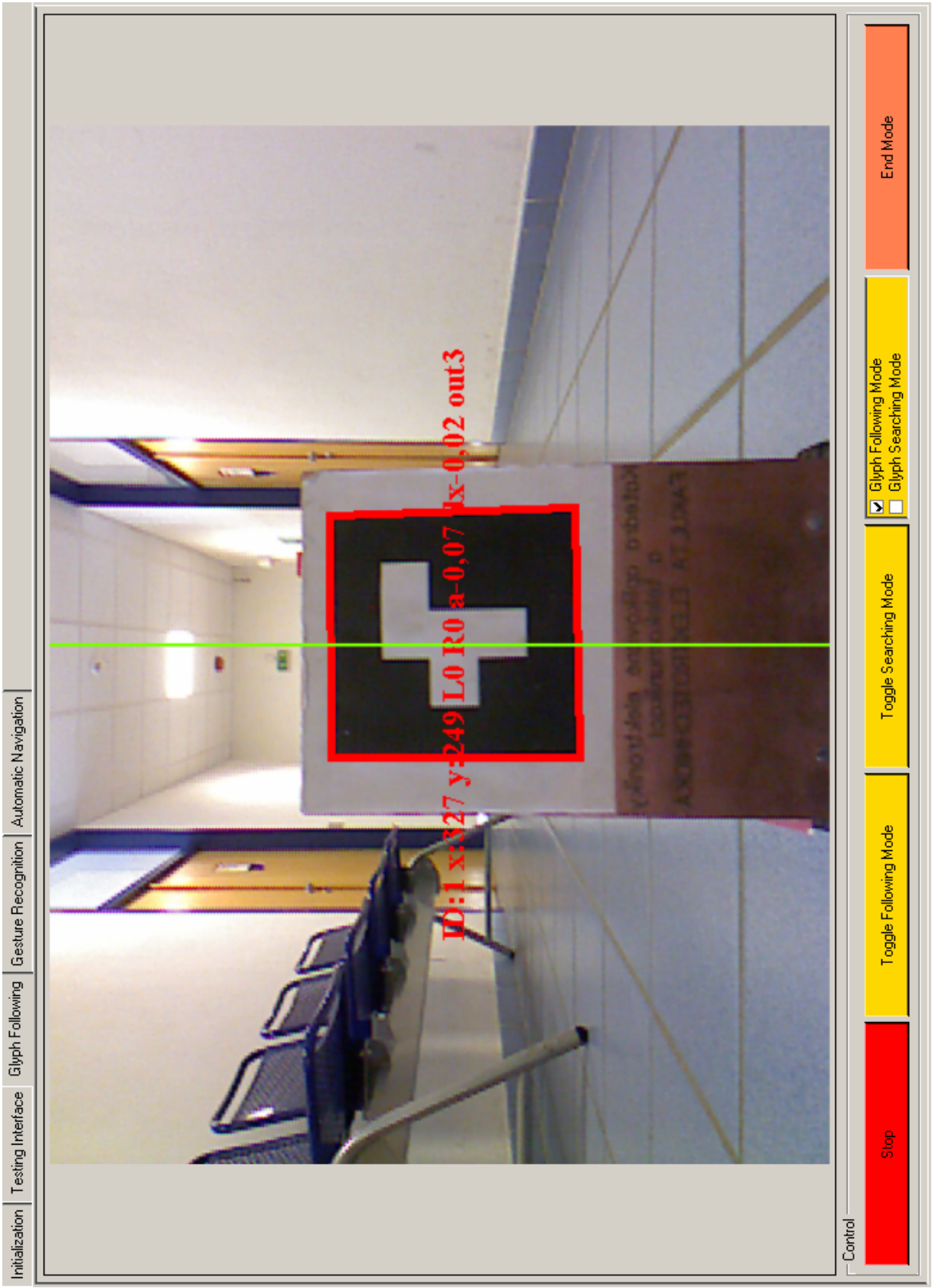
Příloha D

Ukázky z módů řídicí aplikace

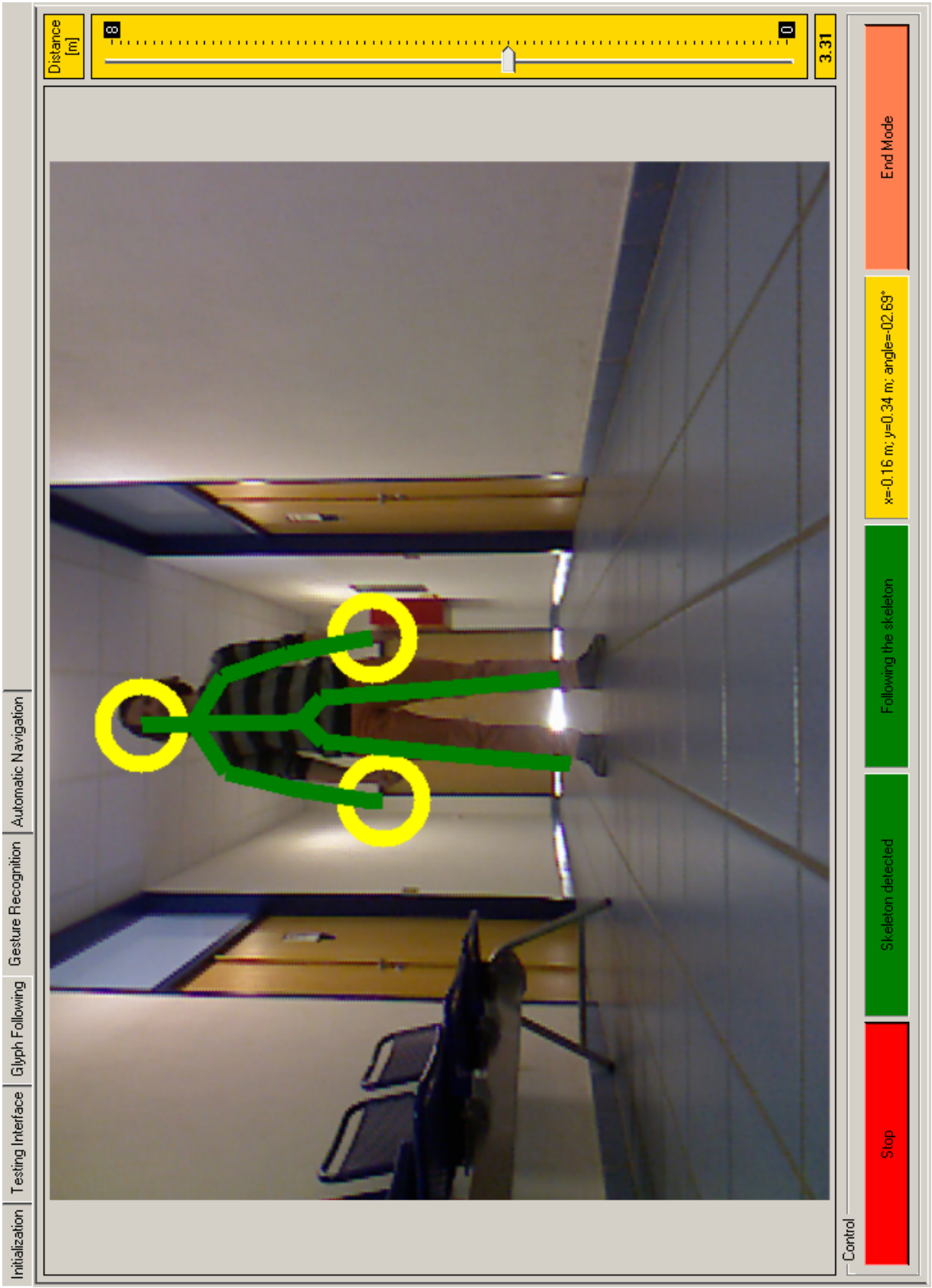


Obr. D.1: Mód inicializace mobilní platformy

Obr. D.2: Mód řízení pomocí ovladače joystick



Obr. D.3: Mód rozpoznávání obrazce a jeho sledování



Obr. D.4: Mód rozpoznávání lidské postavy

Obr. D.5: Mód navigace pomocí GPS a kompasu

Příloha E

Popis tlačítek ovladače joystick

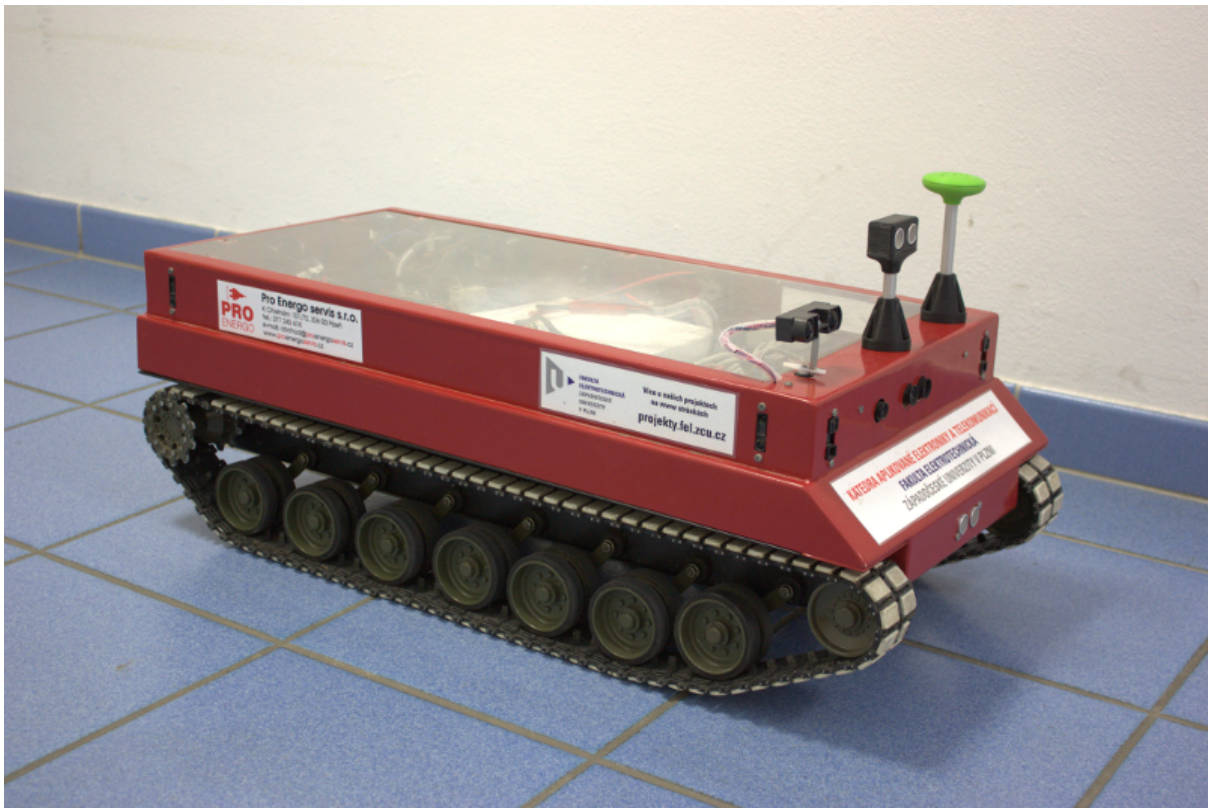
Zde je uveden význam tlačítek na ovladači joystick pro obsluhu řídicí aplikace. Pro správnou funkci musí být joystick přepnut v analogovém módu (zadní přepínač v poloze *X* a tlačítko *Mode* stisknuté tak, aby zelená LED vedle něj byla zhasnuta).

Ovládací prvek	Aktivní v módu	Význam
Tlačítko Start	*Initialization*	Sekvence reset a inicializace
Tlačítko Back	Jiný než *Initialization*	Návrat do módu *Initialization*
Tlačítko Y	*Initialization*	Přechod do módu *Testing Interface*
Tlačítko B	*Initialization*	Přechod do módu *Glyph Following*
Tlačítko A	*Initialization*	Přechod do módu *Gesture Recognition*
Tlačítko X	*Initialization*	Přechod do módu *Automatic Navigation*
Horní levá ovládací páčka	*Testing Interface*	Ovládání levého pásu (vpřed/vzad)
Horní pravá ovládací páčka	*Testing Interface*	Ovládání pravého pásu (vpřed/vzad)
Obě zadní tlačítka najednou	*Testing Interface*	Zapnutí/vypnutí dálkoměrů
Zadní levé tlačítko	*Glyph Following*	Zapnutí/vypnutí sledování obrazce
Zadní pravé tlačítko	*Glyph Following*	Zapnutí/vypnutí hledání obrazce
Zadní levá ovládací páčka	*Initialization*	Smazání chyby H-můstků jednotky motorů
Zadní pravá ovládací páčka	*Initialization*	Smazání chyby „watchdog“ jednotky motorů

Obr. E.1: Tabulka významů tlačítek ovladače joystick

Příloha F

Fotografie mobilní platformy



Obr. F.1: Fotografie mobilní platformy