Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvantitativních metod

Využití Raspberry Pi pro autonomní řízení vozidla

Bakalářská práce

Autor: Martin Polreich
Studijní obor: Aplikovaná informatika

Vedoucí práce: Filip Malý, doc. Ing. Ph.D.

Hradec Králové duben 2017

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne Martin Polreich

Anotace

Práce se zabývá návrhem a implementací systému pro autonomní řízení vozidla pomocí Raspberry Pi. První část obsahuje teoretické informace o programování Raspberry Pi. Druhá část se zabývá problematikou autonomním vozidel a nastiňuje některé problémy. Třetí a čtvrtá část se věnuje samotnému návrhu a implementaci systému, který umožní pohyb vozidla a detekci překážek.

Annotation

Title: Using Raspberry Pi for autonomous car driving

This bachelor thesis deals with designing and implementing of a system for autonomous car driving using Raspberry Pi. First part consists of theoretical information about programming Raspberry Pi. Second part follows up problematics related to autonomous vehicle and outlines some of the major problems. Third and fourth part includes design and implementation of the system itself. This system allows movement of the vehicle and detection of obstacles.

Obsah

1 Úvod

O automatizovaných vozidlech je slyšet stále více a více. Už se nejedná o představy a sny inženýrů a vývojářů, ale skutečně automatická vozidla již v omezeném počtu jezdí v běžném silničním provozu.

Pro příklad lze uvést několik hlavních událostí z poslední doby. V roce 2008 Anthony Levandowski začíná pracovat na projektu autonomního vozidla. Tento projekt přesvědčí vedoucí pracovníky Googlu o tom, že by se této technologii měli věnovat více (1). V červnu 2011 představitelé státu Nevada v USA odhlasovali zákon umožňující testování autonomních vozidel v běžném silničním provozu (2). V roce 2015 Google poprvé předvedl v provozu vozidlo, které nemělo volant ani pedály (3). O rok později uvedla společnost Uber do provozu několik autonomních vozidel, které dopraví zákazníka na místo určení (4).

Z vývoje událostí v tomto segmentu za posledních pár let je jasné, že to nebude trvat moc dlouho a plně automatická vozidla se stanou nedílnou součástí našich každodenních životů.

Velké změny v oblasti automatických vozidel a zároveň možnost seznámení se s technologií Raspberry Pi byly hlavními impulzy k vypracování této práce.

Celý projekt se věnuje návrhu, sestavení a naprogramování autonomního vozidla, které by mělo být schopno rozpoznat překážku v cestě, zastavit a po odstranění této překážky pokračovat v jízdě. Hlavním prvkem celého systému je počítač Raspberry Pi s několika přídavnými moduly, který pomocí senzorů sbírá informace z okolí, vyhodnocuje je a podle naprogramovaného algoritmu poté dává pokyny motorům poháněcím celé vozidlo.

2 Programování Raspberry Pi

Tato kapitola se věnuje představení Raspberry Pi po straně softwaru i hardwaru, možnostem konfigurace a využití.

2.1 Raspberry Pi

Jedná se o jednodeskový počítač, tzv. SBC [1], který byl původně vyvinut charitou Raspberry Pi Foundation za účelem popularizace a zlepšení výuk počítačových věd v rozvojových zemích (5).

První generace byla uvolněna do prodeje počátkem roku 2012, a to ve dvou verzích. Byly jimi Model A a B. Hlavní rozdíl mezi oběma modely je ve velikosti paměti RAM a počtu konektorů. Model A první generace obsahoval 256 MB RAM a jeden USB port. Oproti tomu model B měl dvojnásobnou operační paměť, dva USB porty, a navíc i 100 Mb ethernetový konektor. Už od počátku neměly Raspberry Pi rozhraní pro připojení HDD ani SSD, a proto je nahrávání operačního systému řešeno přes SDHC slot. SD karty slouží také jako trvalé úložiště. V roce 2014 byly vydány vylepšené verze stále patřící do první generace označené jako Model A+ a B+. Oproti svému předchůdci Model A+ obsahoval větší množství GPIO [2], měl nižší spotřebu energie a místo SDHC slotu byl použit menší MicroSDHC slot. Model B+ má ještě více GPIO a další dvojici USB portů (6; 7).

O 3 roky později po uvedení prvního modelu se začala prodávat druhá generace. Oproti první generaci už pouze Model B. Ten nabízel uživatelům vyšší výpočetní výkon díky ARM procesoru s frekvencí 900 MHz, což je o 200 MhZ více než první generace. Kromě toho také obsahoval 1GB sdílenou operační paměť, čtveřici USB portů, HDMI a ethernetový port a také kombinovaný 3,5 mm audio jack (6).

V únoru 2016 byla uvolněna zatím nejnovější verze Raspberry Pi pod označením třetí generace a také pouze v modelu B. Jedná se zatím o nejvýkonnější verzi, která je osazena 64bitovým procesorem ARM verze 8 o frekvenci 1,2 GHz. Oproti Pi v2 má také navíc Wi-Fi kartu pracující na standardu 802.11n a integrovaný adaptér Bluetooth 4.1 (6). Ceny jednotlivých modelů se pohybují od 20 US$ do 35 US$ (8).

[1] Single-board computer
[2] General Purpose Input/Ouput

2.1.1 Příklady využití

Způsobů využití Raspberry Pi je nepřeberné množství. Vzhledem ke své univerzálnosti je nejčastěji využíván jako multimediální centrum, ovládací počítač pro 3D tiskárny, pro ovládání robotů a vozidel, sběr a vyhodnocení dat z různých senzorů a podobně. Mezi nejzajímavější projekty bezpochyby patří lokalizační zařízení navržené v (9), WarBox popsaný v (10) umožňující wardriving[3] pomocí zařízení sestaveného právě z Raspberry Pi a automatizace domova dle návrhu v (11).

2.2 Software

Zakoupené Raspberry Pi je vždy bez operačního systému, a tudíž je potřeba ho nejprve doinstalovat. Je jen na uživateli, jaký operační systém si zvolí, ale Raspbian Foundation oficiálně doporučuje systém Raspbian. Ten je odvozený z Debianu a je speciálně navržen pro použití na Raspberry Pi (12). Dalšími možnostmi jsou například Ubuntu Mate, Snappy Ubuntu Core, Windows 10 IoT Core, OSMC nebo LibreELEC (13).

2.2.1 Raspbian

Jedná se o operační systém vyvinutý Mikem Thompsonem a Peterem Greenem za podpory komunity z řad uživatelů Raspberry Pi. Jak již napovídá název, jedná se o neoficiální port systému Debian, konkrétně verze 7.0 s kódovým označením Wheezy (14). První stabilní verze tohoto systému byla uvolněna v červenci 2012 a od té doby je stále v aktivním vývoji (15). Raspbian je možno, stejně jako většinu ostatních systémů, nainstalovat pomocí instalačního manažeru NOOBS. Použití NOOBS je velice jednoduché a intuitivní. Instalace probíhá z microSD karty, kterou si může uživatel lehce vytvořit, ale dá se také zakoupit již vytvořená a připravená k použití od většiny distributorů Raspberry Pi (16). Raspbian je vybaven předinstalovaným základním softwarem jako například Python, Scratch, Sonic Pi, Java, Mathematica a dalšími. Dále obsahuje desktopové prostředí nazvané PIXEL. Toto prostředí umožňuje uživateli ovládat systém pomocí oken jako například v systému Windows nebo na linuxových distribucích pomocí obdobných desktopových prostředí Unity, GNOME, KDE apod. (12; 17).

[3] Vyhledání bezdrátových Wi-Fi sítí během jízdy ve vozidle.

2.2.2 Ubuntu Mate

Ubuntu Mate je jedna z mnoha linuxových distribucí, která je odvozená od původního Ubuntu. Hlavním rozdílem je použití vlastního desktopového prostředí nazvaného MATE. Tato distribuce míří především na uživatele, kteří chtějí snadno použitelný operační systém s jednoduchým nastavením a klasickým vzhledem plochy. Základním programovým vybavením Ubuntu MATE je mimo jiné webový prohlížeč Firefox, e-mailový klient Thunderbird, přehrávače Rhytmbox a VLC a další (18). K instalaci lze opět použít microSD karty, avšak již ne za pomocí manažeru NOOBS, jako tomu bylo v případě Raspbianu. Instalace je již tedy složitější a vyžaduje základní znalosti příkazů pro práci s terminálem v Linuxu (19).

2.2.3 Snappy Ubuntu Core

Tento operační systém byl poprvé představen Markem Shuttleworthem koncem roku 2014. Jedná se o speciální technologii, která kombinuje minimální systém Ubuntu Core, který běží na fyzickém počítači, s tzv. luskacími transakčními aktualizacemi. Hlavním cílem Ubuntu Core je minimalizovat prostředí jako základ pro obrazy aplikací nebo kontejnery Linuxu jako např. Docker. Tato technologie zajišťuje, aby uživatelské aplikace nemohly zasahovat do systému a aby aktualizace byly co nejrychlejší a nejmenší. Odsud pochází název „luskací“ aktualizace – proběhne rychlostí lusknutí prsty (20). Je tedy evidentní, že tato distribuce je určena pro specializované instalace, ale už nemusí být vhodná pro běžného uživatele.

2.2.4 Windows IoT Core

Jak název napovídá, tento systém je určen pro Internet of Things neboli Internet věcí. Jedná se tedy o snahu Microsoftu proniknout do odvětví tzv. Embedded systems (21).

2.2.5 LibreELEC

LibreELEC je jedna z linuxových distribucí, která je určená k tomu, aby na ní běžel populární systém pro multimediální centra KODI (dříve nazýván XBMC). Projekt je řízen týmem především dobrovolníků, kteří se vedením a vývojem zabývají ve svém volném čase, a samozřejmě také rozsáhlou komunitou uživatelů (22).

2.3 Hardware

Každá verze Raspberry Pi se hardwarově liší. Tato část popisuje hardwarové vybavení třetí verze modelu B, která je použita v celé této práci.

Raspberry Pi 3 Model B je osazeno 64bitovým ARM procesorem s frekvencí 1,2 GHz, Wi-Fi modulem s podporou standardu 802.11n, Bluetooth 4.1 a Bluetooth Low Energy moduly. Velikost operační paměti je 1 GB a jako grafický čip je použit VideoCore IV 3D.

2.3.1 Porty a piny

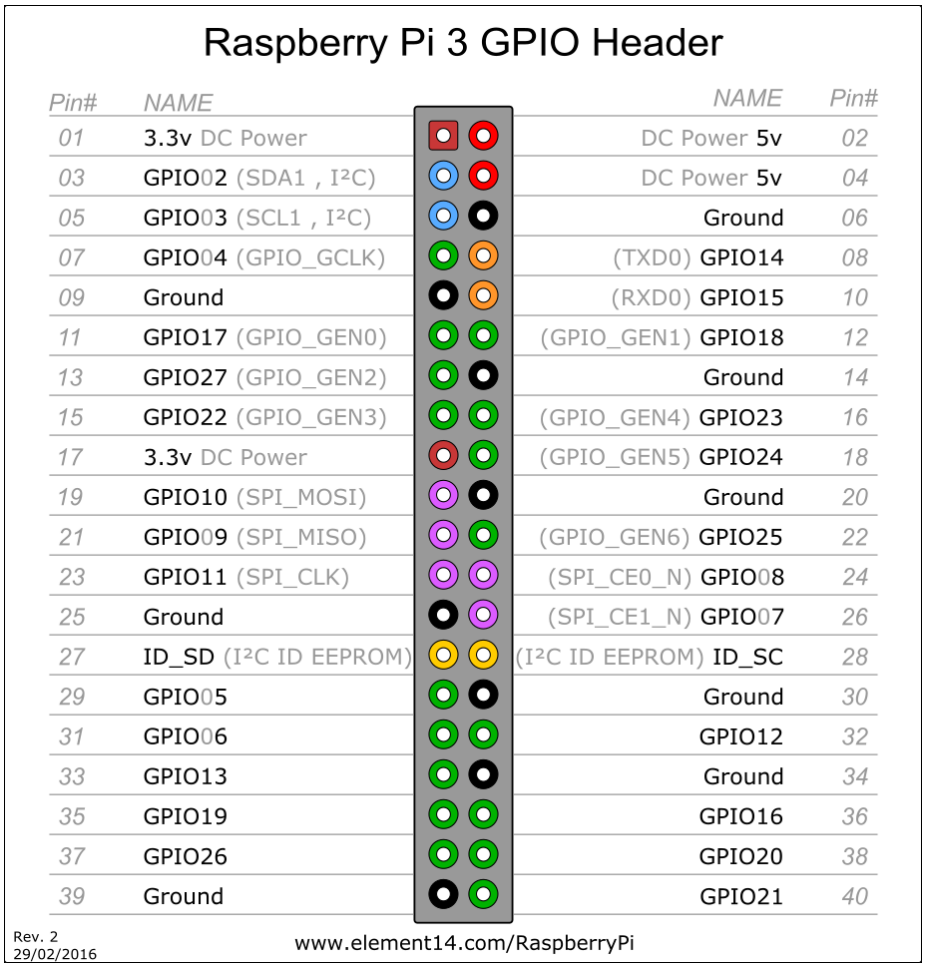
Raspberry Pi 3 Model B poskytuje následující porty a piny:

- 4 USB porty
- HDMI port
- Ethernet port
- MicroSD slot
- Kombinovaný 3,5 mm audio jack konektor
- CSI – pro připojení kamery
- DSI – pro připojení displeje
- 40 GPIO pinů

2.3.2 GPIO piny

Slouží k připojení libovolných zařízení a komponent k Raspberry Pi. U modelu 3 je k dispozici sběrnice se 40 piny a z toho 17 jich je možno použít jako univerzální konektory. Dále je zde dvakrát výstup s napětím 5 V, dvakrát s 3,3 V a osmkrát uzemnění. Jako další je možné využít výstupy sběrnic SPI, I2C a další (23). Více informací o jednotlivých sběrnicích je uvedeno v kapitolách 2.3.3 a 2.3.4. Raspberry Pi používá na GPIO 3,3V logiku, což znamená, že logická 1 je vyjádřena výstupem o tomto napětí. Při logické nule je na výstupu nulové napětí (24).

V našem projektu je využita čtveřice univerzálních GPIO pinů – pro každý senzor dva. Senzory ani motory nejsou napájený z GPIO pinů, ale je použit vlastní zdroj napětí. Více o napájení celého systému je uvedeno v kapitole 5.1.2.



Obrázek 2-I - Přehled GPIO pinů v Raspberry Pi 3 - Zdroj: (25)

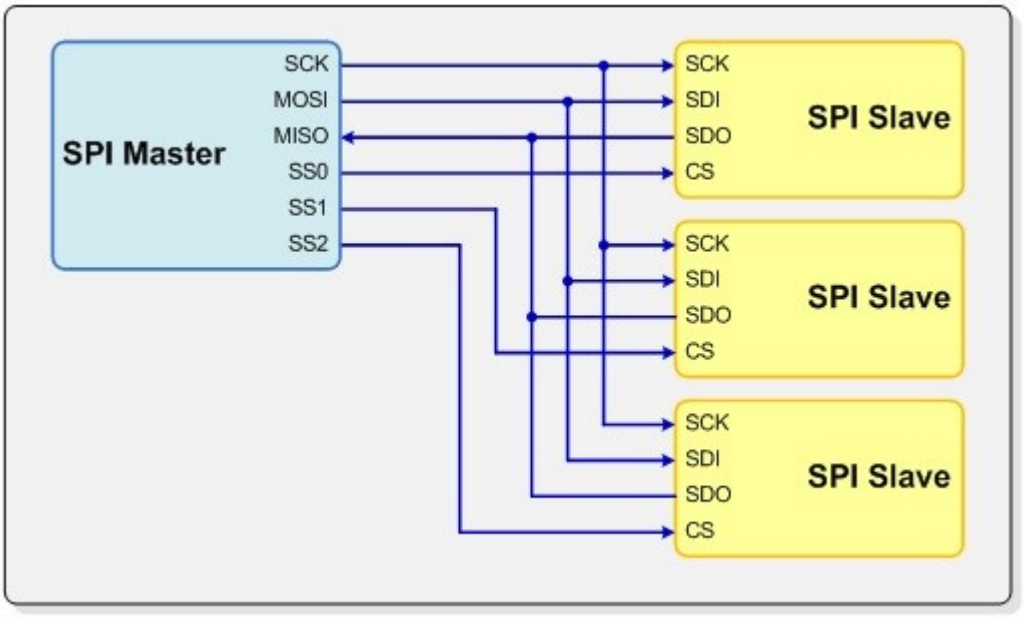
2.3.3 Sběrnice

Jak již bylo zmíněno, Raspberry Pi nabízí několik sběrnic pro připojení zařízení. Nejpoužívanější z těchto jsou sběrnice SPI a I²C, které jsou popsány v následujících dvou podkapitolách.

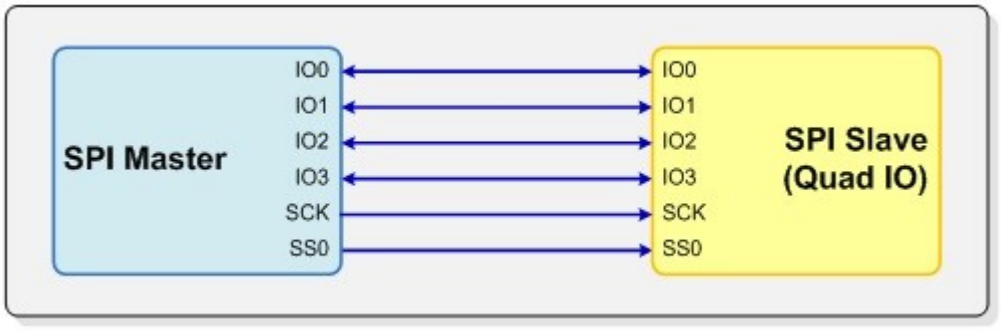
2.3.3.1 Sběrnice SPI

Zkratka SPI pochází z anglického Serial Peripheral Interface. Jak název napovídá, jedná se o sériové rozhraní, které umožňuje synchronní sériovou komunikaci. SPI bylo vyvinuto firmou Motorola. Využívá full-duplex komunikaci, což znamená, že data mohou být přenášena obousměrně zároveň ve stejný okamžik. Komunikace přes SPI může být konfigurována několika způsoby. Prvním z nich je konfigurace, ve které je propojen master a slave pomocí 4 signálů. Jedná se o SCK, který je využíván pro komunikaci mezi master a slave, dále pak MOSI, MISO a SS. V případě zapojení více slaves je možné signály SCK, MOSI a MISO sdílet, jelikož linka SS je pro každý slave oddělena. Detailněji lze toto

zapojení vidět na obrázku 2-II. Další možnou konfigurací je zapojení s využitím více IO propojení. Na obrázku 2-III je zobrazeno zapojení tzv. Quad-IO. Hlavní výhodou využití Multi-IO konfigurací je, že nabízí vyšší datovou propustnost (26).



Obrázek 2-II - Zapojení sběrnice SPI se 3 slaves – Zdroj: (26)

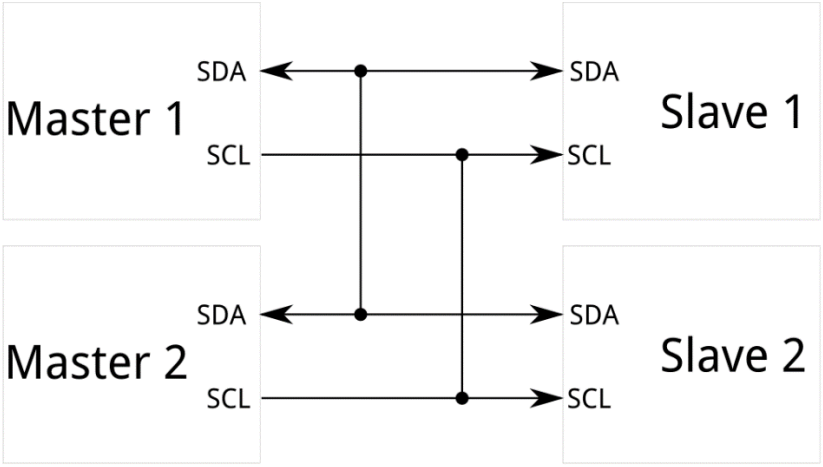


Obrázek 2-III - Zapojení SPI s jedním slave pomocí Quad-IO – Zdroj: (26)

2.3.3.2 Sběrnice I²C

Zkratka znamená Inter-integrated Circuit. Jedná se o sériovou sběrnici vyvinutou firmou Philips Semiconductors (nyní NXP Semiconductors). V principu funkčnosti se velmi podobá sběrnici SPI. Hlavním rozdílem oproti SPI je, že se jedná o tzv. Multi-master sběrnici, což znamená, že je možné zapojit více masterů (27). Dalším rozdílem je využití pouze 2 signálů pojmenovaných SDA a SCL. I²C využívá 7bitové adresování, teoreticky je

tedy možné připojit až 128 různých zařízení. Nicméně specifikace I²C nabízí možnost rozšíření tohoto počtu až na 1024 pomocí 10bitového adresovacího schématu. 10bitová adresa se skládá ze dvou adresních slov. První slovo je tvořeno sekvencí 5 bitů "11110" a prvních dvou bitů adresy. Druhé slovo obsahuje zbylých 8 bitů adresy daného zařízení. Zavolání dané sekvence 5 bitů na začátku volání má za účel informování všech zařízení na sběrnici o tom, že bude následovat 10bitová adresa (28). Složení takové adresy je možné vidět na následujícím obrázku 2-IV.



Obrázek 2-IV - Zapojení I²C sběrnice se dvěma masters a dvěma slaves – Zdroj: (29)

2.3.3.3 Porovnání

Fyzický rozdíl mezi výše zmíněnými sběrnicemi je především v počtu použitých linek. Zatímco SPI vyžaduje 4 signály na propojení jednoho master a slave, sběrnici I²C stačí pouze 2. Dalším rozdílem je počet adresních bitů. Na druhou stranu I²C je omezena počtem adresních bitů, což lze ovšem vyřešit rozšířením zmíněným v předchozí podkapitole (28).

Z pohledu přenosu dat a rychlosti má jednoznačnou výhodu SPI díky tomu, že využívá full-duplex komunikaci. Rychlostní limit SPI sběrnice není definován a implementace často vykazují přenosové rychlosti přes 1 Mb/s. I²C nabízí dva rychlostní módy. Prvním z nich je Fast Mode+ s maximální rychlostí 1 Mb/s a druhým High Speed Mode s rychlostí až 3,4 Mb/s (28).

2.4 Programovací jazyky

Na Raspberry Pi je možné použít libovolné programovací jazyky zkompilovatelné na ARM procesorech, nicméně Raspberry Pi Foundation doporučuje jazyk Python. Ze základu jsou v Raspberry Pi se systémem Raspbian nainstalovány programovací jazyky Python, C, C++, Java, Scratch a Ruby (30). Při zpracování této bakalářské práce je využit programovací jazyk Python 3. Hlavními důvody tohoto výběru je největší podpora komunitou uživatelů Raspberry Pi mezi všemi jazyky a také dostupnost knihoven k některým hardwarovým součástkám taktéž v Pythonu 3.

Python je programovací jazyk, který vznikl roku 1991. Jeho autorem je Guido van Rossum (31). V dnešní době jsou nejpoužívanější dvě verze Pythonu. Jedná se o verzi 3.6 a 2.7, která je označována jako legacy. Hlavním důvodem, proč je stále rozšířena i legacy verze Pythonu je, že verze 3.x nejsou zpětně kompatibilní s 2.x. Z tohoto důvodu bylo potřeba přepsat všechny moduly tak, aby byly kompatibilní s Python 3 (32). Důsledkem toho je, že ještě v roce 2016, tedy 8 let po vydání první verze 3.0, používá stále minimálně polovina uživatelů verzi 2.7 (33). Současná verze obsahuje oproti legacy spoustu změn, například `print` již není příkaz, ale funkce. Dalšími změnami jsou například podpora pouze Unicode řetězců a s tím související zrušení funkce unicode() pro převod textu, sloučení datových typů long a int do jednoho typu int, zrušení symbolu <> jako označení nerovnosti a spoustu dalších (34; 35).

2.5 Možnosti připojení

Nejjednodušší způsob práce s Raspberry Pi je bezpochyby připojení monitoru pomocí HDMI, které má k dispozici a poté klávesnice a myši pomocí USB portů. Nicméně nejjednodušší v tomto případě neznamená nejvhodnější. Jako hlavní důvod lze uvést to, že SBC jsou nejčastěji umístěny v těsných prostorech nebo špatně přístupných místech a často jsou také na pohybujících se předmětech. V případě, kdy je počítač umístěn na špatně dostupném místě, ale nepohybuje se, lze použít připojení kabelem přes ethernetový port a pracovat s ním pomocí tohoto připojení. Pokud se však počítač pohybuje, nejlepší volbou je využít bezdrátové připojení. V dnešní době se nejčastěji používá připojení pomocí protokolů SSH nebo Telnet. Podrobněji se těmto způsobům připojení věnují následující podkapitoly.

2.5.1 Pevné připojení

Připojení takzvaně na pevno, tedy pomocí ethernetového kabelu, nalezne uplatnění v mnoha situacích. Například ve velkých prostorech, kde je rozmístěno více SBC, přičemž každý má nějakou funkci, kupříkladu každý měří pracovní podmínky v jiném místě výrobní haly. Všechny počítače pak mohou být připojeny přes ethernetové kabely a pomocí switche do jednoho řídícího počítače.

2.5.2 SSH

Pro potřeby tohoto projektu se jako nejlepší způsob připojení jeví připojení bezdrátové, a to hlavně z toho důvodu, že se vozidlo pohybuje, a tak je nevhodné, aby bylo připojeno kabelem.

Při programování bylo ve většině případů použito připojení přes SSH protokol. Zkratka SSH pochází z anglického Secure Shell a jedná se o síťový protokol, který umožňuje zabezpečené připojení přes nezabezpečené sítě. K tomu využívá šifrování odesílaných dat tak, aby je případný útočník nemohl nijak použít (36). Komunikace přes SSH protokol probíhá ve výchozím nastavení přes port 22.

Při připojování přes SSH lze použít dva hlavní způsoby autentizace – přihlášení heslem nebo pomocí RSA klíče. V prvním případě se uživatel přihlašuje pomocí kombinace uživatelského jména a hesla, v druhém případě se ověřuje pomocí klíče složeného ze dvou částí – veřejné a privátní. Veřejná je uložena na serveru a je spojena s daným uživatelským účtem. Pro úspěšné přihlášení je nutné připojit se za použití druhé, privátní, části klíče (37). Nejbezpečnější variantou je poté kombinace těchto dvou způsobů, kdy je po připojení s klíčem ještě nutno potvrdit identitu heslem (36).

V případě použití počítače s operačním systémem Linux není s SSH připojením žádný problém, jelikož většina distribucí je již vybavena openSSH klientem, který umožňuje rychlé připojení pomocí příkazu `ssh <username>@<host>`.

Pokud se však chce uživatel připojit přes SSH ze systému Windows, bude potřebovat SSH klienta. K dispozici je jich spoustu a ty nejpoužívanější jsou distribuovány jako freeware. Po nainstalování takového klienta stačí zadat uživatelské jméno a adresu hosta a připojit se.

2.5.3 Telnet

Dalším často používaným protokolem je Telnet. Hlavním rozdílem oproti výše zmíněnému SSH je, že Telnet nepoužívá žádné zabezpečení a data jsou přenášena jako prostý text, a to včetně hesel. Z tohoto důvodu je použití tohoto protokolu vhodné pouze v lokálních sítích (38). Komunikace probíhá ve výchozím nastavení přes port 23.

Telnet má výhodu pro uživatele Windows, jelikož klient je již přítomen v systému. Pro připojení tak stačí do příkazové řádky zadat příkaz `telnet <host>` a potvrdit. I přes to však spousta uživatelů využívá externí klienty.

Stejným příkazem se lze z terminálu připojit i při použití systému Linux.

3 Analýza problematiky autonomního vozidla

Myšlenka autonomního vozidla zde byla již dlouho, avšak teprve nyní se zdá, že by mohla být v brzké době realizována. Autonomní automobily testuje většina největších výrobců v tomto odvětví, a navíc i další hráči z jiných oborů. Bylo jen otázkou času, kdy k tomuto kroku lidé dospějí, však už pojmenování automobil je složenina starořeckého „auto“ tedy „samo“ a latinského „mobile“ což znamená „pohyblivý“. Celé slovo automobil by se tedy dalo počeštit jako „samohyb“. Vše již napovídá tomu, že automobil konečně v brzké budoucnosti dostojí svého jména.

Při přemýšlení o autonomních vozidlech v širším měřítku lze zjistit, že jsou již dávno využívána naprosto běžně. Problém spočívá v tom, že většina lidí si pod pojmem „autonomní vozidlo“ představí právě samořídi;cí automobil, avšak vozidla nejsou jenom auta. Už dávno jsou autonomní vozidla využívána například ve výrobních halách, kde automaticky přesouvají materiál a výrobky po hale z jednoho pracoviště na druhé podle naprogramovaného plánu, a přitom se řídí předem danou trasou. V tomto případě většinou následují čáru vyznačenou na podlaze. Dalším autonomním vozidlem, se kterým se již dnes můžeme běžně setkat, jsou různá kolejová vozidla, například metra nebo nadzemky, které v různých městech po celém světě již jezdí bez řidiče podle předem stanoveného jízdního řádu.

I přesto je však mezi těmito vozidly a „automobily budoucnosti“ velký rozdíl, a to právě v jejich autonomii. Výše jmenované příklady nejsou totiž ani tak autonomní, jako spíše automatické. Mají předem definovaný jízdní řád nebo trasu a tu následují, přičemž umí reagovat na předem naprogramované situace. Co však dělá vozidlo autonomním je schopnost reagovat na nepředvídané situace. Samočinný pohyb po vozovce vyžaduje mnohem více rozhodovacích rovin a mnohem komplexnější program, než je jízdní řád v automatickém vlaku metra. Autonomní automobil musí umět reagovat na nečekané podněty a předvídat chování ostatních účastníků silničního provozu, jako to dělá člověk, například když projíždí kolem školy nebo pracovníků na silnici. V běžném silničním provozu však čeká spousta dalších situací a momentů a nelze všechny naprogramovat do systému. Automobil musí mít určitou úroveň umělé inteligence, aby mohl včas předvídat a reagovat.

Pro autonomní vozidla, která se mají pohybovat v našem běžném světě, je nejdůležitější sběr dat o okolí. Hlavním zdrojem těchto informací jsou především senzory a kamery, které snímají dění kolem vozidla. Teprve pokud vozidlo získá dostatek těchto informací a v dostatečné kvalitě, může se na jejich základě rozhodovat o dalších akcích. Sběr dat a jejich uložení je tedy nezbytný a základní předpoklad pro správné rozhodování. A právě tady může nastat velký problém.

3.1 Omezené vnímání

Senzory na automobilu nejsou ani po připojení k výkonnému hardwaru tak sofistikované jako lidské smysly a vjemy. Řidič při jízdě po silnici instinktivně ví kudy má jet, a to i za zhoršených podmínek. Autonomní automobily toto neví díky instinktu, nýbrž většina z nich se řídí pomocí snímání bílých čar na silnici. Nejenom, že by takové auto bylo v Česku nepoužitelné, jelikož většina silnic nižších tříd nemá z naprosto nepochopitelných důvodů žádné vodicí čáry, ale také by mělo problémy v případě zhoršeného počasí. Velký problém by například způsobilo pokrytí celé vozovky sněhem nebo hustý déšť. Výrobci automobilů tvrdí, že těmto problémům lze předejít. Jedno z možných řešení je kombinace více senzorů tak, aby bylo mnohem těžší nebo nemožné ztratit pojem o vozovce. Mercedes-Benz například nabízí na svých automobilech 23 různých senzorů. Mezi nimi jsou například detektory svodidel, bariér, protijedoucích vozidel a také stromů podél silnice. To vše má napomoci vozidlu udržet se ve správném pruhu na vozovce, i když nejsou k dispozici žádné vodicí čáry (39).

3.2 Etický problém

Podle Bryanta Walkera Smithe, profesora na právnické škole a předsedy Výboru práva rozvíjejících se technologií při Národní akademii, je 90 % všech dopravních nehod způsobeno lidskou chybou. Dále poznamenává, že hlavní podíl na této statistice má vysoká rychlost, řízení v opilosti, agresivní řízení s ohledem na podmínky, ospalost a nesoustředěnost. Všechny tyto klíčové problémy by měla automatizace vozidel odstranit. S prvními plně automatizovanými vozidly se však objevuje úplně nový problém, který do teď neexistoval (40).

Problém spočívá hlavně v tom, že člověk a automatické vozidlo vidí svět kolem sebe odlišně. V momentě, kdy se řidič dostane do situace, kdy je nehoda nevyhnutelná, zareaguje podle svých přirozených reflexů tak, aby co nejvíce minimalizoval škody. Tato

reakce se zdá přirozená a řidič nad ní nemá čas přemýšlet. Otázkou však zůstává, co by mělo udělat automatizované vozidlo, které nemá přirozené reflexy a pouze následuje předem naprogramovaný kód. A právě zde vstupuje onen nový problém, či spíše etická otázka (40).

Jako ukázkový příklad tohoto dilematu je často uváděna následující situace. Vozidlo jede po silnici a náhle mu vjede to cesty cyklista. Vozidlo jede příliš rychle na to, aby stihlo zastavit, nehoda je nevyhnutelná. Jsou dva možné scénáře, jak může celá situace skončit. Jako první se může vozidlo rozhodnout, že chránit zdraví řidiče je důležitější, začne brzdit a srazí cyklistu. Druhou možností, je vyhodnocení, že zdraví cyklisty je ve větším ohrožení než zdraví řidiče, a tak strhne řízení a narazí do sloupu stojícího u silnice. V případě, že by ono vozidlo řídil člověk, jeho reakce by byla závislá na jeho přirozeném reflexu a nebyl by čas přemýšlet o této etické otázce. Automat ale takové reflexy nemá. Jeho chování v takové situaci musí být předem naprogramováno. A zde vzniká zmíněný problém, jak automobil na danou situaci naprogramovat, podle čeho se má vozidlo rozhodnout, čí život je důležitější chránit nebo kdo má větší šanci na přežití.

Patrick Lin, ředitel Skupiny pro etické otázky v oblasti rozvíjejících se technologií při California Polytechnic State University, tvrdí, že vyřešení této otázky je velký úkol, který průmysl musí vyřešit. Navíc podle něj není jasné, kdo má rozhodovat o těchto pravidlech (40).

4 Výběr vhodného hardwaru

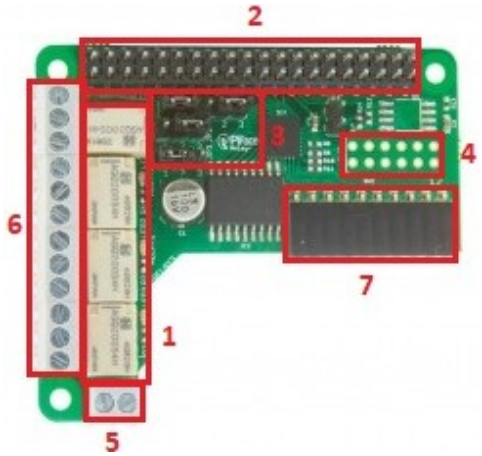
Výběr jednotlivých komponent, ze kterých se bude autonomní vozidlo skládat, je minimálně stejně důležitý jako naprogramování softwarové části, a proto je potřeba předem promyslet všechny požadavky a podle nich vybrat potřebné součástky. Hlavní součástkou je již zmíněný SBC Raspberry Pi, který je řídícím centrem celého zařízení. Dále jsou to dvě doplňující desky PiFace Relay Plus a PiFace Motor Extra, které jsou nezbytné pro ovládání motorů, ultrazvukové senzory poskytující informace o vzdálenosti od překážky, a nakonec samotné šasi vozidla se čtyřmi motory a koly.

4.1 Raspberry Pi

Jedná se o „mozek“ celého systému. Shromažďuje všechny informace, provádí veškeré výpočty a poté posílá signály s pokyny dalším komponentům. Podrobné informace o vybavení a fungování tohoto počítače jsou uvedeny v kapitole 2 a jejích podkapitolách, které se věnují zvlášť softwarové i hardwarové výbavě. V tomto projektu je použita verze 3 Model B.

4.2 PiFace Relay Plus

Jedná se o rozšiřující desku od firmy OpenLX SP Ltd., která vyvíjí a nabízí doplňky k Raspberry Pi pod obchodním označením PiFace.



Obrázek 4-I – PiFace Relay Plus – Zdroj: (41) – s vlastními úpravami autora

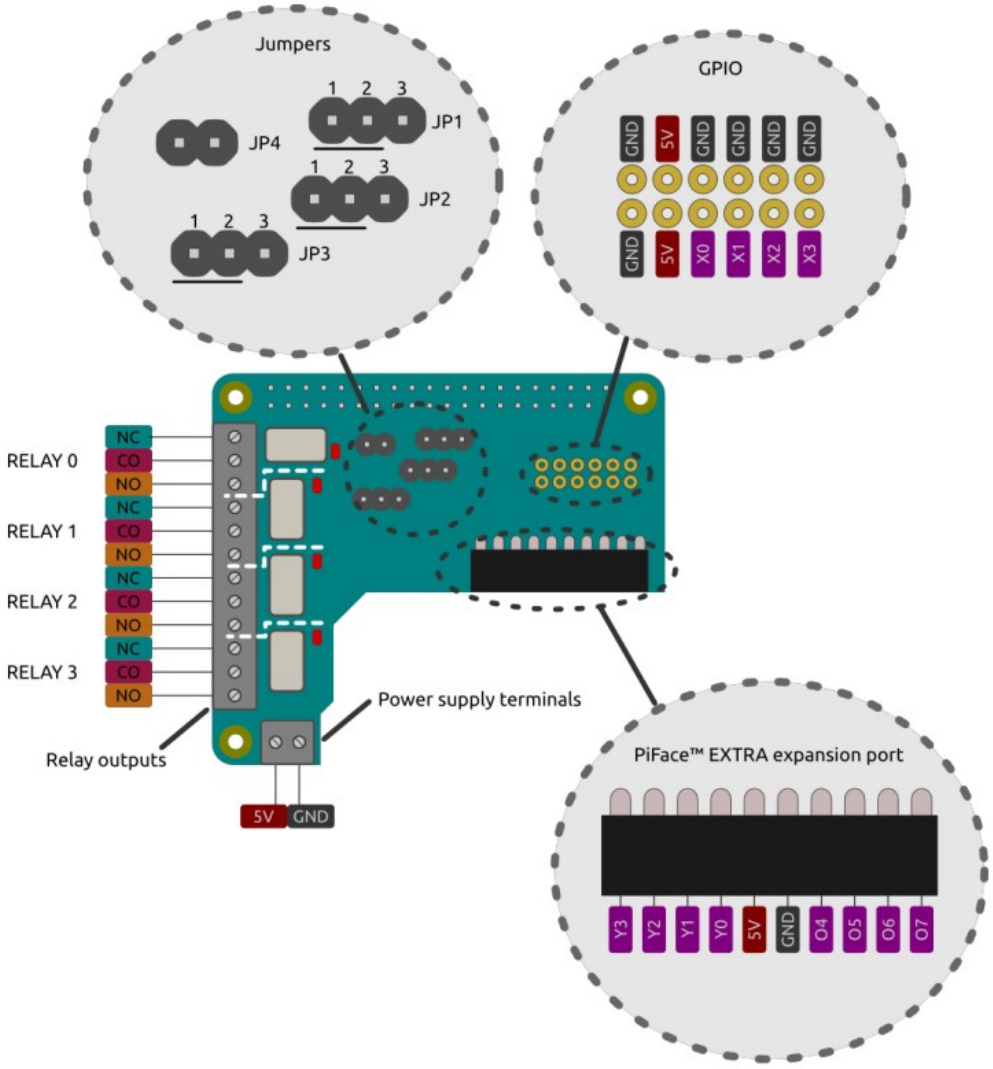
Hlavní součástkou celé desky jsou 4 spínací relé, které lze vidět na obrázku 4-I pod číslem 1. Dále je zde 30 pinů pro připojení do GPIO na Raspberry Pi – na obrázku 4-I označeny číslem 2; 4 jumpery pro základní nastavení funkce – označeny číslem 3; 12 přídavných GPIO pinů – číslo 4; svorky pro přívod napájení – číslo 5; svorky pro připojení spínaných zařízení – číslo 6 a rozšiřující port pro připojení desky PiFace EXTRA – číslo 7.

Jak již bylo zmíněno, hlavní funkci zajišťují relé, kdy každé ovládá jedno zařízení připojené k výstupním svorkám, takže umožňují spínání až 4 zařízení současně s maximálním napětím 20 V a proudem 1 A.

Piny pro připojení k Raspberry Pi přesně kopírují GPIO piny, takže celé připojení proběhne nasunutím desky právě na tyto piny. Zároveň lze pomocí nich připojit stejným způsobem k sobě až 8 PiFace Relay Plus desek a zvýšit tak značně počet připojených zařízení. K adresování jednotlivých desek v případě zapojení více než jedné se používají 3 jumpery, pomocí kterých se na každé nastaví unikátní adresa. Způsoby tohoto adresování lze vidět na obrázku 4-II a tabulce 4-A. Jumper číslo 4 je použit k nastavení napájení. V případě jeho propojení je deska propojena s napájením Raspberry Pi, v opačném případě deska s napájením propojena není a je tedy nutno použít k napájení příslušné svorky.

Pro možnosti dalšího ovládání nebo propojení s jinými periferiemi je možno použít 12 GPIO pinů přítomných na desce v konfiguraci 2x6 - lze detailně vidět na obrázku 4-II. Tyto disponují dvěma piny s +5 V, šesti GND piny a čtyřmi logickými, které používají, stejně jako Raspberry Pi, 3,3 voltovou logiku. GPIO piny mohou poskytnout každý maximálně 20 mA proudu.

Svorky pro ovládání připojených zařízení se skládají ze čtyř jednotlivých svorkovnic. Každá je určena pro jedno zařízení. Prostřední kontakt každé svorkovnice se při přepnutí relé přepíná na levý, respektive pravý. Tyto svorky nebyly v tomto projektu použity, neboť jsou použity ty, které poskytuje přídavný modul PiFace Motor EXTRA zapojený do rozšiřujícího portu. Tento port slouží pro připojení desek z řady PiFace EXTRA a důvod pro použití modulu Motor EXTRA je ten, že obsahuje dvojitý H-můstek a umožňuje tedy efektivní ovládání až čtyř motorů. Tomuto modulu se detailněji věnuje kapitola 4.3.



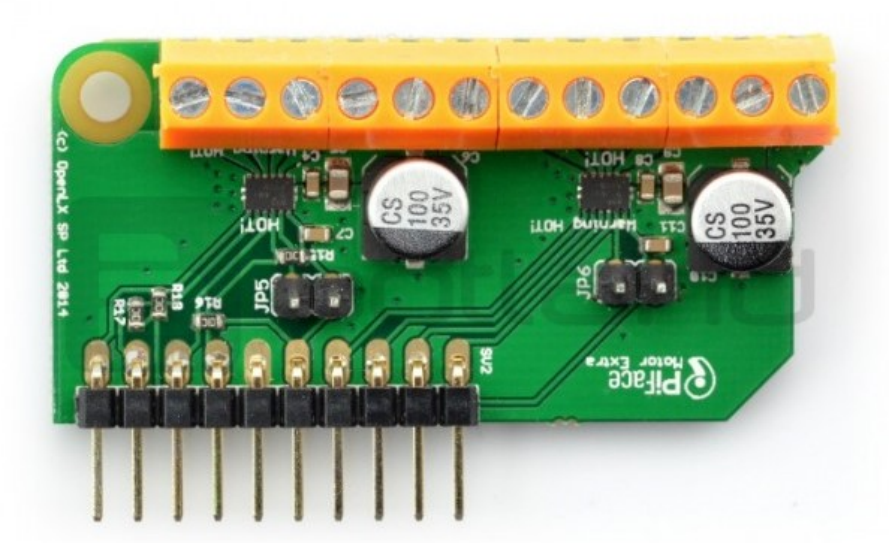
Obrázek 4-II – Detailní pohled na konektory na PiFace Relay Plus – Zdroj: (42)

Adresa	Bin. adresa	Pozice JP3	Pozice JP2	Pozice JP1
0	000	1-2	1-2	1-2
1	001	1-2	1-2	2-3
2	010	1-2	2-3	1-2
3	011	1-2	2-3	2-3
4	100	2-3	1-2	1-2
5	101	2-3	1-2	2-3
6	110	2-3	2-3	1-2
7	111	2-3	2-3	2-3

Tabulka 4-A – Možnosti adresování desek pomocí jumperů – Zdroj dat: (42)

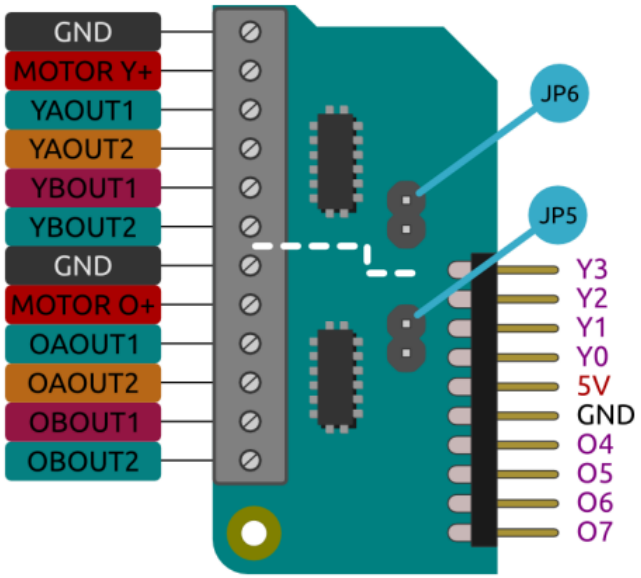
4.3 PiFace Motor EXTRA

Jedná se jednu z několika desek vyvinutých firmou OpenLX SP Ltd. pro použití v kombinaci s moduly PiFace Relay Plus. Jedná se o dvojitý H-můstek se čtyřmi samostatnými spínacími relé.



Obrázek 4-III - PiFace Motor EXTRA – Zdroj: (43)

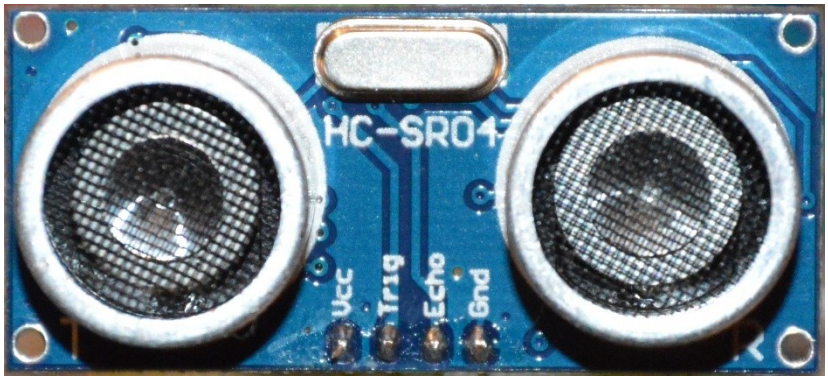
Připojuje se k modulu Relay Plus pomocí 10 pinů, které zapadnou do Extension portu. Dále můžeme na desce vidět čtveřici svorkovnic, každou se třemi výstupy. Tyto slouží k připojení motorů a napájení. Detail připojení je možno vidět na obrázku 4-IV. K dispozici jsou také 2 jumpery, které slouží k povolení fázového módu motorů.



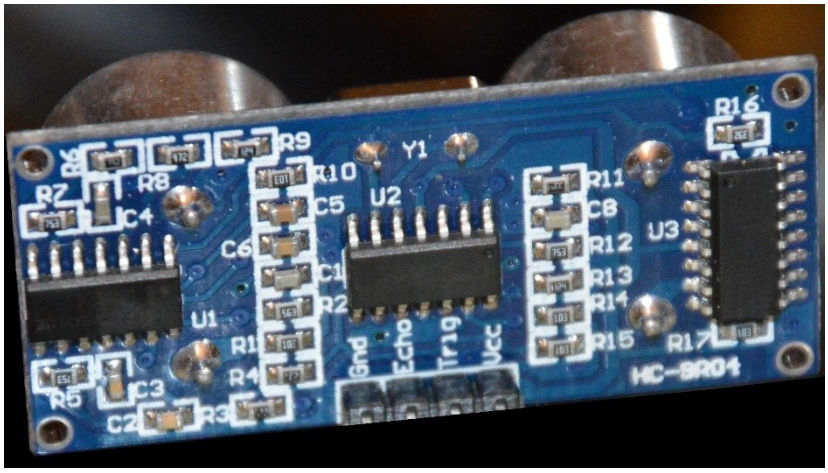
Obrázek 4-IV - Schéma modulu PiFace Motor EXTRA – Zdroj: (44)

4.4 Ultrazvukové senzory

Pro orientaci v prostoru bude auto využívat dvojici ultrazvukových senzorů, které budou poskytovat kontinuální informace o vzdálenosti od překážek. Konkrétně jsou použity senzory HC-SR04, které umožňují určení vzdálenosti v rozsahu od 2 cm do 4 m s přesností až 3 mm. Senzor obsahuje 4 konektory, z nichž 2 jsou určeny k napájení, jeden ke vstupu aktivačního impulzu a jeden jako výstup. Pro zahájení měření je potřeba přivést na vstupní konektor minimálně 10 uS impulz, na který sensor zareaguje vysláním osmi krátkých impulzů o frekvenci 40 kHz. V případě odražení těchto pulzů a jejich navrácení do přijímače se na výstupu objeví impulz, ze kterého lze po přepočtu určit vzdálenost k překážce. Danou vzdálenost v cm lze vypočítat z doby, která uběhla mezi vysláním pulsu a jeho návratem. Tento časový údaj se vydělí rychlostí zvuku a poté dvěma. Pro zajištění odstínění aktivačního pulzu od odezvy je potřeba zaručit odstup mezi měřeními minimálně 60 ms (45).



Obrázek 4-V – Senzor HC-SR04 (přední strana) – Zdroj: archiv autora



Obrázek 4-VI – Senzor HC-SR04 (zadní strana) – Zdroj: archiv autora

5 Návrh a implementace

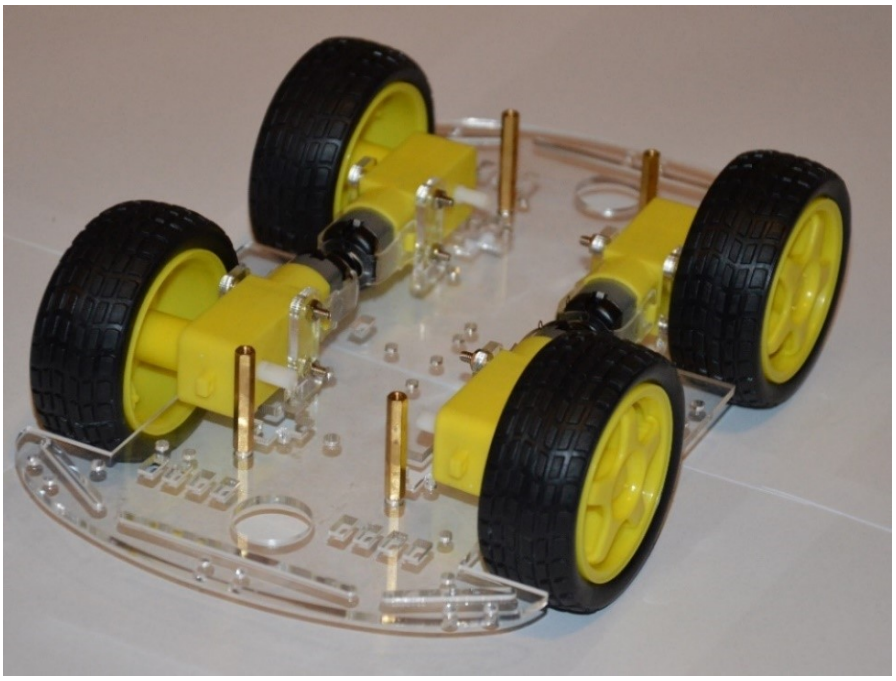
Tato kapitola popisuje jednotlivé kroky návrhu a následné implementace vybraných postupů za účelem sestavení autonomního vozidla. Věnuje se zvlášť hardwarové a softwarové části.

5.1 Hardwarová část

Hlavní a zároveň nosnou částí celého vozidla je šasi vyrobené laserovým vyřezáním z plastu. Jedná se o dvě totožné desky spojené dohromady pomocí šroubů a oddělené distančními sloupky. V obou hlavních deskách je několik perforací a otvorů, které slouží pro připevnění ostatních součástek a pro vedení vodičů a kabelů. K pohonu auta jsou použity 4 stejnosměrné motory, pro každé kolo jeden. Motory jsou připevněny k šasi také pomocí dílů vyřezaných laserem z plastu a šroubů. Sestavené šasi s motory, koly a distančními sloupky lze vidět na obrázku 5-I.

Samotné Raspberry Pi je umístěno na „střeše“ vozidla. Toto umístění se nemusí zdát příliš vhodné, jelikož jsou všechny elektronické komponenty odhaleny, ale z důvodů prototypizace a dobrého přístupu ke všem součástkám je to nejlepší volba. Případná produkční verze produktu by měla všechny elektronické komponenty schované uvnitř šasi tak, aby nebyly vidět, a také, aby nemohlo dojít k jejich náhodnému poškození. Na Raspberry Pi jsou napojeny dva moduly pro ovládání PiFace. Jejich podrobný popis je uveden v kapitole 4. Vpředu i vzadu se nacházejí ultrazvukové senzory, které poskytují informace o vzdálenosti vozidla od překážky. Další informace o funkci těchto senzorů lze nalézt v kapitole 4.4.

Při vývoji hardwaru byla zpočátku použita k propojení komponent nepájivá pole, protože se jedná o nejlepší metodu v raných fázích, kdy se dělají velké změny v zapojení a systém se teprve navrhuje.



Obrázek 5-I - Šasi vozidla s koly a motory (bez horní desky) - Zdroj: archiv autora

5.1.1 Napájení

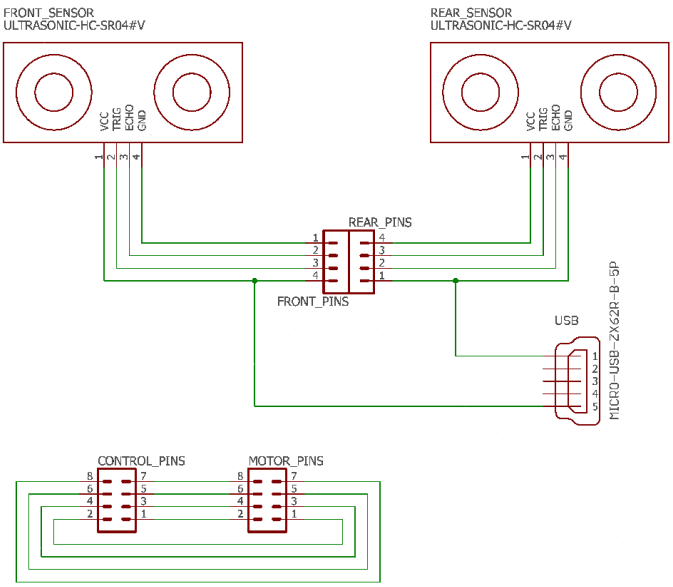
Nejjednodušším způsobem napájení Raspberry Pi je připojení pomocí microUSB konektoru, který je k tomu určen. Vzhledem k tomu, že je vozidlo v pohybu, nelze použít adaptér připojený do elektrické zásuvky, a proto je použita power banka. Pro motory jsou k dispozici dvě možnosti napájení. Zaprvé je lze připojit do druhého portu na power bance, která napájí SBC a druhou možností je připojení k samostatnému zdroji napájení. Výhodou samostatného napájení je kompletní oddělení řídící části od motorové a zároveň snížení zátěže power banky.

Celková kapacita power banky je 16 000 mAh při výstupu 3,75 V[4] což je dostačující pro napájení celého zařízení po dobu několika hodin. K dispozici jsou dva výstupní porty USB, přičemž maximální výstupní proud na každém z nich je 2 100 mA. Při současném zapojení obou výstupních portů je maximální kombinovaný výstupní proud 3 600 mA (46).

[4] Odpovídá 12 000 mAh při výstupu 5 V

5.1.2 Výroba DPS

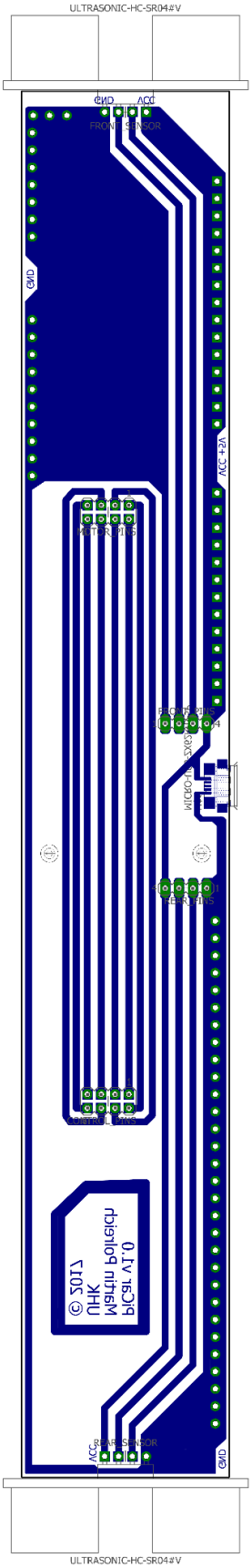
Po navržení konečného zapojení všech komponent bylo přistoupeno k návrhu a výrobě desky plošných spojů, která nahradila dočasná spojení pomocí nepájivých polí. Celý návrh obvodu i desky probíhal v programu Eagle od firmy CADSoft a výsledek lze vidět na obrázcích 5-II, 5-III a 5-IV.



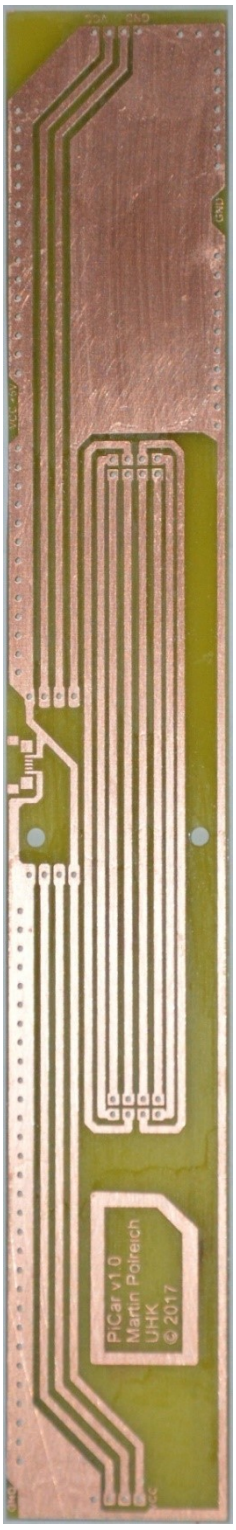
Obrázek 5-II- Schéma obvodu pro výrobu DPS – CADSoft EAGLE – Zdroj: archiv autora

Na prvním obrázku je schéma obvodu. Jedná se o velice jednoduché zapojení. V horní části jsou dva ultrazvukové senzory s vyvedenými konektory na piny pro svorkovnice. V dolní části je poté propojení čtyř dvojic pinů, které slouží pro propojení motorů s Raspberry Pi. Napravo se nachází microUSB konektor pro napájení senzorů.

Po vytvoření schématu je potřeba vytvořit samotnou desku. Její velikost byla zvolena tak, aby se vešla na délku na jednu z hlavních desek šasi vozidla. Její návrh je vidět na další straně na obrázku 5-III a finální podobu na obrázku 5-IV. Na každé kratší straně bude umístěn jeden z dvou ultrazvukových senzorů. Jeden konektor od každého vede vždy do pinů pro svorkovnici a bude připojen na GPIO konektory na Raspberry Pi. Zbývající dva konektory jsou připojeny na napájení pomocí SMD microUSB konektoru, který se nachází v prostřední části delší strany desky.



Obrázek 5-III - Návrh DPS k výrobě



Obrázek 5-IV - Finální podoba vyrobené DPS

Zdroj: Archiv autora

5.2 Softwarová část

Tato část se věnuje postupu při vývoji a implementaci programu pro ovládání vozidla.

Jak je již zmíněno v úvodu práce, celý projekt je napsán v programovacím jazyce Python 3.

Pro ovládání servomotorů na deskách PiFace je použita knihovna „pifacerelayplus“ od této firmy, která je k těmto účelům určena a je volně k dispozici přes server GitHub. Původním záměrem bylo program vyvíjet lokálně přímo na Raspberry Pi s připojeným motorem, avšak nakonec se tento postup moc neosvědčil. Přímo na Raspberry Pi tedy proběhla pouze instalace operačního systému a dodatečných balíků, ale samotný vývoj už probíhal z jiného počítače, který byl připojen k Raspberry Pi pomocí SSH. Další informace o možnostech připojení a jejich výhodách a nevýhodách lze nalézt v kapitole 2.5 a příslušných podkapitolách. Tento postup při vývoji se ukázal jako nejvhodnější, hlavně kvůli tomu, že nebylo potřeba instalovat vývojové prostředí do Raspbianu, ale mohlo být použito to na primárním počítači. Samotné spouštění programu je samozřejmě také nejlepší přes SSH, neboť pohybující se vozidlo nemůže být připojeno napevno k monitoru.

5.2.1 Měření vzdálenosti

```
1.  from __future__ import print_function
2.  import time
3.  import RPi.GPIO as GPIO
4.
5.  FRONT_TRIGGER = 17
6.  FRONT_ECHO    = 18
7.  REAR_TRIGGER = 6
8.  REAR_ECHO = 12
9.
10. temp = 22
11. SoundSpeed = 33140 + (0.6*temp)
12.
13. print("Rychlost zvuku je",speedSound/100,"m/s pri ",temperature,"st.Celsia")
14.
15. time.sleep(0.5)
16.
17. def measure_both():
18.     while True:
19.         measure_front()
20.         measure_rear)
21.
22. def measure_front():
23.     GPIO.setwarnings(False)
24.     GPIO.setmode(GPIO.BCM)
25.     GPIO.setup(FRONT_TRIGGER,GPIO.OUT)  # Trigger
26.     GPIO.setup(FRONT_ECHO,GPIO.IN)      # Echo
27.     GPIO.output(FRONT_TRIGGER, False)
28.     time.sleep(0.2)
29.     GPIO.output(FRONT_TRIGGER, True)
30.     time.sleep(0.00001)
31.     GPIO.output(FRONT_TRIGGER, False)
32.     front_start = time.time()
```

```
33.     while GPIO.input(FRONT_ECHO)==0:
34.         front_start = time.time()
35.     while GPIO.input(FRONT_ECHO)==1:
36.         front_stop = time.time()
37.     front_elapsed = front_stop - front_start
38.     front_distance = front_elapsed * speedSound
39.     front_distance = front_distance / 2
40.     return front_distance
41.     print("Front distance : {0:5.1f}".format(front_distance))
42.
43. def measure_rear():
44.     GPIO.setwarnings(False)
45.     GPIO.setmode(GPIO.BCM)
46.     GPIO.setup(REAR_TRIGGER,GPIO.OUT)
47.     GPIO.setup(REAR_ECHO,GPIO.IN)
48.     GPIO.output(REAR_TRIGGER, False)
49.     time.sleep(0.2)
50.     GPIO.output(REAR_TRIGGER, True)
51.     time.sleep(0.00001)
52.     GPIO.output(REAR_TRIGGER, False)
53.     rear_start = time.time()
54.     while GPIO.input(REAR_ECHO)==0:
55.         rear_start = time.time()
56.     while GPIO.input(REAR_ECHO)==1:
57.        rear_stop = time.time()
58.     rear_elapsed = rear_stop-rear_start
59.     rear_distance = rear_elapsed * speedSound
60.     rear_distance = rear_distance / 2
61.     return rear_distance
62.     print("Rear distance : {0:5.1f}".format(rear_distance))
```

Kód 1 - Soubor "sensor.py" pro měření vzdálenosti senzory – Zdroj: Archiv autora

Na prvních třech řádcích lze vidět příkazy import, kterými se, jak název napovídá, importují moduly. Importování modulů je stěžejní v případě tvorby větších aplikací. Umožňuje totiž volat funkce definované v jiných souborech a knihovnách. Zde se například importuje funkce `print_function` z modulu `__future__`. Dále moduly `time` a `RPi.GPIO`, u kterého můžeme vidět i přiřazení názvu `GPIO`, kterým je na něj poté voláno. Funkce `print_function` opět prozrazuje svůj účel názvem – slouží ke zobrazení výstupu na konzoli. Modul `time` obsahuje velké množství funkcí pro práci s časem. V tomto programu je však z tohoto modulu použita pouze funkce `sleep`, která funguje jako časovač, který odpočítává daný čas v milisekundách, a poté teprve program pokračuje. Modul `RPi.GPIO` je pro funkci také nepostradatelný, jelikož se využívá k nastavení chování GPIO pinů.

Na řádcích 5 až 8 se provádí přiřazení proměnným číslo GPIO pinu. Jsou zde čtyři proměnné – dvě pro každý senzor. Dále můžeme vidět přiřazení hodnoty do proměnné `temp`, která slouží k co nejpřesnějšímu výpočtu rychlosti světla v daném prostředí.

Samotný výpočet probíhá o řádek níže. Pro dosažení maximální přesnosti je možno pevně zadanou hodnotu teploty nahradit proměnnou ze senzoru teploty.

Následuje definování funkce `measure_both`, která obsahuje cyklus pro měření na obou sensorech stále dokola. Pro dosažení tohoto výsledku je použit způsob, který je v Pythonu velmi obvyklý, a to využití podmínky `while True.` Funkce while se sice obvykle používá jen pro spuštění za určitých podmínek, ale deklarování této podmínky jako `True` způsobí její spouštění za každých okolností stále dokola. Tím vznikne nekonečný cyklus.

Na řádcích 22 a 43 začínají definice dvou nejdůležitější funkcí, a to samotné měření a výpočet vzdálenosti. Postup pro oba senzory je totožný a liší se pouze v číslech GPIO pinů, neboť každý senzor je zapojen do jiných.

Na začátku definice můžeme vidět volání funkce `GPIO.setwarnings(False)`, která způsobí potlačení varovných hlášek ve výstupu na konzoli při spouštění programu. Pokud by zde tato funkce nebyla, program by zobrazoval varování o předchozím přiřazení GPIO pinů. Další je funkce `GPIO.setmode(GPIO.BCM)` určující použité číslování GPIO pinů. Jsou zde totiž dva druhy číslování, jedno je `BOARD`, které čísluje piny tak, jak jdou fyzicky popořadě. Druhé je BCM, které odráží logické zapojení v systému. V tomto projektu je použito vždy číslování BCM. Více informací o GPIO se nachází v kapitole 2 a na obrázku 2-I. Na řádcích 25 a 26 následuje přiřazení funkce pinům – tedy `GPIO.IN` nebo `GPIO.OUT`, podle toho, zda budou fungovat jako vstupy nebo výstupy. V našem případě je nutno nastavit Trigger jako výstup a Echo jako vstup. Dále se v programu nastaví Trigger na False, vyčká se 0,2 sekundy a poté se nastaví na True. Program pak zase čeká, ovšem tentokrát pouze 10 μs a poté nastaví výstup opět na False. Tato část kódu způsobí vyslání signálu o délce právě 10 μs ze senzoru směrem do prostoru. O řádek níže můžeme vidět uložení aktuálního času při vyslání signálu do proměnné `front_start`. Na řádcích 35 až 38 je cyklus, který vyčkává na odražený signál, který se vrátí do senzoru a je předán přes konektor Echo. Po jeho příchodu se zaznamená aktuální čas do `front_stop`. Následuje výpočet doby mezi odesláním a přijetím signálu a uložení do proměnné `front_elapsed`. Tato hodnota je poté vynásobena rychlostí zvuku a výsledek je vzdálenost, kterou signál urazil za daný čas. Vzdálenost je nakonec ještě potřeba vydělit dvěma, jelikož signál urazil vzdálenost tam a zpět, ale nás zajímá jen hodnota tam. Finální hodnota obsahující vzdálenost, se kterou se bude dále pracovat, je uložena v proměnné `front_distance`. Pro lepší přehlednost je poté ještě daná hodnota vypsána v konzoli.

5.2.2 Ovládání motorů

Tento soubor obsahuje nastavení spínání motorů pro dosažení požadovaného pohybu.

```
1.  import pifacerelayplus
2.  import time
3.  import sensor as sen
4.
5.  pfr = pifacerelayplus.PiFaceRelayPlus(pifacerelayplus.MOTOR_DC)
6.
7.  def go_front():
8.      pfr.motors[0].forward()
9.      time.sleep(0.15)
10.     pfr.motors[1].forward()
11.     time.sleep(0.15)
12.     pfr.motors[2].forward()
13.     time.sleep(0.15)
14.     pfr.motors[3].forward()
15.     time.sleep(0.15)
16.
17. def stop_all(wait):
18.     time.sleep(wait)
19.     pfr.motors[0].brake()
20.     time.sleep(0.15)
21.     pfr.motors[1].brake()
22.     time.sleep(0.15)
23.     pfr.motors[2].brake()
24.     time.sleep(0.15)
25.     pfr.motors[3].brake()
26.     time.sleep(0.15)
27.
28. def go_back():
29.     pfr.motors[0].reverse()
30.     time.sleep(0.15)
31.     pfr.motors[1].reverse()
32.     time.sleep(0.15)
33.     pfr.motors[2].reverse()
34.     time.sleep(0.15)
35.     pfr.motors[3].reverse()
36.     time.sleep(0.15)
37.
38. def go_left_front(dur):
39.     pfr.motors[1].forward()
40.     time.sleep(0.15)
41.     pfr.motors[0].reverse()
42.     time.sleep(dur)
43.     pfr.motors[1].brake()
44.     time.sleep(0.15)
45.     pfr.motors[0].brake()
46.
47. def go_right_front(dur):
48.     pfr.motors[0].forward()
49.     time.sleep(0.15)
50.     pfr.motors[1].reverse()
51.     time.sleep(dur)
52.     pfr.motors[1].brake()
53.     time.sleep(0.15)
54.     pfr.motors[0].brake()
55.
56. def go_left_rear(dur):
57.     pfr.motors[4].forward()
58.     time.sleep(0.15)
59.     pfr.motors[3].reverse()
60.     time.sleep(dur)
```

```
61.     pfr.motors[4].brake()
62.     time.sleep(0.15)
63.     pfr.motors[3].brake()
64.
65. def go_right_rear(dur):
66.     pfr.motors[3].forward()
67.     time.sleep(0.15)
68.     pfr.motors[4].reverse()
69.     time.sleep(dur)
70.     pfr.motors[3].brake()
71.     time.sleep(0.15)
72.     pfr.motors[4].brake()
```

Kód 2 - Soubor "motor.py" pro ovládání motorů – Zdroj: Archiv autora

I zde jsou na začátku importy modulů. Jako první se importuje knihovna `pifacerelayplus`, což je již zmíněná oficiální knihovna pro ovládání modulů PiFace. Dále je tu modul `time` a nakonec `sensor`, což je program sensor.py, který je podrobně rozebrán v kapitole 5.2.1. Následuje definice funkcí pro pohyb vpřed, zastavení, pohyb vzad a zatáčení. Obsahem těchto funkcí je postupné spouštění, respektive zastavování motorů. Z důvodu ochrany proti přetížení je nutno dodržet čas minimálně 150 ms mezi jednotlivými pokyny `forward`, `brake` a `reverse`. K tomu opět slouží funkce `time.sleep`. Funkce pro zatáčení jsou velice podobné. Funkce `go_left_front` otočí vozidlo doleva, přičemž k tomu jsou použity přední kola, `go_right_rear` otočí vozidlo doprava za použití zadních kol, a tak obdobě pro zbývající dvě kombinace. Tyto funkce na rozdíl o prostého pohybu vpřed a vzad přijímají vstupní parametr `dur`, který určuje délku daného pohybu v sekundách. Hlavní důvod jeho použití je, že se na rozdíl od pohybu vpřed a vzad, předpokládá, že otáčení nebude déletrvající, ale jen jednorázové.

5.2.3 Hlavní program

Hlavní program, který obsahuje rozhodovací algoritmus celého systému a spouštění jednotlivých předem definovaných funkcí.

```
1.  import sensor as sen
2.  import motor
3.  import atexit
4.  import time
5.
6.  last_movement = "none"
7.
8.  def decision_front():
9.      if sen.measure_front() > 50:
10.         motor.go_front()
11.         global last_movement
12.         last_movement = "front"
13.         print("Jedu dopredu")
14.         if sen.measure_front() < 50:
15.             motor.stop_all(0)
16.             decision_rear()
17.             print("Prekazka prede mnou")
18.         else:
19.             motor.go_front()
20.             global last_movement
21.             last_movement = "front"
22.             print ("Pokracuji dopredu")
23.     else:
24.         print ("Prekazka prede mnou")
25.         motor.stop_all(0)
26.         decision_rear()
27.
28. def decision_rear():
29.     if sen.measure_rear() > 50:
30.         motor.go_back()
31.         global last_movement
32.         last_movement = "rear"
33.         print ("Jedu dozadu")
34.         if sen.measure_rear() < 50:
35.             motor.stop_all(0)
36.             decision_front()
37.             print("Prekazka za mnou")
38.         else:
39.             motor.go_back()
40.             global last_movement
41.             last_movement = "rear"
42.             print("Pokracuji dozadu")
43.     else:
44.         print ("Prekazka za mnou")
45.
46. decision_front()
47.
48. while True:
49.     if sen.measure_front() > 50 and last_movement == "front":
50.         decision_front()
51.         print("Stale pokracuji dopredu")
52.     elif sen.measure_front() > 50 and last_movement == "rear":
53.         motor.stop_all(0)
54.         time.sleep(1)
55.         decision_front()
56.         print("Zastavuji a pojedu dopredu")
57.     elif sen.measure_rear() > 50 and last_movement == "front":
58.         motor.stop_all(0)
```

```
59.             time.sleep(1)
60.             decision_rear()
61.             print("Zastavuji a pojedu dozadu")
62.         elif sen.measure_rear() > 50 and last_movement == "rear":
63.             decision_rear()
64.             print("Stale pokracuji dozadu")
65.         elif last_movement == "none":
66.             decision_front()
67.             print("Chyba!!")
68.         else:
69.             decision_front()
70.             print("Nemohu pokracovat")
71.
72.     except (KeyboardInterrupt, SystemExit):
73.         raise motor.stop_all(0.2)
74.     except:
75.         raise
```

Kód 3 – Soubor „start.py“ pro spuštění celého systému – Zdroj: Archiv autora

Stejně jako u ostatních programů, i tento začíná importy jednotlivých modulů. Následuje deklarace globální proměnné `last_movement`, která bude obsahovat informace o posledním směru pohybu vozidla.

Na řádku 8 začíná definice rozhodovacího algoritmu pro jízdu vpřed nazvaného `decision_front`. Jako první se ověří vzdálenost před senzorem. Pokud není detekována žádná překážka v definované vzdálenosti, zavolá se funkce `go_front` z modulu motor. Díky tomu se vozidlo rozjede dopředu. Dokud je podmínka splněna, tj. dokud přední senzor nezaznamená překážku v dané vzdálenosti, vozidlo pokračuje v jízdě vpřed. Jakmile je zaznamenána překážka, zavolá se funkce `stop_all`, která zastaví vozidlo. Následně se zavolá rozhodovací algoritmus `decision_rear`, který zjistí, zda může vozidlo couvat dozadu. Její definice začíná na řádku 28 a funkčně je stejná jako `decision_front` s tím rozdílem, že měří vzdálenost za vozidlem a nevolá funkci `go_front`, ale `go_back`.

Od řádku 48 do 70 je definován cyklus, který na základě výsledků měření a informací o posledním směru pohybu volá jednotlivé rozhodovací algoritmy. Na konci se nachází zachycení výjimky typu `KeyboardInterrupt`. Tato definice při stisku kombinace kláves Ctrl + C zastaví všechny motory.

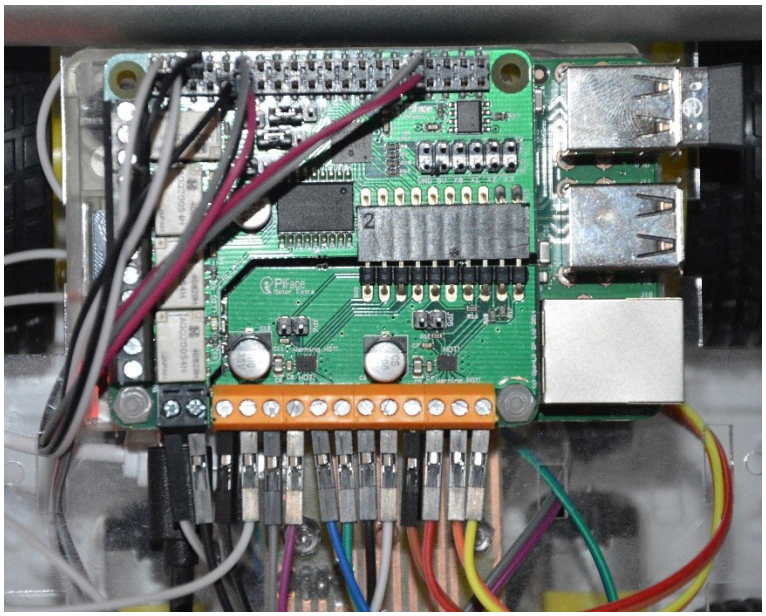
6 Shrnutí výsledků

Tato práce měla za úkol navrhnout a implementovat systém pro automatizované ovládání vozidla pomocí Raspberry Pi.

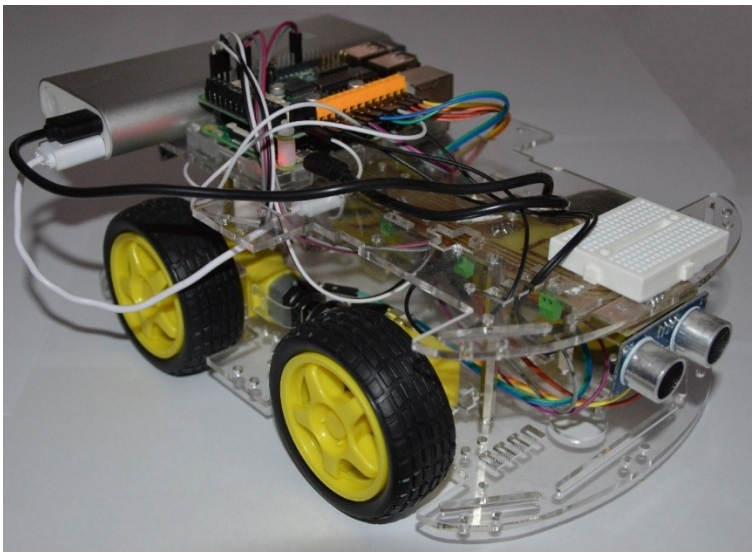
Výsledků bylo dosaženo sestavením pojízdného vozidla poháněného čtveřicí stejnosměrných motorů, které je schopno rozpoznat překážku v prostoru před sebou nebo za sebou, zareagovat na ni zastavením a po jejím odstranění pokračovat v jízdě. Jako řídící počítač celého systému bylo použito Raspberry Pi v3 Model B s dvěma přídavnými moduly značky PiFace. Pro měření vzdálenosti k překážce jsou použity dva ultrazvukové senzory umístěné po jednom v přední a zadní části vozidla. K propojení všech součástek byla použita deska plošných spojů, vyrobená dle vlastního návrhu autora na míru a pro použití s předem navrženými součástkami. Všechny komponenty jsou napájeny z power banky umístěné na vozidle.

Po spuštění hlavního programu se vozidlo rozjede vpřed, dokud nerozpozná překážku. V takovém případě zastaví, zkontroluje prostor za sebou a pokud zde překážku nenalezne, začne couvat. Jakmile je detekována překážka při couvání za vozidlem, zastaví a zkontroluje prostor vpředu. V případě detekce překážek z obou stran vozidlo čeká a jakmile je jedna strana uvolněna, rozjede se. Vozidlo má také nadefinované funkce pro zatáčení vlevo i vpravo za pomocí předních i zadních kol.

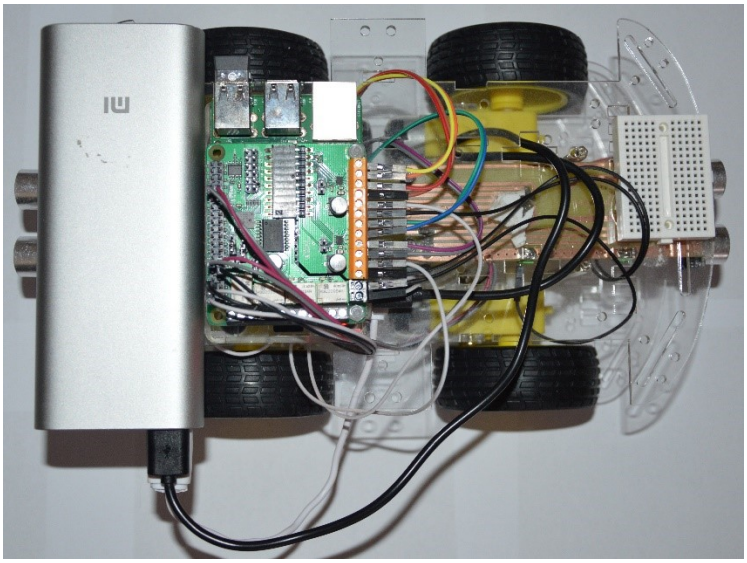
6.1 Fotodokumentace



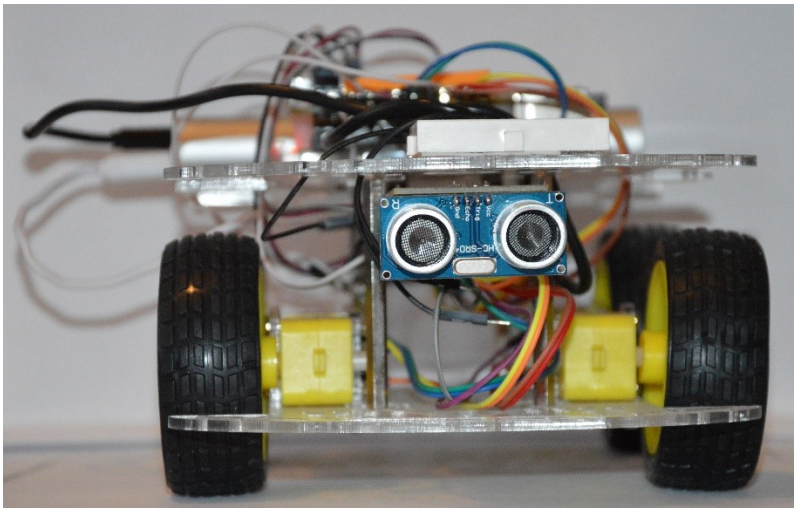
Obrázek 6-I - Detail zapojení Raspberry Pi a modulů PiFace – Zdroj: archiv autora



Obrázek 6-II - Finální vzhled vozidla se všemi komponenty – Zdroj: archiv autora



Obrázek 6-III - Horní pohled na finální vozidlo – Zdroj: archiv autora



Obrázek 6-IV- Čelní pohled na finální vozidlo – Zdroj: archiv autora

7 Závěr

Rozvoj autonomních vozidel v posledních letech naznačuje, že by mohly být tyto automobily běžnou součástí našich životů. Hlavním spouštěčem tohoto rozmachu bylo podstatné zlepšení počítačového vidění a zpracování obrazu, které umožnilo, aby počítače získaly vědomí o dění kolem sebe.

Rychlý nástup nové technologie však vždy kromě kýžených výsledků přináší i nové problémy a stejně tomu je i v oblasti autonomních vozidel. Jedním z takových problémů je rozmanitost prostředí, ve kterých se vozidla pohybují, a s tím spojený problém s univerzálním naprogramováním jejich chování. Dalšími problémy jsou chybějící lidské reflexy umožňující řešení nenadálých situací a nejasné názory na převzetí odpovědnosti v případě nehody.

Tato práce nastiňuje možnosti sestavení a naprogramování vlastního autonomního vozidla za použití běžně dostupného hardwaru. V žádném případě není námi navrhované řešení možno srovnávat s vozidly produkovanými automobilovým průmyslem, ale to ani nebylo účelem této práce. Naše řešení nabízí běžnému uživateli možnost seznámení se s touto novou a stále se rozvíjející technologií v domácích podmínkách.

Další práce v této problematice by mohla rozšířit stávající systém o snímání okolí kamerou a následné zpracování počítačovým viděním. Takový systém by mohl umožňovat rozpoznávání dopravních značek, přechodů pro chodce či různých druhů překážek.

8 Zdroje

(1)	HARRIS, Mark. Who is Anthony Levandowski, and why is Google suing him?. In: *The Guardian* [online]. Londýn: Guardian News and Media Limited, 2017 [cit. 2017-04-08]. Dostupné z: https://www.theguardian.com/technology/2017/feb/23/anthony-levandowski-google-uber-self-driving-cars-lawsuit
(2)	Autonomous Vehicles. In: *National Conference of State Legislatures* [online]. National Conference of State Legislatures, 2017 [cit. 2017-04-08]. Dostupné z: http://www.ncsl.org/research/transportation/autonomous-vehicles-self-driving-vehicles-enacted-legislation.aspx
(3)	KOROSEC, Kirsten. Google unleashes the driverless car it built from scratch. In: *Fortune* [online]. New York City: Time Inc., 2015 [cit. 2017-04-08]. Dostupné z: http://fortune.com/2015/06/25/google-driverless-car/
(4)	HAWKINS, Andrew J. YOU CAN HAIL A SELF-DRIVING UBER IN SAN FRANCISCO STARTING TODAY. In: *The Verge* [online]. Vox Media, Inc., 2016 [cit. 2017-04-08]. Dostupné z: http://www.theverge.com/2016/12/14/13921514/uber-self-driving-car-san-francisco-launch-volvo-xc90
(5)	A 15 pound computer to inspire young programmers. *BBC* [online]. 2011 [cit. 2016-06-03]. Dostupné z: http://www.bbc.co.uk/blogs/thereporters/rorycellanjones/2011/05/a_15_computer_to_inspire_young.html
(6)	Products. *Raspberry Pi* [online]. b.r. [cit. 2016-06-03]. Dostupné z: https://www.raspberrypi.org/products/
(7)	NEW PRODUCT LAUNCH! INTRODUCING RASPBERRY PI MODEL B+. *Raspberry Pi* [online]. 2014 [cit. 2016-06-03]. Dostupné z: https://www.raspberrypi.org/blog/introducing-raspberry-pi-model-b-plus/
(8)	Raspberry Pi 3 on sale now at $35. *Raspberry Pi* [online]. b.r. [cit. 2016-06-03]. Dostupné z: https://www.raspberrypi.org/blog/raspberry-pi-3-on-sale/
(9)	APETROAIE-CRISTEA, Mihaela, Mark SCOTT, Steven J JOHNSTON a Simon J COX. Indoor localisation system based on low-cost commodity hardware. In:

Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing Adjunct - UbiComp '16 [online]. New York, New York, USA: ACM Press, 2016, s. 13-16 [cit. 2017-04-23]. DOI: 10.1145/2968219.2971441. ISBN 9781450344623. Dostupné z: http://dl.acm.org/citation.cfm?doid=2968219.2971441

(10) SAAD, Amna, Ahmad Roshidi AMRAN a M. Naim Abu HASAN. WarBox: Portable wardriving over Raspberry PI. In: *2016 International Conference on Information and Communication Technology (ICICTM)* [online]. IEEE, 2016, s. 227-235 [cit. 2017-04-23]. DOI: 10.1109/ICICTM.2016.7890806. ISBN 9781509004126. Dostupné z: http://ieeexplore.ieee.org/document/7890806/

(11) PATCHAVA, Vamsikrishna, Hari Babu KANDALA a P Ravi BABU. A Smart Home Automation technique with Raspberry Pi using IoT. In: *2015 International Conference on Smart Sensors and Systems (IC-SSS)* [online]. IEEE, 2015, s. 1-4 [cit. 2017-04-23]. DOI: 10.1109/SMARTSENS.2015.7873584. ISBN 9781467393287. Dostupné z: http://ieeexplore.ieee.org/document/7873584/

(12) Raspbian. *Raspberry Pi* [online]. b.r. [cit. 2017-02-12]. Dostupné z: https://www.raspberrypi.org/downloads/raspbian/

(13) Downloads. *Raspberry Pi* [online]. b.r. [cit. 2017-02-12]. Dostupné z: https://www.raspberrypi.org/downloads/

(14) About Raspbian. *Raspbian* [online]. b.r. [cit. 2017-02-12]. Dostupné z: https://www.raspbian.org/RaspbianAbout

(15) Welcome to Raspbian. *Raspbian* [online]. b.r. [cit. 2017-02-12]. Dostupné z: https://www.raspbian.org/

(16) NOOBS. *Raspberry Pi* [online]. b.r. [cit. 2017-02-12]. Dostupné z: https://www.raspberrypi.org/documentation/installation/noobs.md

(17) Introducing PIXEL. *Raspberry Pi* [online]. b.r. [cit. 2017-02-12]. Dostupné z: https://www.raspberrypi.org/blog/introducing-pixel/

(18) FAQ. *Ubuntu MATE* [online]. b.r. [cit. 2017-02-12]. Dostupné z: https://ubuntu-mate.org/faq/

(19) Ubuntu MATE for the Raspberry Pi 2 and Raspberry Pi 3. *Ubuntu MATE* [online]. b.r. [cit. 2017-02-12]. Dostupné z: https://ubuntu-mate.org/raspberry-pi/

(20)	Snappy Ubuntu Core: instalace lusknutím prstů. In: *Root.cz* [online]. 2015 [cit. 2017-02-12]. Dostupné z: https://www.root.cz/clanky/snappy-ubuntu-core-instalace-lusknutim-prstu/
(21)	FROEHLICH, Andrew. 8 IoT Operating Systems Powering The Future. In: *InformationWeek* [online]. 2016 [cit. 2017-02-13]. Dostupné z: http://www.informationweek.com/iot/8-iot-operating-systems-powering-the-future/d/d-id/1324464
(22)	About. *LibreELEC* [online]. b.r. [cit. 2017-02-13]. Dostupné z: https://libreelec.tv/about/
(23)	J, Byron. Raspberry Pi SPI and I2C Tutorial. In: *Sparkfun* [online]. b.r. [cit. 2017-03-19]. Dostupné z: https://learn.sparkfun.com/tutorials/raspberry-pi-spi-and-i2c-tutorial
(24)	GPIO: MODELS A+, B+, RASPBERRY PI 2 B AND RASPBERRY PI 3 B. *Raspberry Pi* [online]. b.r. [cit. 2017-04-23]. Dostupné z: https://www.raspberrypi.org/documentation/usage/gpio-plus-and-raspi2/
(25)	CHAN, Pauline. Raspberry Pi 3 GPIO Header. In: *Element14 Community* [online]. 2015 [cit. 2017-03-19]. Dostupné z: https://www.element14.com/community/docs/DOC-73950/l/raspberry-pi-3-model-b-gpio-40-pin-block-pinout
(26)	SPI Interface. *Corelis* [online]. Corelis, 2017 [cit. 2017-04-20]. Dostupné z: http://www.corelis.com/education/SPI_Tutorial.htm
(27)	UM10204: I2C-bus specification and user manual. *NXP* [online]. NXP Semiconductors N.V., 2014 [cit. 2017-04-20]. Dostupné z: www.nxp.com/documents/user_manual/UM10204.pdf
(28)	Introduction to I^2C and SPI protocols. *Byteparadigm* [online]. Byte Paradigm sprl, b.r. [cit. 2017-04-20]. Dostupné z: http://www.byteparadigm.com/applications/introduction-to-i2c-and-spi-protocols/
(29)	I2C. In: *Sparkfun* [online]. SparkFun Electronics, b.r. [cit. 2017-04-22]. Dostupné z: https://learn.sparkfun.com/tutorials/i2c

(30)	Raspberry Pi FAQs - Frequently Asked Questions. *Raspberry Pi - Teach, Learn, and Make with Raspberry Pi* [online]. b.r. [cit. 2016-06-02]. Dostupné z: https://www.raspberrypi.org/help/faqs/
(31)	LUTZ, Mark. *Programming Python*. 4th ed. Beijing: O'Reilly Media, 2011. programming/Python. ISBN 978-0-596-15810-1.
(32)	Python2orPython3. *Python* [online]. 2017 [cit. 2017-04-09]. Dostupné z: https://wiki.python.org/moin/Python2orPython3
(33)	Python Developers Survey 2016: Findings. *JetBrains* [online]. Praha: JetBrains s.r.o., b.r. [cit. 2017-04-09]. Dostupné z: https://www.jetbrains.com/pycharm/python-developers-survey-2016/
(34)	VAN ROSSUM, Guido. What's New In Python 3.0. In: *Python* [online]. Python Software Foundation, 2017 [cit. 2017-04-09]. Dostupné z: https://docs.python.org/3/whatsnew/3.0.html
(35)	PILGRIM, Mark. *Ponořme se do Python(u) 3: Dive into Python 3*. Praha: CZ.NIC, 2010. CZ.NIC. ISBN 978-80-904248-2-1.
(36)	KEREKI, Federiko. SSH: A Modern Lock For Your Server?. *Linux Journal*. Belltown Media: Houston, Texas, USA, b.r.
(37)	KEREKI, Federiko. More secure SSH connections. *Linux Journal*. Houston, Texas, USA: Belltown Media, 2014, **2014**(237). ISSN 1075-3583.
(38)	REHMAN, Faisal. Difference between Telnet and SSH. In: *LinuxPitStop* [online]. 2015 [cit. 2017-04-02]. Dostupné z: http://linuxpitstop.com/difference-between-telnet-and-ssh/
(39)	BOUDETTE, Neal E. 5 Things That Give Self-Driving Cars Headaches. In: *The New York Times* [online]. New York, NY, USA: The New York Times Company, 2016 [cit. 2017-02-15]. Dostupné z: https://www.nytimes.com/interactive/2016/06/06/automobiles/autonomous-cars-problems.html
(40)	KIRKPATRICK, Keith. The moral challenges of driverless cars. *Communications of the ACM*. 2015, **58**(8), 19-20. DOI: 10.1145/2788477. ISSN 00010782. Dostupné také z: http://dl.acm.org/citation.cfm?doid=2808213.2788477
(41)	PiFace Relay Plus. In: *RPiShop.cz* [online]. b.r. [cit. 2017-04-16]. Dostupné z: http://rpishop.cz/raspberry-pi-prislusenstvi/193-piface-relay.html

(42)	PiFace Relay+ For Raspberry Pi: Getting Started. In: *PiFace* [online]. Londýn: OpenLX SP Ltd., 2014 [cit. 2017-04-16]. Dostupné z: http://www.piface.org.uk/assets/docs/PiFace-Relay-Plus_getting-started.pdf
(43)	PiFace Motor EXTRA. In: *Botland* [online]. b.r. [cit. 2017-04-16]. Dostupné z: https://botland.com.pl/raspberry-pi-hat-ekspandery-wyprowadzen/3366-piface-motor-control-extra-rozszerzenie-do-piface-relay.html
(44)	PiFace Motor EXTRA For Raspberry Pi: Getting started. In: *PiFace* [online]. Londýn: OpenLX SP Ltd., 2014 [cit. 2017-04-16]. Dostupné z: http://www.piface.org.uk/assets/docs/PiFace-Motor-EXTRA_getting-started.pdf
(45)	Ultrasonic Ranging Module HC-SR04. In: *Micropik* [online]. b.r. [cit. 2017-04-16]. Dostupné z: http://www.micropik.com/PDF/HCSR04.pdf
(46)	Mi Power Bank 16000mAh. *Xiaomi: official website* [online]. b.r. [cit. 2017-04-08]. Dostupné z: http://www.mi.com/en/pb16000/

9 Obrázky

10 Tabulky

11 Kódy

Univerzita Hradec Králové
Fakulta informatiky a managementu
Akademický rok: 2016/2017

Studijní program: Aplikovaná informatika
Forma: Prezenční
Obor/komb.: Aplikovaná informatika (ai3-p)

Podklad pro zadání BAKALÁŘSKÉ práce studenta

PŘEDKLÁDÁ:	ADRESA	OSOBNÍ ČÍSLO
Polreich Martin	Žitná 3263/11, Jablonec nad Nisou	I14129

TÉMA ČESKY:

Využití Raspberry Pi pro autonomní řízení vozidla

TÉMA ANGLICKY:

Using Raspberry Pi for autonomous car driving

VEDOUCÍ PRÁCE:

doc. Ing. Filip Malý, Ph.D. - KIKM

ZÁSADY PRO VYPRACOVÁNÍ:

Cílem je seznámit se s problematikou programování Raspberry Pi za účelem programování autonomního vozidla, vybrat vhodný (a dostupný) hardware pro tento účel, navrhnout a implementovat postupy vedoucí k dosažení autonomního pohybu vozidla.

Osnova:

1) Úvod
2) Programování Raspberry Pi
3) Analýza problematiky autonomního vozidla
4) Výběr vhodného hardwaru
5) Návrh a implementace vybraných postupů
6) Závěr
7) Literatura

SEZNAM DOPORUČENÉ LITERATURY:

Lutz M., Ascher D. - Naučte se Python
Dawson M. - Python Programming
Langtangen H.P. - A Primer on Scientific Programming with Python

Podpis studenta: Polreich ..... **Datum:** 12.10.2016

Podpis vedoucího práce: ..... **Datum:** 12.10.2016