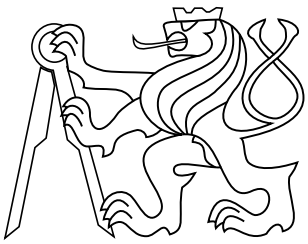
České vysoké učení technické v Praze
Fakulta elektrotechnická



Navigace mobilního robotu ve venkovním prostředí

Bakalářská práce

Adam Heinrich

Praha, 2014

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Adam H e i n r i c h

Studijní program: Kybernetika a robotika (bakalářský)

Obor: Robotika

Název tématu: Navigace mobilního robotu ve venkovním prostředí

Pokyny pro vypracování:

1. Nastudujte dostupné metody navigace mobilních robotů a zpracování senzorických dat.
2. Navrhněte navigační systém vhodný pro účely soutěže "Robo-orienteering", schopný navigovat robot do požadované cílové pozice v obecném venkovním prostředí.
3. Zvolte vhodnou senzorickou výbavu pro vyřešení úlohy a implementujte navigační systém pro mobilní robot.
4. Funkci navigace ověřte při praktických experimentech a proveďte vyhodnocení jeho chování.

Seznam odborné literatury:

[1] Novák Petr: Mobilní roboty - pohony, senzory, řízení. BEN - technická literatura, 2005.
[2] Corke Peter: Robotics, Vision and Control. Springer Berlin Heidelberg, 2011.
[3] Yaghmour Karim, et al.: Building Embedded Linux Systems 2nd ed. O´Reilly Media, 2008.

Vedoucí bakalářské práce: Ing. Jan Chudoba

Platnost zadání: do konce letního semestru 2014/2015

L.S.

doc. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 10. 1. 2014

Prohlášení autora práce

Prohlašuji, že jsem předloženou práci vykonal samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne …..... .....….....

Podpis autora práce

Anotace

Práce se zabývá návrhem a implementací navigačního a senzorického systému mobilního robotu pro navigaci mezi kontrolními body ve venkovním prostředí. Navigační systém využívá znalosti mapy daného prostředí pro naplánování cesty s ohledem na vlastnosti terénu a rozmístění překážek. Pro ověření funkčnosti v reálném prostředí byla navržena robotická platforma, založená na podvozku auta v měřítku 1:10. Tato platforma je řízena jednodeskovým počítačem s operačním systémem Linux, pro který byla vytvořen modulární software, zajišťující zpracování dat ze senzorů a řízení pohybu.

Na platformě byla zatím úspěšně otestována detekce kontrolních bodů na základě obrazu z všesměrové kamery.

Abstract

The bachelor thesis deals with the problem of designing and implementing of a navigation and sensory system for a small mobile robot. The robot's task is to navigate itself through checkpoints in an outdoor environment. The navigation system takes advantage of knowing environment's map to find the best way with respect to terrain properties and placement of obstacles. A robotic platform built on top of 1/10th scale car was designed to prove the functionality in real environment. The platform is controlled by single board Linux computer with custom designed modular software for the sensor processing and a movement control.

It was used to test checkpoint detection based on processing images from an omnidirectional camera.

Děkuji vedoucímu práce, Ing. Janu Chudobovi, za připomínky a pomoc s jejím vypracováním. Děkuji také svým rodičům za trpělivost a podporu během celého studia.

Obsah

1 Úvod

Cílem práce je navrhnout navigační systém a sestrojit autonomní mobilní robot, schopný splnit úlohu danou soutěží RoboOrienteering [5]. Úkolem robotu je v parkovém prostředí projet zadané kontrolní body. Po uplynutí časového limitu je sečteno bodové hodnocení za každý dosažený kontrolní bod, případně za návrat do startovního místa. Kontrolní body jsou vyznačeny oranžovými kužely o výšce 45 cm a za dosažení kontrolního bodu je považováno přiblížení se ke kuželu na vzdálenost menší než 2,5 m. Robot nesmí narazit do kuželu nebo překážky. Přesný počet a rozmístění kontrolních bodů je znám až těsně před startem. Na pořadí průjezdu kontrolními body nezáleží.

Pro lokalizaci robotu v prostředí je možné využít systém GPS, k navigaci na konkrétní souřadnici je ale vhodné použít elektronický kompas, protože azimut vypočítaný GPS přijímačem je při pomalých rychlostech velice nepřesný. Vzhledem k nepřesnosti určení polohy v řádech metrů je také nutné zpřesnit dohledání kontrolních bodů. K tomu lze využít obraz z kamery, jelikož oranžové kužely jsou v parkovém prostředí dostatečně výrazné.

Protože se robot pohybuje v předem známém prostoru, je pro plánování pohybu vhodné využít mapu, tedy znalosti umístění výrazných překážek (stromy, lavičky, voda). Kromě toho musí ale být schopen reagovat na předem neznámé překážky, například chodce nebo další roboty. Měl by tedy být vybaven senzory pro jejich detekci. K tomuto účelu se jako vhodná se jeví kombinace infračervených a ultrazvukových dálkoměrů. Infračervené dálkoměry využívají k detekci překážky úzký paprsek, zatímco ultrazvukové dálkoměry zabírají větší prostor. Jsou ale náchylné na tvar překážky a falešné ozvěny. Jejich kombinací tak je možné vliv nežádoucích vlastností kompenzovat.

2 Robotická platforma

2.1 Konstrukce

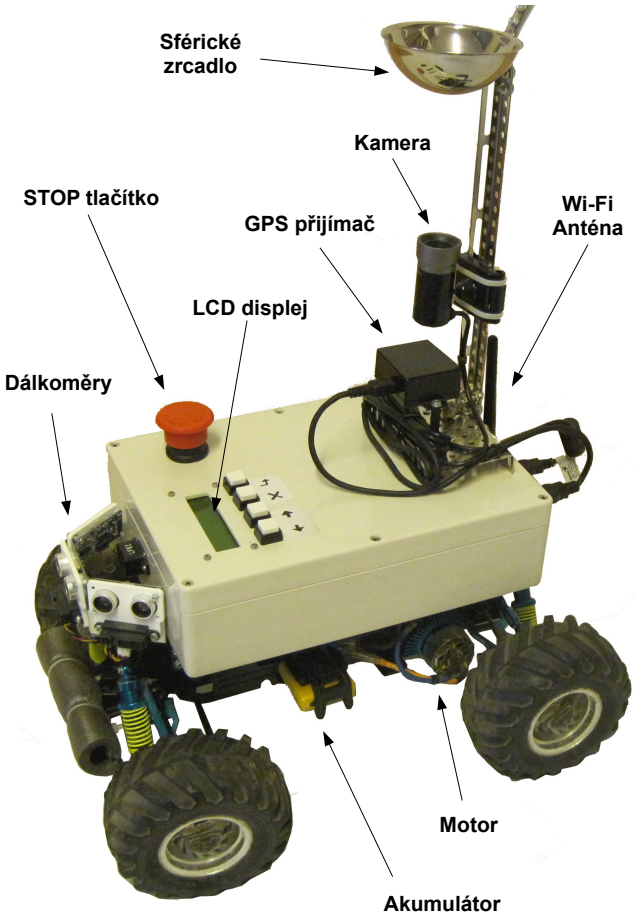
Pro pohyb v parkovém prostředí byl vybrán běžně dostupný rádiem řízený model auta Tamiya TL-01B v měřítku 1:10. Podvozek má náhon na všechna čtyři kola s diferenciálem a jeho nápravy jsou odpruženy olejovými tlumiči. Zatáčení je ovládáno běžným modelářským servomotorem s napájením 5 V a pohon zajišťuje stejnosměrný elektromotor určený pro napětí přibližně 7,2 V. Výhodou podvozku je dobrá průchodnost terénem a dostupnost náhradních dílů.

Model auta byl zbaven čtyř držáků karoserie a na jejich místo byl upevněn rám z profilů stavebnice Eitech. Tato konstrukce slouží k vyvázání důležitých kabelů a k připevnění plastové krabice s elektronikou. Zároveň pomáhá chladit regulátor otáček motoru. Pro větší tuhost rámu a zmírnění namáhání plastové krabice byly zvoleny profily ve tvaru písmene L.

Pro umístění elektroniky byla vybrána krabice o rozměrech 265×185×65 mm. Do víka krabice byly vyřezány otvory pro umístění LCD displeje, ovládacích tlačítek a bezpečnostního vypínače. V zadní straně se nacházejí otvory pro dva USB konektory, napájecí konektor a anténu Wi-Fi modulu. Spodní strana obsahuje dva otvory pro průchod kabelů z elektroniky umístěné mimo krabici. K nosnému rámu je krabice připevněna čtyřmi distančními sloupky.

Rám také nese držák dálkoměrů v přední části robotu. Jeho kostra je vyrobena z ohnutého hliníkového plechu, ke kterému jsou přišroubovány samotné držáky senzorů vyřezané laserem z bílého plexiskla.

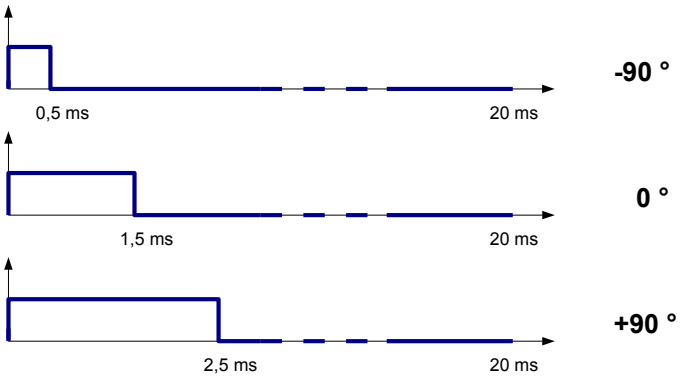
Na obrázku 2.1 je zachycena fotografie hotové robotické platformy spolu s označením některých důležitých částí, popsaných v následujících kapitolách. Další fotografie jsou nahrány na přiloženém CD.



Obrázek 2.1: Fotografie robotu

2.2 Řízení motorů

Poloha modelářského servomotoru je řízena délkou pulsu vstupního signálu. Signál je v 5 V úrovních a opakuje se s periodou 20 ms. Středové poloze odpovídá puls o délce 1,5 ms, výchylkám ±45° pak pulsy o délkách 1 a 2 ms [3]. Použitý servomotor ale zvládá výchylky až ±90° (obr. 2.2).



Obrázek 2.2: Jedna perioda průběhu servosignálu

Pro ovládání trakčního stejnosměrného motoru slouží regulátor[1] Graupner Speed Profi R40, který je řízen podobným signálem jako modelářský servomotor. Regulátor je vybaven funkcí brzdy, což znamená, že při přechodu na zpětný chod je motor nejprve zkratován a až poté se začne otáčet opačným směrem. Další vlastností je automatické nastavení středové polohy (zastavení motoru), aby se lépe přizpůsobil nastavení vysílačky pro ovládání RC modelu. Proto je při řízení nutné nejprve několik cyklů generovat puls o šířce 1,5 ms.

1. Označení regulátor pochází z modelářské terminologie a může být zavádějící, neboť zařízení nepracuje se zpětnou vazbou.

2.3 Napájení

Celý robot je napájen sériově zapojeným dvoučlánkovým lithium-polymerovým akumulátorem s udávaným napětím 7,2 V. Napětí z akumulátoru je přivedeno kabelovou spojkou do regulátoru otáček motoru. Tato spojka zajišťuje mezičlánek mezi dvěma rozdílnými typy konektorů a také přivedení napájecího napětí připájeným slabším kabelem do svorkovnice, umístěné v krabici s elektronikou.

Za svorkovnicí je umístěn modul s DC-DC měničem Traco TSR 1-2450 [7], který vytváří napájecí napětí 5 V pro řídicí počítač. Měnič je schopen dodávat proud až 1 A s maximálním překmitem 1%. Z desky řídicího počítače je pak vyvedeno ještě napětí 3,3 V, využité některými senzory.

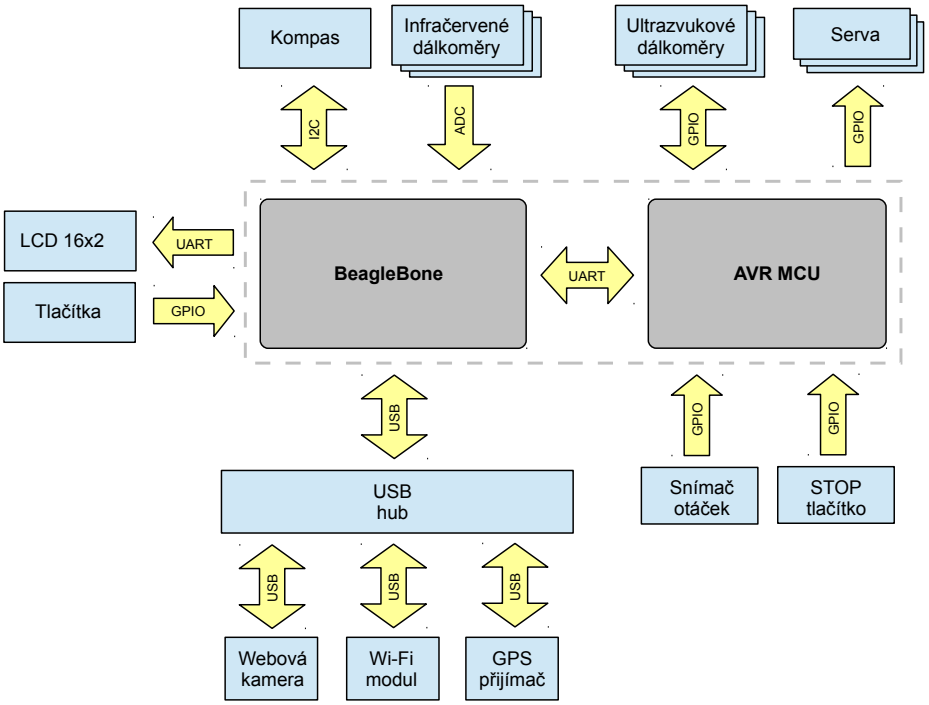
Modelářský servomotor je napájen samostatnou 5 V větví, kterou vytváří modelářský regulátor. Díky tomuto oddělení by nemělo docházet k výpadkům napájení při větších odběrech serva.

Pro účely ladění některých součástí robotu bez použití motoru je možné napájet elektroniku také ze zdroje stejnosměrného proudu se souosým napájecím konektorem 5,5×2,1 mm.

2.4 Řídicí počítač

Pro řízení robotu byl zvolen jednodeskový počítač BeagleBone, který obsahuje čip Sitara AM3358 od výrobce Texas Instruments (procesor ARM Cortex-8), 512 MB paměti RAM a dostatek periferií (GPIO, A/D převodník, sériové porty, I^2C), které jsou vyvedeny na dva 46 pinové konektory. Všechny signály jsou v 3,3 V úrovních. BeagleBone počítá s připojováním vlastních rozšiřujících desek (tzv. *capes*), které jsou v systému identifikovány programem *capemgr* na základě informací uložených v paměti EEPROM.

Podle specifikace [1] byla tedy navržena vlastní rozšiřující deska pro připojení senzorů a ovládání motorů. Kromě zmíněné EEPROM 24C256 deska obsahuje konektor pro připojení dalšího I^2C zařízení na stejnou sběrnici. Ten je využit ke komunikaci s digitálním kompasem.



Obrázek 2.3: Blokové schéma elektroniky připojené k řídicímu počítači

Kvůli možnosti připojení sériového LCD modulu SIC1602A20 [18] je na desce umístěn zámkový konektor, odpovídající konektoru na displeji a invertor realizovaný NPN tranzistorem. Invertor převádí TX signál sériového portu řídicího počítače v 3,3 V úrovních na opačný signál v 5 V úrovních, vyžadovaný LCD modulem.

Přesné generování signálu pro řízení modelářských servomotorů zajišťuje osmibitový mikrokontrolér Atmel ATmega8A [19], který je napájen z 5 V větve. Mikrokontrolér je s řídicím počítačem spojen sériovou linkou (úrovně signálu RX jdoucího z mikrokontroléru jsou sníženy odporovým děličem) a signálem, který umožňuje provést RESET mikrokontroléru. Díky tomu je možné nahrát do mikrokontroléru nový program bootloaderem podle aplikační poznámky AVR109 [13]. Kromě čtyř konektorů pro modelářské servomotory je z mikrokontroléru vyvedeno také pět konektorů pro připojení dalších senzorů, například ultrazvukových dálkoměrů HC-SR04,

snímače otáček nebo bezpečnostního STOP tlačítka.

Pro připojení infračervených dálkoměrů GP2D120 jsou na desce umístěny tři konektory a odporové děliče, které snižují výstupní napětí senzorů. Maximální napětí pro A/D převodník počítače BeagleBone je totiž pouze 1,8 V. Tyto konektory jsou třípinové a mají podobné rozložení jako konektory pro modelářská serva (s napájením uprostřed), aby případné přepólování nezpůsobilo zničení senzorů.

Dále se na desce nachází konektor pro připojení pěti digitálních vstupů/výstupů, konektor pro připojení bzučáku (sirénky), která je spínána NPN tranzistorem a LED dioda pro signalizaci stavu mikrokontroléru.

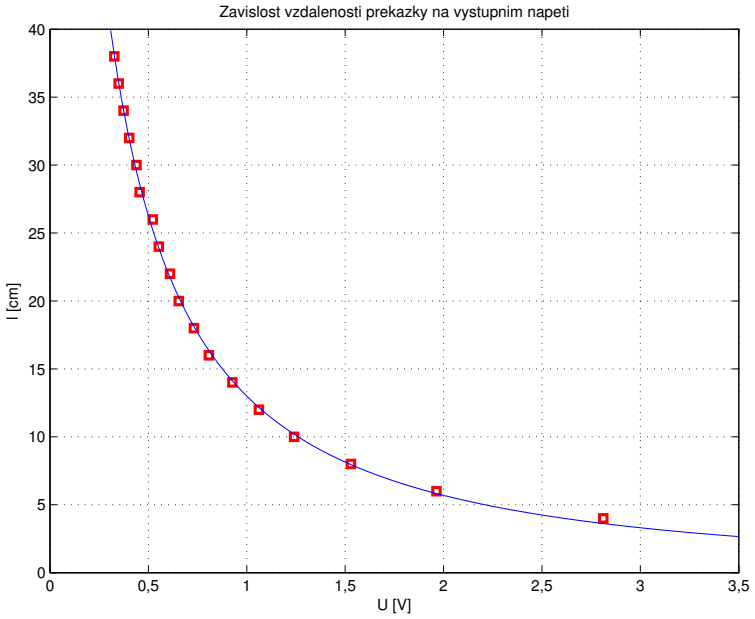
Místo samostatného mikrokontroléru AVR je pro řízení signálů náročných na přesné časování možné využít koprocesor PRU (Programmable Real-time Unit), který se nachází přímo v čipu AM3358. Výhodou by bylo efektivnější využití prostředků a rychlý přístup k registrům pomocí DMA, nevýhodou pak nutnost psát program v Assembleru a převést signály do 5 V úrovní.

Rozšiřující deska byla původně navržena pro první verzi BeagleBone White. Po výměně za novější BeagleBone Black bylo nutné softwarově vypnout některá zařízení, která sdílí piny využívané rozšiřující deskou (konkrétně HDMI framer a eMMC paměť). Mezi konfliktní piny patří například sériový port UART5, který je využitý pro komunikaci s mikrokontrolérem AVR. Konflikt způsobuje, že se naměřená data odeslaná mikrokontrolérem vrací, a způsobují chyby v přijatých příkazech pro ovládání servomotorů. Kvůli tomu nebylo možné ultrazvukové senzory použít.

2.5 Senzory

2.5.1 Infračervené dálkoměry

Pro detekci překážek je v přední části robotu umístěna trojice infračervených dálkoměrů GP2D120 [20], vyráběných firmou Sharp. Senzory měří vzdálenost k překážce na principu triangulace v rozsahu 4-30 cm a mají analogový výstup. Výstupní napětí je nepřímo úměrné naměřené vzdálenosti (obr. 2.4). Velkou výhodou senzorů je malá závislost výstupního signálu na barvě povrchu překážky.



Obrázek 2.4: Naměřené hodnoty napětí a výsledná závislost

Z naměřených dat byl nástrojem *cftool* v prostředí Matlab sestaven vztah pro převod výstupního napětí U [V] na vzdálenost l [cm]:

$$l = \frac{71{,}52}{U^2 + 4{,}061 \cdot U + 0{,}4415}.$$

Vzhledem k poměrně nízké detekční vzdálenosti jsou senzory použity pouze jako nárazníky, které detekují přiblížení k překážce.

2.5.2 Digitální kompas

Digitální kompas slouží k určení azimutu (orientace) na základě hodnot intenzity magnetického pole Země, naměřených magnetometrem ve třech osách. Nevýhodou použití samotného magnetometru je citlivost na natočení senzoru. V robotu je proto použit IMU (Inertial Meassurement Unit) senzor MPU-9150 výrobce Invensense [21], který je kombinací tříosého magnetometru, gyroskopu i akcelerometru. Umožňuje tak měřit intenzitu magnetického pole, zrychlení i úhlového zrychlení. Kromě toho obsahuje ještě

tepelný senzor a tzv. Digital Motion Processor (DMP), který umožňuje data ze senzorů zpracovávat přímo v čipu.

Nadřazený systém s jednotlivými zařízeními v senzoru komunikuje prostřednictvím sběrnice I^2C. Výrobce poskytuje proprietární program do DMP, který při výpočtu zohledňuje natočení senzoru i teplotně závislý drift gyroskopu. Tento program je nutné při každém použití do čipu nahrát, protože je uložen ve volatilní paměti. Dokumentace přesné funkce programu ani DMP bohužel není přístupná, senzor tak funguje jako „black box“ kompas s kompenzací náklonu.

Senzor je umístěn v přední části krabice s elektronikou tak, aby se nacházel co nejdále od motoru, který je velkým zdrojem rušení magnetomeru.

2.5.3 Snímač otáček

Měření otáček je důležité pro řízení rychlosti pohybu. Vzhledem k tomu, že se robot pohybuje terénem a zátěž motoru se mění, je znalost reálných otáček nutná. Moderní rádiem řízené modely aut využívají bezkomutátorové BLDC motory s integrovanými hallovými sondami a elektronikou, která na základě zpětné vazby otáčky motoru reguluje. Nevýhodou tohoto řešení je uzavřenost, informaci z enkodéru totiž není možné použít pro měření ujeté vzdálenosti.

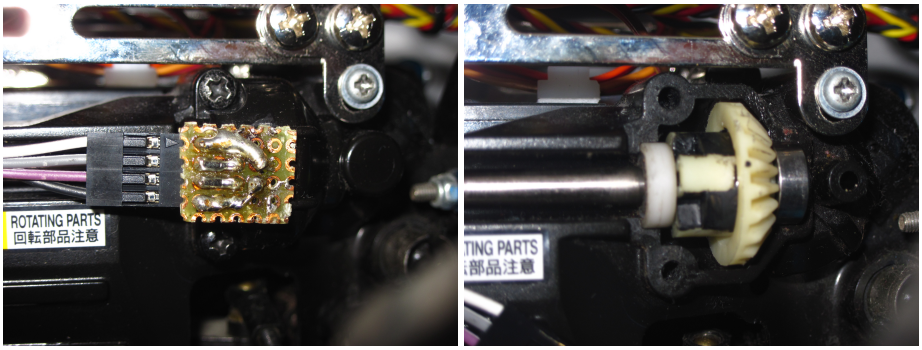
Další možností je použití optického enkodéru s clonkou, která je připevněna k hřídeli motoru. Vzhledem k tomu, že vyčnívající část hřídele na zadní straně použitého motoru není dostatečně dlouhá, musel by být takovýto senzor k motoru připevněn trvale. To by znemožňovalo případnou pozdější výměnu motoru za jiný.

Původně bylo proto zamýšleno řešení s hallovým senzorem připevněným na nápravě a snímajícím přítomnost magnetů vlepených do plastového kola. Byly provedeny experimenty s čidly s digitálním i analogovým výstupem a s různými magnety. Vzhledem k citlivosti senzoru bylo nutné, aby vzdálenost od magnetu nepřesáhla jednotky milimetrů. Uchycení kola ale vykazuje určitou vůli, díky které docházelo k zavadění kola o senzor. Dalším problémem bylo samotné uchycení senzoru na nápravě, které není příliš odolné

k pohybům odpružení. Tato metoda tedy nebyla použita.

Ke snímání otáček slouží odrazové čidlo QRD1114 [22], které v jednom pouzdru obsahuje infračervenou LED a fototranzistor. Čidlo reaguje na odraz od reflexních materiálů přibližně do vzdálenosti 10 mm.

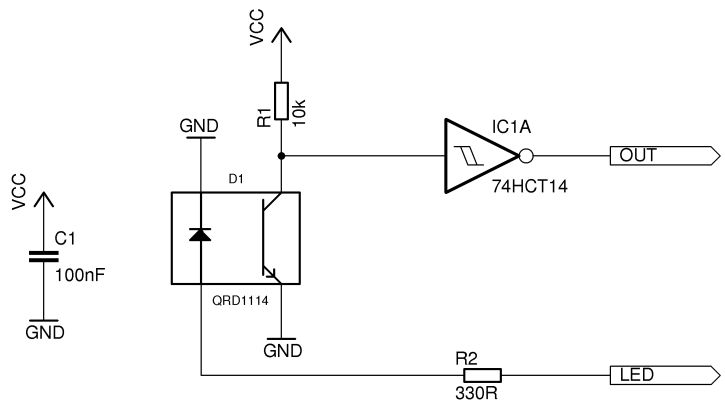
Toto čidlo je umístěno v krytu převodovky a je namířeno na bílé plastové ozubené kolečko, které přenáší pohyb z rozvodné osy do předního diferenciálu. Podobný princip byl použit v [17]. Na kolečku (obr. 2.5) jsou černou matnou barvou namalovány čtyři pruhy o šířce odpovídající rozměrům senzoru tak, aby bylo možné změnu odrazivosti materiálu spolehlivě zaznamenat. Vhodná šířka a počet proužků byly určeny experimentálně. Díky umístění v plastovém krytu převodovky je senzor odolný vůči nečistotám a dennímu světlu, jehož spektrum obsahuje i infračervené záření.



Obrázek 2.5: Fotografie enkodéru

Zapojení senzoru je na obrázku 2.6. Bílá barva způsobí odraz světla, plné otevření tranzistoru a stažení výstupního napětí. Černá barva světlo částečně pohlcuje, takže je výstupní napětí vyšší.

Vzhledem k nedokonalému pohlcení světla je vysoká úroveň signálu pouze 2,2 V, toto napětí navíc kvůli nedokonalosti proužků kolísá. Proto je za fototranzistorem připojen ještě Schmittův invertor 74HCT14, který je díky hysterezi odolný vůči šumu a vytvoří čistý digitální signál pro zpracování mikrokontrolérem.



Obrázek 2.6: Schéma zapojení enkodéru

Rozlišení enkodéru je poměrně nízké, jednomu pulsu odpovídá ujetá vzdálenost 4,55 cm.

2.5.4 Kamera

Kamera slouží k určení polohy oranžového kuželu. Vzhledem k předem neznámému umístění kuželu je nutné, aby byla kamera schopná snímat celé okolí robotu. Toho je možné docílit natáčením kamery nebo použitím speciální optiky, například speciálního nástavce na kameru mobilního telefonu [7] pro tvorbu panoramatických fotografií. Nevýhodou tohoto nástavce je jeho nízká výška, při plánovaném umístění kamery na víko krabice by nebylo možné detekovat kužely v blízkosti robotu.

Proto je na robotu připevněna konstrukce z profilů stavebnice Eitech, která drží kameru nasměrovanou na střed lesklé nerezové kuchyňské naběračky. Tato naběračka slouží jako náhrada sférického zrcadla. Podobné řešení bylo použito například v [8]. Díky této konstrukci je možné experimentálně měnit vzdálenost zrcadla a kamery.

Čip AM3358 na desce BeagleBone je vybaven rozhraním pro připojení kamery přes paralelní sběrnici, to by ale vyžadovalo umístění kamery co nejblíž k řídicímu počítači. Proto je v robotu použita levná USB webkamera Genius FaceCam 1010. Tato kamera má rozlišení 1024×768 pixelů a objektiv s možností ručního ostření.

2.5.5 Snímač proudu

Senzor proudu ACS711 [23] slouží k měření proudu protekajícího motorem. Senzor obsahuje Hallův senzor, který měří magnetické pole generované protékaným vodičem. Výhodou je elektrická izolace měřicí části od části silové a nízké ztráty způsobené použitím senzoru. Měřený proud v rozsahu ± 31 A je přímo úměrný výstupnímu napětí.

Původním důvodem pro použití senzoru byla možnost detekce přetížení motoru, a tedy zaseknutí robotu o překážku. Proud protékaný kotvou stejnosměrného elektromotoru totiž při vyšší mechanické zátěži roste. Vzhledem k využití enkodéru (kap. 2.5.3) ale nebyla tato možnost použita.

2.5.6 GPS přijímač

Pro určení polohy robotu byl zvolen GPS přijímač Ublox NEO6-M [24] na modulu vyráběném firmou Drotek. Tento modul je kromě samotného GPS přijímače vybaven záložní baterií, regulátorem napětí, keramickou anténou a konektory pro připojení USB kabelu nebo sériové linky. Díky malým rozměrům bylo možné modul umístit do plastové krabičky o rozměrech 46×40×20 mm.

Přijímač NEO6-M má udávanou přesnost zjištění polohy 2,5 m, kterou je schopen měřit až pětkrát za sekundu. Přesnost určení polohy by bylo možné zvýšit například použitím identického stacionárního přijímače se známou polohou, umístěného v blízkém okolí robotu, který by předával informaci o aktuální chybě. Pravidla soutěže RoboOrienteering ale bohužel komunikaci se zařízeními umístěnými mimo robot zakazují.

Kromě určení polohy je GPS přijímač použit také pro zjištění přesného času, protože deska BeagleBone neobsahuje zálohovaný RTC obvod. Přesný je potřeba například pro přehlednější logování událostí.

3 Software

3.1 Operační systém

Vývojová deska BeagleBone je dodávána s Linuxovou distribucí Ångström, která je oblíbená i v jiných embedded zařízeních. Tato předkonfigurovaná distribuce je zaměřena především na hobby komunitu, proto obsahuje webové vývojové prostředí Cloud9, které umožňuje vytvářet JavaScriptové programy s využitím frameworku Node.js.

Vzhledem k osobním preferencím byla zvolena distribuce Debian Wheezy [16], která obsahuje pouze nejdůležitější nástroje pro práci s deskou. Jde zejména o program capemgr, který usnadňuje identifikaci rozšiřujících desek (Capes) podle informací uložených v EEPROM, konfiguraci periferií na základě požadavků desek a řešení případných konfliktů. K nastavení periferií je použit standardní nástroj Device Tree Overlays [6], díky kterému není nutné zasahovat do jádra systému.

3.2 Moduly a komunikace mezi nimi

Software robotu byl od počátku zamýšlen jako modulární tak, aby bylo možné psát jednotlivé moduly v různých jazycích (C/C++, Java, PHP, ...), a aby bylo možné tyto moduly spouštět nezávisle a na různých počítačích, zejména z důvodu ladění.

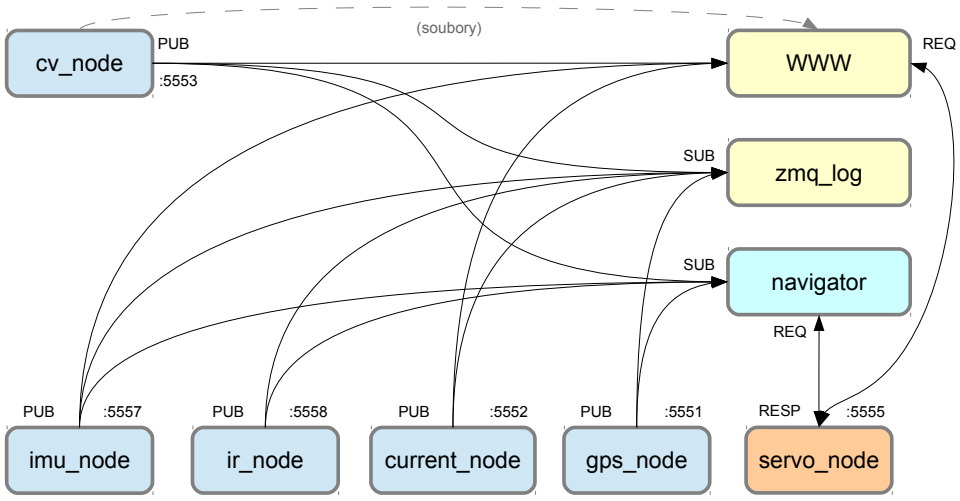
Původním záměrem bylo využití platformy ROS (Robot Operating System) [10], což je sada nástrojů a knihoven, které usnadňují nejen komunikaci mezi programy, logování událostí a distribuované řízení, ale řeší také běžné problémy mobilní robotiky, jako je například navigace v prostoru, lokalizace, zpracování dat z běžných senzorů nebo jejich filtrace. Výhodou je univerzálnost a velká komunita uživatelů, nevýhodou pak poměrně strmá učící křivka, fragmentovaná a nekompletní dokumentace a poměrně náročná kompilace.

Vzhledem k omezenému času byla nakonec tato varianta zavržena a výměnu zpráv mezi jednotlivými programy zajišťuje knihovna ZeroMQ [14].

Tato knihovna je implementována v mnoha programovacích jazycích, zajišťuje konzistenci přenášených dat a disponuje vyčerpávající dokumentací. Komunikace mezi programy probíhá na TCP socketech ve dvou základních režimech:

- **Request/Response** (klient/server) pro programy, které na základě požadavků klientů provádějí akce. Například jde o program zajišťující řízení motorů (server), který přijímá požadavky řídicího programu nebo programu pro dálkové ovládání (klient).
- **Publisher/Subscriber** pro programy, které posílají data k dalšímu zpracování. Typicky jde o program, zajišťující čtení dat ze senzorů (publisher), která mohou být dále zpracovávány řídicími nebo vizualizačními programy (subscriber).

Schéma 3.1 znázorňuje propojení nejdůležitějších modulů:



Obrázek 3.1: Propojení jednotlivých modulů a seznam použitých portů

Jednotlivé moduly si vyměňují zprávy ve formátu JSON (JavaScript Object Notation). Tento formát byl zvolen kvůli nezávislosti na konkrétní plat-

formě (parsery jsou dostupné pro všechny běžné programovací jazyky) a jednoduché čitelnosti, vhodné pro ladění.

- **imu_node** zajišťuje čtení dat z digitálního kompasu (kap. 2.5.2) připojeného ke sběrnici I^2C. K tomu je použit hotový projekt linux-mpu9150 [15] v jazyce C, který byl doplněn o publikování ZeroMQ zpráv.

- **ir_node** zajišťuje čtení výstupního napětí tří infračervených dálkoměrů (kap. 2.4), připojeným k A/D převodníku, a převod naměřených hodnot na vzdálenost k překážce v centimetrech.

- **current_node** zajišťuje čtení dat ze senzoru proudu (kap. 2.5.5), připojeného k A/D převodníku, a převod naměřených hodnot na velikost proudu v Ampérech.

- **servo_node** slouží k ovládání servomotoru a regulátoru otáček motoru, kterým je mikrokontrolér AVR připojený přes sériovou linku UART.

- **gps_node** zajišťuje načítání aktuální polohy z GPS přijímače, připojeného rozhraním USB jako virtuální sériový port.

- **cv_node** využívá obrazu z kamery pro vyhledání kuželu a určení jeho polohy. Použité algoritmy jsou popsány v kap. 4.2.1 a 4.2.2. Tento program je napsán v jazyce C++ s využitím knihovny OpenCV. Před kompilací je možné pomocí direktiv překladače zvolit, zda půjde o verzi použitou v robotu, nebo o verzi s grafickým uživatelským rozhraním, která je použita pro ladění algoritmů a nastavení parametrů kamery.

- **navigator** je řídicí program po jízdu na zadanou zeměpisnou souřadnici, popsaný v kapitole 4.1.3.

- **zmq_log** slouží k ukládání zpráv vyslaných moduly v režimu Publisher. Jedna instance programu slouží k ukládání dat z jednoho zdroje. Přijaté zprávy nejsou interpretovány, jsou pouze doplněny o časovou značku. Uložená data jsou využita k ladění.

3.3 Webové rozhraní

Pro účely dálkového ovládání je na robotu spuštěn webový server Lighttpd, který zpřístupňuje jednoduchou stránku se směrovými šipkami pro ovládání otáček motoru a natočení serva. Webová stránka byla zvolena proto, aby bylo možné robot ovládat z libovolného zařízení vybaveného Wi-Fi a webovým prohlížečem. Šipky reagují jak a kliknutí, tak i na stisk příslušné klávesy.

Rozhraní je vytvořeno ve jazyce HTML s využitím JavaScriptové knihovny jQuery, která zjednodušuje práci s uživatelským prostředím a asynchronní komunikaci se skriptem v PHP. Ten slouží jako prostředník mezi JavaScriptem běžícím v prohlížeči uživatele a knihovnou ZeroMQ.

3.4 Matlab

Prostředí Matlab bylo použito pro návrh a testování algoritmů před jejich implementací v konkrétním programovacím jazyce. K tomu sloužily především knihovny Robotic Toolbox, Machine Vision Toolbox [2] a jednoduchý program napsaný v jazyce C, který přijímá ZeroMQ zprávy na daném portu v režimu subscriber.

Tento program je v cyklu volán funkcí *system* a jeho výstup ve formátu JSON je pak dále zpracováván. Díky tomu je možné na počítači vyhodnocovat a zobrazovat data ze senzorů v reálném čase.

3.5 Mikrokontrolér AVR

Firmware pro mikrokontrolér Atmel AVR ATmega8A je napsán v jazyce C s použitím knihovny Avr Libc [12]. Program v hlavní smyčce vyhodnocuje příkazy přijaté po sériové lince. Příkazy jsou v čitelném formátu, aby bylo možné s mikrokontrolérem komunikovat prostřednictvím běžného terminálového programu:

- Příkaz *SERVOn=poloha* nastaví polohu serva. Číslo serva *n* je v rozsahu 0-3, *poloha* je udávána jako délka pulsu v mikrosekundách v rozsahu

1000-2000.

- Příkaz *SERVOn=OFF* způsobí, že mikrokontrolér přestane generovat pulsy pro dané servo. Tento stav je výchozí.
- Příkaz *REG=perioda* nastaví periodu mezi tiky enkodéru, které se snaží dosáhnout PID regulátor. Perioda je udávána v milisekundách.
- Příkaz *REG=OFF* zastaví PID regulátor a motor je tak možné ovládat „ručně“ bez zpětné vazby. Tento stav je výchozí.
- Příkaz *WDT=OFF* zastaví časovač Watch Dog Timer (WDT). Ve výchozím stavu je časovač zapnutý a způsobí reset mikrokontroléru v případě, že neobdrží žádnou zprávu déle než 1 s. Tato funkcionalita slouží jako záchranná brzda v případě, že řídicí program zkolabuje, případně při výpadku signálu v režimu dálkového ovládání. Vypnutí WDT je vhodné pro ruční testování funkcionality programu.

Program předpokládá připojení regulátoru otáček na pozici *SERVO3.* Vzhledem k vlastnosti regulátoru (2.2) je nutné nejprve spustit generování signálu pro neutrální hodnotu a až poté regulaci otáček.

Například sekvence příkazů pro regulaci na rychlost

$$v = \frac{s}{t} = \frac{4{,}55 \cdot 10^{-2}}{100 \cdot 10^{-3}} = 0{,}46\,\mathrm{m} \cdot \mathrm{s}^{-1}$$

je tedy

```
SERVO3=1500
REG=100
```

Serva jsou řízena integrovaným čítačem/časovačem, který inkrementuje svou hodnotu každou mikrosekundu. Délky pulsů jednotlivých serv jsou uloženy ve vzestupně setříděném poli, což umožňuje efektivní implementaci, která pro jednu periodu generovaných servosignálů (20 ms) potřebuje pouze N+1 přerušení, kde N je počet aktivních serv.

Kromě časovače pro řízení serv v mikrokontroléru běží ještě druhý časovač s periodou 1 ms, který slouží k periodickému spouštění PID regulace a

k měření délky pulsu enkodéru. Pulsy enkodéru jsou zachytávány v přerušením při náběžné hraně výstupního signálu a výsledná délka pulsu je filtrována plovoucím průměrem přes čtyři měření. Tím se eliminují rozdíly mezi jednotlivými ručně namalovanými proužky kódovacího kolečka.

3.6 Uživatelské menu

Uživatelské menu umožňuje obsluhu robotu bez nutnosti připojení počítače. Program *LcdMenu* napsaný v jazyce Java využívá 16x2 znakového LCD displeje pro zobrazení menu a čtyř tlačítek pro pohyb v nabídce, spouštění a vypínání procesů. Každá položka v menu odpovídá jednomu procesu. První řádek displeje zobrazuje název programu a indikaci běhu, druhý řádek pak případný výstup programu.

Jednotlivé položky v menu umožňují například spustit řídicí program a moduly popsané v sekci 3.2 nebo systém restartovat. Kromě toho je možné přepínat Wi-Fi modul mezi režimy Přístupový bod a Klient.

3.7 Reprezentace mapy

Protože se robot pohybuje v předem známém prostředí, je vhodné pro plánování jeho pohybu využít mapu. Původním záměrem bylo použití OpenStreetMap [9], což je vektorový formát založený na XML, který pro reprezentaci mapy používá Uzly, Cesty, Relace a Atributy. Kromě samotného obrázku obsahuje také metadata vhodná pro strojové zpracování. Mapové podklady projektu OpenStreetMap vznikají na komunitní bázi a jsou volně dostupné. Výhodou tohoto formátu je otevřenost, univerzálnost a dostupnost knihoven pro zpracování i softwaru pro editaci, nevýhodou ale nízké detaily dostupných mapových podkladů. Pro použití v robotu by bylo nutné mapové podklady rozšířit o vlastní prvky, zahrnující například lavičky nebo keře, které jsou pro robot překážkou.

Proto je pro reprezentaci mapy v robotu použit tříbarevný bitmapový obrázek, který je z vytvořen z leteckého snímku v grafickém editoru. Červená barva znázorňuje překážku, zelená trávu a bílá asfalt (tedy povrch nejpříznivější pro pohyb):

Obrázek 3.2: Původní letecký snímek a vytvořená mapa. Převzato z [25]

Letecké snímky pocházejí ze serveru Mapy.cz, který umožňuje zaměření bodu na mapě a zobrazení jeho polohy v zeměpisných souřadnicích. Po zaměření dvou referenčních bodů na leteckém snímku je pak možné vytvořit vztah pro přepočet zeměpisných souřadnic na polohu v obrázku a naopak. Tento způsob reprezentace je elegantní, protože není nutné vytvářet programy pro editaci vlastního formátu. Výsledná mapa je navíc ve formátu, který je srozumitelný lidem.

Pro testování algoritmu popsaném v kapitole 4.1.1 a pro zaměření referenčních souřadnic byl v jazyce Java s využitím knihovny Swing vytvořen program *WorldEditor*. Umožňuje načtení obrázku mapy, umístění dvou referenčních bodů a zadání jejich souřadnic. Souřadnice referenčních bodů jsou ukládány do souboru, který je dále využíván navigačním programem.

4 Navigace

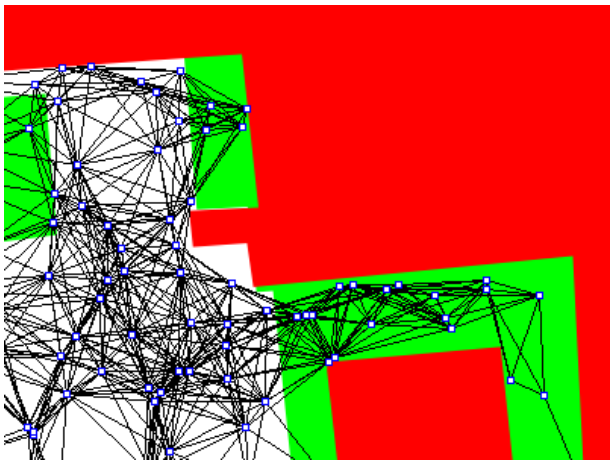
4.1 Plánování trasy pohybu

4.1.1 Vytvoření sítě cest

Aby bylo možné vyhledat trasu mezi dvěma body na mapě (případě naplánovat trasu přes dané kontrolní body), je potřeba vstupní mapu prostředí navzorkovat a vytvořit graf. V počítačových hrách je často používána reprezentace prostředí v mřížce. Každé políčko mřížky má jiné vlastnosti (například druh terénu), které se zohledňují při hledání trasy mezi dvěma body s co nejmenší cenou.

Pro použití v mobilním robotu byla zvolena metoda Probabilistic Roadmap [2], která spočívá v rozmístění několika náhodných bodů do dostupného prostoru v mapě (tedy do takového prostoru, který není vyznačen jako překážka) . Tyto body pak tvoří vrcholy navigačního grafu. Je-li mezi dvěma blízkými body přímá viditelnost, jsou spojeny hranou.

Výsledný navigační graf (obr. 4.1) pak může být použit pro naplánování trasy mezi dvěma body, která je rozdělena na posloupnost rovných úseků. Navigační graf je validní, neobsahuje-li oddělené komponenty (a tedy nedostupné body).

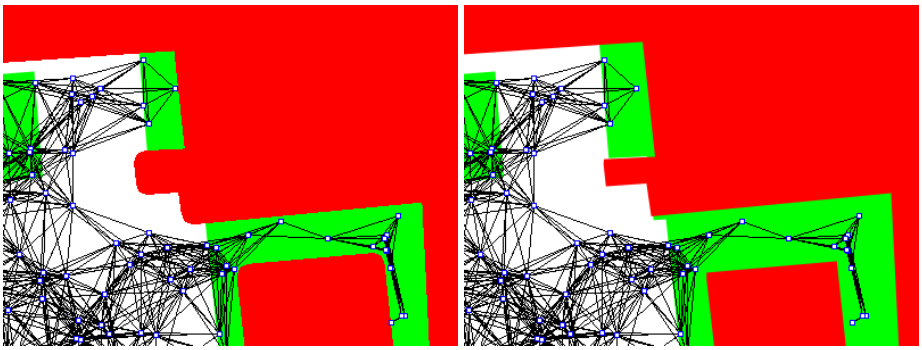


Obrázek 4.1: Příklad výsledného navigačního grafu

Pro praktické užití musí být mezi vrcholy grafu zahrnuty také všechny kontrolní body. Protože reálný robot není bodem, je nevhodné, aby se vrcholy navigačního grafu nacházely v těsné blízkosti překážek.

Proto program před generováním sítě cest využívá morfologickou operaci dilatace, kterou aplikuje na původní obrázek reprezentující mapu prostředí. Dilatace je prováděna na červených pixelech, znázorňujících překážku, kruhovým elementem o zadaném poloměru.

To vede k vytvoření jakési virtuální překážky, která zabírá více prostoru než překážka původní, a k vytvoření příznivějšího navigačního grafu:



Obrázek 4.2: Překážky po dilataci a nový navigační graf

Parametry programu pro tvorbu sítě cest jsou počet rozmisťovaných bodů, prahová vzdálenost mezi dvěma body pro vyhodnocení jejich blízkosti, poloměr dilatace a maximální počet pokusů pro vytvoření validního grafu. Vzhledem k povaze algoritmu je totiž při nízkých počtech bodů vysoká pravděpodobnost, že bude výsledný graf obsahovat více oddělených komponent.

Jednotlivé vrcholy navigačního grafu jsou reprezentovány kartézskými souřadnicemi, které odpovídají poloze vrcholu v mapovém obrázku (v pixelech). Přepočet polohy navigačního vrcholu z/do zeměpisných souřadnic (zeměpisná šířka a délka) je možný díky zaměření dvou referenčních bodů mapy.

4.1.2 Nalezení trasy

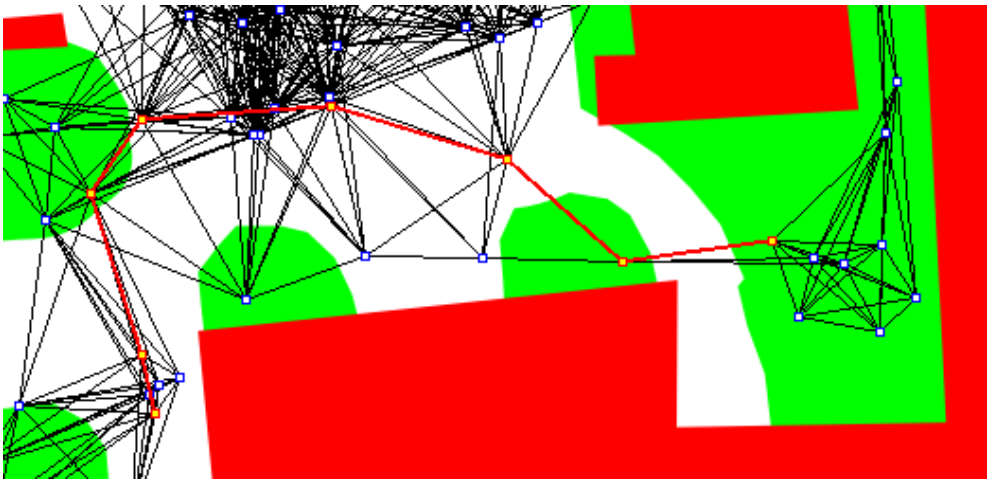
Nejprve je do navigačního grafu vložen vrchol, reprezentující aktuální polohu robotu. Následně je jako cíl vybrán nejblíže umístěný nenavštívený kužel. Pro nalezení trasy ke kuželu je použit Dijkstrův algoritmus, který jako oceňovací funkci přechodu mezi dvěma body p_1, p_2 kombinuje cenu použití hrany v a vzdálenost mezi body v kartézských souřadnicích, odpovídajících poloze bodů v obrázkové mapě:

$$f(p_1, p_2) = v(p_1, p_2) \cdot \sqrt{(x_{p2} - x_{p1})^2 + (y_{p2} - y_{p1})^2}$$

Oproti výpočtu vzdálenosti mezi dvěma body v zeměpisných souřadnicích je tento způsob výpočetně méně náročný. Jednotlivé hrany jsou oceněny na základě informací o terénu, uložených v mapě.

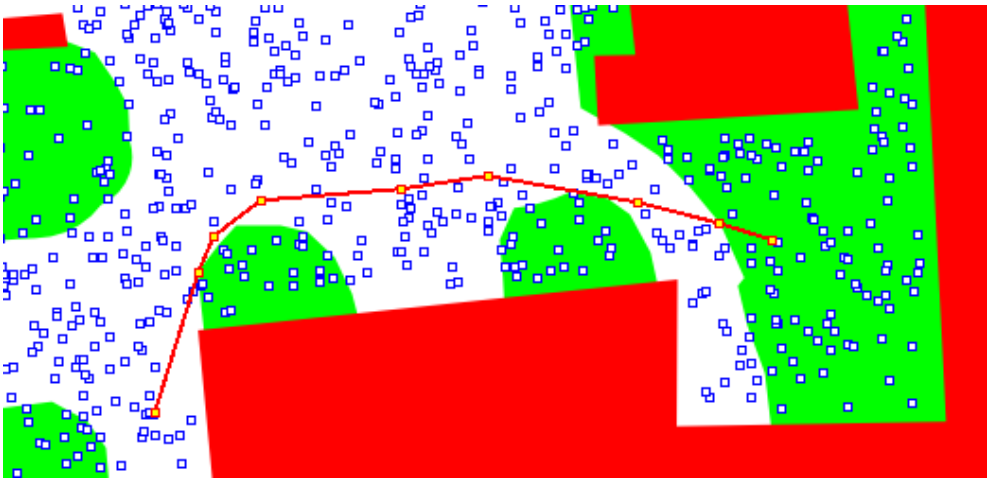
Kříží-li hrana zelenou oblast, značící trávu, je oceněna hodnotou $v = 1{,}5$. Pokud hrana prochází pouze bílou oblastí, značící nejvhodnější povrch, je oceněna hodnotou $v = 1$.

Tvar nalezené trasy závisí na počtu vrcholů navigačního grafu k a na prahu viditelnosti mezi vrcholy d. Při nízkém počtu vrcholů hrozí, že vyhledaná trasa nebude optimální (obr. 4.3).



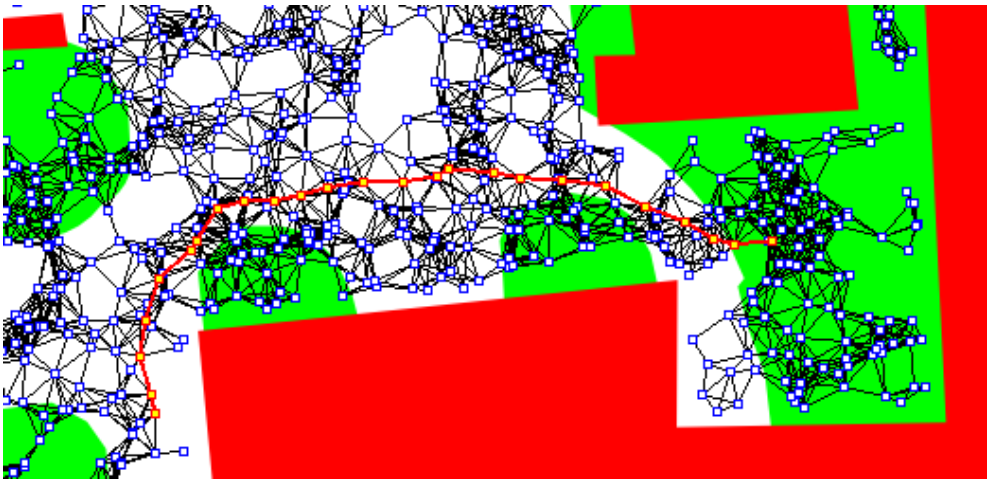
Obrázek 4.3: Příklad nevhodné trasy ($k = 250$, $d = 15\,\mathrm{m}$)

Po zvýšení počtu vrcholů navigačního grafu k je vyhledaná trasa přijatelnější (obr. 4.4). Pro přehlednost obrázku byly skryty hrany.



Obrázek 4.4: Příklad vhodné trasy ($k = 2500$, $d = 15\,\mathrm{m}$)

Vliv nastavení prahu viditelnosti d mezi dvěma vrcholy je zachycen na obr. 4.5. Je-li práh příliš nízký, je výsledná trasa tvořena zbytečně mnoha vrcholy.



Obrázek 4.5: Příklad nevhodné trasy ($k = 2500$, $d = 3\,\mathrm{m}$)

4.1.3 Jízda na zadané souřadnice

Nalezená cesta tvoří posloupnost vrcholů navigačního grafu, kterými musí robot projet. Protože je mezi dvěma po sobě jdoucími vrcholy přímá viditelnost, je možné nalezenou trasu projet opakovaným voláním programu pro rovnou jízdu.

Vstupem programu *navigator* je poloha cílového vrcholu v zeměpisných souřadnicích. Program nejprve vyčká na změření polohy GPS přijímačem, tedy dosažení stavu *2D/3D Fix*[1]. Poté vypočítá azimut mezi cílovým bodem a aktuální pozicí a nastaví otáčky motoru na konstantní rychlost. Vypočítaný azimut je pak porovnáván s azimutem naměřeným digitálním kompasem. Rozdíl požadovaného a změřeného azimutu tak slouží jako regulační odchylka PID regulátoru. Akční veličinou regulátoru je natočení servomotoru pro řízení směru pohybu.

V průběhu jízdy je vyhodnocována vzdálenost mezi cílovým bodem a aktuální polohou, změřenou GPS přijímačem. Je-li tato vzdálenost menší než 2,5 m, došlo k dosažení cílového bodu a pohyb robotu je zastaven.

Je-li v průběhu jízdy některým dálkoměrem detekována překážka, robot vyčká jejího odstranění.

1. Při prvním spuštění GPS přijímače (tzv. Cold Start) to může být až několik minut, později díky zálohovací baterii řádově jednotky sekund (tzv. Warm Start).

4.2 Jízda ke kuželu

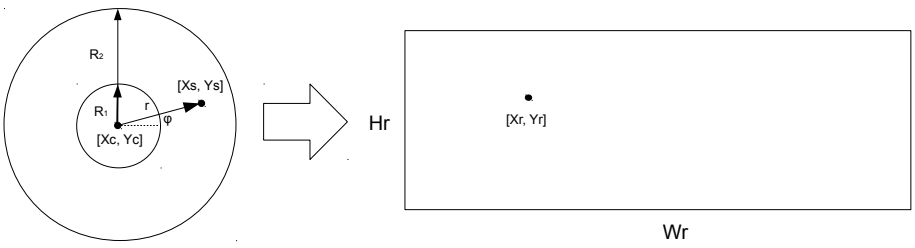
4.2.1 Transformace obrazu

Obrázek z kamery, která snímá odraz na sférickém zrcadle, je potřeba transformovat, aby se dal kužel jednoduše najít, a aby se dala určit jeho poloha. Poloha kuželu v ose x transformovaného obrázku je jednoduše přepočitatelná na azimut, kterým se musí robot vydat.

Obrázek 4.6: Původní záběr z kamery a výřez transformovaného obrázku

V původním obrázku z kamery je potřeba určit několik parametrů. Konkrétně polohu středu objektivu (souřadnice X_c a Y_c), jeho poloměr (parametr R_1) a poloměr sférického zrcadla (parametr R_2). Hodnoty těchto parametrů jsou udány v pixelech.

Pro transformaci pak stačí přepočítat polohu pixelů z polárních souřadnic na souřadnice kartézské:



Obrázek 4.7: Transformace obrazu z kamery

Šířka a výška výsledného obrázku tedy jsou

$$W_r = 2\pi R_2,$$

$$H_r = R_2 - R_1.$$

Pro každý bod $(x_r,\, y_r)$ ve výsledném obrázku pak platí

$$r = \frac{y_s}{H_r} \cdot (R_2 - R_1) + R_1,$$

$$\varphi = \frac{x_s}{W_r} \cdot 2\pi,$$

$$x_r = x_c + r \cdot \sin\varphi,$$

$$y_r = y_c + r \cdot \cos\varphi.$$

Pro transformaci snímků je použita funkce *remap* knihovny OpenCV. Tato funkce využívá dvě mapovací matice, které jsou vytvořeny po startu programu. Díky tomu může operace probíhat v krátkém čase.

4.2.2 Nalezení kuželu

Transformovaný obrázek z kamery je nejprve převeden do barevného modelu HSV, který sestává ze tří složek: Hue (barevný tón), Saturation (sytost barvy) a Value (jas). Tento model je vhodný pro rozeznávání barev, protože odděluje informaci o barevném odstínu a intenzitě - nezáleží tedy na tom, zda se kužel nachází ve stínu či nikoliv.

Následně jsou v obrázku detekovány oblasti, které odpovídají barvě kuželu. Ty jsou uloženy do nového dvoubarevného obrázku D,

$$D(j,j) = \begin{cases} 1 & (\mathrm{H(i,j)} \in [\mathrm{h_1}, \mathrm{h_2}] \wedge \mathrm{S(i,j)} \in [\mathrm{s_1}, \mathrm{s_2}] \wedge \mathrm{V(i,j)} \in [\mathrm{v_1}, \mathrm{v_2}]) \\ 0 & \mathrm{jinak} \end{cases},$$

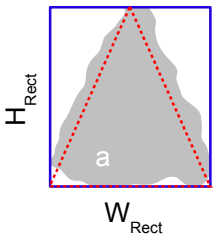
kde H, S a V jsou složky původního obrázku a h_i, s_i a v_i jsou limitní hodnoty získané kalibrací. Pro kalibraci je potřeba program *cv_node* spustit v grafickém režimu a v okně se záběrem z kamery vybrat tři body ohraničující oranžový kužel. Program pak do souboru uloží maximální a minimální hodnoty barevného tónu, sytosti a jasu vybrané oblasti. Kalibraci je vhodné provádět při změně prostředí, ve kterém se robot pohybuje.

Protože výsledný obrázek D obsahuje kromě oblasti odpovídající kuželu také další, menší oblasti, je dále provedeno jejich odfiltrování pomocí morfologických operací otevření a uzavření kruhovým elementem (obr. 4.2.2).

Obrázek 4.8: Detekované barevné oblasti před a po filtrování (invertováno)

Zbylé oblasti jsou kandidáty na kužely. Pro zpracování barevných oblastí je použita knihovna OpenCVBlobsLib [11]. Jako kužel jsou vyhodnoceny ty oblasti s dostatečně velkou plochou a, které jsou orientovány na výšku (tedy výška ohraničujícího obdélníka W_{Rect} je větší než jeho šířka H_{Rect}), a jejichž plocha se od plochy rovnostranného trojúhelníku, vepsaného do ohraničujícího obdélníka (obr. 4.9), liší nanejvýš o 20%, tedy

$$\left| \frac{a}{\frac{1}{2} \cdot W_{Rect} \cdot H_{Rect}} - 1 \right| < 0{,}2.$$



Obrázek 4.9: Klasifikace detekované barevné oblasti

Pro všechny nalezené kužely (obr. 4.10) je pak určen azimut

$$\varphi = -\frac{X_T - X_F}{W_R} \cdot 360°,$$

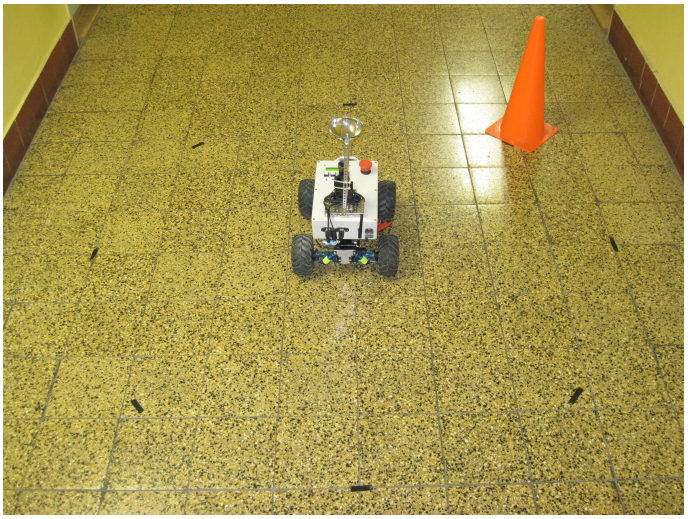
kde X_T je poloha středu obdélníka, ohraničujícího nalezenou oblast, X_F poloha přední části robotu v transformovaném obrázku a W_R šířka obrázku.

Obrázek 4.10: Detekovaný kužel a příslušný azimut (výřez)

5 Experimentální ověření

5.1 Detekce kuželu

Ověření správnosti detekce polohy kuželu bylo provedeno pro osm poloh na obvodu kružnice o poloměru 80 cm. Robot byl umístěn tak, aby se stojan s kamerou nacházel přibližně uprostřed této kružnice (obr. 5.1).



Obrázek 5.1: Nastavení experimentu

Naměřené výsledky jsou uvedeny v tabulce 5.1. Pro polohu 180° není žádný kužel detekován, je totiž zastíněn kovovým nosníkem sférického zrcadla. Výsledky jsou nepřesné a pro řízení nepoužitelné.

Skutečný azimut	0°	45°	90°	135°	180°	225°	270°	315°
Naměřený azimut	2°	47°	93°	147°	-	242°	283°	324°
Chyba	2°	2°	3°	12°	-	17°	13°	9°

Tabulka 5.1: Naměřené polohy kužele

Po změně parametrů pro transformaci obrazu (kap. 4.2.1) se naměřené výsledky výrazně zlepšily. Kromě barvy kuželu je tedy před každou jízdou vhodné znovu nastavit i transformační parametry (zejména polohu středu objektivu). Při transportu nebo manipulaci s robotem totiž může snadno dojít k posunutí kamery.

Skutečný azimut	0°	45°	90°	135°	180°	225°	270°	315°
Naměřený azimut	1°	44°	88°	133°	-	223°	269°	317°
Chyba	1°	1°	2°	2°	-	2°	1°	2°

Tabulka 5.2: Polohy kužele naměřené po kalibraci

6 Závěr

Cílem této práce bylo navrhnout systém pro navigaci mobilního robotu mezi kontrolními body a jeho funkčnost ověřit v reálném prostředí.

Pro ověření funkčnosti jsem sestrojil robotickou platformu, navrhl její senzorickou výbavu a řídicí elektroniku s jednodeskovým počítačem BeagleBone. Software robotické platformy jsem naprogramoval jako soustavu nezávislých modulů, komunikujících mezi sebou prostřednictvím sítě.

Výhodou tohoto řešení je kromě nezávislosti na konkrétním programovacím jazyce také možnost spouštět moduly na více počítačích pro jednodušší vývoj a ladění. Navrhl, naprogramoval a otestoval jsem systém pro detekci polohy kuželu v okolí robotu na základě obrazu z kamery.

Navrhl jsem také navigační systém pro plánování pohybu ve venkovním prostředí s využitím dostupných mapových podkladů, a tento systém ověřil v desktopové aplikaci. Kvůli velké koncentraci na elektroniku a software robotické platformy jsem ale tento systém nestihl plně implementovat v robotu a ověřit v reálném prostředí. Vytvořil jsem program pro rovnou jízdu na zadanou zeměpisnou souřadnici, který je základem této implementace.

Pokračováním práce tedy bude dokončení navigačního systému v robotu, konkrétně sekvenčního spouštění programu rovné jízdy pro jednotlivé vrcholy nalezené trasy, a dále vytvoření programu pro jízdu ke kuželu na základě informace o jeho poloze. Vzhledem k rozšířením pravidel soutěže RoboOrienteering bude také nutné sestrojit podavač golfových míčků, které budou rozváženy ke kuželům.

Literatura

[1] COLEY, Gerald a Robert DAY. BEAGLEBOARD.ORG. *BeagleBone Black System Reference Manual.* 2013.

[2] PETER, Corke. *Robotics, vision and control: fundamental algorithms in Matlab.* 1st ed. New York: Springer, 2011, 570 s. Springer tracts in advanced robotics. ISBN 978-3-642-20143-1.

[3] NOVÁK, Petr. *Mobilní roboty: pohony, senzory, řízení*. Vyd. 1. Praha: BEN - technická literatura, 2004, 247 s. ISBN 80-730-0141-1.

[4] BRADSKI, Gary R. *Learning OpenCV*. Sebastopol: O'Reilly, c2008, xvii, 555 s. ISBN 978-0-596-51613-0.

[5] LOCKER, Martin. *RoboOrienteering* [online]. 2010 [cit. 2014-05-18]. Dostupné z: http://robotika.vosrk.cz/roboorienteering

[6] Cape Manager and Device Tree Overlays. ANTONIOU, Pantelis a Matt PORTER. *eLinux.org: BeagleBone and the 3.8* Kernel [online]. [cit. 2014-03-11]. Dostupné z: http://elinux.org/BeagleBone_and_the_3.8_Kernel

[7] Dot. *Kogeto* [online]. 2012 [cit. 2014-05-18]. Dostupné z: http://www.kogeto.com/say-hello-to-dot

[8] BAUDIŠ, Petr. Brmbot Outdoor: Camera. *Brmlab: Hackerspace Prague* [online]. 2011 [cit. 2014-05-18]. Dostupné z: http://brmlab.cz/project/brmbot_outdoor

[9] OSM XML. *OpenStreetMap Wiki* [online]. 2010 [cit. 2014-05-18]. Dostupné z: http://wiki.openstreetmap.org/wiki/OSM_XML

[10] MARDER-EPPSTEIN, Eitan, Eric BERGER, Tully FOOTE, Brian GERKEY a Kurt KONOLIGE. The Office Marathon: Robust Navigation in an Indoor Office Environment. *International Conference on Robotics and Automation*. 2010. Dostupné z: http://wiki.ros.org/Papers/ICRA2010_Marder-Eppstein

[11] MURGIA, Saverio a Luca NARDELLI. *OpenCVBlobsLib* [online]. [cit. 2014-05-18]. Dostupné z: http://opencvblobslib.github.io/opencvblobslib/

[12] GRUDER, Keith a spol. Homepage. *AVR Libc* [online]. 2002 [cit. 2014-05-18]. Dostupné z: http://www.nongnu.org/avr-libc/

[13] THOMAS, Martin. AVRPROG compatible bootloader for ATMEL ATmega Controllers. *AVR-Projects* [online]. 2008 [cit. 2014-03-11]. Dostupné z: http://siwawi.bauing.uni-kl.de/avr_projects/#avrprog_boot

[14] HINTJENS, Pieter. *ØMQ - The Guide.* [online]. [cit. 2014-04-20]. Dostupné z: http://zguide.zeromq.org/page:all

[15] PANSENTI, LLC. *Linux-mpu9150* [online]. [cit. 2014-05-10]. Dostupné z: https://github.com/mlaurijsse/linux-mpu9150

[16] CLARK, John. BeagleBone Black: Debian Wheezy 7.0.0 Hard Float Minimal Image. *ARMhf: Linux for ARMhf devices* [online]. 2013 [cit. 2014-05-18]. Dostupné z: http://www.armhf.com/index.php/boards/beaglebone-black/

[17] HEINRICH, Adam. Konstrukce mobilního robota schopného pohybu ve venkovním prostředí. *Středoškolská odborná činnost.* 2009, 31. ročník. Dostupné z: http://dir.adamh.cz/soc/mobilni-robot.pdf

[18] HANZAL, Josef. *SIC1602A20: Komunikační protokol.* 2006. Dostupné z: http://www.snailshop.cz/index.php?controller=attachment&id_attachment=2

[7] TRACO POWER AG. *DC/DC Converters: TSR-1 Series 1A* [data sheet]. Zurich, 2010. Dostupné z: http://www.tracopower.com/products/tsr1.pdf

[19] ATMEL CORPORATION. *ATmega8A: 8-bit Atmel Microcontroller with 8KB In-System Programmable Flash* [data sheet]. San Jose, CA,

2013. Dostupné z: http://www.atmel.com/images/atmel-8159-8-bit-avr-microcontroller-atmega8a_datasheet.pdf

[20] SHARP CORPORATION. *GP2D120: Optoelectronic Device* [data sheet]. Osaka, 2006. Dostupné z: http://www.sharpsma.com/webfm_send/1205

[21] INVENSENSE INC. *MPU-9150 Product Specification: Revision 4.3* [data sheet]. Sunnyvale, CA, 2013. Dostupné z: http://www.invensense.com/mems/gyro/documents/PS-MPU-9150A-00v4_3.pdf

[22] FAIRCHILD SEMICONDUCTOR CORPORATION. *QRD1113 / QRD1114: Reflective Object Sensor* [data sheet]. 2005. Dostupné z: http://www.fairchildsemi.com/ds/QR/QRD1114.pdf

[23] ALLEGRO MICROSYSTEMS, LLC. *ACS711: Hall Effect Linear Current Sensor with Overcurrent Fault Output for <100 V Isolation Applications* [data sheet]. Worcester, Massachusetts, 2008. Dostupné z: http://www.allegromicro.com/∼/Media/Files/Datasheets/ACS711-Datasheet.ashx

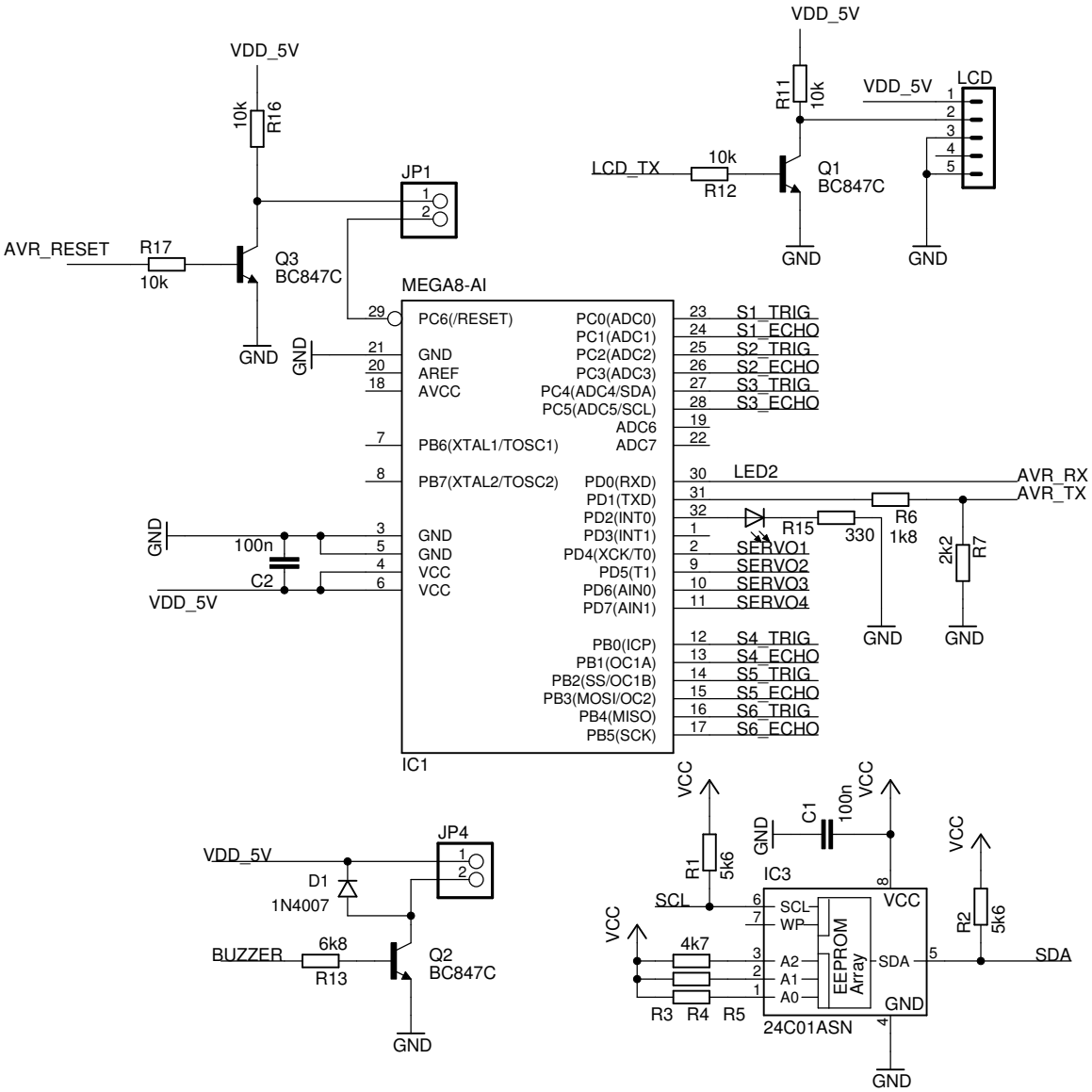
[24] U-BLOX AG. *NEO-6: u-blox 6 GPS Modules* [data sheet]. Thalwil, Switzerland, 2011. Dostupné z: https://www.u-blox.com/images/downloads/Product_Docs/NEO-6_DataSheet_(GPS.G6-HW-09005).pdf

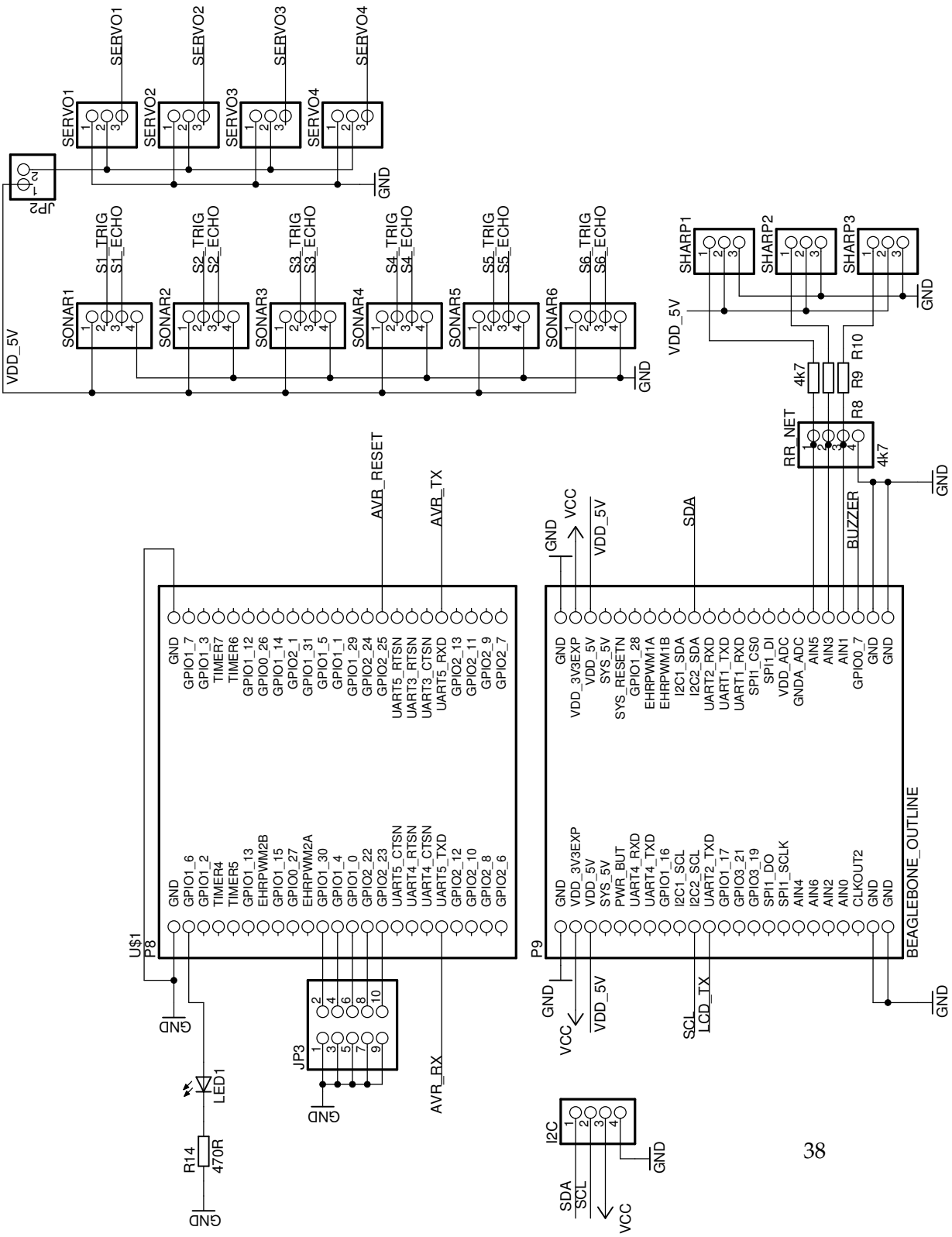
[25] GEODIS BRNO, s.r.o., Seznam.cz, a.s. a NAVTEQ. *Mapy.cz* [online]. [cit. 2014-05-18]. Dostupné z: http://www.mapy.cz

Přílohy

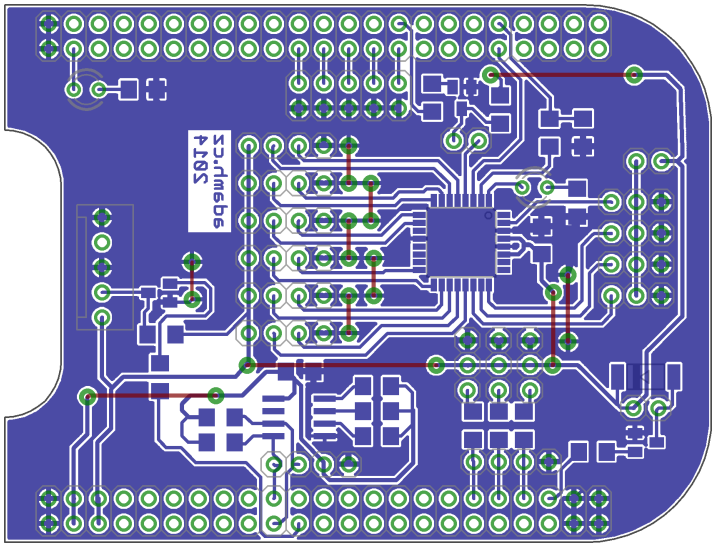
A Rozšiřující deska

A.1 Schéma

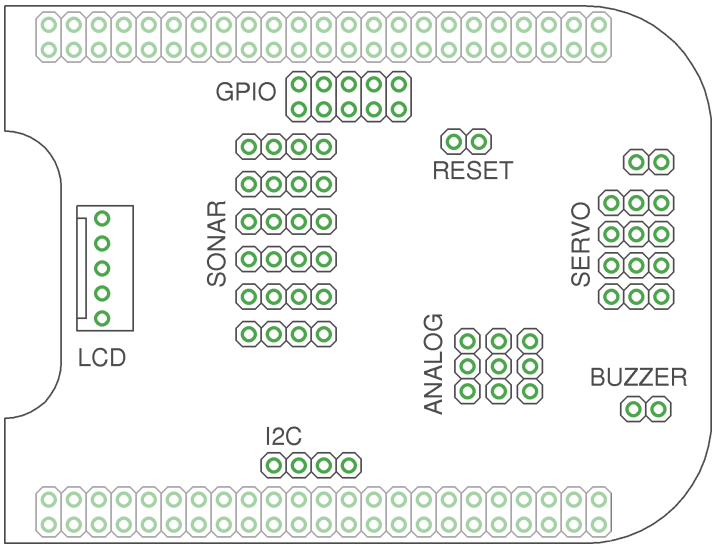



SERVO1
SERVO2
SERVO3
SERVO4
JP2
GND
S1_TRIG
S1_ECHO
S2_TRIG
S2_ECHO
S3_TRIG
S3_ECHO
S4_TRIG
S4_ECHO
S5_TRIG
S5_ECHO
S6_TRIG
S6_ECHO
SONAR1
SONAR2
SONAR3
SONAR4
SONAR5
SONAR6
VDD_5V
SHARP1
SHARP2
SHARP3
4k7
R10
R9
R8
RR_NET
BUZZER
AVR_RESET
AVR_TX
AVR_RX
U$1
P8
P9
JP3
LED1
R14
470R
I2C
SDA
SCL
VCC
LCD_TX
BEAGLEBONE_OUTLINE

A.2 Plošný spoj

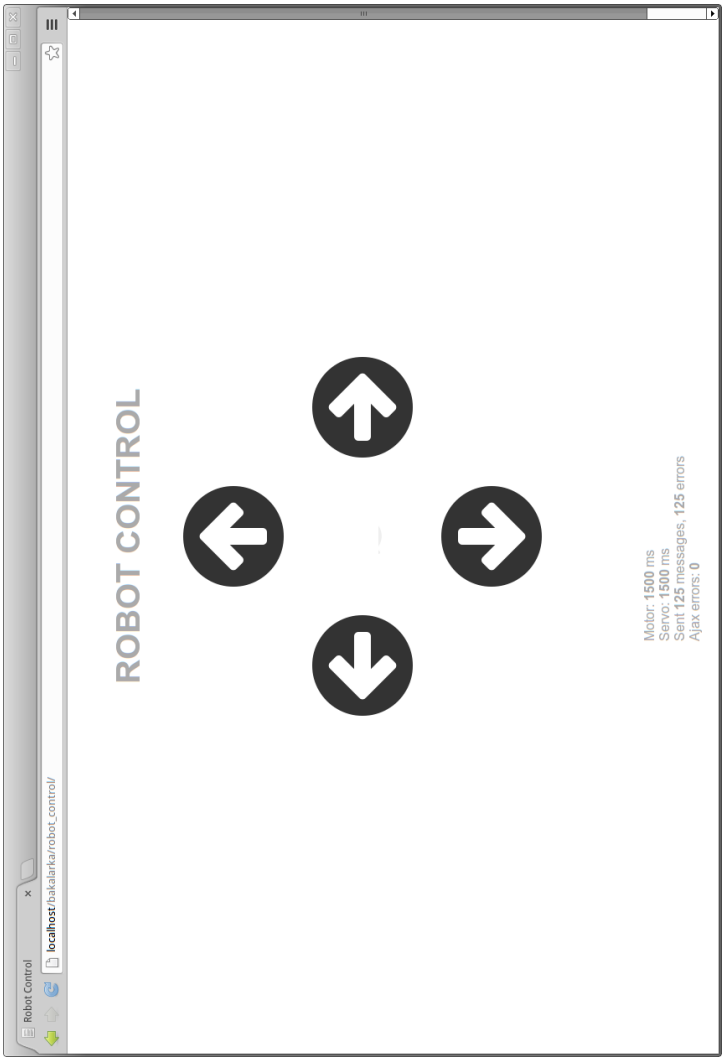


Obrázek A.1: Motiv plošného spoje ze strany součástek (zvětšeno)



Obrázek A.2: Rozložení konektorů

B Snímky obrazovky



Obrázek B.1: Webové rozhraní pro dálkové ovládání robotu

Obrázek B.2: Program *WorldEditor* pro práci s mapou

C Obsah přiloženého CD

- **src/**
  - **avr/** Zdrojové kódy programu mikrokontroléru AVR
  - **matlab/** Zdrojové kódy experimentů v Matlabu
  - **modules/** Zdrojové kódy jednotlivých modulů
  - **WorldEditor/** Zdrojové kódy programu WorldEditor
  - **www/** Zdrojové kódy webového rozhraní
  - **etc/** Další programy a skripty použité v robotu
- **pcb/** Schéma a motiv plošného spoje rozšiřující desky
- **img/**
  - **cv/** Obrázky použité v kap. 4.2
  - **foto/** Fotografie robotu
- **bp.pdf** Tato práce ve formátu PDF
- **obsah.pdf** Obsah CD