Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Vozítko založené na Raspberry Pi

Plzeň, 2016 Jan Hák

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 4.5.2016 Jan Hák,

Abstrakt

Robotic rover based on Raspberry Pi

My work contains a description of the development process of a robotic rover that can be remotely controlled using a Wi-Fi network from a PC. My work includes one program for controlling the I/O pins of Raspberry Pi 2 in C++. It also includes a second program based on Qt framework for C++, which controls trigger and read I/O pins of Raspberry Pi 2 and the movement of the rover remotely via GUI. I will acquaint you with the individual components of the rover, possible extensions, its functions and usage of programs and pitfalls of the Raspberry Pi as controller of the robotic rover.

Má práce obsahuje popis procesu výroby robotického vozítka, které může být ovládáno vzdáleně pomocí Wi-Fi sítě z počítače. Má práce obsahuje program pro ovládání vstupně-výstupních pinů počítače Raspberry Pi 2 napsaný v jazyce C++. Druhý program založený na Qt frameworku pro C++, sloužící k ovládání a čtení vstupně-výstupních pinů Raspberry Pi 2 a ovládání pohybu robotického vozítka vzdáleně pomocí grafického rozhraní. Seznámím Vás s jednotlivými komponentami vozítka, s možnými rozšířeními vozítka, s funkcemi a používáním programů a s nebezpečími použití Raspberry Pi jako ovladače pro robotické vozítko.

Obsah

1 Úvod

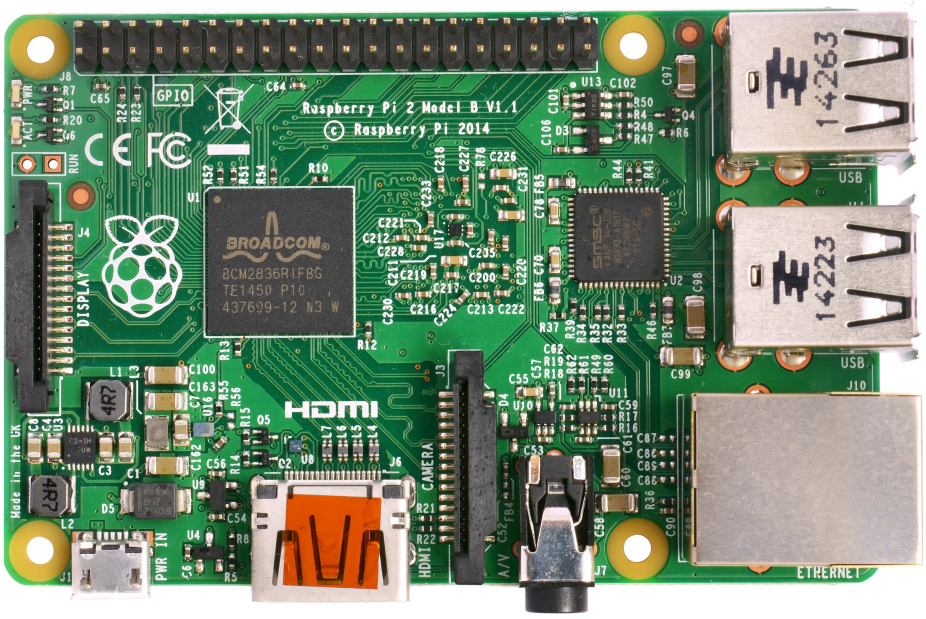
Úkolem této práce je seznámit čtenáře s počítačem Raspberry Pi a s jeho využitím pro vytvoření vozítka, které je ovládané vzdáleně, např. z jiného počítače, prostřednictvím Wi-Fi sítě. Součástí práce je volba potřebných periferií, volba protokolu pro komunikaci po síti a volba knihovny pro spínání vstupně-výstupních pinů, které slouží pro ovládání periferií připojených k Raspberry Pi. Nedílnou součástí práce je popis tvorby aplikace, díky které lze vstupně-výstupní piny vzdáleně nastavovat.

Na začátku práce je nutné zvolit podvozek pro vozítko, motory, které budou pohánět kola, a integrovaný obvod, zvaný H Bridge, díky kterému je možné měnit směr otáčení motoru. Dále je nutné pro komunikaci prostřednictvím Wi-Fi sítě přidat Wi-Fi modul, protože použitý typ Raspberry Pi 2 jím nedisponuje. Vozítko lze doplnit o další periferie, jako jsou LED diodová světla, infračervený reflexní senzor pro sledování vyznačené cesty nebo ultrazvukový senzor vzdálenosti. Námětem pro budoucí práci je např. vybavení vozítka kamerovým modulem.

V kapitole 2 jsou uvedeny informace o Raspberry Pi a možnostech jeho využití a rozšíření. V kapitole 3 zmiňuji použité periferie pro potřeby vozítka a periferie umožňující rozšíření možností vozítka a jeho aplikací. Popis sestavení vozítka a zapojení periferií naleznete v kapitole 4. V kapitole 5 je popsána implementace klienta pro Raspberry Pi a v kapitole 6 je popsána implementace severu pro PC. V 7. kapitole je popsán způsob komunikace mezi klientem a serverem. Souhrn dosažených výsledků se nachází v kapitole 8 a závěr se nachází v kapitole 9. Následují příloha A obsahující uživatelskou dokumentaci, příloha B se seznamem a cenou jednotlivých komponent pro sestavení vozítka a příloha C s grafy funkcí jednotlivých PWM signálů závislých na požadovaném úhlu pohybu vozítka.

2 Raspberry Pi

Raspberry Pi je malý počítač, někdy označovaný jako minipočítač, který pod záštitou *Raspberry Pi Foundation* vytvořila Univerzita v Cambridge pro podporu výuky programování [1].



Obr. 2.1 – Raspberry Pi 2 model B při pohledu ze shora[1].

Momentálně je k dispozici několik variant Raspberry Pi, lišících se od sebe velikostí, vestavěným procesorem, počtem pinů GPIO (Angl. *General Purpose Input/Output*) portu, počtem USB portů nebo přítomností Ethernetového konektoru.

Pro tvorbu projektu jsem zvolil počítač Raspberry Pi 2 model B. Ten obsahuje konektory pro display a kameru, 4 porty USB 2.0, GPIO port o šířce 40 pinů, Ethernetový konektor a HDMI port, viz Obr. 2.1. Je osazen čtyř-jádrovým procesorem ARM Cortex Quad A7, taktovaným na frekvenci 900 MHz [3]. Předností tohoto modelu je právě čtyř-jádrový procesor, díky kterému se dá u aplikace očekávat lepší odezva, a to díky rozdělení běžících procesů mezi více jader, což je vhodné právě pro potřeby vozítka. Předností je také výkonnější ALU[2], která, oproti starším typům, zvládá lépe operace s desetinnými čísly.

1 Zdroj: https://commons.wikimedia.org/wiki/File:Raspberry_Pi_2_Model_B_v1.1_top_new_(bg_cut_out).jpg
2 Angl. *Arithmetic Logic Unit* - součást procesoru provádějící všechny matematické a logické operace.

Pro Raspberry Pi je nutné zvolit operační systém (viz část 2.1) a způsob napájení (viz část 2.2). Dále popisuji možné způsoby využití Raspberry Pi (viz část 2.3), displej a kameru určené pro Raspberry Pi (viz část 2.4) a GPIO port pro připojení dalších periferií (viz část 2.5).

2.1 Operační systém

Raspberry Pi operační systém načítá z paměťové karty. Starší modely používaly klasickou SD kartu, ale novější již využívají micro-SD kartu. Tu je vhodné volit s označením *Class 10* a vyšším. Označení *Class* určuje rychlost a počty zápisů, které se kartě podaří vykonat, než se její paměťové buňky poničí. Zápisů je důsledkem běhu operačního systému prováděno velké množství [1].

Převážná většina operačních systémů pro Raspberry Pi je založena na operačním systému Linux [3]. Pro Raspberry Pi se vyvíjí Linuxová distribuce Raspbian, což je odnož operačního systému Debian. Dalším zajímavým operačním systémem je RiscOS, který je vytvářen pro ARM[3] architekturu procesorů. Díky modernější instrukční sadě ARMv7 procesoru ARM Cortex A7 je možné na Raspberry Pi 2 spustit i Windows 10 IoT, což je operační systém založený na Windows 10, určený pro internet věcí [13].

Při využití Raspberry Pi pro vytvoření vozítka je možné nahrát do Linuxového jádra modul pro nastavení systému do režimu reálného času (Angl. *Real-time Patch*). Systém se pak snaží vyhovět nově vzniklým požadavkům ve stanoveném čase.

2.2 Napájení

Běžný způsob napájení Raspberry Pi je pomocí mini-USB konektoru, který poskytuje napětí 5 V, ale množství proudu závisí na kvalitě zdroje. Zdroj, který Raspberry Pi udává jako doporučený, je schopný poskytnout proud až 1 A. Já zvolil zdroj schopný poskytnout proud až 1,2 A, převážně kvůli možnosti připojit periferie přímo k Raspberry Pi (bez externího zdroje). Další možností je napájet Raspberry Pi skrz 5 V piny na GPIO portu. Tyto piny nalezneme na kraji GPIO portu a jsou označovány jako pin 2 a pin 4, viz dále Obr. 2.2. Tento způsob napájení je ale nedoporučovaný. Je důležité, aby hladina napájení byla stabilní a to

3 Označení architektury procesorů vyvíjených společností *ARM Holdings.*

v blízkém okolí hladiny napětí 5 V. Při přívodu většího napětí dojde k trvalému poškození desky [15].

Pokud by se podařilo externí napájení vyřešit využitím kvalitní baterie a regulátoru napětí, tak je možné navrhnout vozítko tak, aby bylo naprosto bezdrátové, tedy bez datového či napájecího kabelu. Toto řešení je však poměrně finančně náročné. V rámci finančních úspor jsme se proto rozhodli, že budeme pro testovací účely motory napájet čtyřmi 1,2 V tužkovými bateriemi a Raspberry Pi bude napájet mini-USB konektor připojený do sítě elektrického rozvodu.

Napájení periferií skrz USB port je omezeno kvalitou zdroje. Na to je nutné dbát při připojení většího počtu periferií s větším odběrem proudu, jako jsou externí USB harddisk nebo webová kamera. Ty mohou v důsledku nedostatku energie pracovat chybně. V takovém případě je vhodné použít USB Hub s externím napájením.

2.3 Využití

Prvotní idea využití Raspberry Pi byla pro výúkové účely, přičemž snahou bylo umožnit studentům ovládání periferií a tím je dovést k programování. Proto existuje mnoho knížek, které se věnují tvorbě specifických projektů nebo zpracování dat z různých senzorů [1][14]. Většina těchto návodů existuje pro programovací jazyk Python [2]. Kromě využití Raspberry Pi pro výuku či robotiku, popř. jako běžný desktopový počítač, bývá dalším využitím tvorba multimediálního centra [14]. Po připojení Raspberry Pi k televizi nebo monitoru pomocí HDMI portu lze z televize vytvořit „chytrou televizi“ s připojením k internetu a s možností přehrávat filmy z externího harddisku. Malá velikost a nízká energetická náročnost Raspberry Pi vedou k dalšímu zajímavému využití, a to spojení více Raspberry Pi pomocí internetového switche. To umožňuje vytvoření serveru s balancováním přístupů nebo superpočítače, tzv. clusteru, pro distribuované výpočty využívající rozhraní pro zasílání zpráv (MPI). Nevýhodou je pouze nutnost nahrát kopii operačního systému na paměťovou kartu každého uzlu clusteru, tedy na každé Raspberry Pi, protože to nedokáže spustit operační systém ze sítě [12].

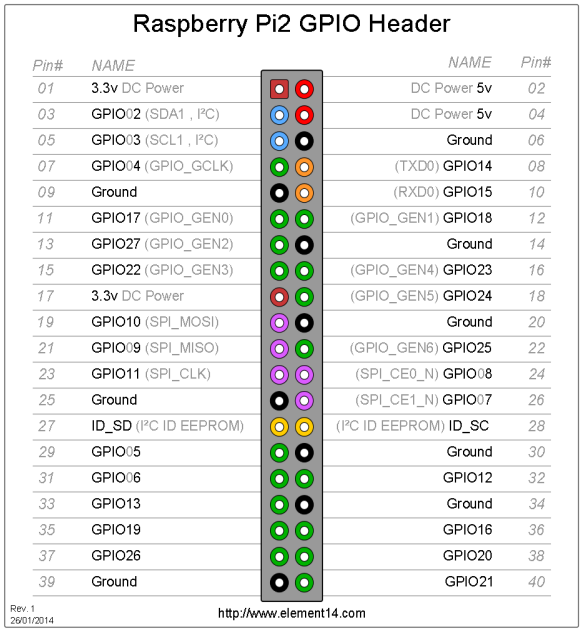
2.4 Displej a kamera

Ačkoli pro připojení displeje lze využít HDMI port a pro připojení webové kamery USB port, Raspberry Pi je navíc osazeno 15pinovými sériovými konektory DSI (*Display Serial Interface*) a CSI (*Camera Serial Interface*) sloužícími pro připojení uzpůsobeného hardware. Je doporučováno použití právě těchto konektorů místo USB nebo HDMI konektorů, což je výhodné i kvůli úspoře omezeného počtu USB portů na desce.

2.5 GPIO port

GPIO port jsou dvě řady pinů, viz Obr. 2.2, sloužících k napájení a ovládání periferií. Některé z pinů, označované jako GPIO, lze softwarově použít jako vstup či výstup. GPIO port má u Raspberry Pi 26 nebo 40 pinů, což je závislé na zvoleném modelu. Port se 40 piny je rozšířením portu 26pinového. Periferie určené pro GPIO port s 26 piny jsou tedy kompatibilní i se širším portem. 26pinový port existuje ve dvou revizích, které mění číslování GPIO pinů. Jeho účel však zůstal nezměněn.

GPIO port obsahuje piny s 5 V nebo 3,3 V stejnosměrným napětím, piny pro uzemnění a sadu GPIO pinů přizpůsobených pro napětí 3,3 V. Pokud bychom hranici napětí 3,3 V překročili, vedlo by to k nevratnému poničení Raspberry Pi [1].

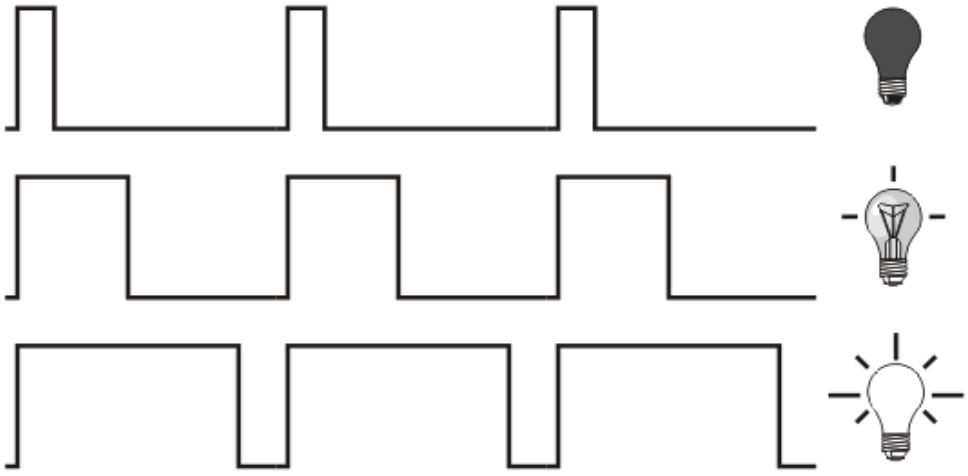


Obr. 2.2 – Schéma pinů GPIO portu Raspberry Pi 2 model B[4].

Piny na schématu z Obr. 2.2 označené jako GPIO jsou vstupně-výstupní piny Raspberry Pi. Tyto piny lze nastavovat přímým přístupem do paměti a změnou bitové mapy sloužící k nastavování pinů. Tím lze nastavit výstup na hladinu napětí 0 V nebo 3,3 V, popř. nastavit pin jako vstup a číst z něj hladinu napětí [1]. Raspberry Pi s 26pinovým GPIO portem má 17 GPIO pinů, 40pinový port obsahuje 26 GPIO pinů.

2.5.1 Pulzně šířková modulace

Pulzně šířkovou modulaci lze využít například k redukci intenzity záření žárovky (viz Obr. 2.3) nebo LED diody, k redukci rychlosti otáčení motoru nebo pro ovládání rychlosti otáčení či úhlu servo motoru [4]. Pulzně šířkově modulovaný signál (PWM signál) je digitální signál, jehož jedna perioda se skládá z doby vyšší a doby nižší hladiny napětí (viz Obr. 2.3). Na základě doby, kdy setrvává signál ve vyšší hladině napětí, a celkové doby periody získáme poměr zvaný střída (Angl. *Duty cycle*). Hladiny napětí ani délka periody nejsou obecně definovány, je však nutné tyto hodnoty vždy zvolit konstantní.



Obr. 2.3 – Ukázka závislost intenzity záření žárovky na PWM signálu[5].

Raspberry Pi má 1 hardwarový modul pro generování PWM signálu. Jeho výstup

4 Zdroj: http://www.element14.com/community/docs/DOC-73950/l/raspberry-pi-2-model-b-gpio-40-pin-block-pinout

5 Zdroj: http://www.mikroe.com/chapters/view/6/chapter-5-ccp-modules/

je připojen na pin 12, který je na Raspberry Pi 2 označován jako GPIO18 (viz Obr. 2.2) [2]. Tento signál je však možné generovat softwarově i na ostatních pinech nebo lze k Raspberry Pi připojit modul, který bude PWM signál generovat, a Raspberry Pi jej bude pouze ovládat pomocí GPIO pinů (přímo nebo pomocí některého z rozhraní popsaných v části 2.5.2).

Existují různé knihovny pro generování PWM signálu na kterémkoli GPIO pinu. Je ale vhodné použít knihovnu, která pro nastavování hodnot pinů využívá dynamický přístup k paměti (DMA). Tento přístup téměř nezatěžuje CPU. I přes to softwarově generovaný PWM signál není vhodné používat u periferií, které jsou citlivé na přesnost signálu [6].

2.5.2 Speciální vlastnosti některých GPIO pinů

Některé z pinů je možné využít jako piny I^2C sběrnice, SPI sběrnice nebo jako UART sloužící pro komunikaci s terminálem pomocí sériové linky.

I^2C (Angl. čteme *I squared C*) sběrnice tvořená 2 vodiči sloužící k připojení některých specifických periferií k základní desce. K I^2C sběrnici je možné se připojit piny 3 a 5. Využitím této sběrnice může být například přístup k datům na paměti EEPROM nebo přístup k analogově-digitálním převodníkům. Zařízení připojená k I^2C sběrnici jsou rozdělena na *master* a *slave*. *Master* zařízení generuje na jednom z vodičů, označovaném jako SCL, hodinový signál a po druhém vodiči, označovaném jako SDA, si vyměňuje data se zařízeními připojenými na sběrnici. Těch může být dle specifikace až 128. Data přijímají všechna zařízení, pro které jsou data určena, což udává adresa vysílaná po SDA [5].

Serial Peripheral Interface (*SPI*) je rozhraní tvořené minimálně 4 vodiči sloužící ke komunikaci řídícího procesoru s periferiemi připojenými ke společné sběrnici. Použití tohoto rozhraní je podobné I^2C (dokáže například komunikovat s EEPROM pamětí). Pro každé *Slave* zařízení je potřeba na *Master* zařízení vyhradit GPIO pin označovaný jako *Slave select* (*SSEL*, někdy označovány jako *CE*) a v jednu chvíli může být aktivní pouze jeden *SSEL* pin. Na Raspberry Pi se nachází 2 *SSEL* nacházející se na GPIO pinech 24 a 26. Další 3 vodiče jsou společné pro všechna zařízení. Vodič označovaný jako *SCLK* (nacházející se na pinu 23) je vyhraněn pro hodinový signál generovaný *Master* zařízením a zbylé dva

vodiče jsou obousměrné komunikační kanály označované *MISO* (Angl. *Master In – Slave Out*) a *MOSI* (Angl. *Master Out – Slave In*) pro přenos dat. *MISO* se nachází na pinu 21 a *MOSI* na pinu 19. Pokud má *Slave* zařízení na *SSEL* pinu logickou 0, tak naslouchá na sběrnici, popř. vysílá data, v opačném případě data přenášená po sběrnici ignoruje [6].

Piny na GPIO portu označované jako *TXD0* a *RXD0* (viz Obr. 2.2), souhrnně nazývané jako *UART*, slouží pro sériovou komunikaci s terminálem Raspberry Pi z počítače. Nachází se na pinech 8 a 9. Toto rozhraní je možné využít, i když zařízení není připojeno k internetové síti, a jeho. Je to tudíž jedna z možností prvotního nastavení operačního systému běžícího na Raspberry Pi. Jedná se o asynchronní blokový přenos. Je proto nutné nastavit stejný Baud přenosu (počet „tiků“ hodinového signálu za sekundu) na obou zařízeních.

Žádná z verzí Raspberry Pi nedisponuje analogově-digitálním, popř. digitálně-analogovým převodníkem. Není tudíž možné jednoduše pracovat s analogovým signálem [7].

3 Periferie vozítka

Ke konektorům Raspberry Pi, zmíněným ve 2. kapitole, je možné připojovat různé elektronické součástky, tzv. periferie. Pro potřeby vozítka je nutné zvolit podvozek, na kterém budou periferie umístěny (viz část 3.1). Dále jsem zvolil 2 stejnosměrné motory (viz část 3.2), H Bridge (viz část 3.3) a Wi-Fi modul (viz část 3.4). K vozítku lze dále připojit senzor vzdálenosti (viz část 3.5), kameru (viz část 3.6) i další periferie (viz část 3.7). Seznam a přibližné ceny zvolených periferií vozítka je možno nalézt v příloze B.

3.1 Podvozek

Podvozek je možné koupit, popř. vytisknout na 3D tiskárně, pro kterou existuje velké množství volně dostupných modelů[6]. Důležitým faktorem při výběru je počet a typ motorů potřebných pro pohon vozítka. Ten ovlivňuje cenu motorů a energetickou náročnost.

Já zvolil podvozek s 2 koly v náhonu a jedním otočným kolečkem pro udržení stability. Kola jsou poháněna 2 stejnosměrnými elektromotory[7].

3.2 Elektromotor

Jsou tři běžně dostupné typy elektromotorů určené pro neprofesionální, tzv. *hobby*, využití a vhodné pro potřeby malého vozítka. Jsou to servomotory pro kontinuální pohyb[8], motory pro stejnosměrné napětí a motory pro střídavé napětí.

Zvolil jsem motory, které byly součástí podvozku, tedy motory pro 3 – 6 V stejnosměrného napětí. Pokud do tohoto motoru přivedeme vhodné napětí, motor se začne otáčet. Rychlost otáčení lze regulovat PWM signálem (viz část 2.5.1). Pro změnu směru otáčení je nutné otočit polaritu napětí. K tomu slouží součástka nazývaná H Bridge, viz část 3.3.

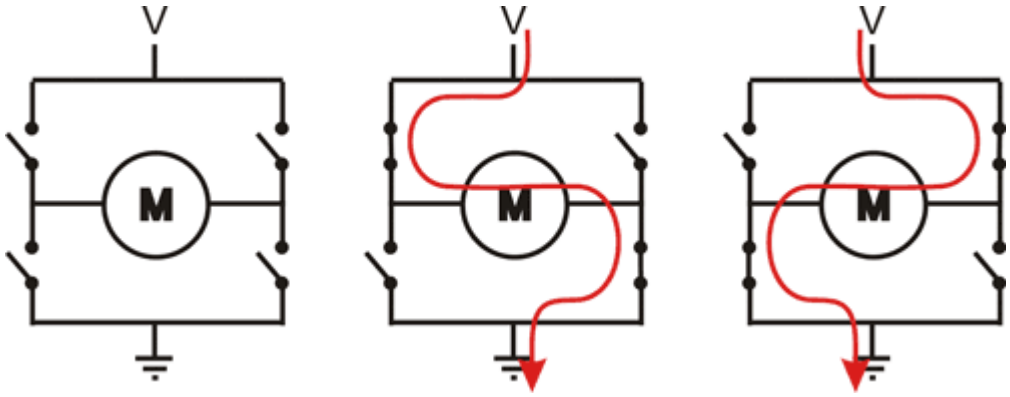
I přes možnost napájet motor 3,3 V napětím, které poskytují GPIO piny, není vhodné napájet motory přímo z Raspberry Pi. Motory a jiné energeticky náročné periferie, které mají

6 Modely podvozků je možné najít například zde: http://www.thingiverse.com/
7 Zakoupený balík s podvozkem a motory zde: http://www.rlx.sk/sk/motor-driver/2968-2wd-mobile-platform-kit-er-rkt00100m.html
8 Motor, kterému lze nastavit přesnou rychlost otáčení osy.

velký odběr proudu, by mohly Raspberry Pi nevratně poničit.

3.3 H Bridge

H Bridge je elektronická součástka sloužící k otáčení polarity napětí. Pro potřeby vozítka dokáže otočením polarity změnit směr otáčení stejnosměrných motorů (viz Obr. 2.4). Dále je tato součástka vhodná, pokud má řídící signál jiné (většinou menší) napětí, než je definováno jako standardní napětí pro napájení motoru. V mém případě ovládám napětím 3,3 V motor napájený napětím 6 V. H Bridge má však v elektrotechnice širší využití. Používá se např. v některých digitálně-analogových převodnících, jako ovladač složitějších motorů (jako jsou krokové motory) nebo v silnoproudé elektrotechnice.



Obr. 2.4 – Princip H Bridge pro otáčení polarity napětí[9].

Pro ovládání 2 motorů je možné zvolit elektronickou součástku zvanou *Quadruple Half-H Bridge*, což je integrovaný obvod obsahující 2 H Bridge, které zapojením mohou tvořit 4 poloviční H Bridge. To umožňuje spínat 4 stejnosměrné motory jedním směrem, 2 stejnosměrné motory oběma směry nebo jeden krokový motor. Tento integrovaný obvod jsem zakoupil společně s deskou *RTK Motor Controller Board Kit for Raspberry Pi*, která je přizpůsobena pro Raspberry Pi. Obsahuje patici pro integrovaný obvod H Bridge, zdířky sloužící k připojení drátů pro napájení motorů, přívod napětí z externího zdroje napětí a patici pro upevnění desky na GPIO port Raspberry Pi.

9 Zdroj: http://www.dr-iguana.com/prj_marchbreakrobot2012/page2.html

Při práci s deskou je nutné striktně dodržet značení kladného a záporného pólu pro napájení motorů externím napájením. V případě, že bychom póly otočili, by se přes uzemnění mohl dostat proud do připojeného Raspberry Pi a došlo by k jeho nevratnému poškození. V případě drátů vedoucích z desky k motorům na polaritě napětí nezáleží. Pokud používáte již hotový program pro ovládání vozítka a jedno kolo (popř. obě kola) se točí opačným směrem než očekáváte, tak stačí prohodit dráty pro kladný a záporný pól motoru.

3.4 Wi-Fi modul

Pro komunikaci Raspberry Pi 2 po Wi-Fi síti je nutné přikoupit Wi-Fi modul. S Raspberry Pi jsou kompatibilní moduly s USB nebo RJ-45 konektorem (standard dnešního internetového kabelu). RJ-45 konektor nemá pin pro napájení externích zařízení, proto je potřeba jejich napájení vyřešit dalším vodičem nebo připojením do elektrické sítě. Proto se pro potřeby vozítka více hodí připojení přes USB, které standardně poskytuje 5 V napětí.

Zvolil jsem modul *Edimax Wireless Nano*[10], který dokáže komunikovat na Wi-Fi sítích standardu IEEE 802.11b/g/n a je podporován operačním systémem Raspbian (linuxovou distribucí vytvořenou pro Raspberry Pi).

3.5 Senzor vzdálenosti

Senzor vzdálenosti je elektronická součástka měřící vzdálenost na základě některých fyzikálních vlastností. Existuje několik typů, z nichž nejdostupnější pro použití vozítka jsou ultrazvukový senzor vzdálenosti a infračervený senzor vzdálenosti.

Pro tento projekt jsem zvolil ultrazvukový senzor. Ten má 4 piny sloužící k ovládání, reproduktor a mikrofon. Jeden pin je spoušť (z Angl. *Trigger*), druhý je ohlas (z Angl. *Echo*) a poslední dva piny slouží k napájení periferie. Když na pin spouště přivedeme krátký impulz napětí, tak reproduktor přemění elektrický impulz na ultrazvukový impulz. Ten se šíří prostředím a po odrazu o překážku dorazí zpět k senzoru, kde ho zachytí mikrofon, který vyšle elektrický impulz do pinu ohlasu. Z času zpoždění mezi vysláním a přijmutím ultrazvukového signálu lze vypočítat vzdálenost překážky. Vzorec, který neuvažuje změny vlastností prostředí,

10 Specifikace: http://rpishop.cz/raspberry-pi-prislusenstvi/100-wi-fi-usb-adapter-usb-edimax-wireless-nano-.html

ve kterém se senzor nachází, jako např. vlhkost vzduchu, je vzorec (1), kde *x* je vzdálenost od překážky v centimetrech, která je rovna polovině doby *t* mezi vysláním a přijmutím signálu (protože se jedná o cestu s odrazem), vynásobené konstantou 1/29, což je přibližná rychlost zvuku v cm za mikrosekundu [9]. Zabývat se změnou rychlosti zvuku vlivem prostředí není účelem této práce. Další informace lze nalézt například v [8].

$$x = \frac{t}{2}\frac{1}{29}[cm;\mu s] \qquad (1)$$

3.6 CSI Kamera

Přestože práce s obrazem není součástí mé bakalářské práce, další periferie, o kterou jsem vozítko rozšířil, je CSI kamera pro Raspberry Pi. Ta využívá CSI rozhraní, viz část 2.4, a proto, oproti USB kameře, dokáže přenášet větší množství dat. Data navíc nejsou procesoru předávána po USB sběrnici. USB kamery sběrnici vytěžují, což může být znatelné například při současném použití USB Wi-Fi modulu, protože v případě odesílání obrazu z USB webové kamery po síti dochází k obousměrnému přesunu videa po USB sběrnici. Zvolená kamera dokáže pořídit snímek o rozlišení 5 megapixelů nebo video o rozlišení 1080p (počet prokládaných řádků obrazu) o frekvenci 30 snímků za vteřinu[11]. Poskytuje obraz v nekomprimovaném (RAW) formátu a dokáže přes snímky aplikovat různé filtry, jako jsou černobílý obraz, sépiový odstín apod.

3.7 Další možná rozšíření vozítka

Vozítko se dá rozšiřovat o další senzory dle toho, pro jaké užití je určeno. Existuje mnoho způsobů, jak vozítko rozšířit a mnoho využití vozítka:

- Přidáme-li na předek vozítka řady infračervených reflexních senzorů směřujících kolmo dolů k povrchu, po kterém se vozítko pohybuje, a bílou LED diodu pro získání dostatečného osvětlení, tak můžeme vozítko přizpůsobit sledování čáry na podlaze, popř. řešení bludiště.
- Zakoupený podvozek umožňuje přidání malých magnetů na osu kola. Pokud do jejich

11 Specifikace: http://www.rlx.sk/sk/raspberry-pi/1450-rpi-camera-board-raspberry-pi-camera-board-5mp.html

blízkosti přidáme senzor magnetického pole, tak můžeme zjistit, zda se kola točí a jakou rychlostí.

- Na vozítko je připojena kamera. Tu lze použít k rozšíření projektu například o rozpoznávání obrazu umožňující autonomní pohyb. Také je možné obraz posílat uživateli ovládajícímu vozítko, aby mohl vozítko řídit bez nutnosti přímého dohledu.
- Dále existují periferie s detektorem kouře a jedovatých plynů, GPS modulem pro určení aktuální polohy, senzorem seismických otřesů nebo čtečkou NFC, popř. RFID tagů [11].

4 Stavba vozítka

Pro stavbu vozítka je nutné obstarat si podvozek, pohon, zdroj napětí, Wi-Fi USB dongle[12] (pokud nepoužíváte Raspberry Pi 3) a počítač (popř. mikrokontrolér) pro ovládání elektronických součástek (viz část 4.1). Další periferie jsou volitelné. Pro stavbu vozítka (viz část 4.2) je též vhodné být vybaven multimetrem, cínovou pájkou (včetně cínu a kalafuny) a zhruba 20cm dlouhými dráty pro spojení jednotlivých komponent. Pro snížení napěťové hladiny mezi některými periferiemi a GPIO piny Raspberry Pi (viz část 4.3) bude potřeba sestrojit dělič napětí.

4.1 Získání potřebných součástí

Podvozek lze vytisknout na 3D tiskárně, nebo je možné zakoupit stavebnici. Je potřeba, aby bylo možné na podvozek umístit všechny potřebné komponenty jako jsou baterie a Raspberry Pi (viz část 3.1). Pro potřeby vozítka jsem podvozek zakoupil společně s náhonem a spínačem pro přerušení přívodu napětí pro motory[13]. Náhon podvozku je tvořen 2 stejnosměrnými 3 - 6 V motory, které napájíme 4,8 V. Motory lze zvolit i jiné, například motory pro střídavé napětí nebo servomotory pro kontinuální pohyb (viz část 3.2). Uvažuji však použití stejnosměrných 3 - 6 V motorů. Při použití jiného druhu motoru je potřeba zapojit vlastní iniciativu pro sestavení vozítka a navrhnout vlastní napájení motorů.

Pro možnost pohybu zpět u stejnosměrných motorů je potřeba zakoupit H Bridge. Ten dokáže otočit polaritu napětí přicházejícího na motor, a tím donutit motor otáčet se opačným směrem (viz část 3.3). Často slouží i jako napěťový most mezi 2 napěťovými hladinami (5 V pro motory a 3,3 V pro GPIO piny Raspberry Pi). Pro potřeby vozítka jsem zvolil H Bridge na plošném spoji vytvořeném přímo pro potřeby Raspberry Pi 2[14]. Ten obsahuje 26 pinovou patici (26 pinovou pro zpětnou kompatibilitu se všemi dosavadními typy Raspberry Pi), kterou lze připojit přímo na GPIO port Raspberry Pi. To usnadní zapojení (vyhneme se tak pájení) a zvyšuje modularitu celého systému.

12 Připojitelný Wi-Fi přijímač – vysílač, který lze k zařízení připojit např. pomocí USB.
13 Zakoupený balík s podvozkem a motory zde: http://www.rlx.sk/sk/motor-driver/2968-2wd-mobile-platform-kit-er-rkt00100m.html
14 K dostání zde: https://www.adafruit.com/products/1687

Dále je nutné zvolit vhodný zdroj napájení. Pro cenovou dostupnost jsem zvolil 2 zdroje napětí. Běžný zdroj napětí pro Raspberry Pi (přes mini-USB) a 4 tužkové 1,2 V baterie (akumulátory) zapojené do série pro napájení motorů. Zapojením baterií do série získáme napětí 4,8 V, které je pro potřeby motorů dostatečné. V případě, že bychom chtěli tyto zdroje sjednotit, je nutné zakoupit 5 V regulátor napětí a jako zdroj napětí zvolit kvalitní baterii, která je schopná dodat dostatečný proud pro napájení Raspberry Pi i motorů (např. Li-Pol akumulátor) a případných periferií.

Poslední, neméně důležitá část vozítka, je samotné Raspberry Pi. V projektu využívám Raspberry Pi 2. Je možné použít i jiný typ Raspberry Pi, avšak implementace programu počítá s použitím Raspberry Pi s 40 pinovou paticí GPIO portu (momentálně to jsou verze 2 a 3). Program je napsán obecně pro kterékoli Raspberry Pi, avšak počet pinů je nutné změnit přímo v kódu přepsáním definované proměnné udávající maximální počet GPIO pinů. Součástí kódu není procedura, která by dokázala rozpoznat různé typy Raspberry Pi, protože jsem měl pouze jediný typ Raspberry Pi pro testování. Pro Raspberry Pi je navíc vhodné obstarat krabičku, kterou lze přizpůsobit pro přimontování Raspberry Pi k podvozku. Krabičku lze zakoupit nebo vytisknout na 3D tiskárně[15].

4.2 Sestavení vozítka

Při sestavování podvozku postupujte dle přiloženého návodu. Sestavení každého podvozku se může lišit. K podvozku přimontujeme držák na baterie, spínač zdroje energie a Raspberry Pi propojené s ovladačem motorů (viz část 4.1). Je vhodné kontrolovat, zda po namontování součástí k podvozku nebude nic překážet volnému otáčení kol. Setkal jsem se s problémem, kdy připojený USB Wi-Fi dongle drhl o pravé kolo vozítka. Proto jsem byl nucen prohodit pozici držáku na baterie a Raspberry Pi a podvozek přizpůsobit (vyvrtat další díry pro šrouby).

Zdroj napětí pro napájení motorů připájíme vodiči ke spínači a z něj vyvedeme vodiče do ovladače motorů a do zdířek určených pro napájení desky *RTK Motor Controller for Raspberry Pi*. Je nutné sledovat polaritu napájení vedoucího na desku, abychom předešli zničení Raspberry Pi, viz část 3.3. Dále připájíme vodiče ke každému z motorů a přivedeme

15 Modely např. zde: www.thingverse.com/search?q=raspberry+pi+case&sa=

je do zdířek ovladače motorů určených pro napájení motorů. Zde na polaritě napětí nezáleží. Motory jsou konstruovány tak, aby se do nich dalo připojit libovolně polarizované napětí 5 V. Napětí mění pouze směr otáčení osy motoru. Na tomto principu H Bridge mění směr otáčení kol vozítka.

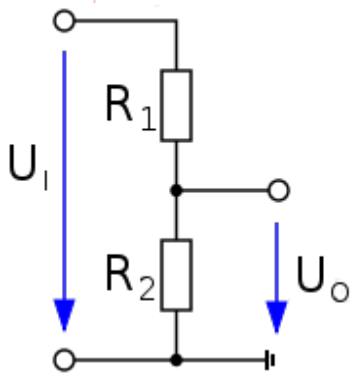
4.3 Připojení dodatečných periferií

Periferie ovládané GPIO piny připojujeme přímo na piny prostupující destičkou ovladače motorů z Raspberry Pi. Není vhodné periferie připojovat na piny, které využívá ovladač motorů pro ovládání kol. V případě desky, kterou jsem v projektu zvolil já, se jedná o GPIO piny 0, 1, 3 a 4.

Dále je vhodné zkontrolovat napětí, které vstupuje z periferií do Raspberry Pi. To nesmí přesáhnout hodnotu 3,3 V (viz část 2.5). V případě, že bychom měli součástku, která vrací vyšší napětí, je možné zkonstruovat na jejím výstupu tzv. dělič napětí (viz část 4.3.1).

4.3.1 Dělič napětí

Dělič napětí získáme zapojením dvou odporů R_1 a R_2 tak, abychom na výstupu získali požadované napětí. V případě vstupů Raspberry Pi požadujeme 3,3 V. Potřebujeme tedy zvolit odpory tak, abychom napětí U_I vstupující do děliče napětí rozdělili na napětí $U_O \leq 3{,}3\ [V]$ a zbytkové napětí U', které přivedeme na zemnící vodič, viz Obr. 4.1.



Obr 4.1 – Schéma děliče napětí.

Samotný výpočet vhodných odporů popisuje vzorec *(2)*, který využívá značení z Obr. 4.1. Je potřeba zvolit jeden z odporů jako parametr, zadat vstupní napětí U_I, požadované výstupní napětí U_O a vypočítat zbylý odpor [17]. Pro výpočet odporu R_2 získáme vztah *(3)*.

$$U_O = U_I \cdot \frac{R_2}{R_1 + R_2} \qquad (2)$$

$$R_2 = R_1 \frac{U_O}{U_I - U_O} \qquad (3)$$

Pokud bychom tedy chtěli připojit např. senzor vzdálenosti založený na TTL[16], tak musíme jeho 5 V výstup *Echo* (viz část 3.5) připojit k Raspberry Pi přes dělič napětí. Není vhodné volit odpory tak, aby bylo vstupní napětí přesně 3,3 V. Hrozilo by zničení Raspberry Pi při nečekaném výkyvu napětí. Pro jistotu je lepší volit napětí nižší, které Raspberry Pi také vyhodnotí jako hodnotu 1, a navíc nehrozí zničení samotné desky. Pro R_2 tedy zvolíme odpor, který bude v blízkém okolí vypočtené hodnoty, ale pokud možno vyšší, abychom náhodou nepřekročili zvolenou hranici výstupního napětí. Též je vhodné hlídat si dolní hranici napětí, které Raspberry Pi detekuje jako logickou 1. Tato hranice napětí je určena na 1,8 V, ale tato hodnota napětí není vždy garantována a může pohybovat mezi 0,8 a 2 V. Proto je vhodné se řídit horní hranicí 2 V. Vlastnosti děliče se mění dle jeho zatížení. Při větším zatížení napětí na výstupu klesá [16].

Pro Raspberry Pi je vhodné volit dvojice odporů, které při nezatíženém děliči napětí na výstup přivedou napětí přibližně 3 V. Takovou dvojici představují např. odpory $R_1 = 330\ [\Omega]$ a $R_2 = 470\ [\Omega]$.

16 Digitální integrovaný obvod založený na tranzistorově-tranzistorové logice (Angl. *Transistor-transistor logic*) napájený 5 V.

5 Aplikace pro Raspberry Pi

Aplikace pro Raspberry Pi je složen z části zajišťující komunikaci se serverem po síti a z části zajišťující o objektový přístup k GPIO pinům. Přiložené DVD obsahuje zdrojové soubory aplikace a instalační *Bash*[17] skript který slouží k instalaci potřebných komponent pro kompilaci programu a vytvoření potřebných konfiguračních souborů v souborovém systému. Program se kompiluje z následujících tříd a hlavičkových souborů:

- **ConfigLoader** – Třída starající se o načtení konfiguračního souboru uloženého v '/etc/raspberrycpp/network.conf' (viz část 5.1).
- **ConfigMotorController** – Třída dle návrhového vzoru *Přepravka* (Angl. *Messenger*) obsahující identifikátory 4 GPIO pinů sloužících k ovládání motorů.
- **ConnectionErrorException.h** – Definice vyjímky, kterou vrací třída *TCPRemoteClient,* pokud se nezdaří navázat spojení se serverem.
- **GPIOManager** – Třída obsahující pole tříd *GPIOPin* umožňující přístup k jednotlivým pinům. Tato třída je uložena ve sdílené paměti a je tak dostupná více procesům (viz část 5.2).
- **GPIOPin –** Třída reprezentující GPIO pin. Stará se o nastavení typu pinu, nastavení výstupu, a čtení vstupu. Tyto operace obsahují ochranu proti souběhu ve více vláknech, popř. procesech.
- **IllegalGPIOStateException.h** - Definice vyjímky, kterou vrací třída *GPIOPin*, pokud se uživatel snaží nad piny provádět operace, které nejsou určené pro aktuální stav pinu.
- **LoadConfigurationException.h –** Definice vyjímky, kterou vrací třída *ConfigLoader*, pokud detekuje chybu při načítání konfiguračního souboru.
- **MessageUnresolvedException.h** - Definice vyjímky, kterou vrací třída *TCPRemoteClient,* pokud obdrží zprávu, kterou nedokáže klasifikovat, nebo zprávu ve vadném formátu.

17 Program, tzv. *Shell*, interpretující příkazy příkazového řádku.

- **MotorController –** Třída obsahující referenci na instance *GPIOPin* pro ovládání motoru, která umožňuje převedení vstupních hodnot úhlu a rychlosti na signály PWM vysílané z GPIO pinů na motor.
- **RPIConstants.h** – Soubor obsahující definici konstant, které se využívají na různých místech projektu. To usnadňuje úpravu těchto konstant a jejich dostupnost v kódu.
- **RPIManager** – Třída implementovaná dle návrhového vzoru *Jedináček* obsahující ukazatel na sdílenou paměť, s instancí třídy *GPIOManager* (viz část 5.2).
- **RPIRover.cpp** – Soubor obsahující *main*[18] funkci programu. Obsahuje inicializaci tříd starajících se o načtení nastavení, navázání spojení a inicializaci knihovny pro správu GPIO pinů.
- **TCPRemoteClient** – Třída, která navazuje spojení se serverem. Poskytuje odesílání zpráv serveru a inicializuje vlákno starající se o příjem zpráv.
- **WrongArgumentsException.h** – Definice obecné vyjímky, kterou vrací třídy, pokud obdrží vadné argumenty konstruktoru.

Následuje popis načtení konfiguračního souboru aplikace (viz část 5.1), knihovny pro přístup k GPIO pinům (viz část 5.2), části kódu obstarávajícího komunikaci se serverem (viz část 5.3) a skriptu pro překlad programu na spustitelný soubor a instalaci (viz část 5.4).

5.1 Konfigurace

Konfiguraci načítá *ConfigLoader* ze souboru *'/etc/raspberrycpp/network.conf'* a slouží k zadání informací potřebných pro navázání spojení a nastavení modulů. Modul je třída, spravující skupinu GPIO pinů jako jeden celek. Může jím být např. třída reprezentující I^2C sběrnici, SPI sběrnici nebo senzor vzdálenosti. Momentálně aplikace akceptuje jako modul pouze ovladač motorů.

Aplikace se při inicializaci pokusí otevřít soubor *'/etc/raspberrycpp/network.conf'*. V případě neúspěchu (tj. neexistující umístění nebo vadný soubor) vrátí výjimku *LoadConfigurationException*, v opačném případě začne číst soubor po řádcích. Každý textový

18 Funkce, kterou kompilátor bere jako výchozí funkci celého programu.

řetězec načteného řádku ořízne ze začátku a konce od *bílých znaků*[19], aby bylo možné dát uživateli volnost při formátování konfiguračního souboru. Takto oříznutý řádek pak testuje, zda není prázdný nebo nezačíná znakem *křížek*, tj. znak '#' (Angl. *Hashmark*). Takto označené řádky jsou v konfiguračního souboru brány jako komentáře.

Konfigurační soubor akceptuje řetězce začínající příkazy *Bind*, *Port* a *MotorController*. Řetězec začínající slovem *Bind* nastavuje IP adresu, se kterou má Raspberry Pi navázat spojení. Ta může být zadána jako IP adresa nebo jako doménové jméno. Příkaz *Port* nastavuje síťový port, po kterém budou aplikace komunikovat. Tyto dva příkazy jsou povinné a v souboru se mohou nacházet pouze jednou. Pokud v konfiguračním souboru nejsou, třída *ConfigLoader* vrátí výjimku *LoadConfigurationException*. Příkaz *MotorController* je reprezentant modulu. Následují ho identifikátory 4 pinů, které tento modul spravuje. GPIO piny, které jsou spravovány modulem, mohou být spravovány vždy pouze jedním modulem a nejsou zobrazovány v GUI serverové aplikace pro ruční změnu jejich hodnot. Ukázka konfiguračního souboru je zobrazena na obrázku 5.1.

```
# [ Networking ]
	Bind 10.10.115.39
	Port 31415
# [ Modules ]
	MotorController 3 4 0 1
```

Obr. 5.1 – Ukázka konfiguračního souboru.

5.2 Knihovna pro přístup k GPIO pinům

Pro nízkoúrovňový přístup k GPIO pinům využívám knihovnu *WiringPi*. *WiringPi* je knihovna napsaná v programovacím jazyce C a syntakticky se její použití podobá programování pomocí Wiring pro Arduino. Tato knihovna je defaultně předinstalována na novějších verzích operačního systému *Raspbian* a slouží pro přístup k GPIO pinům. Obsahuje též implementaci softwarového PWM výstupu, díky kterému je možné generovat PWM výstup na všech GPIO pinech, nebo funkce pro přístup k I^2C a SPI sběrnici.

19 Znak představující prázdné místo v textu, jako je mezera nebo tabulátor.

Nad knihovnou *WiringPi*, starající se o obsluhu GPIO pinů a PWM výstupů, jsem vytvořil objektově orientovanou nadstavbu v jazyce C++. Ta obsahuje třídu *RPIManager* implementovanou dle návrhového vzoru *Jedináček* (Angl. *Singleton*). Tato třída načte segment sdílené virtuální paměti pomocí systémového volání *mmap*. Toto volání připojí procesu virtuální adresní prostor, který odkazuje na soubor nacházející se v souborovém systému na adrese *'/tmp/RPI-Manager'*. Tento soubor je uložen v dočasném adresáři a tudíž se smaže dle nastavení operačního systému (většinou při vypnutí). Pozici souboru je možné změnit přepsáním konstanty *FILEPATH* definované v souboru *RPIManager.h*. Daný blok paměti program reprezentuje jako instanci třídy *GPIOManager*. Při prvním spuštění programu, kdy soubor sdílené paměti neexistuje, soubor vytvoří a jeho virtuální adresu přiřadí do nové instance třídy *GPIOManager*. Ta obsahuje pole instancí třídy *GPIOPin*. Velikost pole je určena počtem pinů GPIO portu Raspberty Pi, který je nastaven pevně na hodnotu 26. Program neobsahuje proceduru, která by dokázala rozeznat počet pinů GPIO portu, protože jsem neměl více typů Raspberry Pi. Tato konstanta může být změněna v hlavičkovém souboru *GPIOManager.h*.

```
enum GPIOType {
        RPI_OUTPUT = 0,
        RPI_INPUT = 1,
        RPI_OUTPUT_PWM = 2
};
```

Obr. 5.2 – Ukázka výčtového typu GPIOType.

Třída *GPIOPin* ukládá číslo pinu, informaci o nastavení pinu (výstup, PWM výstup nebo vstup) a hodnotu na pinu. Dále umožňuje operace s pinem pomocí metod *read*, s*etType, getType*, *setValue* a *getValue*. Pokud je pin vstupem, metoda *read* přečte napěťovou úroveň na pinu. Metoda *setType* nastavuje chování vstupně-výstupního pinu, tj. vstup nebo výstup. Možné nastavení je digitální vstup, digitální výstup nebo PWM výstup a lze ho najít ve výčtovém typu *GPIOType* definovaném v hlavičkovém souboru *RPIConstants.h* (viz Obr. 5.2). Metoda *getType* vrací hodnotu tohoto výčtového typu dle nastavení pinu. Metoda *setValue* slouží k nastavení hodnoty na pinu nastaveném jako výstup nebo PWM výstup. Pokud je pin klasickým výstupem, interpretuje nenulovou hodnotu jako vysokou

hladinu napětí, v opačném případě jako nízkou napěťovou hladinu. Pokud je pin nastaven jako PWM výstup, očekává hodnoty v rozmezí od 0 do 100. Hodnota 0 je rovna nízké hladině napětí, hodnota 100 je rovna vysoké hladině napětí. Hodnoty mezi reprezentují velikost střídy (viz část 2.5.1), tj. rychlost otáčení motoru. Metoda *getValue* vrátí hodnotu nastavenou na výstupním GPIO pinu, nebo přečte hodnotu na pinu nastaveném jako vstup. V nedefinovaných situacích pin vrátí výjimku *IllegalGPIOStateException*.

5.3 Komunikace se serverem

Pro síťovou komunikaci slouží třída *TCPRemoteClient*. Ta iniciuje vlákno starající se o přijímání zpráv od serveru a obsahuje metody pro odesílání zpráv serveru. Dále spustí vlákno, které v intervalu 2 sekund odesílá aktuální nastavení GPIO portu, aby zaručil správné zobrazení nastavení GPIO pinů na straně serveru.

Jako argumenty se konstruktoru předají IP adresa a port serveru. Vytvářená instance se pokusí navázat spojení se serverem na zadané IP adrese a portu. V případě, že neuspěje, vrátí výjimku *ConnectionErrorException*. Poté, co je spojení navázáno, je nutné instanci třídy *TCPRemoteClient* definovat, se kterými GPIO piny a moduly může manipulovat a spustit vlánko pro přijímání zpráv a vlákno pro cyklické odesílání stavu zařízení.

Přidávání GPIO pinů a modulů probíhá pomocí metod *addGPIO* a *addMotorController*. Třída *TCPRemoteClient* obsahuje mapu[20] ukazatelů na instanci třídy *GPIOPin*, kde klíčem je číslo pinu. Metoda *addGPIO*, která jako argument obdrží identifikátor pinu, si ukazatel na instanci třídy *GPIOPin* uloží do této mapy. Stejným způsobem se přistupuje k modulům pro ovládání motorů. Třída *TCPRemoteClient* obsahuje další mapu pro ukládání ukazatelů na ovladače motorů, kde klíčem je identifikátor daného modulu. Metoda *addMotorController* obdrží jako argumenty identifikátor modulu a čtveřici identifikátorů GPIO pinů, které modul spravuje. O vytvoření a nastavení instance třídy *TCPRemoteClient* se stará funkce *initConnection* deklarovaná v souboru *RPIRover.cpp*, která získá nastavení z reference na instanci třídy *ConfigLoader* popsané v části 5.1.

Spuštění naslouchání zpráv obdržených od serveru probíhá pomocí metody *start*.

20 Abstraktní datový typ – asociativní pole dostupné jako *std::map*, implementované jako ADT *Red-Black tree.*

Ta vytvoří a spustí vlákno pro příjem zpráv a vlákno pro odesílání aktuálního stavu GPIO pinů. Vlákno pro příjem zpráv naslouchá pomocí funkce *recv* na socketu. Tato operace je blokující, je ale nastaven *timeout* této operace na 2 sekundy. Pokud je vyslán požadavek na vypnutí serveru pomocí metody *stop*, tak *timeout* zapříčiní, že nejdéle do 2 sekund bude vlákno validně ukončeno. Při příjmu zprávy se přečte první byte a podle něj se rozhodne o typu zprávy. Následně se vybere příslušná metoda, která přečte zbytek informace a dle příslušných argumentů (popsaných v kapitole 7) provede příslušnou akci. Vlákno odesílající informace o GPIO pinech každé 2 sekundy odešle informace o typu a hodnotě každého pinu, který byl přidán do správy instanci třídy *TCPRemoteClient* pomocí metody *addGPIO*. Po zbytek času je uspané.

5.4 Skript pro překlad

Skript *install.sh* pro *Bash* po spuštění ověří, zda byl spuštěn s právy superuživatele. Pokud má dostatečná práva, ve výkonu kódu pokračuje, a ověří, zda je v daném operačním systému nainstalován *CMake*[21], který je potřebný pro překlad. V případě, že není, zeptá se uživatele, zda ho má nainstalovat pomocí nástroje pro správu balíčků *apt-get*[22]. Pokud je vše připraveno, zeptá se, jestli má nainstalovat (popř. přeinstalovat) knihovnu *WiringPi* ze zálohy této knihovny dodané s programem. Poté se skript pokusí přeložit samotný program. Kompilace probíhá ve složce *build*, kterou si skript v případě potřeby vytvoří. Po dokončení kompilace přesune výsledný spustitelný soubor do složky *'/opt/RPIRover/'*, a ve složce *'/usr/bin/'* vytvoří symbolický *link* (odkaz) na spustitelný soubor (to umožní spustit program z příkazové řádky z libovolného umístění). Dále vytvoří konfigurační soubor, který uloží jako *'/etc/raspberrycpp/network.conf'* a v případě, že uživatel při výzvě povolí, přidá se skript pro spuštění programu po startu systému.

21 Nástroj generující *Makefile* skript pro překlad pomocí libovolného software pro automatický překlad.
22 Program sloužící pro správu software v distribucích založených na distribuci Debian.

6 Desktopová aplikace

Desktopová aplikace je implementována v jazyce C++ s využitím *Qt frameworku*, který se stará o portabilitu kódu na různé platformy a vykreslování GUI. Základem aplikace jsou zdrojové soubory, složka s ikonami pro jednotlivá tlačítka a *QMake*[23] skript pro překlad.

Aplikace slouží ke vzdálenému ovládání vozítka a jeho GPIO pinů z PC. Skládá se z GUI a knihovny pro síťovou komunikaci s Raspberry Pi dle definovaných pravidel popsaných v kapitole 7. GUI umožňuje ovládat libovolný počet Raspberry Pi vozítek.

Program se kompiluje z následujících tříd a hlavičkových souborů:

- **CircleIndicator** – Widget vykreslující kruh, který slouží jako indikátor nastavení jednoho GPIO pinu. Obsahuje 4 definované stavy – výstup vysoký/nízký a vstup vysoký/nízký.
- **ControllerOverviewWidget** – Widget s horizontálním posuvníkem, sloužící k zobrazení jednotlivých modulů a aktuálního stavu GPIO pinů.
- **ControllerWidget** – Widget obsahující kruh, který slouží k ovládání pohybu.
- **GpioPinWidget** – Widget sloužící k nastavování stavu GPIO pinu.
- **GpioSettingsWidget** – Widget s vertikálním posuvníkem, sloužící k zobrazení prvků určených k ručnímu ovládání GPIO pinů.
- **HeaderWidget** – Widget obsahující legendu a indikátory jednotlivých GPIO pinů Raspberry Pi.
- **main.cpp** – Soubor obsahující *main* funkci programu. Obsahuje pouze zobrazení hlavního okna programu.
- **RaspberrySocket** – Třída reprezentující komunikaci s jedním Raspberry Pi. Obsahuje signály emitované při obdržení konkrétní zprávy od klienta a metody sloužící k odesílání zpráv klientovi.
- **RaspberryTabWidget** – Widget sloužící k vykreslení všech ovládacích prvků

23 Nástroj pro automatizaci generování *Makefile* skriptu (skript pro překlad aplikace).

náležících Raspberry Pi klientovi.

- **RaspberryTcpServer** – Třída spravující komunikaci s jednotlivými klienty. Navazuje nová a ukončuje přerušená spojení.
- **Window** – Hlavní okno celé aplikace. Obsahuje panel nástrojů pro obsluhu programu a zobrazuje obslužné nástroje jednotlivých klientů.

Následuje popis knihovny pro komunikaci a způsob jejího použití (viz část 6.1), popis funkce GUI (viz část 6.2) a popis funkce skriptu pro překlad (viz část 6.3).

6.1 Knihovna pro síťovou komunikaci

Knihovna pro síťovou komunikaci je složena z tříd *RaspberryTcpServer* a *RaspberrySocket*. Pro použití v aplikaci je potřeba vytvořit instanci třídy *RaspberryTcpServer*, které jako parametr předáme ukazatel na rodičovský *Qt objekt*[24], a adresu a port, na které má naslouchat. Následně stačí naslouchání spustit metodou *start*. Ukončit server lze metodou *stop*.

Pokud server běží a klient se k němu pokusí připojit, tak server emituje signál *newSocket* obsahující ukazatel na nově vzniklou instanci třídy *RaspberrySocket*. *Qt framework* umožňuje spojení signálu se vstupním bodem (Angl. *Slot*) metodou *connect*, kterou zdědí každý objekt od *Qt objektu*. Proto je možné tento signál spojit s libovolnou obslužnou metodou. Např. v mé ukázkové aplikaci používám signál pro zobrazení dialogového okna sloužícího pro přijmutí nebo odmítnutí nového klienta, popř. k jeho pojmenování.

Třída *RaspberrySocket* pro komunikaci využívá množinu signálu a slotů. Signál *destroyed* je vyvolán, pokud je ukončeno spojení. Obsahuje ukazatel na instanci třídy *RaspberrySocket*, která tento signál vyvolala, aby byla usnadněna identifikace socketu. Pokud *RaspberrySocket* přijme zprávu od klienta, pokusí se z ní získat příkaz spolu s jeho argumenty. V případě, že neuspěje, vyvolá signál *messageUnresolved*. V opačném případě vyvolá jeden z následujících signálů, dle typu příkazu:

24 Objekt v C++ dědící od třídy *QObject*. Tvoří hierarchický strom a každá třída pak spravuje prostor paměti svých potomků. Navíc může obsahovat definice signálů a vstupních bodů (tzv. *Slot)*, což je obdoba metody běžné třídy.

- **newComonGPIO** – Tento signál je vyvolán, pokud socket přijme informace o novém GPIO pinu, který je možné ručně ovládat. Spolu se signálem předá identifikátor pinu.
- ***GPIOTypeChange*** – Tento signál je vyvolán, pokud server obdrží oznámení o změně nastavení typu některého z GPIO pinů. Signál předá identifikátor pinu a informaci o aktuálním nastavení GPIO pinu.
- ***GPIOValueChange*** - Tento signál je vyvolán, pokud server obdrží oznámení o změně hladiny napětí (nastavené či obdržené) na některém z GPIO pinů. Signál předá identifikátor pinu a hladinu napětí na pinu.
- ***newMotorController*** – Tento signál je vyvolán, pokud server obdrží oznámení o novém modulu pro ovládání motorů. Tento signál předá pouze identifikátor daného modulu. Tento identifikátor musí být následně předán jako jeden z argumentů při nastavování směru pohybu vozítka.

Pokud požadujeme odeslání některého z požadavků do Raspberry Pi, je možné použít definované sloty. Ty jsou určeny pro spojení s některým ze signálů metodou *connect*, ale je možné je vyvolat i jako běžné metody. Jen je nutné pro spuštění použít statickou metodu *invokeMethod* třídy *QmetaObject*, která v *Qt frameworku* slouží právě k tomuto účelu. Třída *RaspberrySocket* má definovány následující sloty:

- **sendSetGPIOType** – Tento slot slouží k odeslání příkazu pro změnu typu GPIO pinu. Je nutné mu předat identifikátor pinu a typ, na který chceme pin nastavit.
- **sendSetGPIOValue** – Tento slot slouží k odeslání příkazu pro změnu napěťové hladiny GPIO pinu. Je nutné mu předat identifikátor pinu a hodnotu, kterou chceme na pinu nastavit.
- **sendMotorControllerDritection** – Tento slot slouží k odeslání směru, kterým se má vozítko pohybovat. Musí se mu předat jako argumenty identifikátor modulu, úhel pohybu a rychlost pohybu.
- **sendShutdown** – Tento slot odešle požadavek na vypnutí Raspberry Pi.
- **sendPing** – Tento slot odesílá příkaz *ping*, který slouží k testování spojení s klientem.

Pokud klient obdrží tento příkaz, odešle zpět odpověď v podobě příkazu *pong*. Pokud server tuto odpověď obdrží, spojení je stále dostupné.

- **sendPong** – Je svázán s přijmutím zprávy *ping*, na kterou reaguje odesláním zprávy *pong* klientovi.

Po povolení připojení nového klienta server odešle využitím slotu *sendGetSettings* požadavek o zaslání popisu nastavení klienta. Podrobný popis odesílaných a přijímaných zpráv, jejich velikost a způsob formátování naleznete v kapitole 7.

6.2 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní (*GUI*) je vytvořeno pomocí *Qt frameworku*. Ten na Linuxu využívá desktopové prostředí *KDE*[25]. Je možná i kompilace pro Windows, pokud jsou na počítači nainstalovány potřebné knihovny.

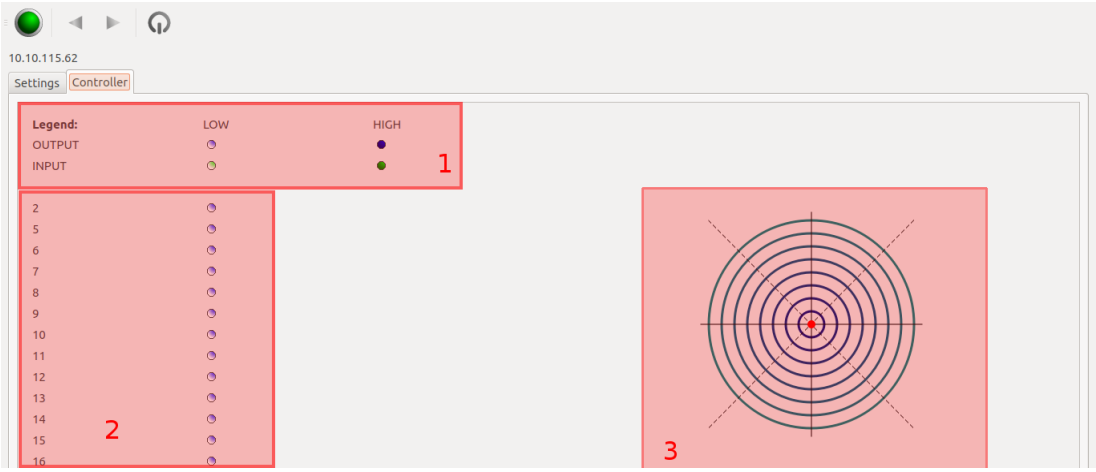
Obr. 6.1 – Ukázka panelu Settings s označenými oblastmi pro odkazování v textu.

Po spuštění aplikace se zobrazí hlavní okno programu (třída *Window*), které obsahuje panel nástrojů pro ovládání aplikace (viz Obr. 6.1). Je na něm umístěno tlačítko pro spuštění naslouchání na síti (na Obr. 6.1 označeno 1). Po kliknutí na něj se emituje signál na slot *start* nebo *stop*, což záleží na aktuálním stavu serveru. V případě, že server naslouchá a pokusí se k němu připojit klient, tak server zobrazí dialogové okno. To slouží k pojmenování nového klienta nebo jeho odmítnutí. Název klienta je ve výchozím stavu nastaven dle jeho IP adresy. Pokud nového klienta uživatel přijme, tak se zobrazí v hlavním okně aplikace nový panel

25 KDE je spolu s Gnome nejrozšířenější desktopové uživatelské prostředí na platformě Linux. Slouží k vykreslování GUI aplikací na pracovní ploše operačního systému.

pro ovládání klienta. Mezi dostupnými klienty lze přepínat tlačítky umístěnými na panelu nástrojů (na Obr. 6.1 označeno 2). Panel klienta je reprezentován instancí třídy *RaspberryTabWidget*. Instance je odstraněna z paměti ve chvíli, kdy se ukončí spojení s klientem. K odeslání příkazu pro ukončení klienta slouží poslední tlačítko nacházející se v ovládacím panelu (na Obr. 6.1 označeno 3).

RaspberryTabWidget obsahuje popisek s se jménem klienta (na Obr. 6.1 označeno 4), který zvolil uživatel v dialogovém okně a okno záložek (na Obr. 6.1 označeno 5). První záložka s názvem *Settings* odkazuje na instanci třídy *GPIOSettingsWidget* slouží k zobrazení widgetů určených pro změnu nastavení GPIO pinů. Dále obsahuje seznam všech GPIO pinů, které má server povolené ovládat. Každý GPIO pin je reprezentován instancí třídy *GPIOPinWidget* obsahující popisek s identifikátorem pinu, výběrové menu pro nastavení typu a tlačítko pro změnu napěťové hodnoty na pinu v případě, že je výstupní (na Obr. 6.1 označeno 6). Tato instance metodou *connect* zřetězuje volání signálů končících odesláním příkazů do Raspberry Pi třídou *RaspberrySocket*.



Obr. 6.1 – Ukázka panelu Controller s označenými oblastmi pro odkazování v textu.

Druhá záložka v panelu *RaspberryTabWidget* s názvem *Controller* (viz Obr. 6.1) obsahuje instanci třídy *ControllerOverviewWidget*. Ta obsahuje instanci třídy *HeaderWidget* a instance třídy *ControllerWidget*. *HeaderWidget* obsahuje pole indikátorů jednotlivých pinů (na Obr. 6.2 označeno 2) spolu s jejich popisem (na Obr. 6.2 označeno 1). Tyto indikátory jsou reprezentovány třídou *CircleIndicator*, která vykreslí do okna kružnici, jejíž barva indikuje,

zda je pin nastaven jako vstup či výstup, a sytostí barvy indikuje napěťovou hladinu na daném pinu. Tyto indikátory metodou connect zřetězuje volání signálů končících u třídy *RaspberrySocket*, od které získávají oznámení o změně hladiny napětí na pinu nebo změně nastavení pinu (na vstup nebo výstup).

Instance třídy *ControllerWidget* vykreslí v okně soustavu soustředných kružnic, osy indikující klíčové úhly pro ovládání vozítka a bod indikující aktuální nastavení směru a rychlosti pohybu vozítka (na Obr. 6.2 označeno 3). Pozici tohoto bodu je možné změnit myší. Pozice je posléze převedena funkcí *atan2*[26] na úhel a Pythagorovou větou[27] je vypočítána rychlost. Vypočítané hodnoty jsou posléze odeslány klientovi. Postup odesílání je popsán v kapitole 7.

6.3 Skript pro překlad

Program obsahuje *Bash* skript pro překlad programu na platformě Linux. Ten otestuje, zda je nainstalovaný *QMake*, který slouží pro překlad programu napsaného pomocí *Qt frameworku*. Pokud nainstalovaný je, tak vytvoří složku *build*, ve které přeloží program na spustitelný soubor, ten přesune do složky '/opt/RaspberryPiServer/' pod názvem *RaspberryPiServer* a vytvoří symbolický link (odkaz) do adresáře '/usr/bin/' (to umožní spouštět program z libovolného umístění).

26 Arctg2, v programování často používaný jako Atan2(y, x) je funkce 2 proměnných, jejíž hodnota je shodná s úhlem sevřeným mezi osou x a průvodičem bodu [x, y].

27 Pythagorova věta popisuje vztah, který v eukleidovské rovině umožňuje dopočítat délku třetí strany pravoúhlého trojúhelníka. Je to běžně používaný vztah pro výpočet vzdálenosti dvou bodů.

7 Komunikační protokol

Pro komunikaci jsem použil TCP protokol s bytovou reprezentací zpráv. Každá zpráva obsahuje 1 byte reprezentující typ zprávy, sekvenci hodnot daného příkazu a ukončení nulovým bytem. Takto reprezentované zprávy jsou úsporné, co se datového přenosu týká, a lehce se spojují do složitějších sekvencí příkazů. Server i klient mají definovány stejné typy zpráv (viz Tab. 7.1).

Protokol obsahuje příkazy pro manipulaci s GPIO Piny Raspberry Pi, ovládání modulu pro spínání motorů a příkaz pro vypnutí Raspberry Pi.

Tabulka 7.1 - Seznam typů použitých zpráv (Identifikátor je dekadicky zapsané 1bytové číslo).

Název příkazu	Identifikátor	Data
GET NEW GPIO	1	GPIO pin [1B]
GET GPIO TYPE	2	GPIO pin [1B], GPIO typ [1B]
GET GPIO VALUE	3	GPIO pin [1B], GPIO hodnota [1B]
SET GPIO TYPE	4	GPIO pin [1B], GPIO typ [1B]
SET GPIO VALUE	5	GPIO pin [1B], GPIO hodnota [1B]
MODULE MOTOR CONTROLLER NEW	64	ID modulu pro ovládání motorů [1B]
MODULE MOTOR CONTROLLER DIRECTION VECTOR	65	ID modulu pro ovládání motorů [1B], úhel směru pohybu [2B], rychlost pohybu [1B]
TOOL GET SETTINGS	128	-
TOOL SHUTDOWN	129	-
TOOL PING	130	-
TOOL PONG	131	-

Raspberry Pi se pokusí navázat spojení se serverem (viz část 7.1). Pokud je server dostupný a povolí připojení Raspberry Pi, tak je možné ovládat ze strany serveru vzdáleně GPIO piny Raspberry Pi (viz část 7.2), pohyb vozítka (viz část 7.3) nebo lze vzdáleně Raspberry Pi vypnout (viz část 7.4).

7.1 Navázání spojení

Navázání spojení inicializuje Raspberry Pi. Po navázání spojení rozhoduje server, zda připojení Raspberry Pi povolí. Pokud ne, tak uzavře spojení. V opačném případě pošle klientovi zprávu *TOOL GET SETTINGS*, viz Tab. 7.1. Tou dává server Raspberry Pi najevo, že povoluje připojení. Při obdržení této zprávy Raspberry Pi odešle seznam pinů a modulů, které se zobrazí v GUI serveru, aby byly možné měnit vzdáleně jejich stav. Seznam se skládá s příkazů *GET NEW GPIO* a *MODULE MOTOR CONTROLLER NEW*, viz Tab. 7.1.

Příkaz *GET NEW GPIO* obsahuje identifikátor pinu o maximální velikosti 1 byte (tj. max. 256 GPIO pinů). Tento návrh umožňuje rozšíření aplikace v případě jejího převodu na jinou platformu než je Raspberry Pi či použití možných budoucích typů Raspberry Pi. Aktuální verze však disponují maximálně 26 GPIO piny.

Příkaz *MODULE MOTOR CONTROLLER NEW* obsahuje identifikátor daného modulu. Toto číslo může nabívat též 256 hodnot. Volba identifikátoru modulu může probíhat sekvenčně i náhodně, avšak každý modul musí mít identifikátor unikátní.

7.2 Vzdálené operace s GPIO piny

Na Rasbperry Pi běží cyklus, který informuje server o aktuálním nastavení pinů, nebo, v případě vstupních pinů, čte a odesílá jejich aktuální hodnoty. Tuto informaci posílá každé 2 vteřiny příkazy *GET GPIO TYPE* a *GET GPIO VALUE*, viz Tab. 7.1. Pokud uživatel změní nastavení pinů v grafickém rozhraní serveru, tak se odešle zpráva Raspberry Pi neprodleně. Tyto zprávy mají formát příkazů *SET GPIO TYPE* a *SET GPIO VALUE*.

Zpráva *GET GPIO TYPE čte* a zpráva *SET GPIO TYPE* mění nastavení GPIO pinů na vstupní a výstupní. Tyto příkazy obsahují 1 bytový identifikátor pinu a 1 byte značící nastavení GPIO pinu. Hodnota 0 značí výstup a hodnota 1 značí vstup. Zprávy *GET GPIO VALUE* a *SET GPIO VALUE* slouží k čtení a nastavení hodnot GPIO pinů. Pokud chceme vstup nastavit na vysokou či nízkou hladinu napětí, pošleme příkaz *SET GPIO VALUE* s identifikátorem pinu a 1 bytovou hodnotou. Nenulová hodnota je brána jako vysoká hladina napětí, nulová pak jako nízká hladina napětí. Stejně pracuje příkaz pro nastavení PWM výstupu. Ten očekává hodnotu v rozmezí od 0 do 100 (tj. střída v procentech) dle které určí

střídu výstupního PWM signálu (viz část 2.5.1).

7.3 Modul pro ovladač motorů

Součástí síťové komunikace je i třída starající se o převod úhlu a rychlosti obdržených pomocí příkazem *MODULE MOTOR CONTROLLER DIRECTION VECTOR* na PWM signály určující rychlost a směr otáčení motoru. Identifikátor modulu musí být odeslán spolu s konfigurací GPIO pinů, aby se ovládací prvek vykreslil v GUI desktopové aplikace. Zpráva obsahuje 1 bytový identifikátor modulu, 2 bytové celé číslo reprezentující úhel pohybu v rozmezí od 0 do 360 stupňů a rychlost pohybu určenou 1 bytovým celým číslem v rozmezí od 0 do 100.

Modul obsahuje reference na 4 GPIO piny nastavené jako PWM výstup. Dvojice pinů určuje pohyb jednoho motoru. Pokud je PWM signál přiveden na první z dvou pinů, motor se točí vpřed, v opačném případě se motor otáčí vzad. Rychlost otáčení je dána střídou signálu přivedeného na pin (viz část 2.5.1). Je nutné úhel a rychlost převést na dané hodnoty PWM výstupů, k čemuž jsem navrhl čtveřici funkcí vyobrazených na obrázku C.1.

Jedná se o funkce dle předpisu (4) kde $A=\sqrt{2}$ a pro levý motor je φ_L=45° a pro pravý φ_R=135°. Pro piny obsluhující pohyb vpřed funkci omezíme na interval $\langle 0;1\rangle$, pro piny osluhující pohyb vzad pak funkci omezíme na interval $\langle -1;0\rangle$ a výstup uvedeme v absolutní hodnotě. Tím získáme požadovanou střídu. Vynásobením střídy stem získáme střídu v procentech, což je hodnota, kterou nastavíme na PWM výstup.

$$f_i(x)=A.\sin(x+\varphi_i) \qquad (4)$$

Směr otáčení (vyjádřený v procentech) motorů pro významné úhly lze vyjádřit dle tabulky 7.2, hodnoty mezi těmito úhly se odvíjí od hodnot vztahu (4), popř. je lze aproximovat lineárně.

Tab. 7.2 – Směr otáčení motorů významných hodnot úhlů

Úhel	Levý motor	Pravý motor
0°	+ 100 %	+ 100 %

Úhel	Levý motor	Pravý motor
45°	0 %	+ 100 %
90°	- 100 %	+ 100 %
135°	- 100 %	0 %
180°	- 100 %	- 100 %
225°	0 %	- 100 %
270°	+ 100 %	- 100 %
315°	+ 100 %	0%

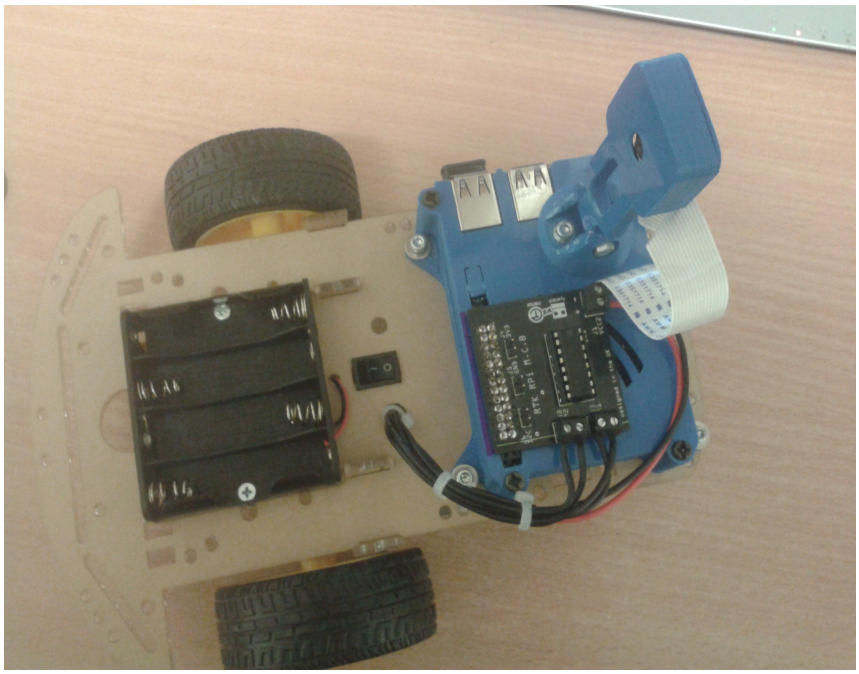
7.4 Vzdálené vypnutí

Na ovládacím panelu je tlačítko, kterým lze požádat o vzdálené vypnutí celého Raspberry Pi. To se provede posláním příkazu *TOOL SHUTDOWN* Raspberry Pi. Ten ukončí spojení se serverem a systémovým voláním požádá o *fork*[28] procesu. Rodič se ukončí běžným způsobem. Potomek se uspí na 5 sekund a posléze provede systémové volání *execvep* na program *shutdown*, což zapříčiní přepsání paměti potomka procesem *shutdown* zajišťujícím vypnutí operačního systému Raspberry Pi.

28 Systémové volání, které provede duplikaci procesu. Každý proces pak může vykonávat různé procedury bez ovlivnění druhého z procesů.

8 Dosažené výsledky a možná rozšíření

Podařilo se mi splnit všechny body zadání Bakalářské práce. Nakoupil jsem vhodné periferie pro stavbu vozítka a vozítko zkonstruoval, viz Obr. 8.1. Vytvořil jsem dvojici programů sloužících ke vzdálenému ovládání GPIO pinů Raspberry Pi a pohybu vozítka z PC prostřednictvím Wi-Fi sítě. Pro možné využití v jiných aplikacích (pro PC či mobilní zařízení) jsem vytvořil knihovnu v C++ s využítím Qt frameworku, určenou pro komunikaci s Raspberry Pi. Tuto knihovnu jsem zakomponoval do aplikace s grafickým rozhraním.



Obr. 8.1 – Vzhled vozítka po sestavení (vlastní fotografie).

8.1 Praktická použitelnost

Vozítko je plně funkční a umožňuje vzdálené ovládání prostřednictvím Wi-Fi sítě. Praktická použitelnost vozítka momentálně není velká a odvíjí se od periferií, o které se vozítko v budoucnosti rozšíří. Celá konstrukce však trpí některými nedostatky. I přes napájení motorů z baterií je nutné počítač Raspberry Pi napájet přes kabel připojený do elektrické sítě. Toto řešení jsme zvolili v rámci finančních úspor. Tento nedostatek je možné vyřešit přikoupením 5 V regulátoru napětí a Li-Pol akumulátoru (viz část 4.1).

Tím by se napájení motorů a Raspberry Pi sjednotilo. Dalším nedostatkem je, že vozítko neposílá na server obraz z kamery, a proto je vhodné mít na vozítko přímý výhled.

8.2 Rozšíření práce

Nad rámec zadání jsem rozšířil knihovnu WiringPi sloužící k ovládání GPIO pinů o blok sdílené virtuální paměti, kterou sdílí všechny programy využívající tuto knihovnu. V případě, že běží současně více procesů ovládajících GPIO piny, tak jejich akce nad piny jsou zabezpečeny zámky proti souběhu. Rozdílné procesy také mohou mít díky bloku sdílené virtuální paměti vždy aktuální informaci o nastavení GPIO portu Raspberry Pi.

Z časových důvodů jsem již nestihl rozšířit vozítko o kameru a odesílání obrazu. K získání snímků z kamery jsem použil knihovnu RaspiCam. Narazil jsem však na problém velkého datového toku sítí při odesílání nekomprimovaného obrazu. Z důvodu pouze krátké části kódu jsem se rozhodl ji k finálnímu projektu nepřipojit. Kód je však součástí přílohy na DVD ve složce */other/camera*.

Dále jsem se pokusil rozšířit vozítko o senzor vzdálenosti. Na TTL ultrazvukový senzor vzdálenosti jsem napájel dělič napětí. Z časových důvodů a poničení některých potřebných pinů[29] na ovladači motorů jsem však práci nedokončil.

29 Pozn: poničení nevzniklo mou vinou.

9 Závěr

Hlavním cílem práce bylo vytvořit vzdáleně ovládatelné vozítko, k čemuž bylo zapotřebí seznámit se s počítačem Raspberry Pi, vybrat vhodné periferie a seznámit se s jejich fungováním, zapojením a napájením. Pro potřeby vozítka jsem zvolil podvozek, motory, ovladač motorů a další periferie. Podařilo se mi vyřešit i napájení motorů. Z finančních důvodů je však stále nutné připojení vozítka jedním kabelem k rozvodné síti elektrického napětí pro potřeby napájení Raspberry Pi. Pro implementaci softwarového vybavení vozítka zvolil jazyk C++ a potřebné knihovny.

Dále bylo nutné implementovat programy pro vzdálené ovládání vozítka. Vytvořil jsem knihovnu pro práci s piny GPIO portu a kód pro interpretaci příkazů, které Raspberry Pi obdrží ze sítě. Dále jsem vytvořil knihovnu pro vzdálené ovládání Raspberry Pi a grafické rozhraní nad touto knihovnou na straně serveru, za použití Qt frameworku. Vytvořené aplikace umožňují vzdálené ovládání vozítka, a tím splňují cíle práce.

9.1 Možná budoucí vylepšení vozítka

Pokud by se vozítko rozšířilo o napájení Raspberry Pi bez kabelu a kameru a obraz by se posílal na server, tak by bylo možné vozítko ovládat vzdáleně až do dosahu Wi-Fi sítě. Pokud by měl server navíc veřejnou IP adresu, tak by bylo možné vozítko ovládat prostřednictvím internetové sítě, a tak pořádat např. vzdálené prohlídky objektů, aniž by se člověk nacházel na daném místě. Obraz z kamery je též možné analyzovat přímo na Raspberry Pi, a vytvořit algoritmus pro autonomní pohyb vozítka.

Další možné rozšíření vozítka je doplnění senzorů pro detekci kolize nebo infračervených senzorů pro detekci barvy podkladu. Pokud by se vozítko rozšířilo o senzor nebo skupinu senzorů vzdálenosti, bylo by možné implementovat některý z algoritmů umělé inteligence pro autonomní pohyb, detekci a řešení kolizí (např. pomocí *multiagentního systému*[30] nebo *neuronové sítě*[31]). V případě rozšíření o infračervené senzory lze

30 Simulované prostředí, kde interagují jednotlivé subsystémy (agenti) mezi sebou a s prostředím, a řeší tak problém, který by sám žádný ze subsystémů vyřešit nedokázal.

31 Výpočetní model, který se svým chováním podobá neuronových sítí v biologických strukturách (síti neuronů v mozku).

implementovat algoritmus pro pohyb po čáře nebo řešení bludiště.

Přehled zkratek

ALU – Arithmetic Logic Unit

ARM – Advanced RISC Machine

CPU – Central Processing Unit

CSI – Camera Serial Interface

DMA – Direct Memory Access

DSI – Display Serial Interface

EEPROM – Electrically Erasable Programmable Read-Only Memory

GPIO – General-Purpose Input/Output

GPS – Global Positioning System

GUI – Graphical User Interface

HDMI – High-Definition Multimedia Interface

I/O – Input/Output

I^2C – I-square C

IoT – Internet of Things

LED – Light-Emitting Diode

Li-Pol – Lithium Polymer

MPI – Message Passing Interface

NFC – Near Field Communication

PC – Personal Computer

PWM – Pulse-Width Modulation

RFID – Radio-Frequency Identification

SPI – Serial Peripheral Interface Bus

TTL – Transistor-Transistor Logic

UART – Universal Asynchronous Receiver/Transmitter

USB – Universal Serial Bus

Literatura

[1] UPTON, Eben a HALFACREE, Gareth. *Raspberry Pi User Guide*, 2nd edition. Cambridge: Wiley, 2013, 314 s. ISBN 978-1-118-79548-4.

[2] COX, Tim. *Raspberry Pi Cookbook for Python Programmers*. Birmingham: Packt Publishing , 2014, 388 s. ISBN 978-1-84969-662-3.

[3] RASPBERRY PI FOUNDATION. *Raspberry Pi* [online]. 2008 [cit. 2015-12-12]. Dostupné z: https://www.raspberrypi.org/

[4] DH SERVIS. *Pulzně šířková modulace* [online]. 2002 [cit. 2015-12-12]. Dostupné z: http://www.dhservis.cz/psm.htm

[5] NXP SEMICONDUCTORS N.V. *UM10204 I^2C-bus specification and user manual*, version 6 [online]. 2014 [cit. 2015-12-12].
Dostupné z: http://www.nxp.com/documents/user_manual/UM10204.pdf

[6] FREESCALE SEMICONDUCTOR, INC. *SPI Block Guide*, version 3.06 [online]. 2003 [cit. 2015-12-12]. Dostupné z: http://www.datasheetarchive.com/dlmain/Datasheets-12/DSA-222613.pdf

[7] BECHYNSKÝ, Štěpán. *Raspberry Pi a GPIO – co nefunguje* [online]. 2015 [cit. 2015-12-12]. Dostupné z: https://www.zdrojak.cz/clanky/raspberry-pi-a-gpio-co-nefunguje/

[8] CARULLO, Allesio a PARVIS, Marco. An Ultrasonic Sensor for Distance Measurement in Automotive Applications. *IEEE SENSORS JOURNAL*. 2001, 1(2), 143-147. ISSN 1530-437X. Dostupné z: http://porto.polito.it/1398684/1/2001_TR_AnUltrasonicSensorForDistance_MeasurementIn_Automotive.pdf

[9] ARDUINO SA. *Arduino* [online]. 2005 [cit. 2015-12-12]. Dostupné z: *https*://www.arduino.cc/

[10] THE QT COMPANY. Application *Development* [online]. 2015 [cit. 2015-12-12]. Dostupné z: http://www.qt.io

[11] NORRIS, Donald. *Raspberry Pi Projects for Evil Genius*. New York: McGraw-Hill Education , 2014. 288 s. ISBN 978-0-07-182158-2

[12] DENNIS, Andrew K. *Raspberry Pi Super Cluster*. Birmingham: Packt Publishing, 2013. 126 s. ISBN 978-1-78328-619-5

[13] MICROSOFT CORPORATION, *Windows 10 IoT* [online]. 2015 [cit. 2015-12-12]. Dostupné z: https://dev.windows.com/en-us/iot

[14] HORAN, Brendan. *Practical Raspberry Pi*. New York: Apress, 2013. 239 s. ISBN 978-1-4302-4972-6

[15] PARTNER, Kevin. *Ultimate Guide to Raspberry Pi*. London: Magbook, 2014. 161 s. ISBN 1-78106-312-5

[16] MOSAIC INDUSTRIES, *GPIO Electrical Specifications* [online]. 2014 [cit. 2016-4-7].

Dostupné z: http://www.mosaic-industries.com/embedded-systems/microcontroller-projects/raspberry-pi/gpio-pin-electrical-specifications

[17] BLAHOVEC, Antonín. *Elektrotechnika I.* Praha: Informatorium, 2005. 191 s. ISBN 80-7333-043-1

Příloha A - Uživatelská příručka

A.1 Překlad a instalace programů

Informace o instalaci či překladu programů též naleznete v *REAME.txt* souborech umístěných v příslušných adresářích na přiloženém DVD.

A.1.1 Raspberry Pi

Program pro Raspberry Pi naleznete na DVD v příloze nebo je ke stažení online na https://bitbucket.org/payne-x6/raspberrypi-rover-gpio-layout. DVD obsahuje přeloženou verzi programu (umístění *binary/RPi/*) a zdrojový kód (umístění *source/RPi/*). Oba adresáře obsahují instalační skript a soubory potřebné pro instalaci.

```
?> cd binary/RPi/
binary/RPi/?> chmod u+x install.sh
binary/RPI/?> sudo ./install.sh
...
prog/RPi/?> sudo RPIRover
```

Obr. A.1 - Instalace a spuštění programu pro Raspberry Pi

Pro instalaci pomocí binární (spustitelné) verze programu se pomocí příkazu *cd* nasměrujte na složku s projektem (adresář *binary/RPi/*). Zde se nachází spustitelná verze programu. Tento program obsahuje závislosti ke knihovně *wiringPi*. Pro spuštění musí být nainstalována tato knihovna. To se projeví, pokud se pokusíte spustit aplikaci přímo z tohoto adresáře (příkaz *./RPIRover*). Program však nebude mít vytvořené potřebné konfigurační soubory. Pokud se program podaří spustit ale program se ukončí z důvodu chybějících konfiguračních souborů, pak pro jeho instalaci do systému slouží skript *install.sh*, který se nachází v aktuálním adresáři. Tomu pomocí příkazu *chmod* skriptu *install.sh* nastavte práva pro spuštění a spusťte (viz Obr. A.1). Pokud instalační průvodce proběhne správně, je možné program spustit z libovolného umístění (samotný program naleznete ve složce */opt/RPIRover/*). Pro spuštění programu jsou potřeba práva superuživatele (uživatel *root* nebo skupina *sudo*) (viz Obr. A.1). Instalační průvodce nabízí zaregistrování aplikace mezi

programy spuštěné po startu systému. Pokud tuto možnost zvolíte, program se pokusí přihlásit k serveru nastaveném v konfiguračním souboru při každém spuštění.

Pokud se však program spustit nepodaří z důvodu chybějících závislostí, je třeba program přeložit pro dané Raspberry Pi. Instalační skript se nachází na DVD ve složce '*source/RPi*'. Ten zobrazí průvodce instalací popsanou v části 5.4. Skriptu je nutné nastavit práva pro spuštění. Pro překlad je třeba mít nainstalován nástroj *CMake* a knihovnu *wiringPi*. V případě, že tyto závislosti chybí, se skript pokusí tyto závislosti doinstalovat. Spuštění po instalaci je možné pomocí příkazu *sudo RPIRover* (viz Obr. A.2), nebo (pokud jste tak zvolili při instalaci) se spustí při startu systému.

```
?> cd source/RPi/
source/RPi/?> chmod u+x install.sh
source/RPI/?> sudo ./install.sh
...
source/RPi/?> sudo RPIRover
```

Obr. A.2 – Překlad, instalace a spuštění programu pro Raspberry Pi

A.1.2 Osobní počítač

Program pro vzdálené ovládání Raspberry Pi z PC naleznete na přiloženém DVD nebo je ke stažení online na https://bitbucket.org/payne-x6/raspberrypi-rover-remote-control.

Pro 64bitovou verzi operačního systému Linux jsem vytvořil spustitelnou verzi programu a instalační skript. Pokud máte nainstalované knihovny *Qt*, je možné program spustit přímo díky přiloženému spustitelnému souboru (na DVD soubor *'binary/Desktop/linux_x64/RaspberryPiServer/RaspberryPiServer'*). Pokud knihovny *Qt* předinstalovány nemáte, je možné pro spuštění použít skipt nacházející se na stejném umístění s názvem *RaspberryPiServer.sh*. Ten však stále závisí na knihovnách, které se v systému nemusí nacházet. V tento moment je třeba program přeložit přímo pro daný počítač.

Pro operační systém Windows se mi spustitelnou verzi programu (která by byla nezávislá na Qt knihovnách, nebo by je dokázala připojit z adresáře programu) vytvořit nepodařilo. Pokud však Qt knihovnami váš operační systém disponuje, je možné program

spustit pomocí souboru v adresáři *binary/Desktop/win_x64/*.

Pro přeložení programu na osobním počítači je potřeba mít nainstalovaný nástroj *Qt*. Pokud nemáte, je potřeba si *Qt* doinstalovat ručně. To se na různých platformách podstatně liší, návod lze nalézt zde: http://doc.qt.io/qt-5/gettingstarted.html.

Pro překlad na operačním systému Linux se pro překlad nachází na přiloženém DVD v adresáři *source/Desktop/* skript *install.sh*. Je potřeba udělit mu práva pro spuštění pomocí příkazu *chmod*. Po spuštění skriptu, se zkompilují zdrojové soubory do spustitelného programu s názvem *RaspberryPiServer*, který skript přesune do složky */opt/RaspberryPiServer/* a vytvoří na něj symbolický *link* (odkaz) ve složce */usr/bin/*. Tento program spustíte příkazem *RaspberryPiServer*.

Pro překlad na operačním systému Windows je třeba využít nástroj QMake. Pokud se nasměrujete do složky se zdrojovými soubory programu (adresář *source/Desktop/src*), je možný překlad pomocí příkazu '*qmake -config release'* (viz. Obr. A.3).

```
?> cd source
source/?> cd Desktop
source/Desktop/?> cd src
source/Desktop/src?> qmake -config release
```

Obr. A.3 – Překlad desktopové aplikace pro platformu Windows

A.2 První spuštění

Po instalaci programu na Raspberry Pi je třeba před prvním spuštěním v konfiguračním souboru *'/etc/raspberrycpp/network.conf'* nastavit adresu serveru, který bude Raspberry Pi ovládat. To provedete nastavením IP adresy nebo doménového jména serveru příkazem *bind* (viz část 5.1). Nyní je možné Raspberry Pi restartovat, nebo spustit aplikaci příkazem *RPIRover*.

A.3 Vzdálené ovládání

Před spuštěním aplikace na Raspberry Pi je nutné mít naslouchající server na osobním počítači. Po spuštění aplikace se zobrazí okno s panelem ovládacích nástrojů. Naslouchání

serveru spustíte tlačítkem na Obr. 6.1 označeném 1. Server naslouchá, pokud je tlačítko zelené. Po připojení klienta k serveru se zobrazí dialogové okno. Pomocí tohoto dialogového okna lze komunikaci s klientem přijmout či odmítnout, popř. klienta pojmenovat. Tlačítka na Obr. 6.1 označené 2 slouží k přepínání mezi připojenými klienty. Tlačítko na Obr. 6.1 označené 3 slouží ke vzdálenému vypnutí Raspberry Pi.

Nastavení GPIO pinů klienta lze měnit pomocí ovládacích prvků na Obr. 6.1 označených 6. Po přepnutí na ovladač (záložka *Controller* na Obr. 6.1 označena 5), zobrazí se okno se seznamem pinů (na Obr. 6.2 označena 2) a ovladačem pohybu (na Obr. 6.2 označen 3). Před začátkem ovládání pohybu vozítka zkontrolujte, zda má vozítko spuštěné napájení motorů. Pohyb se ovládá pomocí myši. Stiskem tlačítka myši v oblasti kruhu ovladače motorů (na Obr. 6.2 označen 3) a následným pohybem myši měníte směr pohybu vozítka.

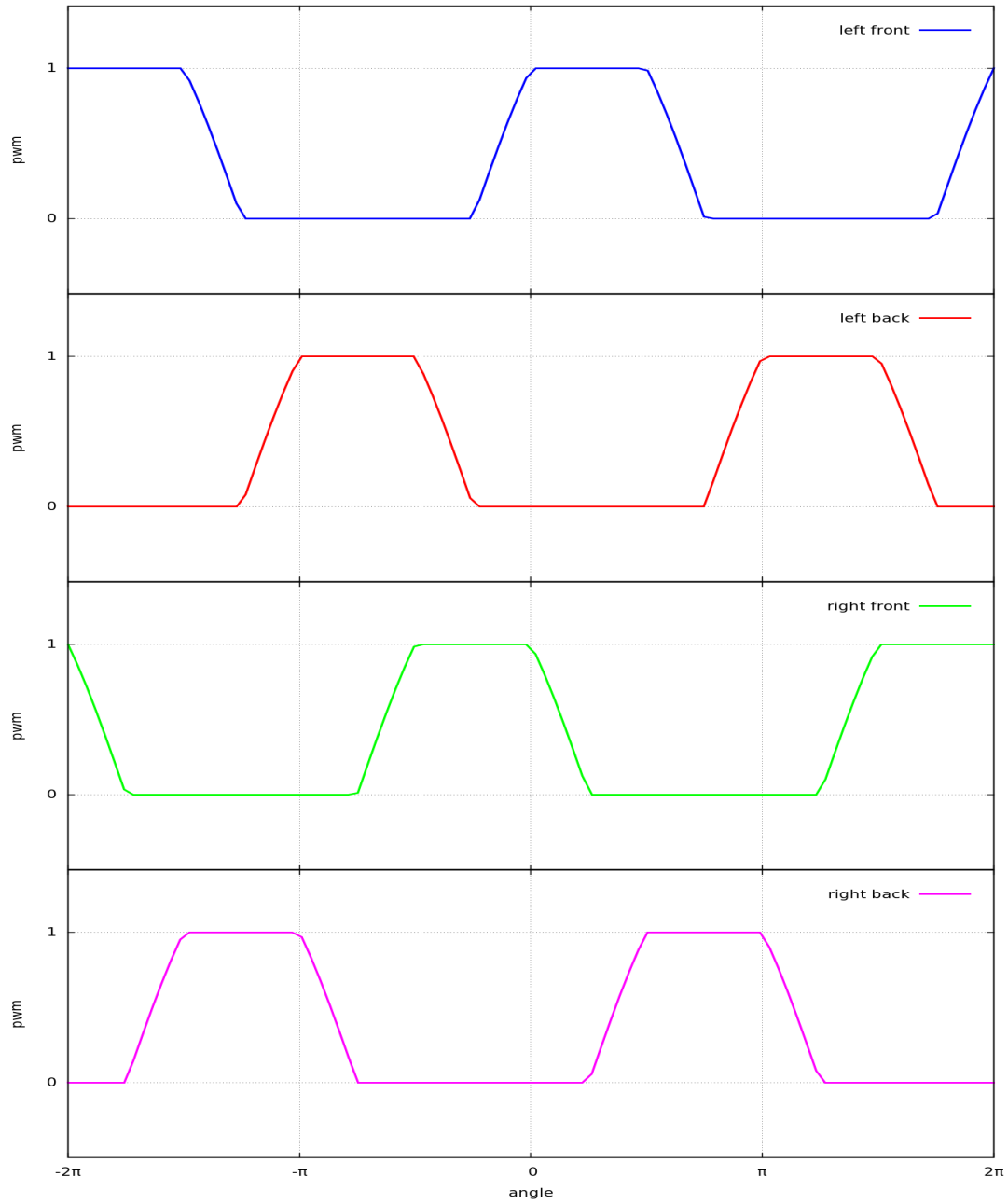
Příloha B – Přibližná cena komponent vozítka

Tab. B.1 - Zvolené komponenty pro sestavení vozítka včetně jejich přibližné ceny.

Kompletní robotický podvozek[32]	16,70 EUR
CSI Camera pro Raspberry Pi[33]	21,50 EUR
RTK Motor Controller Board[34]	19,95 EUR
Ultrazvukový senzor vzdálenosti[35]	100 CZK
Raspberry Pi 2 model B[36]	1 039 CZK
Napájecí kabel pro Raspberry Pi[37]	189 CZK
Wi-Fi USB Adapter[38]	389 CZK
Celkem:	**1 717 CZK + 58,15 EUR (~ 1 572 CZK)**
	~ 3 289 CZK

32 http://www.rlx.sk/sk/motor-driver/2968-2wd-mobile-platform-kit-er-rkt00100m.html
33 http://www.rlx.sk/sk/raspberry-pi/1450-rpi-camera-board-raspberry-pi-camera-board-5mp.html
34 https://www.adafruit.com/products/1687
35 http://www.easyduino.cz/Ultrazvukovy-senzor-HC-SR04-d43.htm?tab=description
36 http://rpishop.cz/raspberry-pi-pocitace/170-raspberry-pi-2-1024-mb-ram.html
37 http://rpishop.cz/kategorie/21-1a-originalni-napajeci-zdroj-pro-raspberry-pi-.html
38 http://rpishop.cz/raspberry-pi-prislusenstvi/100-wi-fi-usb-adapter-usb-edimax-wireless-nano-.html

Příloha C – Funkce výstupních PWM signálů



Obr. C.1 - Graf stříd PWM signálů na jednotlivých pinech závislé na požadovaném úhlu pohybu[39].

39 Graf vygenerován pomocí GNUPlot. Skript je součástí přiloženého DVD (v adresáři *other/*).