VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

AUTONOMNÍ VOZIDLO PRO MODEL DOPRAVNÍ SITUACE
AUTONOMOUS VEHICLE FOR TRAFFIC SITUATION MODEL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Dominik Schneiderka**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Ilona Janáková, Ph.D.**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Kybernetika, automatizace a měření**

Ústav automatizace a měřicí techniky

Student: Bc. Dominik Schneiderka ***ID:*** 186180
Ročník: 2 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Autonomní vozidlo pro model dopravní situace

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je navrhnout a realizovat autonomní model vozidla pro model dopravní situace. Model vozidla bude na základě vyhodnocení dat z kamery (případně jiných senzorů) simulovat některé dopravní situace.

1. Seznamte se s detailními požadavky kladenými na úlohu a s aktuálními možnostmi stávajícího modelu dopravní situace.
2. Navrhněte a vyberte vhodný hardware pro umístění na model vozidla. Předpokládá se umístění kamerového modulu s vhodnou optikou přímo na autíčku. Vyhodnocovací jednotka bude buď také přímo součástí vozidla, nebo bude zajištěn přenos snímků do nadřazeného systému.
3. Realizujte fyzický model vozidla.
4. Navrhněte možné úlohy, které by autonomní vozidlo mohlo řešit, např. rozpoznání stavu semaforu ze snímku, detekce překážek a jiných vozidel atd. a k jednotlivým situacím navrhněte i adekvátní reakci vozidla.
5. Daným úlohám vhodně upravte/přizpůsobte stávající model dopravní situace. Řešte s ohledem na snadný transport, rychlou instalaci a nenáročnou obsluhu celé platformy.
6. Navrhněte a implementujte algoritmy řešící vybrané úlohy.
7. Vytvořte ukázkovou aplikaci a otestujte její funkčnost.
8. Výsledky zpracujte, stanovte omezující podmínky, zhodnoťte.

DOPORUČENÁ LITERATURA:

SOJKA, Eduard, Jan GAURA a Michal KRUMNIKL. Matematické základy digitálního zpracování obrazu. VŠB Ostrava, ZČU v Plzni, 2011. Dostupné z: http://mi21.vsb.cz/modul/matematicke-zaklady-digitalniho-zpracovani-obrazu

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 1.6.2020

Vedoucí práce: Ing. Ilona Janáková, Ph.D.

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

ABSTRAKT

Tato práce se zabývá vývojem autonomního autíčka na autodráze Carrera 143. Předmětem ovládání autíčka je samovolné zastavení na semaforu při svítící červené, či zastavení před překážkou. Práce popisuje vyvinuté elektrické obvody pro autíčko a jejich usazení na model vozidla. Algoritmy vyvinuté pro sledování vozovky před vozidlem jsou psány v jazyce C/C++ a výpočetní jednotku tvoří Raspberry Pi Zero. K práci s obrazovými informacemi je využito knihovny OpenCV. Veškeré zdrojové kódy byly vyvinuty ve vývojovém prostředí Microsoft Visual Studio 2019.

KLÍČOVÁ SLOVA

autonomní řízení, autonomní autíčko, Carrera 143, C++, počítačové vidění, segmentace obrazu, detekce objektů, Raspberry Pi Zero

ABSTRACT

This thesis describes development of autonomous car for Carrera 143 racing track. Main objective of a car is to stop when traffic light shows red, or when there is an obstacle infront of a car. This paper also describes electric schemes used to control the car and their placement on the car. Algorithms developed for image processing are developed for processing unit Raspberry Pi Zero and are written in C/C++ programming language. OpenCV library is used for image processing. All source codes were developed in Microsoft Visual Studio 2019.

KEYWORDS

autonomous control, autonomous car, Carrera 143, C++, computer vision, image segmentation, object detection, Raspberry Pi Zero

SCHNEIDERKA, Dominik. *Autonomní vozidlo pro model dopravní situace*. Brno, 2020, 84 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce: Ing. Ilona Janáková, Ph. D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Autonomní vozidlo pro model dopravní situace“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno 1.6.2020 .....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucí diplomové práce paní Ing.Iloně Janákové, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci. Také bych rád poděkoval své rodině za podporu při studiu a nesmírnou trpělivost.

Brno 1.6.2020 .....

podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Seznam symbolů, veličin a zkratek

HW	hardware
SW	software
OSI	Open System Interconnection - referenční model pro účely propojování systémů
DPS	deska plošných spojů
PWM	pulse-width modulation - pulsně šírková modulace
RISC	Reduced Instruction Set Computer - procesory s redukovanou instrukční sadou
GPIO	General Purpose Input and Output - Univerzální vstupně výstupní pin
DC	Direct Current - Stejnosměrné napětí
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor - polem řízený tranzistor
OS	Operating System - Operační Systém
OTG	On-The-Go
USB	Universal Serial Bus - univerzální sériová sběrnice
SSH	Secure Shell
FPS	Frames Per Second - snímky za sekundu

Úvod

Tato práce se věnuje HW (hardware) a SW (software) řešení autonomního autíčka pro autodráhu Carrera 143. Jejím úkolem je vytvoření modelu reálné dopravní situace.

Práce se snaží přiblížit problematiku počítačového vidění ve spojení s autonomními vozidly. V reálných případech jsou již tyto kamerové systémy nasazeny na nespočet vyráběných vozidel snad všech značek. Počítačové vidění v automobilech se používá na sledování jízdních pruhů, čtení dopravních značek, sledování překážek před vozidlem, či kontrolu mrtvého úhlu. V některých případech se však nedá spolehnout pouze na kamerové systémy, ty bývají často doplňovány dalšími senzory, jako jsou radary, lidary, nebo ultrazvukové senzory. K doplňování kamerových systémů dochází z toho důvodu, že nejsou schopny spolehlivého snímání okolí vozidla při zhoršených podmínkách viditelnosti (špatné světlo, déšť, mlha, atp.). Například autonomní vozidla od společnosti Tesla využívají osmi kamerových systémů, které jsou doplněny o dvanáct ultrazvukových senzorů a radar. Tato sada senzorů a kamer umožňuje vozidlu snímat a správně vyhodnocovat své okolí i za zhoršené viditelnosti [1].

Úkolem práce bylo vytvořit autonomní autíčko na autodráhu Carrera 143, které by využívalo kamerový systém na sledování vozovky před sebou. Je potřeba, aby auto správně vyhodnotilo semafor na křižovatce a správně zareagovalo, popřípadě aby zpomalilo, či úplně zastavilo při přibližování se k překážce. Zároveň by autíčko mělo správně vyhodnotit zatáčky před sebou a dle prudkosti stanovit rychlost, která je bezpečná pro projetí.

Ze zadání práce plynou následující požadavky na sestavení autonomního autíčka.

Požadavky na autonomní autíčko

- Nezávislost na ovládání autodráhy
- Rozpoznání překážky a jiných vozidel
- Rozpoznání stavu semaforu

V práci jsou popsány základní teoretické znalosti, které byly využity pro řešení. Zejména teorie ohledně počítačového vidění a teorie elektrických obvodů. Po teoretických kapitolách se práce věnuje správnému výběru architektury pro řešení autonomního modelu vozidla. Velkou součástí je návrh elektrických obvodů potřebných pro řízení rychlosti a světel. V práci jsou popsány 3D modely vytvořené pro autíčko a model autodráhy. V závěrečné části se práce věnuje programovému vybavení Raspberry Pi Zero a vyvinutým algoritmům pro sledování vozovky před vozidlem.

1 Elektrické obvody - teorie

V následující kapitole se nachází teoretické podklady, které byly využity při návrhu elektrických obvodů v práci.

1.1 Stabilizace napětí

Stabilizátory napětí se využívají pro stabilizaci a snížení úrovně napětí na vstupu. Snížení napětí je například potřeba pro napájení mikrokontrolérů. V této práci bylo potřeba převést napětí z 15 V na 5 V, aby bylo možné napájet Raspberry Pi Zero a LED diody umístěné na autíčku.

1.1.1 Lineární stabilizátory napětí

Lineární stabilizátory jsou široce rozšířenou součástkou a dokážou přesně stabilizovat napětí na danou úroveň. Pro stabilizaci využívají lineárního prvku, kde vzniká přesně definovaný úbytek napětí (například tranzistor). Přebytečná elektrická energie je přeměněna na tepelnou a vyzářena do okolí. Lineární stabilizátory se proto v praxi (ve většině případech) neobejdou bez použití pasivního chlazení. Právě ona přeměna energie má také za následek nižší účinnost lineárních stabilizátorů [2].

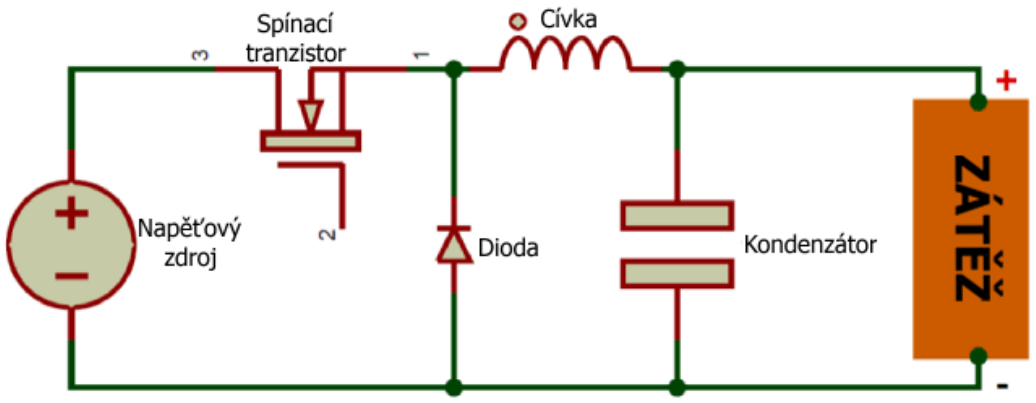
Vyrábí se v několika řadách a existují stabilizátory s nastavitelným výstupním napětím i pevně daným výstupním napětím. Nejznámější používaný stabilizátor napětí je 7805, který převádí napětí na 5 V.

1.1.2 DC/DC měniče napětí

Tyto měniče využívají spínaného obvodu pro převod napětí. Pro spínání se využívá tranzistor, který je spínán s velkou frekvencí (kHz). Principem měniče je sériové zapojení cívky a kondenzátoru. Pomocí cívky je nabíjení kondenzátoru limitováno, nabíjí se pomaleji, a zároveň je také energie ukládána v cívce. Tato uložená energie je poté použita pro napájení obvodu [2].

V případě sepnutého tranzistoru dochází k ukládání energie v cívce a nabíjení kondenzátoru. Při rozepnutém tranzistoru je energie z cívky využita pro napájení obvodu. Tento princip je využit u DC/DC měničů napětí [2].

DC/DC měniče napětí nevyužívají přeměny elektrické energie na tepelnou a tak se nezahřívají jako lineární stabilizátory, není potřeba využití pasivního chlazení. Tyto měniče napětí ovšem neposkytují stejné stabilní napětí jako lineární stabilizátory, díky ukládání a uvolňování energie je napětí lehce zvlněno. DC/DC měniče mohou také rušit okolní součástky [2].



Obr. 1.1: Schéma DC/DC měniče napětí [3]

1.2 Ochrana proti přepólování

Ochrana chrání elektroniku před zničením v případě připojení opačné polarity napájecích svorek. Tato ochrana může být realizována mnohými způsoby, od využití konektorů, které lze zapojit jen jedním způsobem až po komplexnější řešení jako jsou integrované obvody. Integrované obvody jsou kombinovány i s ochranami proti přepětí, zkratu a další (např. TPS2660X od TexasInstruments).

1.2.1 Ochranná dioda

Jedná se o jeden z nejjednodušších a nejlevnějších způsobů ochrany. Diodu je potřeba připojit sériově k zátěži. Při zapojení správné polarity bude dioda propouštět proud, jelikož její PN přechod bude zapojen v propustném směru. Při obrácení polarity bude PN přechod diody zapojen v závěrném směru a obvodem nebude protékat proud [4].

Nevýhodou tohoto řešení je úbytek napětí na diodě, na zátěži tak získáme napětí snížené o tento úbytek. Toto řešení má také relativně nízkou účinnost. Jako ochranná dioda se mohou používat například Schottkyho diody, mají nižší úbytek napětí a krátkou zotavovací dobu [4].

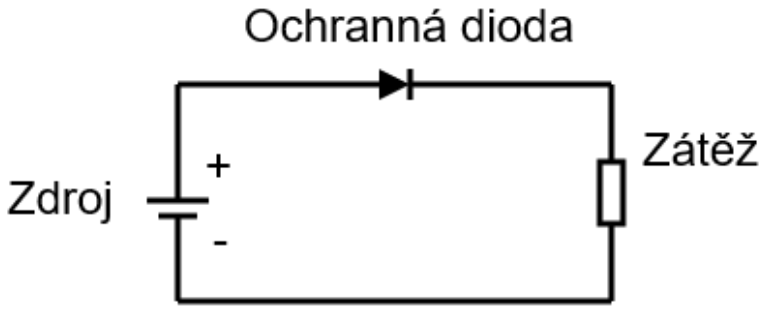
1.2.2 Ochranný tranzistor

Jako ochranný tranzistor se využívá MOSFET (Metal-Oxide-Semiconductor Field-Effect Transistor - polem řízený tranzistor) tranzistor s P vodivým kanálem. Drain

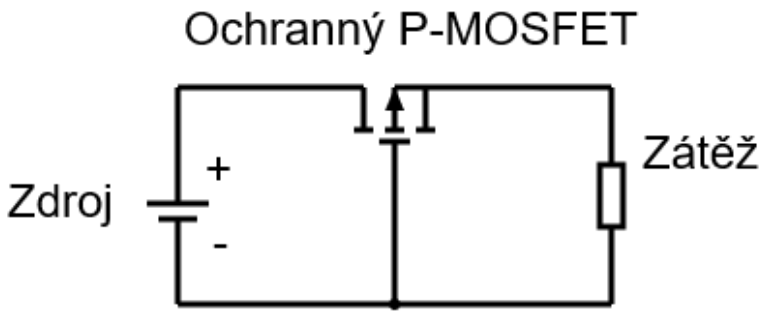
(D) elektroda tranzistoru je připojena na kladnou svorku přívodu a Gate (G) elektroda je připojena na zápornou svorku. Elektroda Source (S) slouží jako „výstup“ tranzistoru. Takto zapojený tranzistor chrání obvod proti přepólování.

V případě přiložení správné polarity napětí (plus svorka na D a mínus svorka na G) tranzistor vytvoří vodivý kanál mezi D a S a obvodem protéká proud. Při přiložení opačné polarity není vytvořen vodivý kanál mezi elektrodami D a S a obvodem proud neprotéká.

Výhodou tohoto řešení oproti využití diody je malý odpor tranzistoru při zapnutém stavu a tedy zanedbatelný úbytek napětí mezi svorkami D a S [4].



Obr. 1.2: Schéma zapojení ochranné diody



Obr. 1.3: Schéma zapojení ochranného tranzistoru

1.3 Záložní zdroj napájení

Záložní zdroje napájení se ve většině případech používají za účelem udržení napájení dostatečně dlouhou dobu, aby došlo k bezpečnému vypnutí napájeného zařízení. Mohou se k tomu využít baterie, nebo záložní super kondenzátory.

1.3.1 Baterie

Baterie jsou dnes používány téměř v každé elektronice. Jejich velkou výhodou oproti super kondenzátorům je větší poměr uložené energie vůči velikosti baterie. Dokážou tak uchovat větší množství energie při menších rozměrech [5].

Baterie se pro jejich nabíjení neobejdou bez nabíjecích obvodů. V dnešní době se již tyto nabíjecí obvody dají koupit ve formě integrovaných obvodů a značně tak ulehčují implementaci akumulátorů v různých projektech. Nabíjecí cykly baterií se liší dle výrobce a je potřeba tyto cykly dodržovat, jinak může dojít k poškození [6].

Výhoda baterií spočívá v možnosti uchovat větší množství energie na delší dobu, baterie samovolně ztrácí svou energii v mnohem menší míře než super kondenzátory. Jejich nevýhoda však tkví v potřebě dalších obvodů pro napájení [6].

1.3.2 Super kondenzátor

Super kondenzátory využívají jiné technologie uchování energie. Jejich výhoda spočívá v rychlosti vybíjení a nabíjení. Dokážou se nabít téměř okamžitě a to samé platí i o vybíjení.

Na rozdíl od baterií kondenzátory nepotřebují nabíjecí obvody. Nabíjí se samovolně a napětí na kondenzátoru nemůže dosáhnout jeho nominální hodnoty. Kondenzátor se také nemůže poškodit v případě, že by byl zapojen do obvodu s větším napětím.

Další výhodou je jednoduchost elektrických obvodů. Při využití super kondenzátorů není potřeba žádných nabíjecích obvodů.

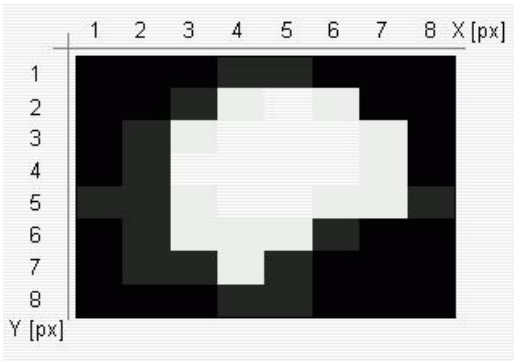
Nevýhodou je velká míra samovolného vybíjení, nejsou tedy vhodné pro dlouhodobé uchovávání energie [5].

2 Zpracování obrazu - teorie

V následující kapitole se nachází teoretické podklady, které byly využity při zpracování obrazu v této práci.

2.1 Reprezentace obrazu v paměti

V počítačové technice je obraz reprezentován pomocí matic, které jsou dále děleny na jednotlivé pixely. Pixely v sobě uchovávají barevnou hodnotu obrazu. V případě barevné hloubky obrazu 8 bitů mohou jednotlivé pixely nabývat hodnoty 0 (0x00) až 255 (0xff), jeden pixel tak může nabývat až 256 různých barev. U šedotónového obrazu bude matice druhého řádu s barevnou hloubkou 8 bitů. Řádky a sloupce matice pak určují obrazovou rovinu. Pomocí těchto souřadnic jsme schopni určit hodnotu jednotlivých pixelů v matici [7].



(a)

	1	2	3	4	5	6	7	8	X [px]
1	0	0	0	10	10	0	0	0	
2	0	0	10	60	100	60	0	0	
3	0	10	60	100	250	100	60	0	
4	0	10	100	250	250	100	60	0	
5	10	10	60	100	100	60	60	10	
6	0	10	60	60	60	10	0	0	
7	0	10	10	60	10	0	0	0	
8	0	0	0	10	10	0	0	0	
Y [px]									

(b)

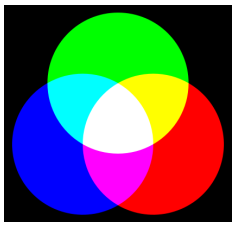
Obr. 2.1: a Šedotónový obraz [7]; b Matice reprezentující šedotónový obraz [7]

Nejedná-li se o šedotónový obraz, ale o obraz RGB, pak matice uchovávající pixely bude třetího řádu. Přidaný rozměr matice nám umožňuje pohybovat se mezi jednotlivými barevnými složkami. Barevné složky jsou tedy oddělené a každá barevná složka je reprezentována pomocí pixelů stejně jako šedotónový obraz [7].

2.2 Barevné prostory

Existuje několik barevných prostorů, které lze využít pro zpracování obrazu. Tato práce se omezí na popis jen několika z nich. Nejčastějším a nejznámějším barevným prostorem je prostor RGB, který dělí obraz do tří barevných složek, červená

(R - red), zelená (G - green) a modrá (B - blue). V RGB prostoru je obraz představován pomocí tří matic, kde každá matice představuje jinou barvu. Výsledný obraz je pak získán součtem těchto tří matic, proto je RGB prostor nazývám jako aditivní prostor [8].

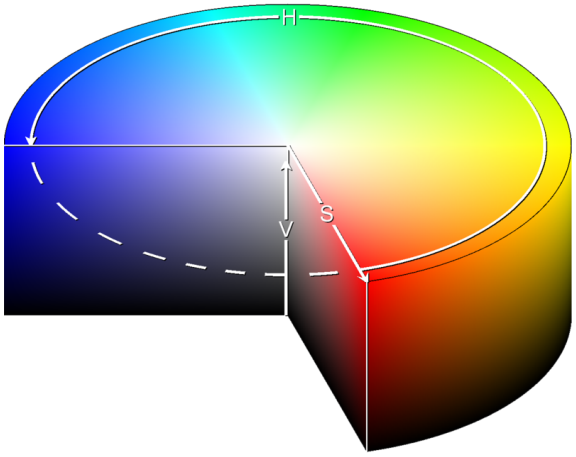


Obr. 2.2: RGB barevný model [8]

CMY(K) model je opakem RGB. V tomto barevném modelu se jednotlivé barvy od sebe odčítají, jedná se tedy o subtraktivní model. CMYK se opět dělí do tří barevných složek, azurová (C - cyan), purpurová (M - magenta), žlutá (Y - yellow) a černá (K - key). S černou barvou pracují tiskárny, aby došlo k úspoře toneru [8].

Obr. 2.3: CMYK barevný model [8]

Modely HSV a HSL představují tzv. intuitivní modely. Intuitivní proto, že v představování barev se nejvíce blíží lidskému vnímání barev. Modely představují barvu ve třech složkách, pro model HSV je to odstín (H - hue), sytost (S - saturation) a hodnota (V - value). Model HSL je totožný až na poslední hodnotu, kterou je světelnost (L - lightness). Tyto modely jsou nejbližší k našemu vnímání barev, jelikož počítají s tím, že při malé světelnosti/hodnotě nejsme schopní rozlišovat barvy [8].



Obr. 2.4: HSV barevný model [8]

2.3 Potlačení šumu

Při zpracování obrazové informace se, u vstupního obrazu, může vyskytovat šum. Šum ztěžuje zpracování obrazu, proto je potřeba využít filtrací pro jeho odstranění. Filtračních metod je mnoho, tato práce popisuje jen některé z nich.

Nejjednodušším způsobem filtrování je prosté průměrování jasových hodnot. Výsledná hodnota je dána průměrem hodnot okolních pixelů. Jedná se o konvoluci s konvoluční maskou. Konvoluční maska může mít různé hodnoty, například masky h1 a h2. U masky h2 je zvýšená váha středu. Nevýhodou průměrování je, že dochází k rozmazání hran v obraze [9].

$$h1 = \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} \quad h2 = \frac{1}{10}\begin{bmatrix}1 & 1 & 1\\ 1 & 2 & 1\\ 1 & 1 & 1\end{bmatrix}$$

Můžeme také použít filtr s Gaussovým rozložením. Princip filtru je stejný jako u průměrování, konvoluční maska má Gaussovo rozložení (h3 - příklad masky 5x5). Maska s Gaussovým rozložením je vypočítána pomocí rovnice 2.1 pro 1D, nebo 2.2 pro 2D, kde x a y jsou souřadnice obrazu a σ představuje směrodatnou odchylku [9].

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} \cdot \exp\frac{-x^2}{2\sigma^2} \tag{2.1}$$

$$G(x,y) = \frac{1}{2\pi\sigma^2} \cdot \exp -\frac{x^2+y^2}{2\sigma^2} \tag{2.2}$$

$$h3 = \frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

2.4 Detekce hran v obraze

Pro získání potřebné informace z obrazových dat lze využít hranové detekce. Pomocí hran lze poté jednotlivé objekty v obraze klasifikovat. Detekce hran spočívá ve zvýraznění částí obrazu, kde je velká změna jasové hodnoty (velký gradient). Jednotlivé hrany v obraze jsou určené velikostí a směrem. Pro detekci hran se podobně jako u filtrace šumu využívá konvolučních masek [9].

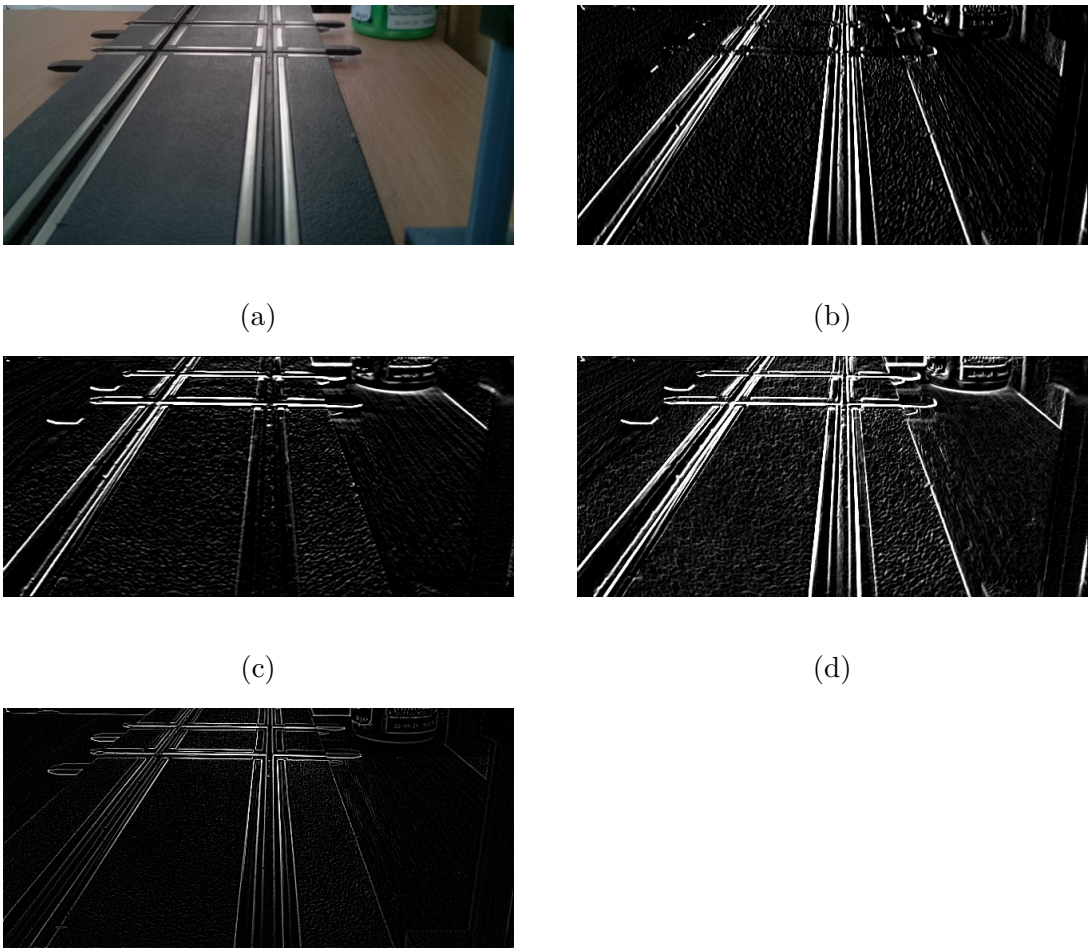
Rozlišujeme konvoluční masky neinvariantní vůči rotaci a invariantní vůči rotaci. Neinvariantní masky vůči rotaci dokážou rozpoznat hrany jen v jednom směru. Pro detekci hran ve všech směrech (horizontální, vertikální, diagonální) je zapotřebí použít více masek. Konvoluční masky invariantní vůči rotaci dokážou rozpoznat hrany ve všech směrech. Konvolučních masek existuje mnoho, níže jsou uvedeny pouze dva příklady [9].

Příklad konvoluční masky pro Sobelův hranový detektor (neinvariantní vůči rotaci). Matice h1 detekuje horizontální hrany a matice h2 hrany vertikální [9].

$$h1 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 2 \end{bmatrix} \quad h2 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Níže je uveden příklad konvoluční masky Laplaceova hranového detektoru (invariantní vůči rotaci) pro čtyř-okolí [9].

$$h1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$



(a) (b)

(c) (d)

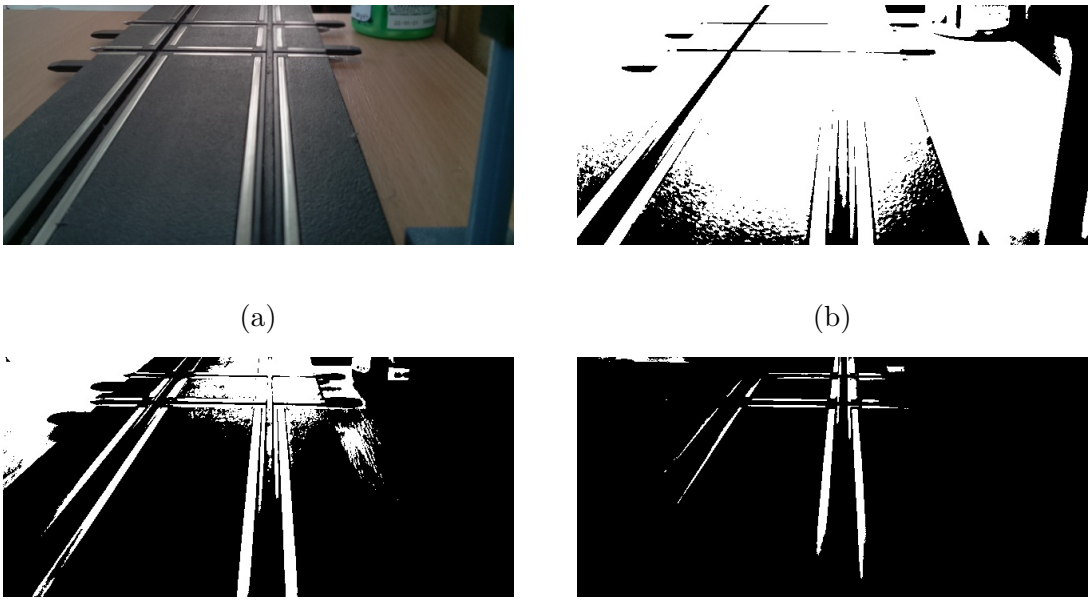
(e)

Obr. 2.5: a Vstupní obraz hranového detektoru; b Sobeluv detektor - vertikální směr; c Sobeluv detektor - horizontální směr; d Sobeluv detektor, součet horizontálního a vertikálního směru e Laplaceův detektor

2.5 Prahování obrazu

Prahováním lze z obrazu odlišit objekt zájmu od okolí (segmentace obrazu). Jedná se o operaci, na jejímž vstupu se nachází vstupní obraz f(i,j) a na výstupu získáme binární obraz g(i,j). Obraz prahujeme na základě stanoveného prahu T. V případě, že hodnota pixelu je nižší než práh, zapíše se do pixelu hodnota 0, jinak 1. Práh pro správné naprahování lze získat z histogramu (graf zobrazující četnost jednotlivých jasových úrovní). Prahovat lze s jedním prahem (viz rovnice 2.3), nebo i s více prahy (přesnější segmentace), popřípadě využít adaptivního prahu. Při adaptivním prahování je práh počítán vždy pro část obrazu [9].

$$f(i,j) = \begin{cases} 1, & \text{pro } g(i,j) \geq \mathrm{T} \\ 0, & \text{pro } g(i,j) < \mathrm{T} \end{cases} \tag{2.3}$$



(a) (b)

(c) (d)

Obr. 2.6: a Vstupní obraz prahování; b Práh 50; c Práh 100; d Práh 150

2.6 Morfologické operace

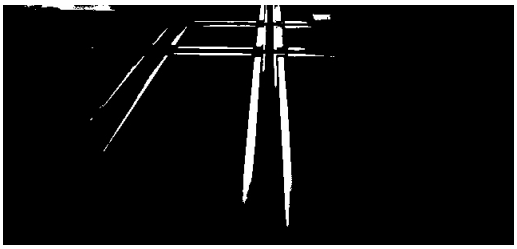
Morfologické operace můžeme využít pro předzpracování obrazu (např. odstranění šumu), zdůraznění objektů (např. zesílení získaných hran v obraze), popřípadě pro popis objektů. Pro morfologické operace se využívá tzv. strukturní element. Každý strukturní element má definovaný svůj počátek. Mezi základní operace patří dilatace a eroze [10].

Dilatace obrazu na základě strukturního elementu přidá určitý počet pixelů k hranici nalezeného objektu. Udělá tak z tenčího objektu tlustší objekt. Dilatace provádí vektorový součin strukturního elementu s obrazem [10].

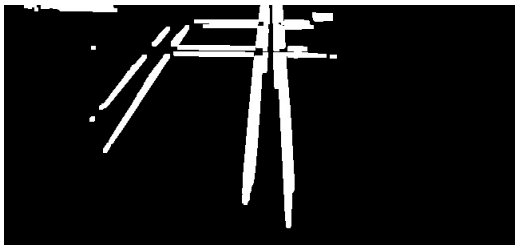
Eroze je opak dilatace. Na základě strukturního elementu ubere určitý počet pixelů z hranice nalezeného objektu. Udělá tak z tlustšího objektu objekt tenčí, může se stát, že objekt úplně zanikne (filtrace šumu). Eroze provádí vektorový rozdíl strukturního elementu s obrazem [10].

Příklad čtvercového strukturního elementu 5x5 s počátkem ve středu:

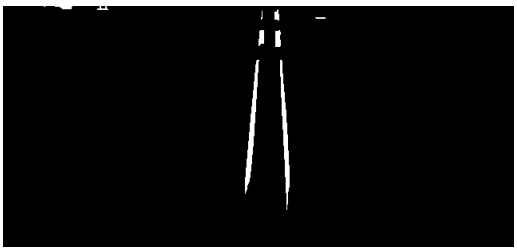
$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$



(a)



(b)

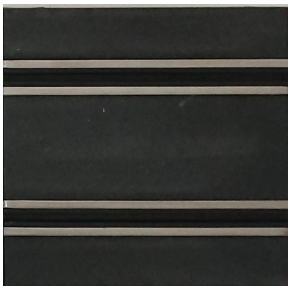


(c)

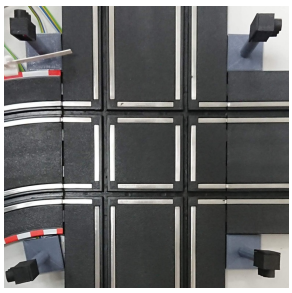
Obr. 2.7: a Naprahovaný obraz; b Dilatace s elementem 5x5; c Eroze s elementem 5x5

3 Seznámení s autodráhou Carrera 143

Jak již bylo zmíněno, práce se zabývá vytvořením autonomního autíčka pro autodráhu Carrera 143. Tato autodráha je digitální a můžou na ní současně jezdit až tři autíčka. Veškeré modely autíček a dílů autodráhy jsou vyrobeny v poměru 1:43 vůči realitě. Napájení autíček probíhá přes kartáčky na spodní straně (viz 3.1c), které jsou při nasazení autíčka spojené s kolejnicemi autodráhy (viz 3.1a). Kolejnice autodráhy mají stejnosměrné napětí U = +14,8 V. Autodráha byla následně vybavena mikrokontrolérem Arduino, který nadále celou autodráhu řídí, díky němu je možné provádět různé modulární úpravy autodráhy. K autodráze byly také přidány čtyři tříbarevné semafory (červená, oranžová, zelená), rovněž ovládané mikrokontrolérem Arduino, které se nachází na křižovatce (viz 3.1b).



(a)



(b)

(c)

Obr. 3.1: a Kolejnice autodráhy Carrera143; b Křižovatka se semafory; c Kartáčky na autíčku sbírající napětí z kolejnic

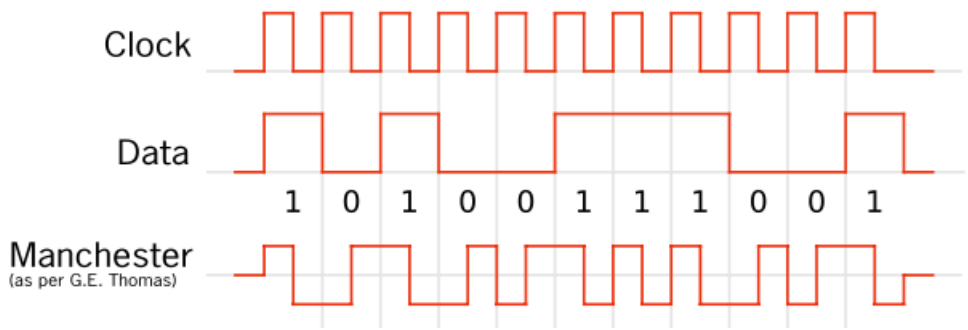
3.1 Ovládání autodráhy

Komunikace je zajištěna pomocí kódování v Manchester kódu. Arduino kóduje data na kolejnicích autodráhy pomocí odpojení napětí z kolejnice. To znamená, že jedna

z kolejnic je stále připojena na zem (GND) a druhá kolejnice je ovládána pomocí Arduina a tedy odpojována od napětí +14,8 V. Tato skutečnost znamená, že v případě, kdy komunikace neprobíhá, je na kolejnicích stále napětí +14,8 V.

Manchester kódování je synchronní kódování, které je implementováno na fyzické vrstvě OSI (Open System Interconnection - referenční model pro účely propojování systémů) modelu. Signál mění svou hodnotu vždy v půl periodě hodinového signálu, v případě vyslání dvou stejných bitů za sebou (například dvakrát log. 0) může signál změnit hodnotu i na začátku periody. Pomocí tohoto kódování jsme schopní komunikovat synchronně bez potřeby synchronizačního pulsu. Pro Manchester kódování nám tedy stačí jeden vodič. Logická 1 je detekována přechodem signálu z vysoké (high) hodnoty do nízké (low) hodnoty a logická 0 je detekována opačně, čili z nízké (low) hodnoty do vysoké (high) hodnoty. V našem případě se hodnoty v Manchester kódování pohybují mezi +14,8 V (high) a 0 V (low). Manchester kódování názorně popisuje obrázek 3.2.

Výhodou Manchester kódování je nepřítomnost synchronizačního (hodinového) signálu, komunikace je tedy levnější. Nevýhoda tohoto řešení spočívá v případě, kdy za sebou vysíláme dva stejné bity (například dvakrát log. 0). V tomto případě se v signálu objeví jeden puls navíc, který nenese žádnou informaci, pouze nastavuje sběrnici do potřebné logické úrovně, aby mohl poslat druhý (stejný) bit. Tuto nevýhodu odstraňuje diferenciální Manchester kódování.



Obr. 3.2: Manchester kódování [11]

Každé autíčko je naprogramováno na svůj ovladač a jelikož v sobě obsahuje HW, který je schopný dekódovat Manchester kód, je schopné se pohybovat jen na pokyny příslušného ovladače. Autíčko detekuje signál, který je pro něj určený a ostatní signály ignoruje. Signál, který je pro autíčko určen je poznán podle toho, který ovladač akci vyvolal (zrychlení, nebo zpomalení). V případě ovládání pouze pomocí mikrokontroléru Arduino je v příslušném datovém slově zapsána logická 1, aby se pohybovalo jen jedno autíčko. Podrobnější popis HW autíčka viz podkapitola 3.2.

Předtím, než začneme ovládat pouze jedno autíčko je potřeba autíčko s ovladačem spárovat. To se provede následující sekvencí [12]:

1. Položit autíčko na autodráhu
2. Dvakrát zmáčknout tlačítko pro změnu koleje
3. Zvednout autíčko z autodráhy a položit jej zpátky
4. Dvakrát zmáčknout tlačítko pro změnu koleje

Po provedení této sekvence bude autíčko spárováno s daným ovladačem a nebude reagovat na žádný jiný. Stejnou sekvenci lze provést také pomocí mikrokontroléru Arduino, stačí akorát poslat správná datová slova pro párování. Podrobný popis datových slov a jejich bitů lze nalézt zde [13].

Výše popsaný text popisuje autodráhu nacházející se v laboratoři. Autodráha byla modifikována o výše zmíněný mikrokontrolér Arduino, znamená to tedy, že autodráha nepotřebuje ke svému ovládání ovladač, vše je řízeno pomocí SW zapsaném v mikrokontroléru Arduino. Schéma zapojení s mikrokontrolérem Arduino viz příloha A. Pro podrobný popis schématu a jeho vysvětlení viz [13].

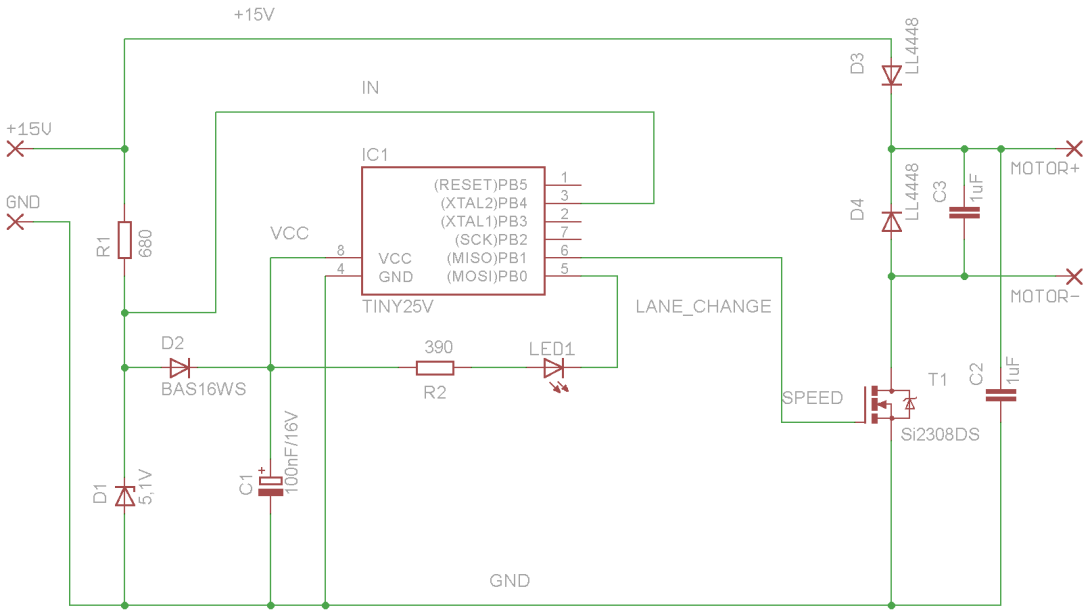
3.2 Dekodér autíčka

Digitální autíčka pro autodráhy Carrera v sobě obsahují své vlastní dekodéry, které dekódují příchozí komunikaci popsanou v předešlé kapitole 3.1. Dekodér se nachází na malé DPS (deska plošných spojů), která je napájena z kolejnic autodráhy.

Hlavní součástí dekodéru je mikrokontrolér ATtiny25V od společnosti Atmel [14]. Tento mikrokontrolér vyhodnocuje probíhající komunikaci na kolejnicích a tedy určuje, zda je komunikace určena právě pro dané autíčko, či nikoliv. V případě, že se jedná o komunikaci určenou danému autíčku, je z datových slov přečtena rychlost, jakou se má autíčko pohybovat. Na základě rychlosti mikrokontrolér zapne na pinu PB1 PWM (pulse-width modulation - pulsně šířková modulace), pomocí které je řízena rychlost motorku autíčka. Příchozí komunikace je přiváděna na pin PB4. Pin PB0 je využit pro spínaní infračervené diody LED1, která slouží k přepínání výhybek na autodráze, aby autíčko mohlo přejet na vedlejší kolejnici (v této práci nejsou výhybky využívány). Zbylé piny mikrokontroléru jsou využity pro programování mikrokontroléru a tedy nemají vliv na funkčnost autíčka. Mikrokontrolér je napájen napětím 5 V, kterého je docíleno pomocí napěťového regulátoru vytvořeného zenerovou diodou D1 (vstupní napětí je 15 V), tato dioda zároveň slouží jako ochrana proti obrácení polarity napájení (v případě, že autíčko položíme na kolejnice opačně). Napájení je následně filtrováno pomocí filtračního kondenzátoru C1, aby nedocházelo k výpadku napájení z důvodu využití kolejnic nejen jako napájení, ale

také pro komunikaci.

Pinem PB1, který je využíván jako PWM, je spínán unipolární tranzistor T1, jedná se o tranzistor typu MOSFET s N vodivým kanálem. Na G (Gate) tohoto tranzistoru je přiveden signál z pinu PB1, na D (Drain) je připojena záporná svorka elektromotorku a pomocí vývodu S (Source) je tranzistor připojen k zemi (GND). Tranzistor v tomto zapojení spíná když je na G svorce kladné napětí a pomocí tohoto se mezi svorkami D a S vytvoří vodivý kanál, díky němuž je záporná svorka elektromotorku připojená k zemi. Sepnutím tranzistoru T1 začne obvodem protékat proud a elektromotorek se začne otáčet. Jelikož je na pin PB1 přiváděn PWM signál, je rychlost autíčka ovládána pomocí střídy tohoto signálu, pro podrobnější popis viz kapitola 4.3. Na kladnou svorku motorku je připojen kondenzátor C2, který funguje jako filtrační kondenzátor. Paralelně k motorku je také připojen kondenzátor C3, jehož funkce je zpomalení náběhu motorku, při sepnutí motorku tedy nedojde k nárazovému sepnutí. Dioda D4 paralelně spojená s elektromotorkem v závěrném směru slouží jako ochrana proti napěťovým špičkám. Stejně tak i dioda D3 slouží jako ochrana, ale vůči tomu, aby nedošlo ke změně vstupní zakódované informace. Schéma popsaného zapojení je zobrazeno na obrázku 3.3 [1].



Obr. 3.3: Schéma zapojení dekodéru autíčka

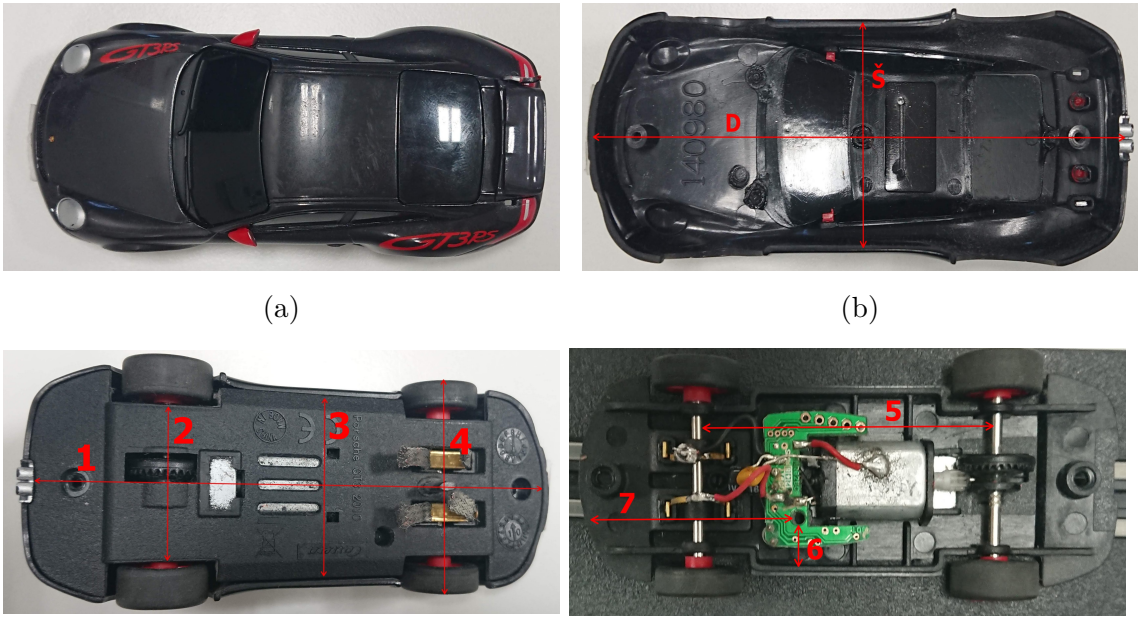
[1] schéma bylo zjištěno měřením DPS, nejedná se o originální schéma od výrobce, jednotlivé hodnoty a typy součástek se mohou lišit

3.3 Popis autíčka

Veškerá autíčka pro autodráhu Carrera 143 jsou vyráběna v poměru 1:43 vůči skutečnosti. Jedná se o nejmenší autodráhu od společnosti Carrera. Rozměry černého modelu Porsche GT3 RS (viz obrázek 3.4a), použitého v této diplomové práci, jsou popsány v tabulce a na obrázcích 3.4b, 3.4c, 3.4d.

Čára	Rozměr	Obrázek
Š	38,65 mm	3.4b
D	104,10 mm	
Výška	29,60 mm	
1	99,50 mm	3.4c
2	27,15 mm	
3	35,65 mm	
4	43,05 mm	
5	57,25 mm	3.4d
6	12,80 mm	
7	22,75 mm	

Tab. 3.1: Rozměry autíčka Porsche GT3 RS, pro popis obrázků 3.4b, 3.4c, 3.4d



(a) (b)

(c) (d)

Obr. 3.4: a Model autíčka Porsche GT3 RS; b Rozměry uvnitř karoserie autíčka; c Rozměry podvozku autíčka; d Vnitřní rozměry autíčka

4 Návrh řízení autonomního autíčka

Následující kapitola popisuje různé možnosti na trhu, pomocí kterých by bylo možné ovládat autonomní autíčko a jaká architektura řešení byla zvolena.

4.1 Architektura autonomního autíčka

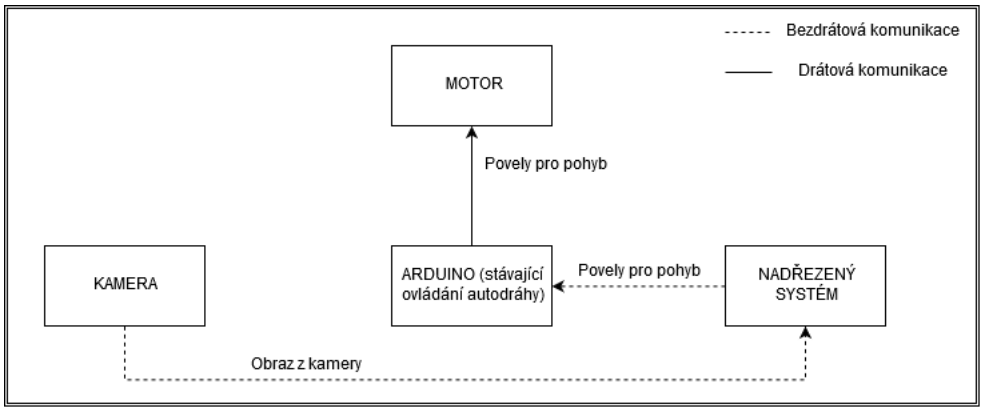
Před výběrem vhodného mikrokontroléru pro ovládání autonomního autíčka bylo potřeba prvně stanovit samotnou architekturu. Existuje několik možných architektur, které připadaly v úvahu.

4.1.1 Využití stávajícího ovládání autodráhy

Jako jedno z řešení se nabízelo využití stávajícího ovládání autodráhy, kdy by autonomní autíčko bylo ovládáno pomocí stejné šifrované komunikace jako ostatní autíčka na autodráze (viz kapitola 3). Toto řešení by spočívalo v tom, že by se na autíčku nacházela kamera, snímající situaci před autíčkem. Obraz z kamery by byl přenášen do nadřazeného systému. Na tomto systému by docházelo k vyhodnocení obrazu a výsledný povel k jízdě, nebo zastavení autíčka by byl poslán do mikrokontroléru Arduino, který celou autodráhu řídí. Mikrokontrolér Arduino by povel interpretoval pro naše autonomní autíčko. Toto řešení by tedy vyžadovalo buďto bezdrátovou IP kameru schopnou bezdrátově posílat obraz do nadřazeného systému, nebo na autíčko instalovat mikrokontrolér, který by obraz z kamery posílal do nadřazeného systému. Dále by bylo zapotřebí onoho nadřazeného systému, v tomto případě by se mohlo jednat například o standardní počítač. Architektura je znázorněna na obrázku 4.1.

Výhodou řešení je jednodušší ovládání autonomního autíčka, možnost vyhodnocování obrazu na jakémkoliv (dostatečně výkonném) počítači, také by nebylo zapotřebí zasahovat do stávajícího HW řešení autíčka, bylo by potřeba akorát přidat bezdrátovou IP kameru na autíčko. Ani u tohoto řešení není třeba se obávat, že by vyhodnocování obrazu z kamery trvalo moc dlouho a že by to počítač nezvládal.

Nevýhodou řešení je závislost na stávajícím ovládání autodráhy. V případě, že by došlo k přechodu ovládání z mikrokontroléru Arduino zpět na ovládání pomocí ovladačů (originální ovládání autodráhy), autíčko by nefungovalo. Další nevýhodou je potřeba externí výpočetní jednotky. Pokud by se jednalo o počítač, tak by se řešení mohlo prodražit a přemisťování modelu by také nebylo moc praktické. Jednou z nevýhod je rovněž využívání bezdrátového přenosu informací, kdy může docházet k neočekávaným výpadkům, nebo zpomalením.



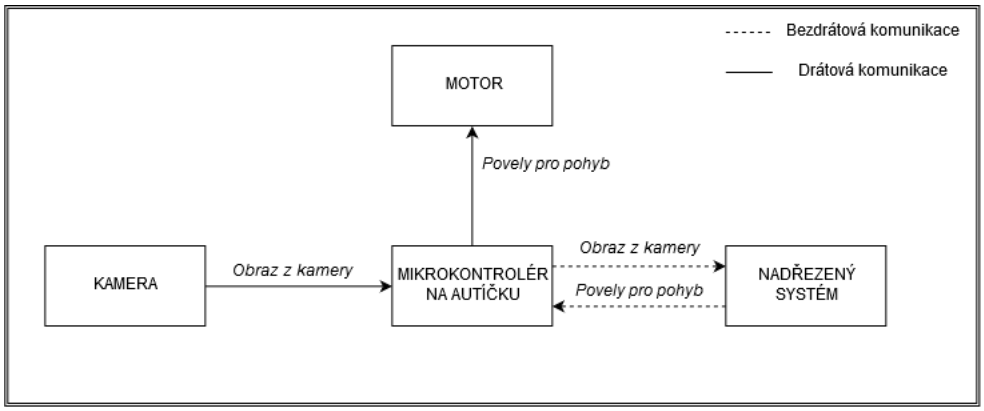
Obr. 4.1: Blokový diagram architektury s využitím stávájícího ovládání autodráhy

4.1.2 Využití vlastního ovládání autíčka

Následující řešení je velice podobné předešlému, hlavní rozdíl spočívá v oproštění od stávajícího ovládání autodráhy. K tomuto řešení by bylo zapotřebí kamery, jejíž obraz by byl posílán do nadřazeného systému, který by opět představoval standardní počítač. Nadřazený systém by opět vyhodnotil obraz, ale, na rozdíl od výše uvedeného řešení, povel k zastavení, či rozjetí autonomního autíčka by byl poslán přímo autíčku. Nevyužívalo by se tak stávajícího ovládání autodráhy. Znamenalo by to ovšem, že na autíčku musí být přítomna vyhodnocovací jednotka, která by uměla bezdrátově poslat snímaný obraz z kamery do nadřazeného systému a poté, opět bezdrátově, přijímat povel pro autíčko a vykonat jej. Architektura je znázorněna na obrázku 4.2.

Toto řešení má výhodu v nezávislosti na stávajícím ovládání. Autonomní autíčko by bylo funkční na každé autodráze s napájením kolejnic U = +15 V. Ostatní výhody jsou shodné s architekturou, kde by bylo využito stávajícího ovládání autodráhy.

Nevýhody řešení jsou opět shodné s výše uvedenou architekturou, až na nevýhodu závislosti na stávajícím řešení.



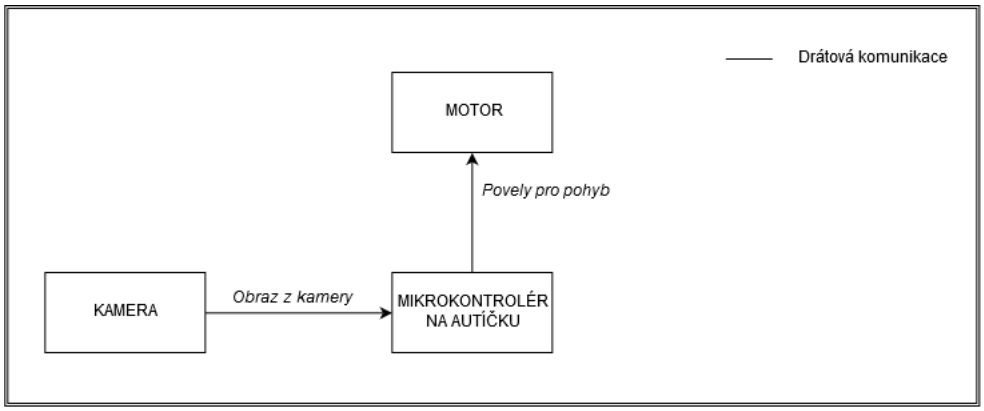
Obr. 4.2: Blokový diagram architektury s využitím vlastního ovládání autíčka

4.1.3 Vlastní ovládání integrované na autíčku

Poslední z řešení, která připadala v úvahu, je řešení, kdy by k veškerému vyhodnocování obrazu a zpracování povelů docházelo přímo v samotném autonomním autíčku (viz 4.3). K tomuto řešení by bylo zapotřebí pouze kamery a vyhodnocovací jednotky, která by byla schopna zpracovávat obraz z kamery, tento obraz vyhodnotit a na základě zjištěných dat správně interpretovat povel pro autonomní autíčko a tak zrychlit, či zpomalit elektromotorek.

Výhoda tohoto řešení spočívá v nepotřebnosti nadřazené vyhodnocovací jednotky a tedy v integrování řešení do samotného autíčka. Autíčko by tak bylo nezávislé na stávajícím ovládání autodráhy a bude funkční i na autodráze s jiným typem řízení, podmínkou ale je, že na kolejnicích bude napětí U = +15 V. Velká výhoda je nepotřebnost bezdrátového přenosu.

Nevýhodou tohoto řešení jsou malé rozměry autíčka. Bude třeba zvolit výpočetní jednotku s dostatečně malými rozměry, aby se na autíčko vešla. S malými rozměry se dá ale bohužel předpokládat i menší výkon výpočetní jednotky, může to tedy s sebou nést následky, že výpočetní výkon nebude dostačující pro dostatečně rychlé zpracování obrazu.



Obr. 4.3: Blokový diagram architektury s vyhodnocovací jednotkou na autíčku

4.1.4 Vybraná architektura

Ačkoliv se může zdát nevýhoda architektury, kdy je ovládání integrováno na samotné autíčko, jako velice velká, autor práce se i přesto rozhodl pro toto řešení. Tato architektura totiž představuje elegantní řešení jak autonomní autíčko sestavit. Navíc se bude tento model více blížit realitě, kdy nejsou posílány obrazy z kamer do nadřazeného systému, ale veškeré výpočty probíhají v samotném autě. Dnešní mikrokontroléry by měly poskytovat dostatek výkonu na základní zpracování obrazu z kamery, tím pádem by se měla eliminovat nevýhoda tohoto řešení.

4.2 Výběr výpočetní jednotky

Vhodných výpočetních jednotek, které by mohly ovládat autíčko je na trhu mnoho, autor práce se však zaměřil pouze na nejznámější dvě a to mikrokontrolér Arduino a „mini“ počítač Raspberry Pi. Mezi jiné výpočetní jednotky by se mohly řadit mikrokontroléry BeagleBone, nebo návrh vlastního řídicího systému s využitím mikrokontrolérů, například od společnosti Atmel.

4.2.1 Arduino

Arduino je mikrokontrolér s volně přístupným (open-source) HW řešením, znamená to tedy, že jsou k dostání schémata zapojení tohoto mikrokontroléru se všemi jeho periférie mi. Mikrokontrolér byl vyvinut v Itálii ve městě Ivrea na škole „Interaction Desing Institute Ivrea“. Prvním a hlavním účelem Arduina bylo zpřístupnění prototypování studentům. Po zpřístupnění mikrokontroléru široké veřejnosti se začaly modely adaptovat na poptávku a vyráběly se tak zaměřené na určité problémy (například 3D tisk), nevyráběly se už jen 8-bitové [15].

Arduino využívá ARM procesorů převážně od společnosti Atmel (některé modely využívají procesory například od společnosti Intel, nebo ARM Cortex procesory od společnosti Arm Holdings). Procesory využívají RISC (Reduced Instruction Set Computer - procesory s redukovanou instrukční sadou) architekturu a jsou nenáročné na napájení, avšak dosahují menších výkonů. Operační paměť se pohybuje v rozmezí kilobajtů. Mikrokontroléry Arduino mají instalovanou malou paměť pro ukládání programu a dat. Pro napájení využívají konektoru USB, přes který je rovněž prováděno programování. Mezi periférie patří pouze digitální a analogové vstupy a výstupy. Mikrokontroléry Arduino neobsahují žádné standardní výstupy pro monitor, či připojení externích periférií jako klávesnici, nebo myš [16].

Arduina jsou více určené pro jednoúčelové aplikace, které nevyžadují větších výpočetních výkonů a jsou nenáročné na výpočetní paměť. Typickým příkladem jejich využití může být sledování hodnot z více senzorů a na jejich základě vyhodnotit, co se má provádět dále. V případě teploty by se buďto začalo, nebo přestalo topit v místnosti. Mikrokontroléry Arduino se nehodí na aplikace, které vyžadují větší výpočetní výkon a hlavně více operační paměti, nedají se tedy využít například na zpracování velkého množství dat, nebo pro provoz operačních systémů [17].

Parametr	Arduino UNO Rev3	Arduino Micro	Arduino Zero
CPU	16 MHz	16 Mhz	48 MHz
RAM	2 kB SRAM	2,5 kB SRAM	32 kB SRAM
Napájení	5 V	5 V	3,3 V
Paměť	32 kB	32 kB	256 kB
Rozměry (ŠxD)	53,4 mm x 68,6 mm	18 mm x 48 mm	53 mm x 68 mm
Cena	599 Kč	299 Kč	1 020 Kč

Tab. 4.1: Porovnání mikrokontrolérů Arduino [16, 18]

4.2.2 Raspberry Pi

Podobně jako mikrokontroléry Arduino vznikly „mini“ počítače Raspberry Pi jako výukový nástroj, který si klade za úkol naučit lidi programovat. Raspberry Pi vzniklo ve Spojeném království Velké Británie a Severního Irska a bylo vyvinuto díky nadaci „The Raspberry Pi Foundation“ [19].

Na rozdíl od mikrokontrolérů Arduino jsou „mini“ počítače Raspberry Pi výkonnější, využívají více jádrové ARM procesory, od společnosti Broadcom, taktované již na gigahertze (počet jader procesoru závisí na modelu). Procesory v sobě mají integrován také grafický čip. Procesoru sekunduje větší operační paměť, od 0,5 GB

po 4 GB paměti. Raspberry Pi má i více periférií, počet a druh se liší dle modelu, všechny však mají výstup pro připojení monitoru a alespoň jeden USB konektor pro připojení periférií. Raspberry Pi „mini“ počítače tak představují miniaturní náhradu stolního počítače s omezeným výkonem, avšak také nižší cenou. Operačním systémem může být například volně dostupný Raspbian OS (existuje však více operačních systémů). Jedná se o Linuxový operační systém, který si uživatelé mohou nainstalovat buď s grafickým rozhraním, či bez něj. Oproti mikrokontrolérům Arduino neobsahují analogové vstupy a výstupy. GPIO (General Purpose Input and Output - Univerzální vstupně výstupní pin) piny fungují pouze jako digitální vstupy a výstupy, které však disponují různými módy (např. PWM, RS232 sběrnice, či I2C sběrnice). Ke všem modelům Raspberry Pi lze také připojit kameru [19].

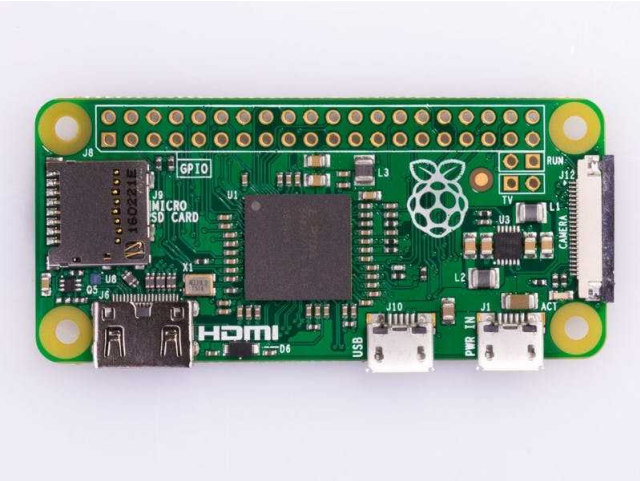
Raspberry Pi „mini“ počítače se převážně používají na složitější aplikace, kde je zapotřebí větších výpočetních výkonů. Často se využívají například v autonomních RC modelech aut, kde vyhodnocují obraz z kamery a řídí tak celý model auta. Naopak v případě, kdy je potřeba analogového řízení, či řízení „real-time“ aplikací (řízení aplikací v reálném čase - můžeme přesně říct, jak dlouho potrvá vykonat danou instrukci a tento čas bude neměnný), je lepší pracovat s mikrokontrolérem Arduino [17].

Parametr	Raspberry Pi 3B	Raspberry Pi Zero	Raspberry Pi 2B
CPU	4x1,2 GHz	1 GHz	4x900 MHz
RAM	1 GB RAM	512 MB RAM	1 GB RAM
Napájení	5 V 2,5 A	5 V 1 A	5 V 2,5 A
Paměť	nemá (mikro SD)	nemá (mikro SD)	nemá (mikro SD)
Rozměry (ŠxD)	85 mm x 56 mm	65 mm x 30 mm	85 mm x 56 mm
Cena	888 Kč	157 Kč	1 029 Kč

Tab. 4.2: Porovnání mikrokontrolérů Raspberry [19, 20]

4.2.3 Vybraná výpočetní jednotka

Na základě výše popsaných vlastností mikrokontrolérů Arduino (viz podkapitola 4.2.1) a „mini“ počítačů Raspberry Pi (viz podkapitola 4.2.2) se autor práce rozhodl pro využití „mini“ počítače Raspberry Pi, konkrétně modelu Raspberry Pi Zero bez zabudovaného Wi-Fi přijímače. Tento model byl vybrán na základě rozměrů (viz tabulka 4.2 a tabulka 3.1), bylo potřeba zajistit, aby se výpočetní jednotka vešla na autíčko, popřípadě, aby autíčko nebylo příliš vysoké v případě, že by výpočetní jednotka byla umístěna na autíčko vertikálně.



Obr. 4.4: Raspberry Pi Zero [19]

Zároveň toto řešení vyhovuje zvolené architektuře (viz podkapitola 4.1.4), jelikož „mini“ počítače jsou dostatečně výkoné pro zpracování obrazu z kamery a zároveň ovládání modelu autíčka. Zpracování obrazu bylo hlavní kritérium, proč nebyl vybrán mikrokontrolér Arduino.

Nevýhodou zvoleného řešení je pracnější implementace Raspberry Pi Zero na autonomní autíčko. Raspberry Pi Zero neobsahuje žádný jiný zdroj napájení, než přes mikro-USB konektor a neobsahuje ani regulátory napájecího napětí. Po této stránce bude zapotřebí navrhnout vlastní DPS pro napájení Raspberry Pi Zero, stejně tak i pro ovládání elektromotorku autíčka. Jelikož Raspberry Pi „mini“ počítače neobsahují analogové výstupy, bude muset být použito řízení motorku pomocí PWM.

4.2.4 Vybraný kamerový modul

Kamer pro Raspberry Pi existuje mnoho, lze k nim totiž připojit i web kamery pomocí USB konektoru. Lepší je však připojit kameru, která je přímo určená pro Rasbperry Pi „mini“ počítače. Výběrem takové kamery je zajištěna kompatibilita se systémem.

Pro řešenou úlohou v této práci připadá nejvíce v úvahu kamera RPI Zero V1.3 Camera a to hlavně kvůli svým rozměrům. Zároveň je kamera specifická tím, že ji lze připojit pouze do Raspberry Pi Zero. Tento model jako jediný z vyráběných má CSI konektor pro kameru s menšími rozměry. Je zapotřebí, aby kamera byla dostatečně malá a bylo jí tak možné umístit na výsledný model autíčka. Zároveň není potřeba zbytečně dlouhých propojovacích vodičů. Porovnání kamer, jejich velikosti a schopnosti záznamů, včetně délky přívodních vodičů se nachází v tabulce níže.

Kamera	Zorný úhel (diagonální)	Rozměry (ŠxV) [mm]	Rozlišení videa a FPS
RPI Camera V2	62,2 °	25 x 24	1080p 30 FPS
RPI Zero V1.3 Camera	69,1 °	11,5 x 9*	1080p 30 FPS
RPI Camera (I)	170 °	32 x 32	1080p 30 FPS
RPI Camera (M)	200 °	25 x 24	1080p 30 FPS

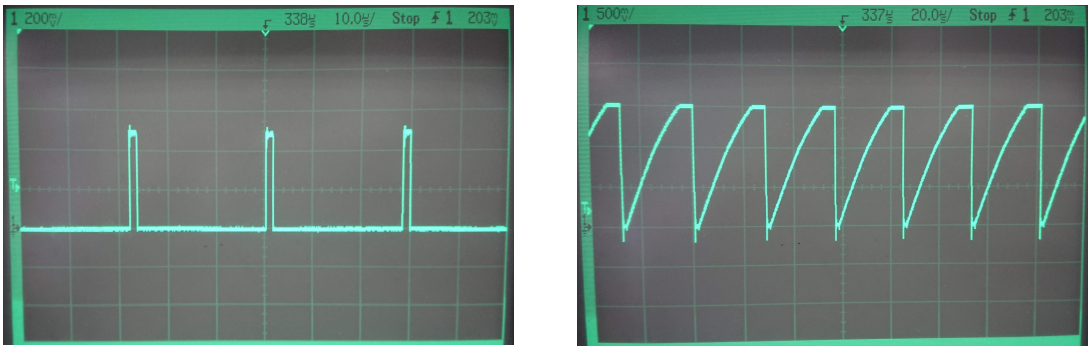
Tab. 4.3: Porovnání kamer k Raspberry Pi [21]; *uvedeno bez přívodních vodičů (na rozdíl od zdroje)

4.3 Způsob ovládání autonomního autíčka

Jak již bylo zmíněno v podkapitole 4.2.3 při volbě vhodné výpočetní jednotky, tak elektromotorek modelu autonomního autíčka bude muset být řízen pomocí PWM signálu. Pro řízení bylo využito stávajícího řízení, které použili tvůrci modelu autíčka. Z dekodéru autíčka bylo použito tranzistoru T1 a příslušných součástek pro spínání elektromotorku (diody D3 a D4 a kondenzátory C2 a C3, viz obrázek 3.3).

Elektromotorek je ovládán pomocí tranzistoru T1, kdykoliv je na G tranzistoru přivedeno kladné napětí, tranzistor sepne a spojí tak zápornou svorku motorku se zemí, obvodem tak začne protékat proud a motorek se začne točit. Řízení pomocí PWM zajišťuje regulaci proudu protékajícího elektromotorkem. Zvětšování střídy PWM signálu znamená zvyšování proudu protékajícího elektromotorkem (zvyšuje se totiž četnost proudových pulsů na motorku). Pomocí tohoto ovládání jsme schopní rychlost autíčka plynule ovládat až na maximální hodnotu, která se zdá být 80 % střídy, jelikož při plné rychlosti autíčka (ovladač zmáčknut na maximum) hodnota střídy tuto hodnotu nepřekročí. Výrobce autíčka se tak nejspíše rozhodl pro zvýšení životnosti motorku, jelikož nikdy nedosáhne své maximální rychlosti. Motorek je řízen pomocí PWM signálu s frekvencí f = 35 kHz, napětím U = 15 V (spínáno tranzistorem T1, ovládání tranzistoru probíhá při U = 3,3 V - výstupní napětí PWM signálu z Raspberry Pi Zero) a volitelnou střídou od 0 % do 80 %.

Na obrázku 4.5 lze vidět dva průběhy PWM signálu. Obrázek potvrzuje, že při přivedení kladné hodnoty na G tranzistoru T1 dojde k propojení minusové svorky elektromotorku s GND, motorkem tak začne protékat proud a začne se otáčet. Tohoto si lze všimnout na obrázku 4.5b. Při kladné hodnotě napětí na tranzistoru je elektromotorek uzemněn. Jakmile se tranzistor uzavře, dochází k nabíjení kondenzátoru C3 na svorkách elektromotorku a motorek se přestane otáčet.



(a) (b)

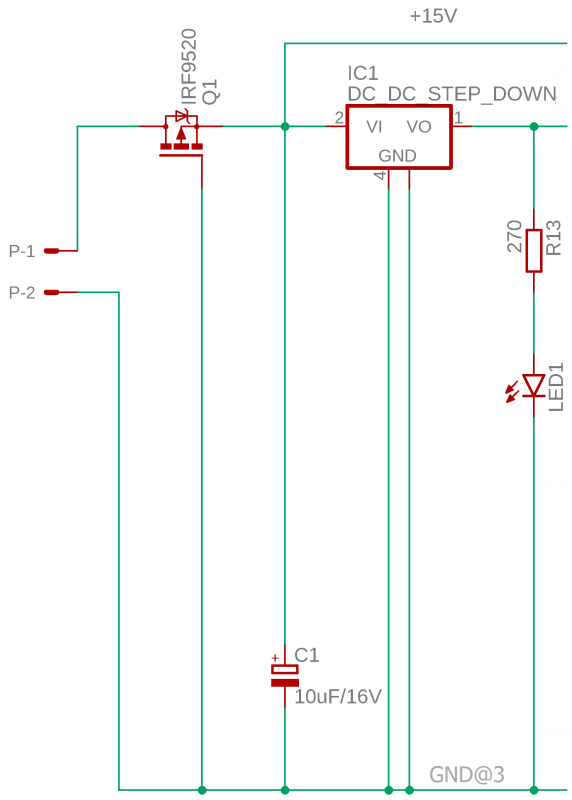
Obr. 4.5: a PWM signál naměřený na G tranzistoru T1, při nejnižší rychlosti; b PWM signál naměřený na mínus svorce elektromotorku při nejnižší rychlosti

5 Návrh DPS s periferiemi

Následující kapitola popisuje návrh DPS vytvořené pro ovládání motorku autíčka, světel a napájení Raspberry Pi Zero. Elektrické návrhy a DPS byly provedeny v programu Eagle 9.5.2 od společnosti Autodesk. Firma nabízí software zdarma na omezenou dobu pro nekomerční využití a zdarma neomezeně pro studenty.

5.1 Elektrické schéma

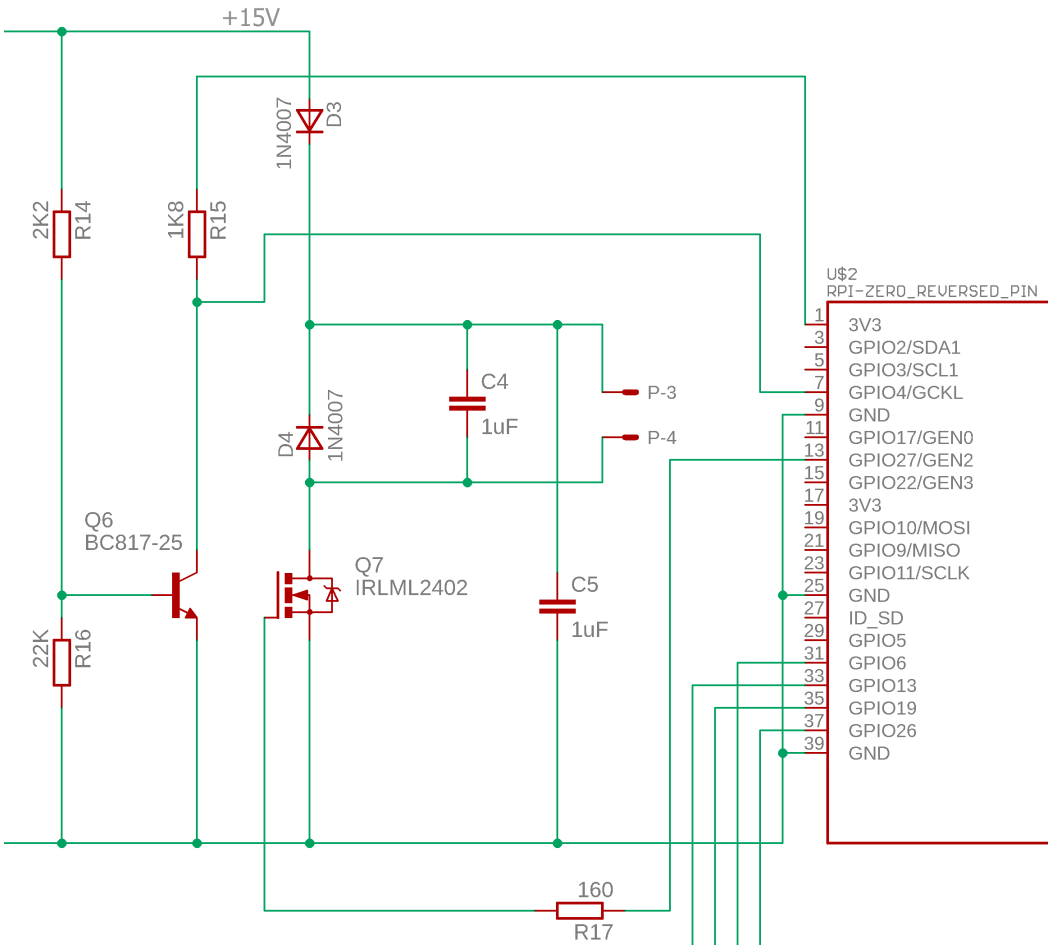
V elektrickém schématu se pracuje se dvěma úrovněmi napětí, 15 V a 5,7 V. Vyšší napětí je napětí napájecí, které je získáno z kolejnic autodráhy. Nižšího napětí je docíleno pomocí DC/DC měniče napětí (IC1). DC/DC měnič byl zvolen z důvodu malého zahřívání měniče, není tak potřeba využívat pasivní chlazení (viz podkapitola 1.1.2).



Obr. 5.1: Napájecí část elektrického obvodu

Jelikož je možnost položit autíčko na kolejnice různými způsoby a tím pádem také obrátit polaritu napájení z kolejnic, bylo potřeba vyřešit ochranu proti obrácené polaritě. Tuto ochranu zajišťuje tranzistor Q1 (viz obrázek 5.1), jedná se o MOSFET tranzistor s P vodivým kanálem. Výhody využití tranzistoru jsou zmíněny v kapitole 1.2. Elektroda D (Drain) je připojena k vstupnímu napětí 15 V, G (Gate) elektroda je připojena k zemi a S (Source) elektroda rozvádí napětí 15 V dále do obvodu. Při připojení správné polarity bude tranzistor otevřený a bude jím protékat proud, v opačném případě bude uzavřen a proud nebude protékat.

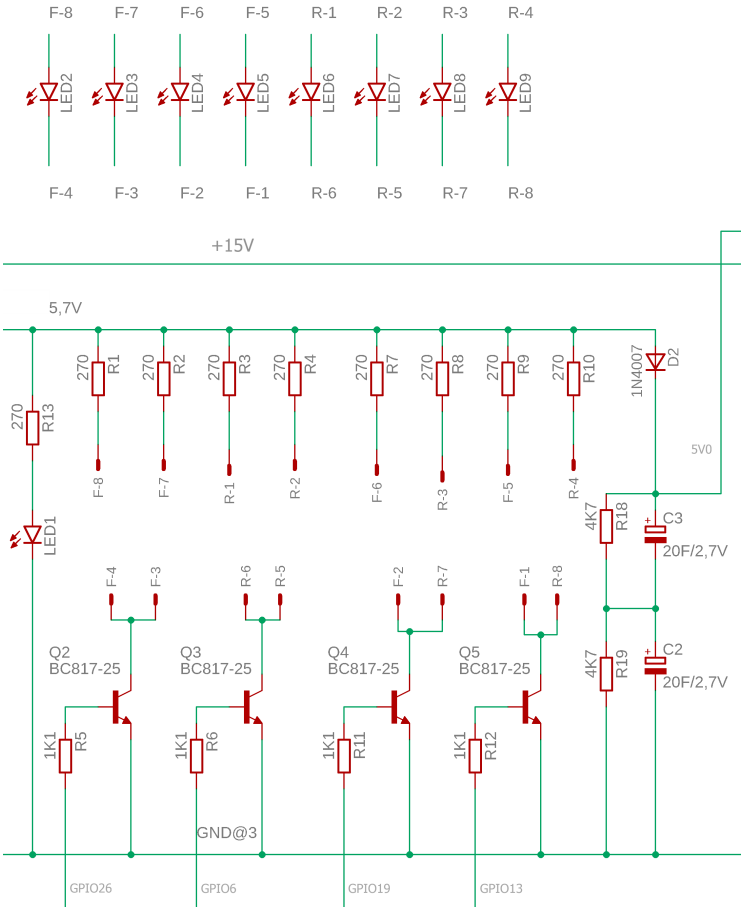
Ovládání motorku je popsáno v podkapitole 3.2, v konečném návrhu je ovládání totožné, ale s jinými typy součástek. Jiné typy součástek byly zvoleny z důvodu snadné dostupnosti, změna typu nemá vliv na funkčnost. Diody D3 a D4 zajišťují filtraci proti nechtěným napěťovým špičkám. Tranzistor Q7 zajišťuje spínání motorku k zemi. Kondenzátor C5 zajišťuje filtraci vstupního napětí motorku a kondenzátor C4 funguje jako náběhový kondenzátor. Gate tranzistoru Q7 je před rezistor R17 připojen k PWM výstupu „mini“ počítače Raspberry Pi Zero.



Obr. 5.2: Část obvodu pro ovládání elektromotorku a signalizaci výpadku napájení

Pro indikaci výpadku vstupního napájení je využito bipolárního NPN tranzistoru Q6 (viz obrázek 5.2). Jedná se o jednoduchý spínací obvod. Báze tranzistoru je přes rezistor R14 připojena k napájecímu napětí 15 V. Kolektor je přes rezistor R15 spojen s výstupním napětím 3,3 V, které generuje Raspberry Pi Zero, napětí je poté přivedeno na GPIO pin 4, emitor tranzistoru je připojen k zemi. Zapojení zajišťuje, že v případě, kdy je přivedeno napájecí napětí na bázi tranzistoru, dojde k otevření tranzistoru a na GPIO pinu 4 se objeví nulové napětí (log. 0). V případě výpadku napájecího napětí je báze tranzistoru připojena přes tzv. pull-down rezistor R16 k zemi a tranzistor bude uzavřen. Na GPIO pinu 4 se tak objeví napětí 3,3 V (log. 1).

Vzhledem k tomu, že autíčko může z autodráhy vypadnout a tím pádem ztratit napájení, je nezbytné opatřit jej záložním zdrojem napájení. Tento zdroj zajistí bezpečné vypnutí Raspberry Pi Zero při ztrátě napájení. Eliminuje se tak riziko poškození dat. Vzhledem k jednoduchosti řešení a omezenému prostoru zvolil autor práce jako záložní zdroj napájení dva super kondenzátory (C3 a C2, viz obrázek 5.3). O výhodách a nevýhodách pojednává kapitola 1.3.



Obr. 5.3: Část obvodu pro ovládání LED diod a zálohování napájení

Tranzistory Q2 až Q5 slouží pro ovládání LED diod na modelu autíčka (jak lze vidět na obrázku 5.3). Tranzistory jsou pomocí báze přes rezistor připojeny k GPIO pinům 6, 13, 19 a 26. Na kolektor tranzistoru je přivedena katoda LED diod a pomocí emitoru je tranzistor spojen se zemí. Rezistory připojené k bázi tranzistoru mají za úkol limitovat proud odebíraný z Raspberry Pi Zero. Proud odebíraný ze všech GPIO pinů dohromady nesmí překročit 50 mA (limit Raspberry Pi Zero [22]). Při sepnutí tranzistoru dojde k otevření tranzistoru a katoda LED diody se spojí ze zemí a diodou i tranzistorem začne protékat proud.

Elektrolytický kondenzátor C1 slouží pro filtraci vstupního napětí od Manchester kódování. Rezistory R1 až R4; R7 až R10 a rezistor R13 limitují proud, který protéká LED diodami. Usměrňovací dioda D2 zabraňuje, aby super kondenzátory napájely LED diody, napětí ze super kondenzátorů slouží pouze pro napájení Raspberry Pi Zero. Z tohoto důvodu bylo zvoleno napětí 5,7 V za DC/DC měničem. Napětí za diodou D2 je totiž sníženo o její úbytek (0,7 V) a napětí na Raspberry Pi Zero je pak 5 V. Rezistory R18 a R19 slouží pro rovnoměrné rozložení napětí mezi super kondenzátory C3 a C2 [23]. Rovnoměrného rozložení napětí je dosaženo vybíjením kondenzátoru, který je nabitý na vyšší hodnotu, přes jeho paralelní rezistor. Tímto způsobem dochází k vyrovnání napětí na obou kondenzátorech. Rezistor R17 slouží, stejně jako rezistory R5, R6, R11 a R12, k limitování odebraného proudu z GPIO pinů.

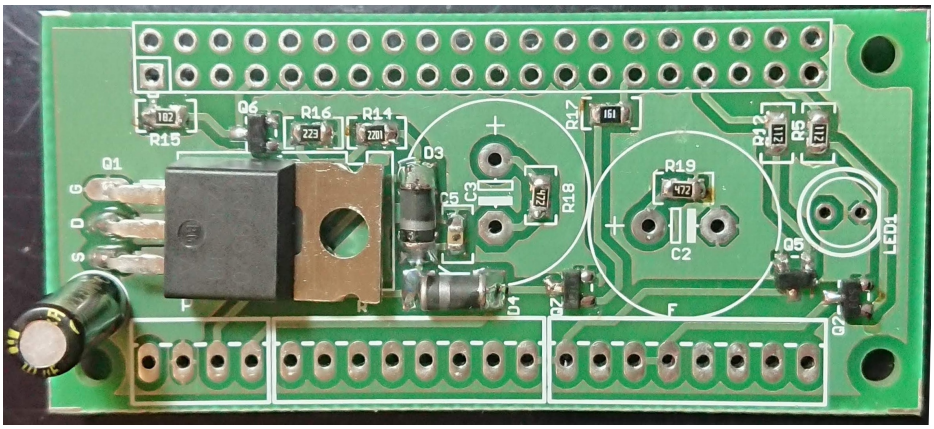
Elektrické schéma a seznam součástek se nachází v příloze B.

5.2 Deska plošných spojů

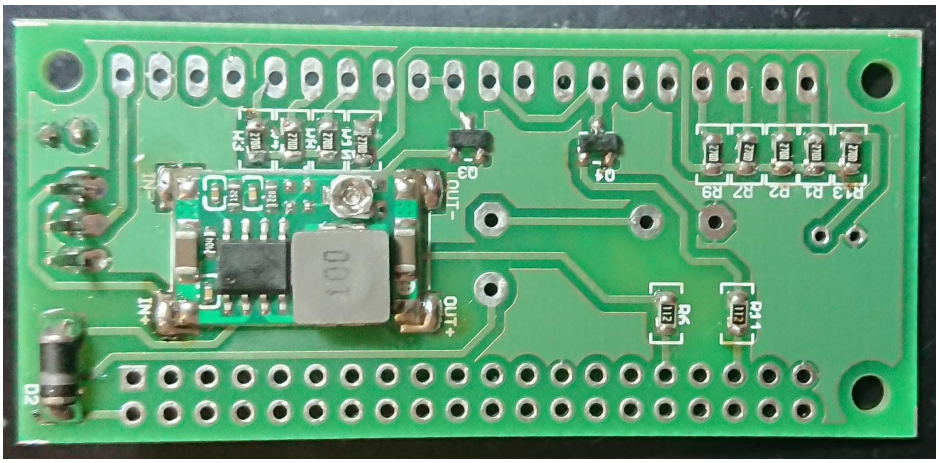
Deska plošného spoje byla vyrobena firmou APAMA sídlící v Brně a osazována ručně. DPS byla navržena jako dvouvrstvá a se stejnými rozměry jako Raspberry Pi Zero (viz 4.2). Díky malé velikosti je deska navržena pro povrchové osazování většiny součástek.

Deska plošného spoje je napájena přes konektor P - piny 1 a 2. Ze stejného konektoru také vedou řídící vodiče k motoru autíčka - piny 3 a 4. Pomocí konektorů F a R jsou do obvodu připojeny LED diody. Napájení Raspberry Pi Zero pak probíhá přes GPIO pin - konkrétně pin 2.

Návrh DPS se nachází v příloze B.



Obr. 5.4: Částečně osazená horní vrstva DPS



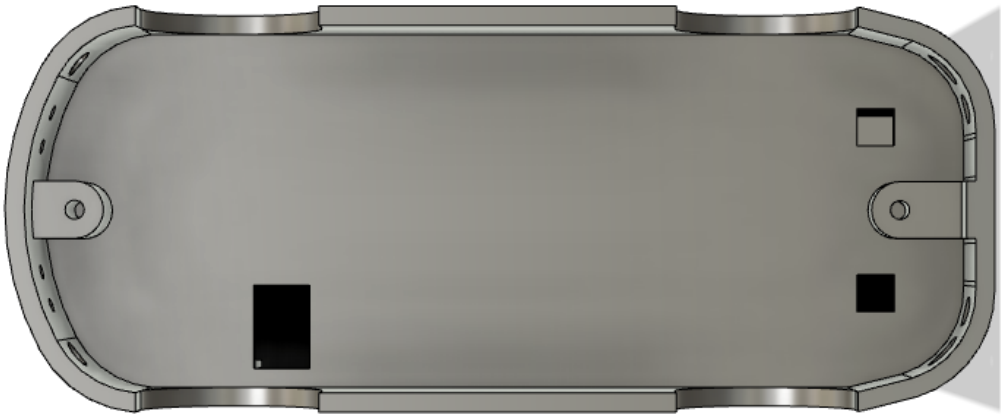
Obr. 5.5: Částečně osazená spodní vrstva DPS

6 Model autíčka a úprava autodráhy

Následující kapitola popisuje vytvořený 3D model autonomního autíčka, krabičky pro stávající ovládání autodráhy a úpravy, které byly provedeny na autodráze. Veškeré 3D modely byly vytvořeny pomocí 3D modelovacího programu Fusion 360 od společnosti Autodesk. Společnost Autodesk nabízí program zdarma pro studenty. Modely byly následně vytisknuty pomocí 3D tiskárny.

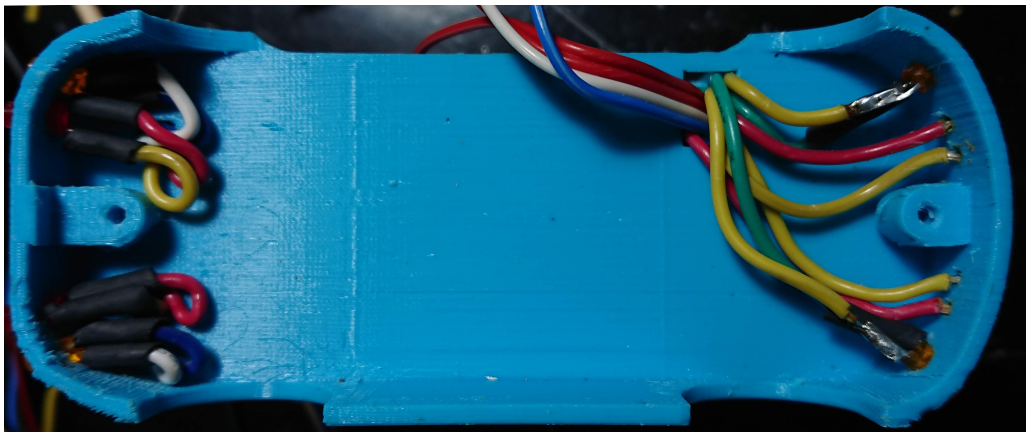
6.1 Model autíčka

Pro autíčko byla navrhnuta zcela nová karoserie, která umožňuje uložení „mini“ počítače a výše zmíněných elektrických součástek. Model je navržen tak, aby byla umožněna relativně snadná výměna součástek v případě poškození. Z původního autíčka Porsche GT3 RS (popsaného v podkapitole 3.3) byl zachován pouze podvozek (viz obrázek 3.4d). Nová karosérie byla navržena s respektem k podvozku.



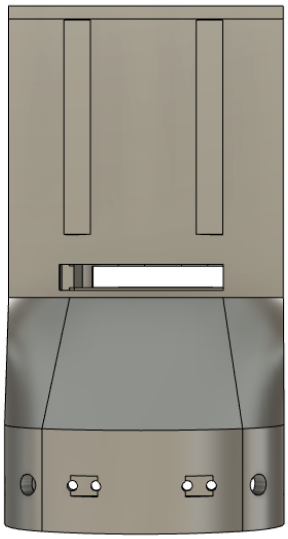
Obr. 6.1: Spodní část navrženého modelu

Na spodní části modelu se nachází díry pro šroubky, díky nimž dojde ke spojení modelu s podvozkem auta. Jsou zde vytvořeny také tři díry pro elektrické vodiče, pomocí kterých bude možné propojit světla (LED diody) autíčka s periferní deskou.



Obr. 6.2: Spodní část modelu autíčka s vodiči

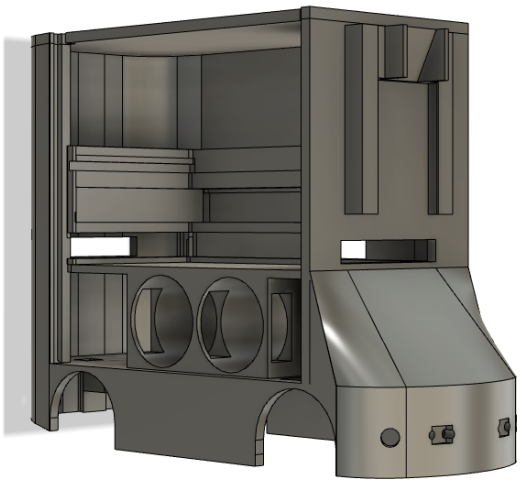
Přední část modelu obsahuje drážky pro držák na kameru. Drážky jsou záměrně umístěny na středu autíčka. Tímto bude zajištěno, že kolejnice, které slouží k jízdě autíčka, se v obraze budou nacházet v blízkosti středu osy X. Pod drážkami je otvor pro konektor, aby bylo možné kameru připojit k Raspberry Pi Zero. Ve spodní části obrázku si lze také všimnout otvorů pro přední světla a blinkry.



Obr. 6.3: Přední část navrženého modelu

Uvnitř modelu se nachází otvory pro uložení super kondenzátorů. Před kondenzátory se nacházejí průchody pro vedení drátů ke světlům, motoru a napájení od kolejnic. Dráty jsou umístěny před kondenzátory z toho důvodu, aby nedocházelo k jejich pohybu během jízdy. Na zadní stěně je umístěn otvor pro připojení USB

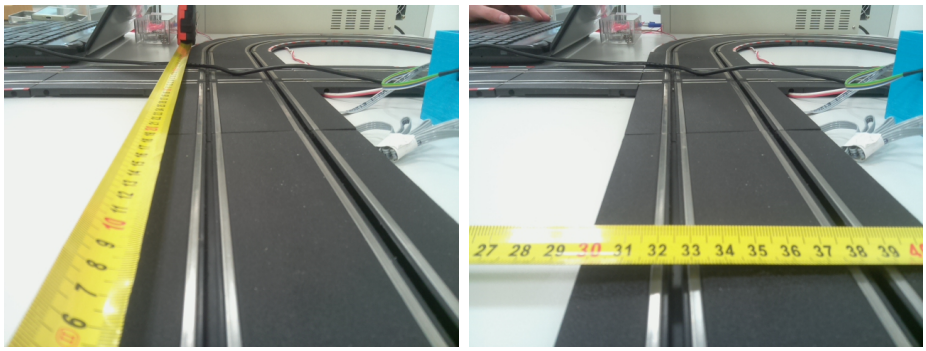
konektoru k Raspberry Pi Zero. Desky plošných spojů jsou v modelu uloženy do dvou drážek, které je drží na místě.



Obr. 6.4: Boční část navrženého modelu

Zadní část modelu má poté jen výřez pro vyjmutí paměťové karty z Raspberry Pi Zero a otvory pro zadní světla a blinkry. K modelu byly navrženy také „dveře“, které zakryjí vnitřní část autíčka. „Dveře“ jsou odnímatelné a v modelu drží pomocí dvou drážek.

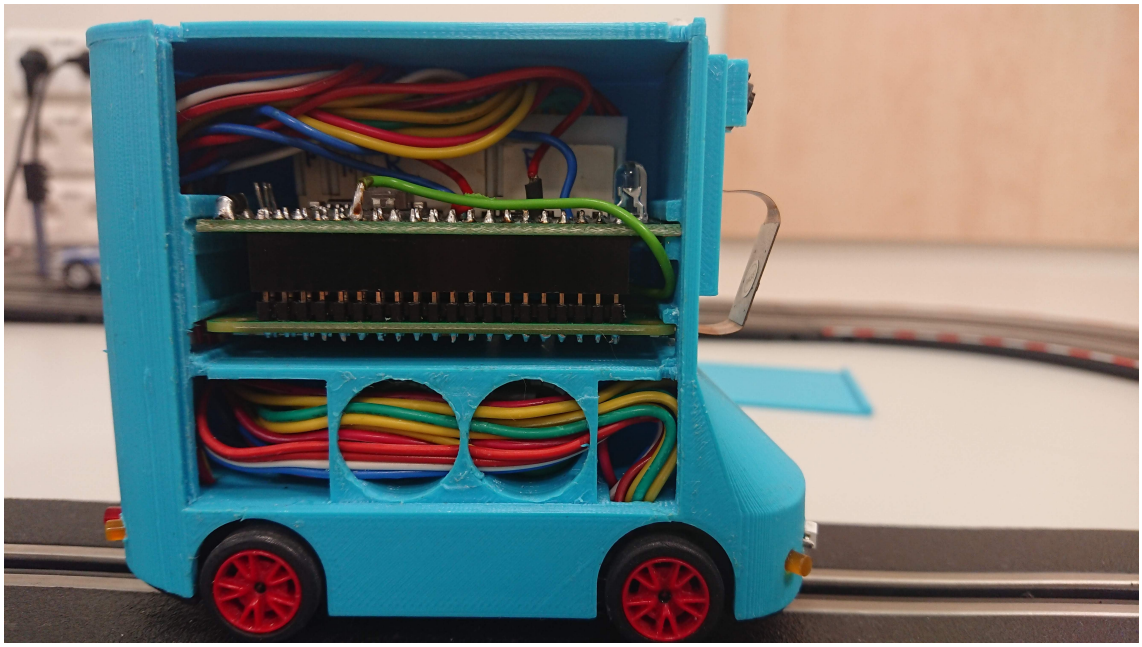
Držák pro kameru je navržen pod úhlem 26°. Tento úhel je zvolen proto, aby kamera dokázala zabírat prostor alespoň 7 cm před vozidlem. Vzdálenost 7 cm odpovídá zhruba jedné délce autíčka na autodráze. Držák je za pomocí drážek polohovatelný. Je možné měnit výšku umístění kamery. Kamera musí být umístěna minimálně ve výšce semaforů, aby je mohla rozpoznat. Zorné pole kamery lze vidět na obrázku 6.5. Držák kamery je umístěn stejně jako je vyobrazeno na obrázku 6.3.



(a) (b)

Obr. 6.5: Zorné pole autíčka

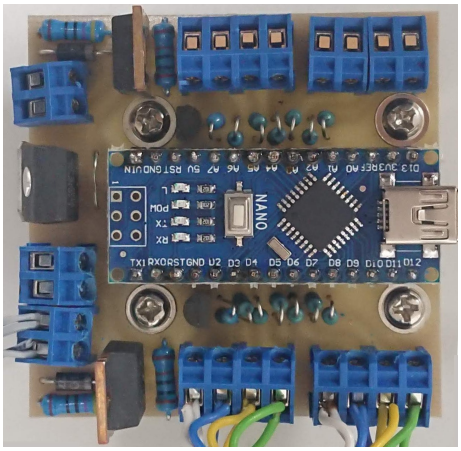
Nově navržené autíčko je vysoké 8 cm, široké 4,2 cm a dlouhé 10,4 cm. Při porovnání rozměrů s tabulkou 3.1, je vidět, že se nejvíce změnila výška. Tuto změnu rozměrů bylo možné předpokládat, jelikož byl použit původní podvozek. Rozdíly v délce a šířce jsou způsobeny tloušťkou tisku. Po zkompletování autonomního autíčka však razantně stoupla váha a to čtyřnásobně, původní autíčko vážilo 40 gramů a nové váží 160 gramů.



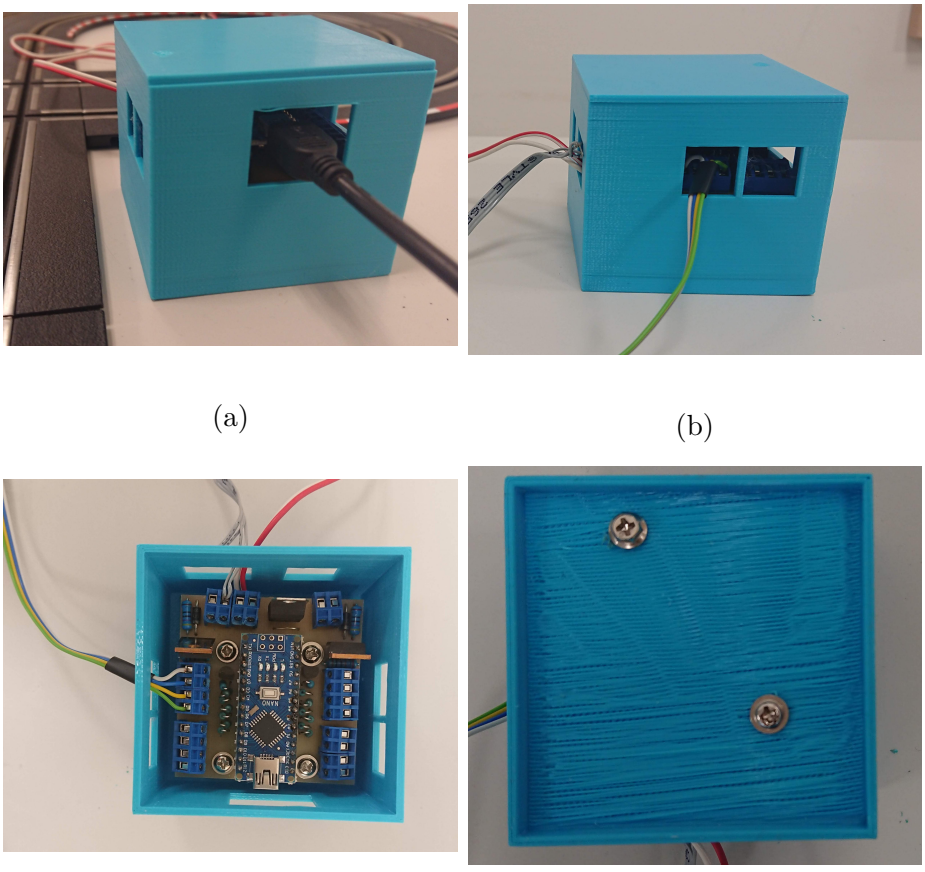
Obr. 6.6: Autíčko s veškerou elektronikou

6.2 Model krabičky pro ovládání autodráhy

Ke stávajícímu ovládání autodráhy pomocí mikrokontroléru Arduino byla navržena krabička. Krabička umožňuje připojení všech nezbytných periferií pro její funkčnost, včetně USB konektoru pro programování Arduina. Krabička tak na svých bočních stěnách disponuje výřezy pro připojení napájení, semaforů, autodráhy a výše zmíněného USB konektoru. Deska s ovládáním je ke krabičce připevněna pomocí dvou šroubků na spodní straně. Bylo vytvořeno také víko, které umožní zakrytí veškeré elektroniky uvnitř.



Obr. 6.7: Mikrokontrolér Arduino ovládající autodráhu



(a) (b)

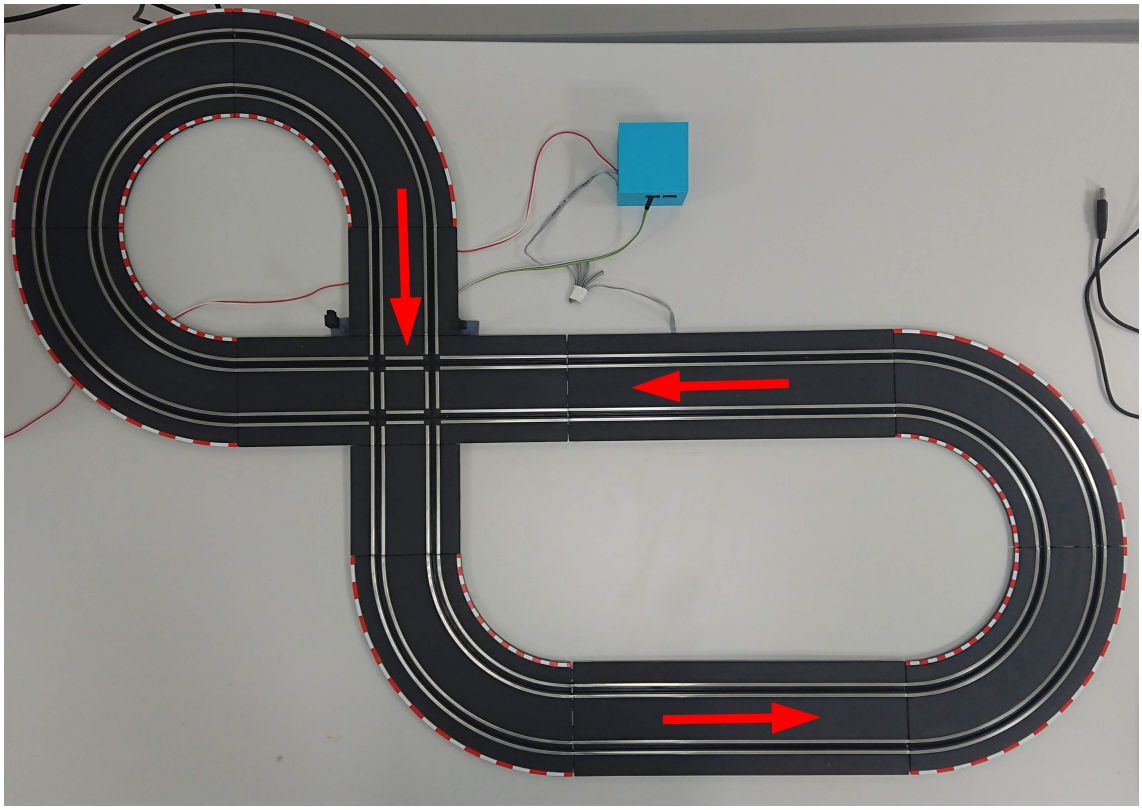
(c) (d)

Obr. 6.8: Různé pohledy na krabičku pro ovládání autodráhy

6.3 Autodráha

Autodráha byla upravena, aby vyhovovala HW omezením autíčka - zornému poli kamery, a aby bylo možné názorně demonstrovat vyvinuté algoritmy, popsané v kapitole 8. Na dráze se vyskytuje pravá, i levá zatáčka. Díky požadavku zastavení na semaforu (při svítící červené) se zde nachází i díl s křižovatkou.

Je předpokládáno, že autíčko bude po autodráze jezdit ve směru šipek uvedených na obrázku. Zároveň, aby autíčko bylo schopné nalézt semafor, je potřeba autíčko umístit do pravé kolejnice. Pro nalezení semaforu je využito předpokladu, že se bude vždy nacházet v pravém horním rohu snímaného prostoru (viz podkapitola 8.3.1).Krátká rovinka před křižovatkou zajišťuje spolehlivější detekci semaforu. Při výjezdu ze zatáčky by se totiž mohlo stát, že se semafor nedostane do zorného pole kamery.



Obr. 6.9: Mikrokontrolér Arduino ovládající autodráhu

Autodráha je umístěna na bílé desce, pomocí které lze s celým modelem manipulovat. Pro plnou funkčnost autodráhy je potřeba, společně s autodráhou, mít k dispozici také zdroj. Zdroj musí být schopný dodat napětí 15 V a proud alespoň 1 A.

7 Programové vybavení

Pro vykonávání operací zpracovávání obrazu je využito knihovních funkcí volně dostupné knihovny (open-source) OpenCV 3.4.3 [24]. Pro ovládání kamery z C/C++ programu je využito knihovny RaspiCam 0.1.8, která je volně šířená pod licencí BSD a dostupná z [25]. Pro práci s GPIO piny je využito knihovny WiringPi.h sestavené panem Gordonem Hendersonem. Knihovna je i s popisem jednotlivých funkcí dostupná z těchto webových stránek [26]. Následující kapitola popisuje správnou instalaci operačního systémy a zvolených knihoven na Raspberry Pi Zero.

7.1 Operační systém

Jak již bylo zmíněno v podkapitole 4.2.2 „mini“ počítače Raspberry Pi Zero jsou dostatečně výkonné na zprovoznění operačního systému. Pro tuto práci byl pro zvolený systém Raspberry Pi Zero vybrán operační systém Raspbian Buster Lite ve verzi 2019-09-30 (dostupné z [27]).

Operační systém Raspbian Buster Lite představuje volně dostupný operační systém využívající Linuxového jádra. Verze Lite znamená, že se jedná o operační systém bez grafického rozhraní. Vzhledem k tomu, že v této práci nebude potřeba vůbec grafického prostředí, ani výstupu pro připojení monitoru k Raspberry Pi Zero, je tento operační systém vyhovující. Navíc, díky absenci grafického rozhraní bude systém rychlejší, jelikož nebude muset tyto informace zobrazovat. Díky oficiální distribuci tohoto OS je zajištěna maximální kompatibilita s dostupným HW (konkrétně Raspberry Pi Zero).

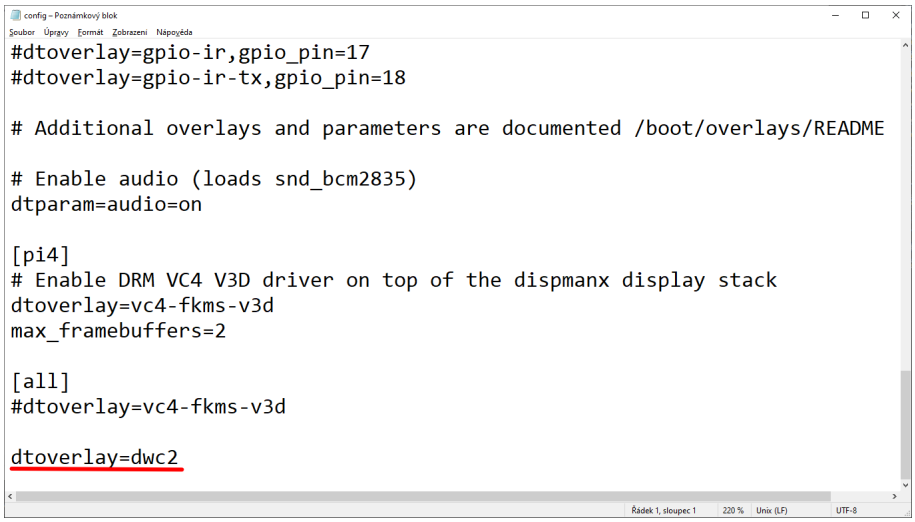
Žádná verze Raspberry Pi nedisponuje vlastní pamětí programu a dat (viz tabulka 4.2). Z toho důvodu je pro účely běhu operačního systému, ukládání dat a programování nezbytné využít externí mikro SD paměťové karty. Tvůrci Raspberry Pi doporučují pro operační systém Raspbian Buster Lite paměťovou kartu o velikosti alespoň 4 GB (viz [28] sekce Installation). Pro tuto práci je použita paměťová karta o velikosti 16 GB. Karta tak disponuje dostatkem místa pro instalaci všech potřebných programů.

7.1.1 Instalace operačního systému

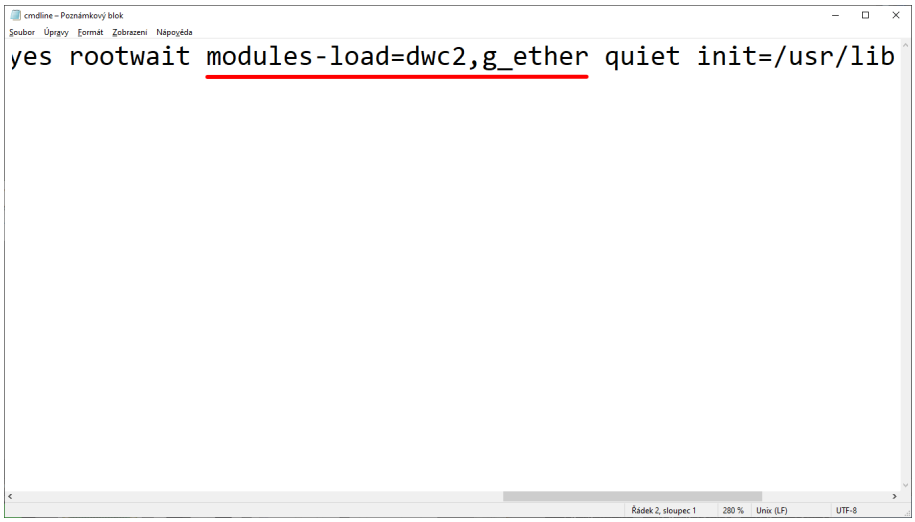
Pro instalaci operačního systému je potřeba z oficiálních stránek Raspberry Pi stáhnout požadovaný operační systém. Operační systém lze stáhnout pouze jako komprimovaný soubor ve formátu **.zip*. Je třeba soubory uložené v komprimovaném formátu extrahovat (například pomocí nástroje *7-Zip*) a získat tak soubor ve formátu **.img*.

Nyní je zapotřebí vytvořit „bootovatelné“ médium ze zvolené mikro SD paměťové karty. Pro tyto účely lze pod OS Windows 10 využít nástroje *Win32 Disk Imager*. Nástroj je zcela intuitivní, stačí zvolit SD paměťovou kartu pro vytvoření „bootovacího“ média a dříve extrahovaný soubor **.img* obsahující vybraný operační systém Raspbian Lite. Velkou výhodou nástroje *Win32 Disk Image* oproti jiným (např. *balenaEtcher*) je možnost vytvoření zálohovacího souboru. Zálohování je velmi důležité a doporučené, jelikož může ušetřit spoustu času při potřebě reinstalace operačního systému, nebo při selhání HW (např. selhání mikro SD paměťové karty).

Aby bylo možné se připojit k Raspberry Pi Zero (bez bezdrátového připojení) přímo z počítače pomocí USB OTG (On-The-Go), je potřeba upravit soubory vytvořené na paměťové kartě. Soubory pro úpravu se jmenují *cmdline.txt* a *config.txt*, lze je nalézt na části paměťové karty označené jako *boot*. V souboru *config.txt* je potřeba na konec souboru přidat nový řádek s textem *dtoverlay=dwc2* (viz obrázek 7.1). V druhém souboru je potřeba vložit text *modules-load=dwc2,g_ether* mezi slova *rootwait* a *quiet* (viz obrázek 7.2) - je nutné přesně dodržet počet mezer, tzn. jednu mezeru před a za přidaným textem. [29]



Obr. 7.1: Úprava souboru *config.txt*



Obr. 7.2: Úprava souboru *cmdline.txt*

Po úpravě souborů je zapotřebí povolit SSH (Secure Shell) připojení. To se provede vytvořením *ssh* souboru (bez přípony) v kořenové části *boot* paměťové karty. Soubor lze vytvořit například tímto příkazem z příkazové řádky ve Windows 10: Po zadání příkazu:

```
> echo >E:\ssh
```

Kde E představuje část *boot* na paměťové kartě.

Takto připravený operační systém Raspbian Buster Lite je připraven pro první spuštění. Stačí paměťovou kartu zasunout do Raspberry Pi Zero, připojit napájení a počkat, než se provedou prvotní operace.

7.1.2 První spuštění systému

Po spuštění upraveného systému popsaného v kapitole výše a připojení Raspberry Pi Zero pomocí USB do počítače, by měl systém Windows 10 indikovat nové připojené zařízení a nainstaloval potřebné ovladače (viz obrázek 7.3). Pokud se tak nestane a ovladače nebudou nainstalovány, je potřeba je nainstalovat ručně (návod na instalaci se nachází zde [29]).

Obr. 7.3: Správně připojené Raspberry Pi Zero

Pro připojení k Raspberry Pi Zero pomocí SSH lze využít například program *PuTTY*. Adresa pro připojení je *raspberrypi.local* a port má číslo *22*. Po úspěšném připojení se objeví příkazový řádek s požadavkem zadat uživatelské jméno a heslo. Při prvním spuštění je uživatelské jméno „pi“ a heslo „raspberry“. Je doporučeno toto nastavení později změnit. V rámci této práce byly změněny přihlašovací údaje dle následující tabulky, rovněž bylo povoleno připojení k systému pomocí SSH jako administrátor.

Typ uživatele	Jméno	Heslo	PORT pro SSH
Administrátor	root	root	2222
Uživatel	pi	autonomous	2222

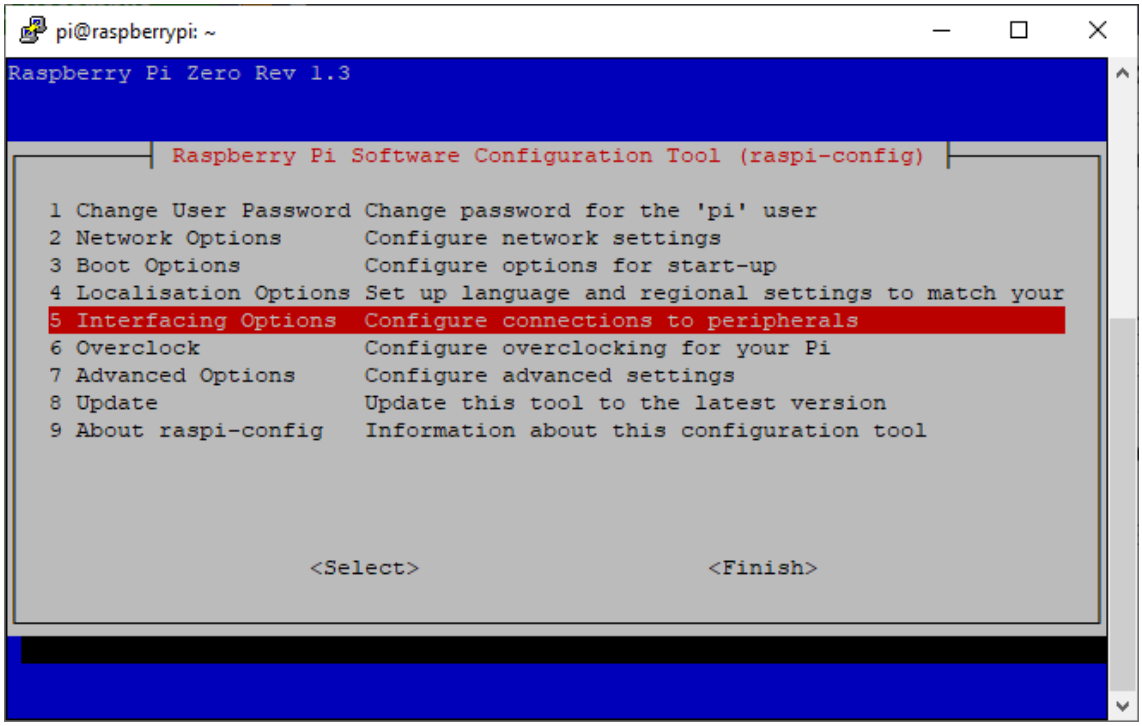
Tab. 7.1: Přihlašovací údaje pro Raspberry Pi Zero

Jelikož k systému bude připojena kamera, je potřeba ji nejprve povolit. Po zadání příkazu:

```
$ sudo raspi-config
```

se otevře konfigurační obrazovka Raspberry Pi Zero. *Sudo* před příkazem znamená, že příkaz bude spuštěn s oprávněním správce. V nastavení lze upravovat základní vlastnosti systému, jako heslo, systémový čas, přetaktování procesoru, nebo také určovat jaké periferie budou k zařízení připojeny. Nabídku lze prohlížet pomocí šipek a potvrzovat klávesou „enter“. Pro nastavení připojení kamery je třeba zvolit

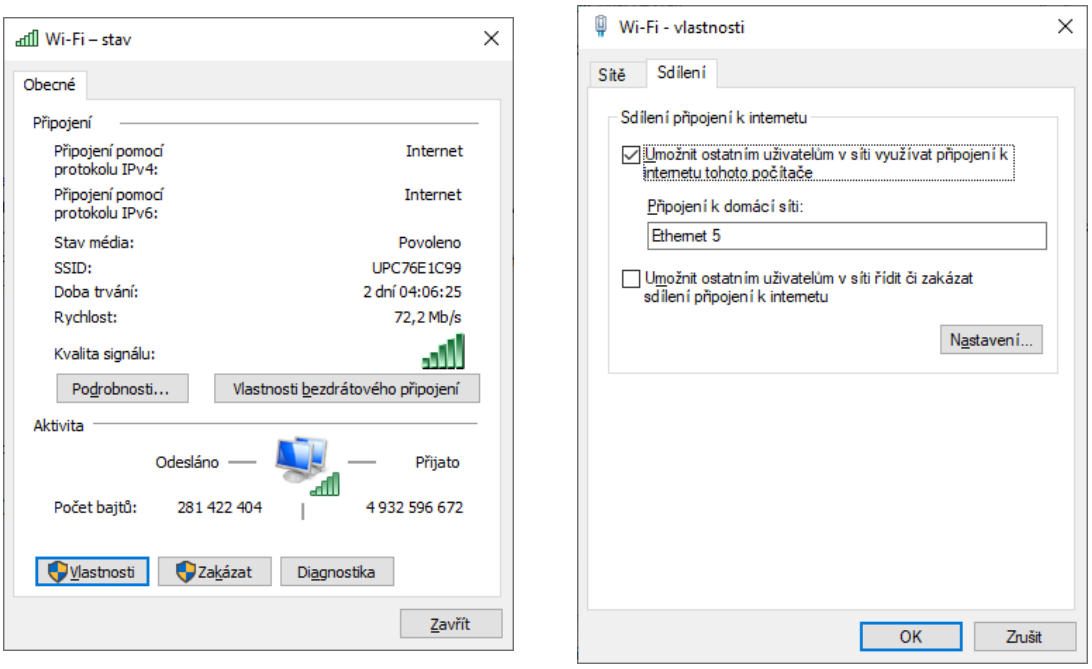
nabídku „Interfacing Options“ (viz obrázek 7.4) a potvrdit volbu klávesou „enter“. Dále stačí vybrat z nabídky kameru a potvrdit povolení připojení.



Obr. 7.4: Základní nastavení systému Raspbian Buster Lite

7.2 Instalace knihoven

Před instalací knihoven do systému, je potřeba systém připojit k internetu. Ve Windows 10 lze sdílet internetové připojení s dalšími zařízeními připojenými k počítači. V centru síťových připojení (ovládací panely -> centrum síťových připojení a sdílení) stačí vybrat aktivní připojení k internetu. Poté ve vlastnostech, na kartě sdílení, povolit sdílení připojení. V poli „Připojení k domácí síti:“ musí být zvoleno ethernetové připojení, které odpovídá připojenému Raspberry Pi Zero a nastavení potvrdit. Nastavení popisuje obrázek 7.5a a 7.5b.



(a) (b)

Obr. 7.5: a Aktivní internetové připojení; b Nastavení sdílení internetového připojení

Po připojení systému k internetu je potřeba aktualizovat operační software Raspbian Buster Lite pomocí příkazu:

```
$ sudo apt-get update && sudo apt-get upgrade
```

Jakmile bude systém aktuální, lze přistoupit k instalaci programových balíčku. Jak je zmíněno v úvodu této kapitoly, ke zpracování vstupu z kamery a obrazových dat je využito knihoven WiringPi, OpenCV a RaspiCam. Knihovna WiringPi je již předinstalována na všech systémech Raspbian. Pro její instalaci stačí zadat příkaz:

```
$ sudo apt-get install wiringpi
```

Instalace OpenCV a RaspiCam již není tak snadná, jelikož je potřeba stáhnout jejich soubory z externích zdrojů. Návod na instalaci OpenCV je možné najít například zde [30]. Vzhledem k tomu, že knihovna potřebuje více externích programů a potřebuje také zkompilovat, je nejrychlejší využít předkompilovaného balíčku uvedeného v návodu. Je třeba si také uvědomit, že instalace probíhá na nejméně výkonnějším Raspberry Pi. I s využitím předkompilovaného balíčku instalace trvá více jak 5 hodin.

Pro instalaci knihovny RaspiCam je potřeba knihovnu stáhnout a nakopírovat do Raspberry Pi Zero. Následně pomocí příkazu níže bude obsah komprimovaného souboru extrahován do aktuálního adresáře.

```
$ unzip raspicam-0.1.8.zip
```

Dále stačí zadat sérii následujících příkazů pro vytvoření kompilačního souboru knihovny.

```
$  cd raspicam-0.1.8
$  mkdir build
$  cd build
$  cmake ..
```

Poté již stačí jen knihovnu zkompilovat, nainstalovat a aktualizovat systémové cesty, aby operační systém mohl knihovnu nalézt a pracovat s ní. Operace lze provést těmito příkazy:

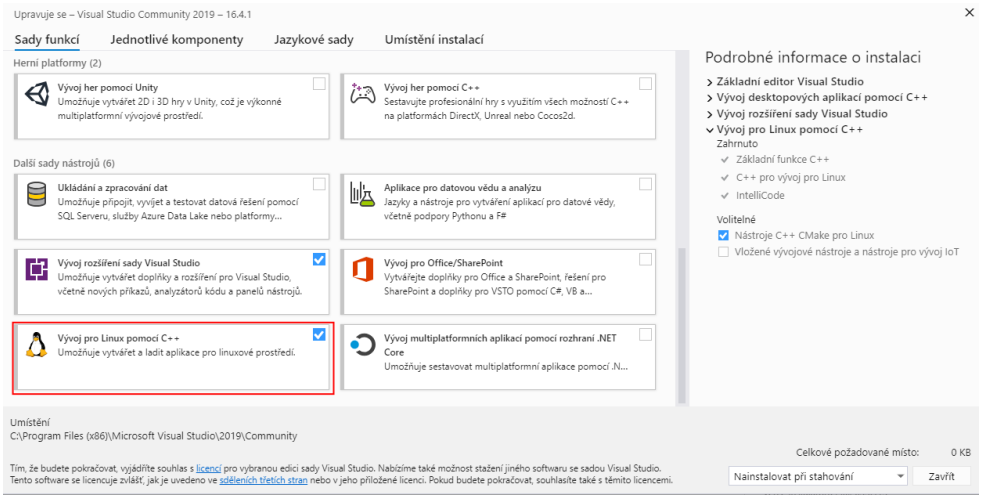
```
$  make
$  sudo make install
$  sudo ldconfig
```

7.3 Nastavení vývojového prostředí

Vyvíjení programů pro Raspberry Pi systémy může probíhat přímo na těchto systémech. Obsahují v sobě zabudované vývojové prostředí, nebo se dá vývojové prostředí nainstalovat. Ovšem vyvíjení na těchto systémech bez grafického výstupu může být horší a pro někoho nepřehledné. Proto se autor rozhodl jít alternativní cestou.

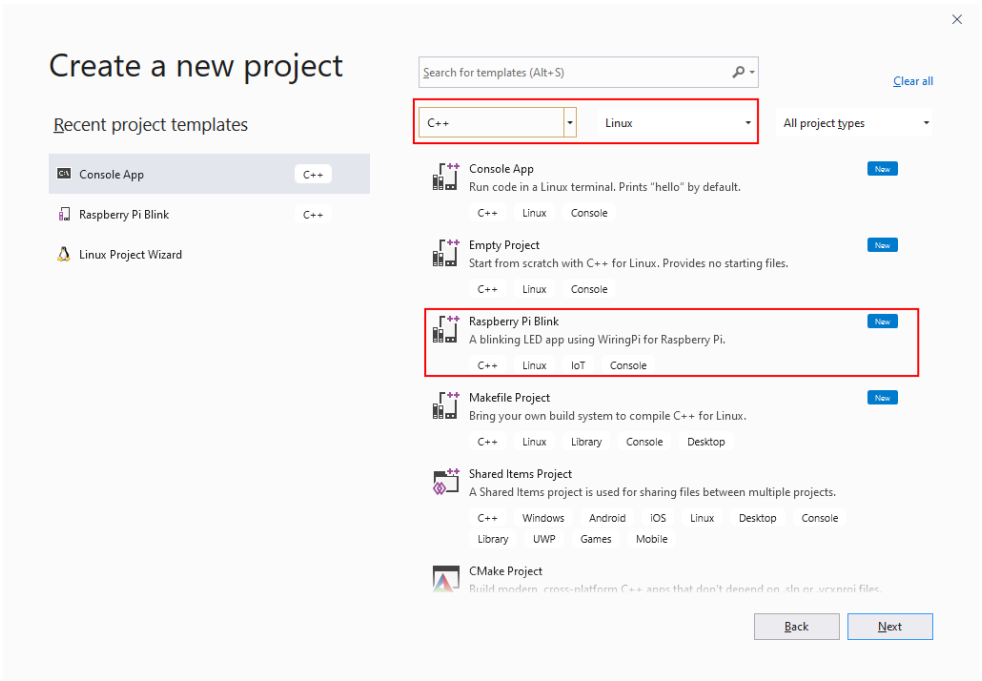
Veškeré vyvinuté programy v této práci byly vyvinuty pomocí vývojového prostředí Visual Studio 2019. Vývojové prostředí podporuje kompilování napříč více platforem. To znamená, že lze programovat ve Visual Studiu pro více cílových platforem (Linux, Android, atd.). Lze se totiž připojit do výchozího systému, např. pomocí SSH spojení, a vyvíjený program přeložil a odladit na cílovém systému. V podstatě je poté program vyvíjen na jednom systému, ale překládán a spouštěn na druhém. Vývojové prostředí dokonce umožňuje i odlaďování (debuging) na těchto cílových systémech, lze tedy program krokovat, podívat se na aktuální hodnoty proměnné atp.

Pro zpřístupnění tohoto režimu ve Visual Studio 2019 je třeba mít nainstalován balíček pro vývoj na platformě Linux (viz obrázek 7.6).



Obr. 7.6: Přídavný balíček pro Visual Studio 2019

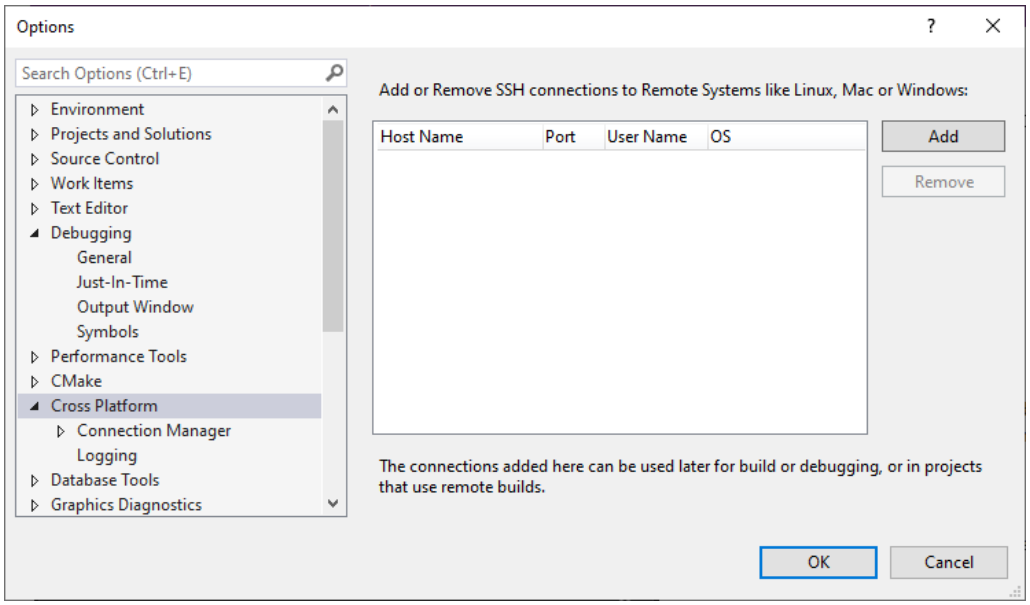
Poté se ve vytváření nového projektu objeví volba pro vytvoření programu na platformě Linux. Je zde již před vytvořen projekt pro Raspberry Pi (Raspberry Pi Blink). Pro snadnější implementaci stačí vytvořit projekt s využitím před vytvořeného programu (viz obrázek 7.7).



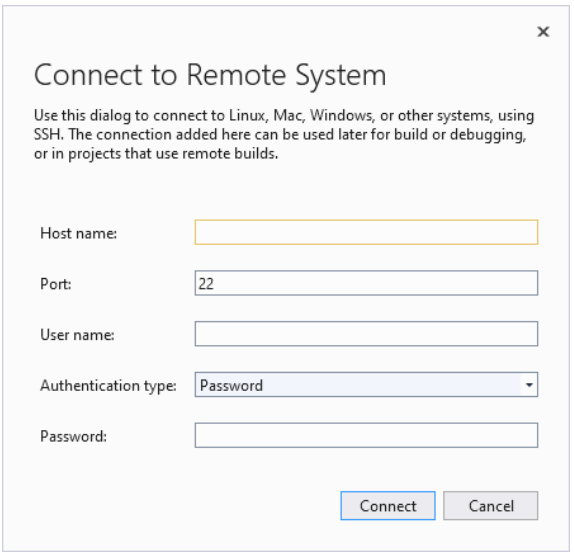
Obr. 7.7: Volba správného projektu ve vývojovém prostředí

Poslední fází je nastavení připojení pro externí systém. V záložce „Tools“ se nachází volba „Options“, pomocí které se otevře okno nastavení celého vývojového

prostředí. V liště vlevo se nachází nastavení „Cross Platform“, které umožňuje nastavení připojení k cílovému systému. Volbou „Add“ dojde k otevření okna, které vyzve uživatele k zadání cílové adresy, portu, přihlašovacího jména a hesla (viz obrázek 7.8 a 7.9). Po vyplnění údajů se Visual Studio 2019 k cílovému systému připojí, a tímto je umožněno vyvíjení na jiné platformě.



Obr. 7.8: Nastavení vývojového prostředí



Obr. 7.9: Připojení k cílovému systému

8 Řídící program autonomního autíčka

Zvolená výpočetní jednotka Raspberry Pi Zero umožňuje programovat v jakémkoli programovacím jazyce. Autor práce se rozhodl pro využití C/C++, tyto jazyky jsou pro vykonávání operací rychlejší, než například C#, nebo Python. Jazyky C a C++ totiž pracují na nižších úrovních a práci s pamětí programu musí nastavovat sám programátor. Pomocí těchto jazyků je tedy možné programy optimalizovat tak, aby jejich vykonávání bylo co nejrychlejší. Program byl vyvíjen ve vývojovém prostředí Visual Studio 2019 od společnosti Microsoft.

Následující kapitola popisuje algoritmy zpracovávající obrazový vstup z kamery. Rozhodování autonomního vozidla na základě získaných dat a také zajímavé části C/C++ kódu.

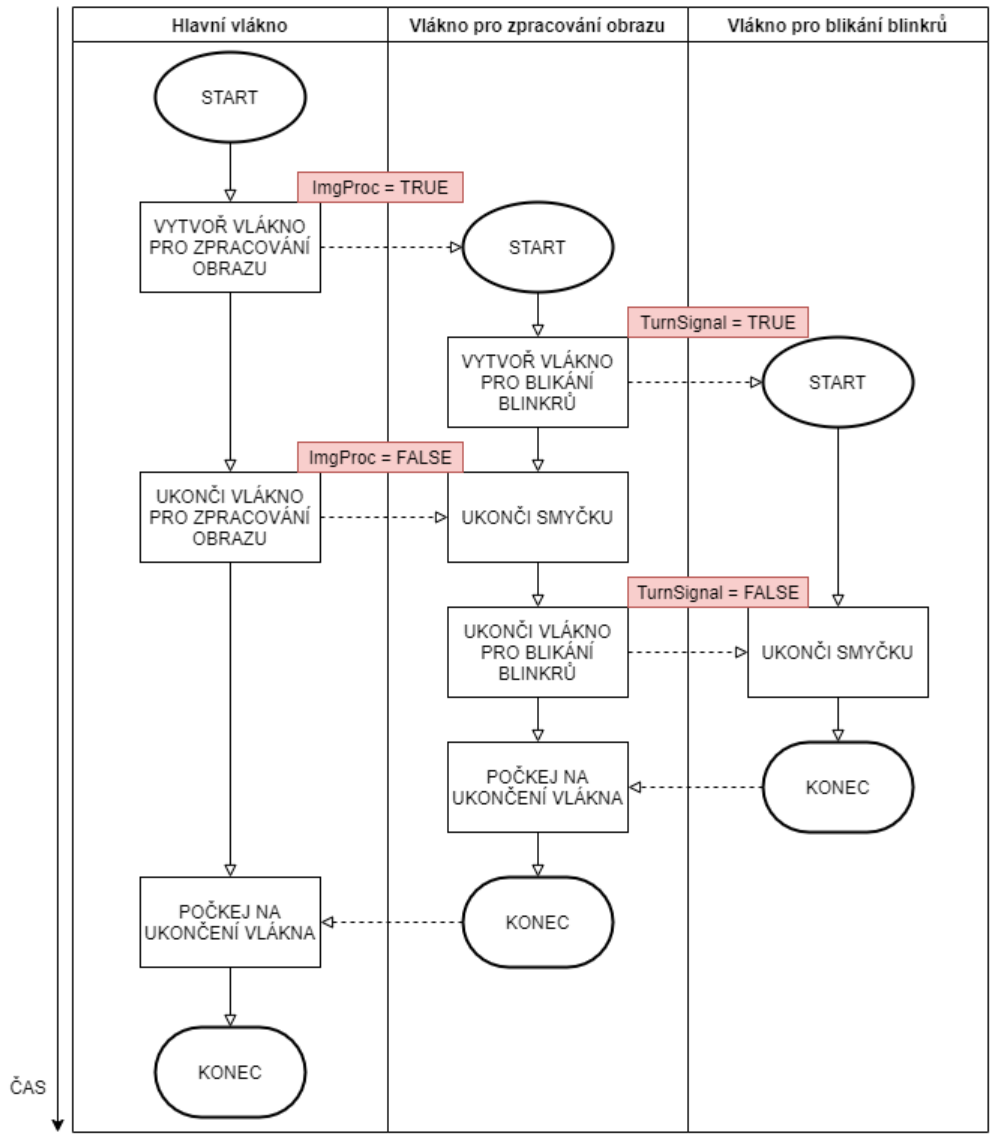
8.1 Hierarchie programu

Aplikace ovládající celé autonomní autíčko je rozdělena do tří vláken, která běží souběžně (paralelně). Každé vlákno má na starost jinou operaci. Hlavní vlákno, které je spuštěno okamžitě se spuštěním systému, slouží jako vstupní bod aplikace. Vytváří nové vlákno pro zpracování obrazu a zároveň hlídá výpadek napájení. V případě dostatečně dlouhého výpadku ukončuje vytvořená vlákna a vypíná celý systém.

Vlákno pro zpracování obrazu je zodpovědné za zisk obrazové informace z kamerového modulu. Obraz dále zpracovává a na základě získaných dat je vyhodnocena akce. Výsledkem tohoto vlákna je zrychlení, zpomalení, zastavení, či rozjetí autíčka.

Poslední vlákno je pomocné a slouží pouze pro blikání oranžových LED diod umístěných na vozidle jako blinkry. Samostatné vlákno pro ně bylo vytvořené proto, že rychlost blikání nesmí být ovlivněna zpracováním obrazu.

Ve všech vláknech jsou implementovány cykly pomocí *while()*. V těchto smyčkách je kontrolován povel na jejich ukončení a tedy i ukončení celého vlákna. Diagram na obrázku 8.1 popisuje v jakých časových okamžicích jsou jednotlivá vlákna vytvořena, nebo ukončena. Vlákna pro vytvoření a zánik využívají dvou proměnných (*ImgProc* a *TurnSignal*). Všechny proměnné jsou implementovány jako *atomic* a uchovávají pouze binární hodnotu. Výhoda tohoto datového typu spočívá v tom, že není potřeba využívat kontroly přístupu k proměnným mezi vlákny (například odmítnutí přístupu na základě zámku - *mutex*). Diagram nezohledňuje funkce, které jsou v rámci vláken vykonávána. Funkce vláken jsou popsány v následujících podkapitolách.



Obr. 8.1: Diagram popisující tvorbu a zánik vláken

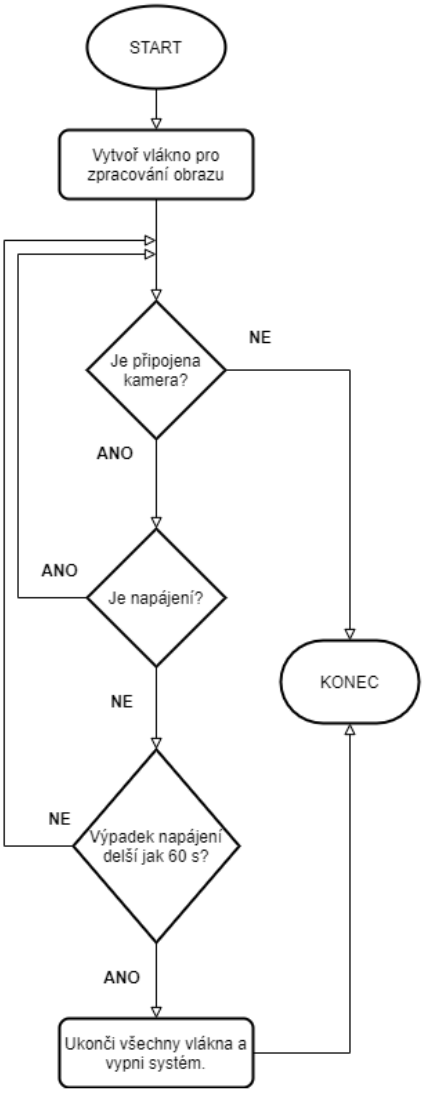
8.2 Vlákno pro kontrolu výpadku napájení

Jak již bylo zmíněno toto vlákno je vytvořeno na začátku aplikace. Hlavním úkolem vlákna je vytvoření vlákna pro zpracování obrazu a dále hlídání výpadku napájení. Napájení je hlídáno z toho důvodu, aby bylo možné včas zareagovat a bezpečně vypnout celý systém (nedojde tak ke ztrátě dat, či k jiným HW chybám). Bezpečné vypnutí je zajištěno pomocí super kondenzátorů (viz podkapitola 5.1).

Při ztrátě napájení (autíčko vypadlo z autodráhy) je zaznamenán čas a následně je porovnáván s aktuálním časem. Jakmile časový interval překročí 60 sekund, do-

jde k ukončení všech vytvořených vláken a následnému vypnutí systému. Kdykoliv při čítání časového intervalu může dojít k opětovnému připojení napájení (autíčko bylo vráceno na autodráhu). V takovém případě dojde k ukončení čítání intervalu a nedojde k vypnutí systému.

Vytvořené vlákno pro zpracování obrazu může být ukončeno dříve, a to tehdy, není-li k Raspberry Pi Zero připojena kamera. Pomocí nepřipojené kamery je indikován stav, kdy je potřeba například upravit stávající program, nebo změnit nastavení systému. V tomto případě totiž nedojde k vypnutí celého systému, ale jen k ukončení aplikace. Nepřipojení kamery kontroluje hlavní vlákno aplikace. Popsanou funkci vlákna zobrazuje následující vývojový diagram.



Obr. 8.2: Vývojový diagram hlavního vlákna programu

8.3 Vlákno pro zpracování obrazu

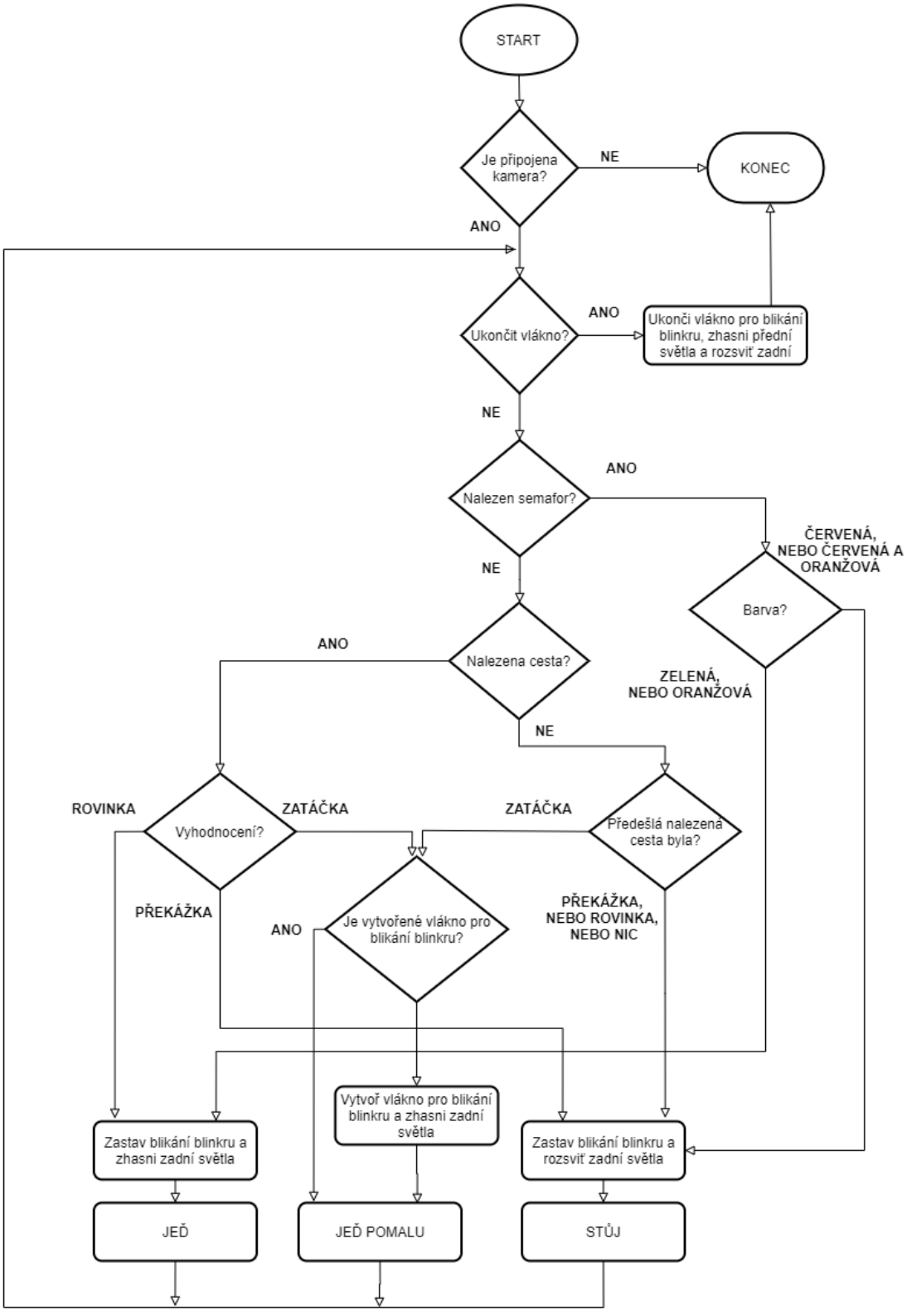
V tomto vlákně probíhá stěžejní část vyvinuté aplikace. Dochází zde k získání obrazové informace z kamerového modulu a k jejímu následnému zpracování. Na základě vyhodnocení obrazových dat vlákno rozhoduje jakou rychlostí má autíčko jet, či úplně zastavit. Algoritmy pro vyhledání semaforu a cesty v obraze jsou popsány v podkapitole 8.3.1 a 8.3.2.

V případě, že nebyla k systému připojena kamera, je vlákno okamžitě ukončeno (viz podkapitola 8.2). Na začátku vyhledávacího algoritmu je také kontrolováno, zda má být vlákno ukončeno (povel z hlavního vlákna), či nikoliv. Dále je spuštěno vyhledávání pro nalezení semaforu. Hledání semaforu je zařazeno jako první, jelikož má nejvyšší prioritu. A to z důvodu, že v případě zastavení autíčka na červenou na tvaru cesty nezáleží. Je také předpokládáno, že semafor je umístěn před křižovatkou a že v době průjezdu se v křižovatce nenachází žádná překážka. Je-li semafor nalezen, je vyhodnocena barva semaforu. Na zelenou a oranžovou může autíčko projet. Svítí-li na semaforu červená, nebo červená s oranžovou zároveň, autíčko zastaví.

Nevyskytuje-li se semafor v prohledávaném obrázku, je spuštěn algoritmus pro vyhledávání cesty. Výstupem algoritmu může být povel pro jízdu rovně, autíčko může jet vysokou rychlostí. Dalším výstupem může být zatáčka, autíčko musí zpomalit, aby ze zatáčky nevyjelo. Poslední z možností je, že se před autíčkem nachází překážka, autíčko zastaví. V případě, že se nepodařilo cestu najít, algoritmus vyhodnocuje jaký povel byl v předešlém kroku. Byl-li povel pro zatáčku, algoritmus bude dále pokračovat v zatáčce, v jiných případech zastaví.

Kdykoliv je v obraze nalezena zatáčka, je vytvořeno vlákno pro blikání blinkrů (nebylo-li vlákno již vytvořeno) a je mu předána informace, zda se jedná o levou, či pravou zatáčku. Blikání blinkrů je ukončeno vždy, když po zatáčce byla nalezena rovinka, nebo na základě vyhodnocení semaforu. Z vývojového diagramu (viz obrázek 8.3) si lze všimnout, že autíčko nebude reagovat na překážku stojící v zatáčce, nebo bezprostředně za ní. Tento zdánlivý nedostatek je záměrný, jelikož zorný úhel kamery není dostačující pro snímání zbytku zatáčky při jejím průjezdu. Potřeba opětovného nalezení rovinky tak zamezuje vyhodnocení chybných snímku při průjezdu zatáčkou (autíčko při průjezdu nezastaví).

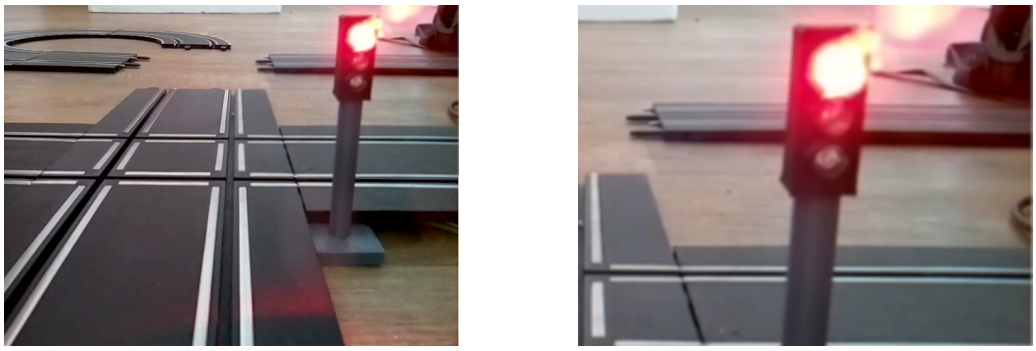
Vlákno také ovládá světla umístěná na autonomním autíčku. Při zastavení autíčka se rozsvítí zadní světla a při opětovném rozjetí zhasnou. Na začátku vlákna jsou také rozsvícena přední světla. Ovládání směrových LED diod je zřejmé z výše uvedeného textu.



Obr. 8.3: Vývojový diagram vlákna pro zpracování obrazu

8.3.1 Algoritmus nalezení semaforu

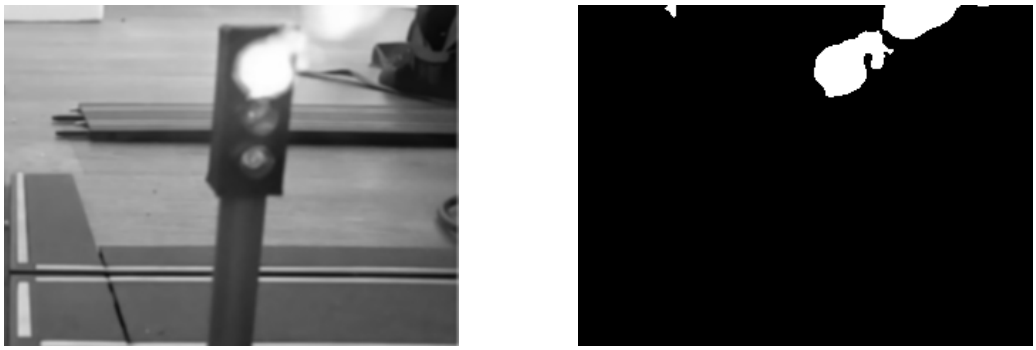
Semafor je hledán na každém pořízeném snímku. Na základě požadavku, aby algoritmus proběhl co nejrychleji, je předpokládáno, že se bude semafor nacházet ve druhém kvadrantu obrázku.



(a) (b)

Obr. 8.4: a Vstupní obraz pro nalezení semaforu; b Oříznutý obraz semaforu, ve kterém se předpokládá výskyt semaforu

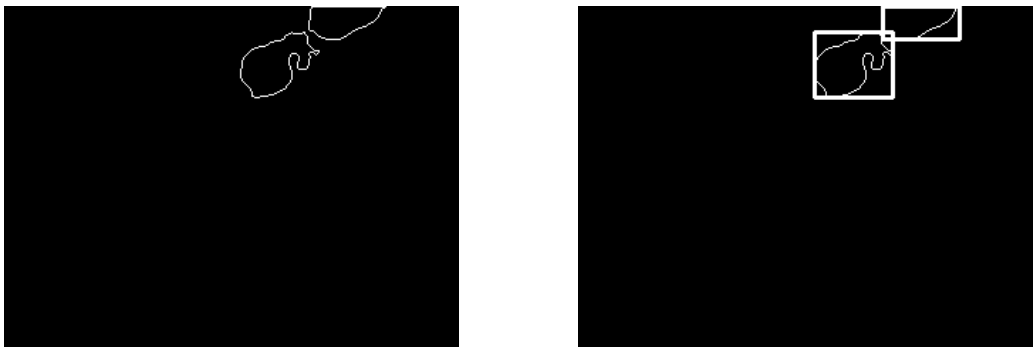
Obrázek po provedení operace „oříznutí“ je převeden do stupňů šedi a také do HSV barevného modelu. Jak si lze všimnout na obrázku 8.4b, jas semaforu vytváří tzv. „přepal“. Červená barva se tak nejeví jako červená, ale jako bílá. Bylo tak využito obrázku ve stupních šedi pro detekci bílé barvy. Bílá barva je získána pomocí prahování obrazu s vysokou hodnotou prahu. Tím je získán binární obraz reprezentující bílou barvu.



(a) (b)

Obr. 8.5: a Šedotónový obraz semaforu; b Naprahovaný obraz semaforu obsahující bílou barvu

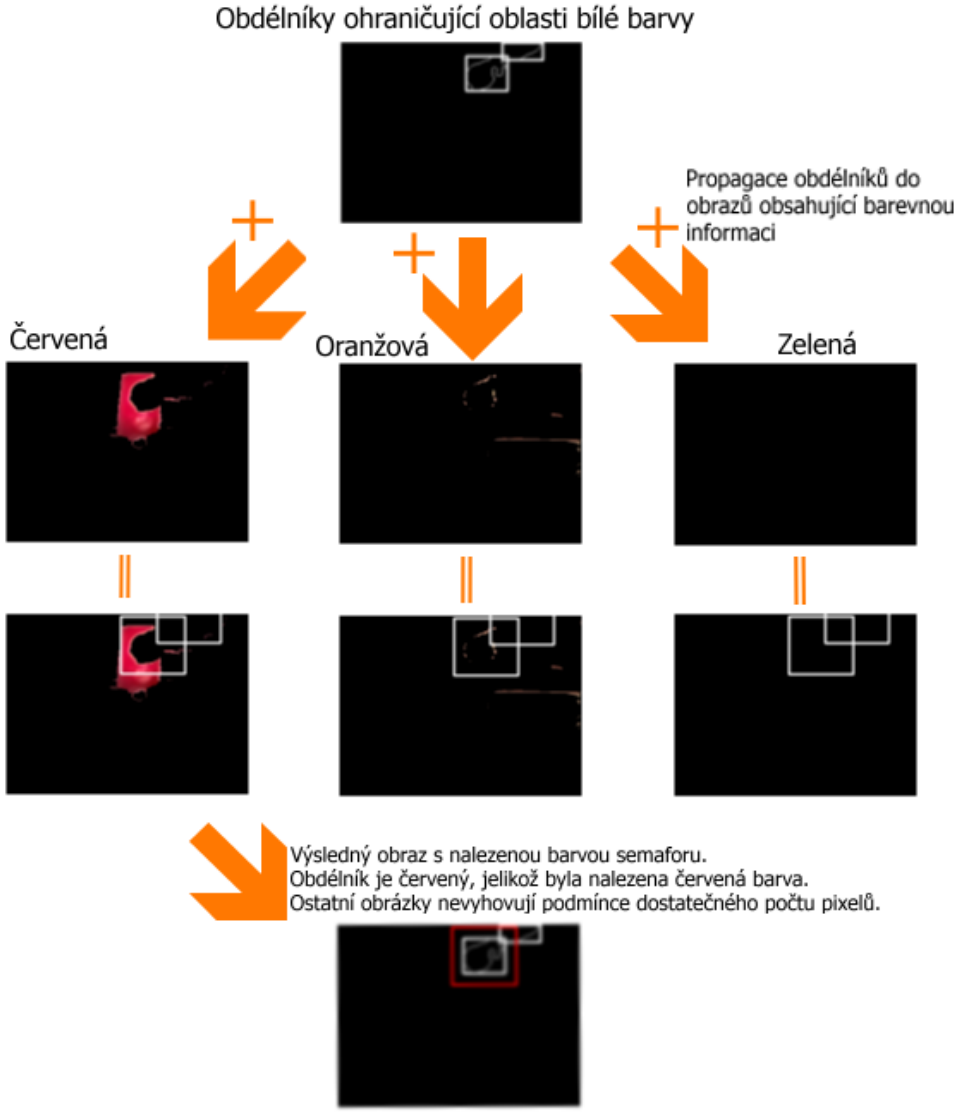
Po získání jen bílé barvy, je využito OpenCV funkce *findContours()*. Funkce pomocí algoritmu hledání hranic objektu nalezne ohraničení bílé barvy v obrazu a vrátí hraniční body (pro bližší popis funkce viz dokumentace OpenCV [24]). Pomocí nalezeného tvaru objektu, jsou následně filtrovány objekty s malou plochou. Odstraní se tak šum z obrazu. Následně získané bílé objekty jsou ohraničeny obdélníkem.



(a) (b)

Obr. 8.6: a Hranice oblastí bílé barvy; b Obdélníky ohraničující bílé oblasti

Obdélník je zvětšen o 10 pixelů do všech stran a vložen do obrazu obsahující barevnou informaci. Dochází tak k detekci barevné korony, která je tvořena kolem jasně bílých pixelů. Tato korona již obsahuje správnou barevnou informaci. Obrazy pouze s pixely představující červenou, oranžovou, nebo zelenou barvu jsou získány z HSV barevného modelu. Uvnitř obdélníku jsou spočteny pixely s těmito barvami. Podle toho, kterých pixelů se v obdélníku nachází nejvíce je prohlášeno jaká barva na semaforu svítí. Není-li však splněna podmínka dostatečného počtu pixelů s danou barvou, je výsledkem vyhodnocení, že se ve snímku semafor nenachází. Tento postup popisuje obrázek 8.7. Algoritmus je na Raspberry Pi Zero proveden přibližně za 40 milisekund.



Obr. 8.7: Propagace nalezených hraničních obdélníků do obrazů s barevnou informací a určení výsledné barvy semaforu

8.3.2 Algoritmus nalezení cesty

Algoritmus je založený na detekci počátku cesty (ve spodní části obrázku). Tento počátek je následně ohraničen omezujícím obdélníkem. Obdélník je dále propagován v záporném směru osy Y po obraze. Na základě výskytu kolejnic v obdélníku je upravována poloha v rámci osy X. Pro detekci rovinky nebo zatáčky je poté využita diference v poloze počátku jednotlivých obdélníků na ose X. V rámci posuvu obdélníku není měněna jeho výška, ani šířka. Pro rychlý průběh algoritmu byl zvolen přístup, kdy operace s obrazem (jako hranový detektor, převod do šedotónového obrazu, atd.) jsou vykonány jen na částech obrazu, které nás zajímají. Díky tomu nejsou náročné operace vykonávány na celém získaném obrázku. Výsledkem je vykonání méně matematických operací a celkové zrychlení algoritmu.

(a) (b)

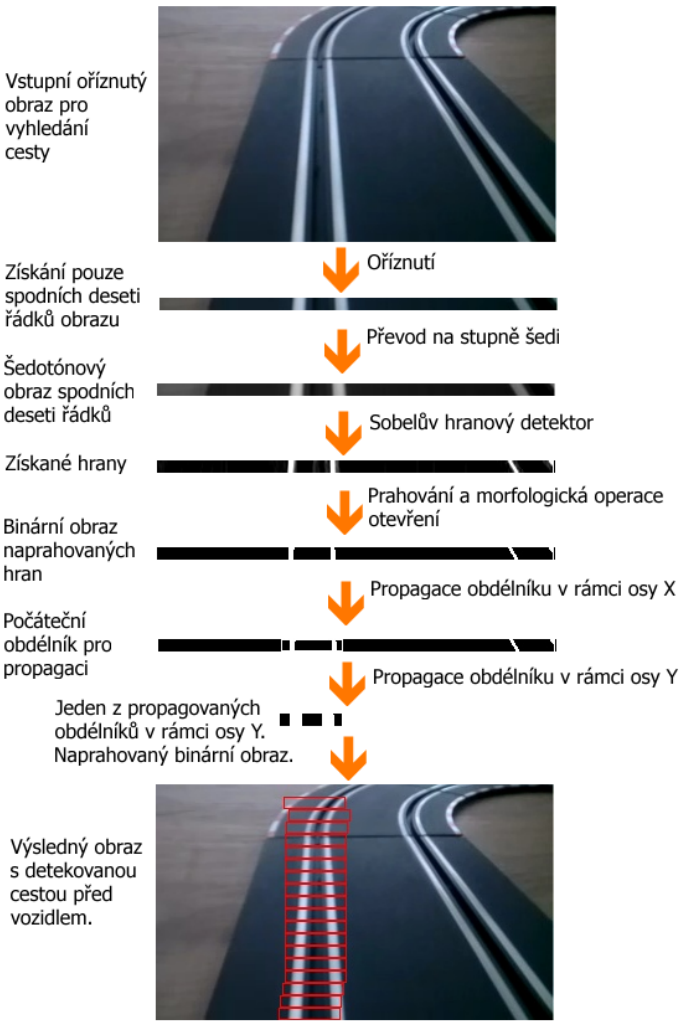
Obr. 8.8: a Vstupní obraz pro detekci cesty; b Oříznutý obraz na kterém bude vykonán algoritmus pro detekci cesty

Obrázek 8.8a představuje vstupní obrázek do algoritmu pro nalezení cesty, který je následně oříznut o části obrazu, ve kterých by se cesta neměla nacházet (viz obrázek 8.8b). Pro detekci lze využít skutečnosti, že autíčko je po autodráze naváděno pomocí kolejnic. Pro správnou detekci je třeba nalézt počátek kolejnic a poté správně propagovat obdélníky skrz obraz, jak je napsáno výše. Kolejnice jsou nalezeny pomocí Sobelova hranového detektoru. Jelikož mezi kolejnicí (šedá) a zbytkem autodráhy (černá) se nachází gradientní přechod. Hrany z hranového detektoru jsou poté naprahovány, aby bylo docíleno binárního obrazu a pomocí morfologické operace otevření (dilatace následována erozí) je odstraněn šum. Tyto operace jsou prováděny pouze na prvních deseti pixelech v ose Y od spodní části (prvních deset řádků obrazu).

Následný počátek kolejnic a obdélník, ohraničující obě kolejnice je získán propagací obdélníku po ose X. Opět je pro zrychlení využito faktu, že kolejnice se budou

nacházet blízko středu obrazu. Obdélník je tedy propagován zleva doprava kolem středu osy X s danou odchylkou. Prvním kritériem pro nalezení kolejnic je fakt, že se v propagovaném obdélníku nalézá více bílých pixelů než daný práh. Druhou podmínkou je, že kolejnice (hrany) vyhovují stanovené šířce a mezeře mezi sebou.

Po nalezení počátku kolejnic je stejný obdélník položen nad počáteční. Propagace obdélníku nastává v oříznutém obraze obsahujícím autodráhu před vozidlem. V novém obdélníku je obraz převeden na stupně šedi. Naprahován pomocí dynamicky získaného prahu. Práh odpovídá změně pixelů mezi kolejnicí a zbytkem autodráhy. Je spočítán počet bílých pixelů a pokud je vyšší než 50 pixelů je prohledávaná oblast prohlášena za správnou (kolejnice se v ní nacházejí). Následně je spočítána průměrná souřadnice X bílých pixelů a tato hodnota je nový střed obdélníku na ose X. Výše popsaný algoritmus znázorňuje obrázek 8.9. Algoritmus je na Raspberry Pi Zero proveden přibližně za 30 milisekund.



Obr. 8.9: Nalezení cesty v obraze

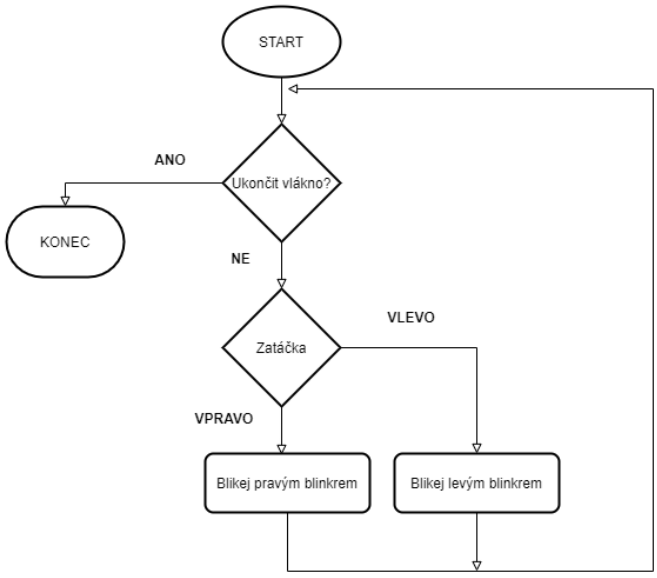
Pro nalezení cesty bylo zvažováno více algoritmů. Z toho pouze jeden odpovídal potřebné, malé, výpočetní náročnosti. Nejdříve byl vyzkoušen způsob, který k nalezení cesty využíval Houghovu transformaci, pomocí které lze nalézt čáry v obraze. Tyto čáry by, po správné filtraci, měly odpovídat kolejnicím, nacházejících se v obraze. Tento přístup byl ale později zamítnut, jelikož se jednalo o příliš výpočetně náročné operace a na dostupném HW by nalezení cesty trvalo dlouho.

Dalším zamítnutým způsobem byla detekce podobná výslednému algoritmu. Veškeré práce s obrazem byly prováděné na celém (nezmenšeném) obraze. Tato skutečnost však vedla ke zbytečnému zpomalení algoritmu.

Rovněž bylo vyzkoušeno více rozlišení pro optimalizaci algoritmu. Jako nejvhodnější rozlišení bylo zvoleno 320x240 pixelů. Zpracovávaný obraz je díky tomu menší a dojde k požadovanému zrychlení. Zároveň obraz není natolik malý, aby byla ztížená detekce požadovaných prvků.

8.4 Vlákno pro ovládání blinkrů

Jak již bylo zmíněno v podkapitole 8.1, toto vlákno slouží pro správnou funkci LED diod sloužící jako blinkry. Vlákno na základě zjištěné cesty před vozidlem vyhodnocuje zda zapnout pravé, nebo levé blinkry, či blikání ukončit. Funkce vlákna je zobrazena na vývojovém diagramu níže. LED diody blikají s periodou 0,5 sekundy.



Obr. 8.10: Vývojový diagram vlákna pro blikání blinkrů

8.5 Třída PWM

Pro ovládání rychlosti autíčka byla vytvořena třída PWM. Jak již název napovídá, třída má za úkol přepnutí příslušného pinu do módu PWM a nastavení potřebných hodnot signálu (frekvence a střída). Zdrojový kód třídy se nachází na přiloženém CD.

8.5.1 Nastavení PWM signálu

Pro práci s GPIO piny pomocí této knihovny je zapotřebí inicializovat GPIO piny pomocí funkce *wiringPiSetup()*. Vrátí-li funkce hodnotu -1 znamená to, že se nepodařilo správně inicializovat GPIO piny a nebude tak umožněno s nimi pracovat. V opačném případě, funkce nevrátila hodnotu -1, vše proběhlo v pořádku.

Následně je potřeba nastavit GPIO pin do módu PWM. Nastavení se provede funkcí *pinMode(int PIN, int MODE)*, této funkci jsou předávány dva parametry, PIN a MODE. Parametr PIN představuje číslo pinu, který chceme nastavit do módu PWM, je potřeba použít číslování pinů, které využívá knihovna „WiringPi.h“ (viz [26]). Raspberry Pi Zero může mít nastavený HW signál PWM na pinech GPIO 12 (WiringPi 1), GPIO 13 (WiringPi 2), GPIO 18 (WiringPi 5), GPIO 19 (WiringPi 12) [19]. Parametr MODE představuje mód, na který se má daný pin nastavit. V knihovně jsou tomuto přednastaveny makra, která nesou dané číselné hodnoty, stačí napsat „PWM_OUTPUT“ pro nastavení módu PWM.

Po nastavení pinu na požadovaný mód, stačí již nastavit pouze samotný PWM signál. První je potřeba nastavit mód PWM signálu a to buďto „mark:space“, který se chová jako standardní PWM, nebo „balanced“ mód, který je speciálně používán „mini“ počítači Raspberry Pi a je nastaven jako výchozí mód pro PWM signál. Pro nastavení PWM módu je třeba využít funkce *pwmSetMode(int MODE)*, funkce má jednu hodnotu a tou je hodnota MODE. Stejně jako při předešlém vybírání módu, jsou v knihovně přednastaveny hodnoty módu a tak stačí zadat „PWM_MODE_MS“ (1) pro „mark:space“ mód, nebo „PWM_MODE_BAL“ (0) pro „balanced“ mód.

Dále je potřeba nastavit děličku základní frekvence (19,2 MHz - základní frekvence by správně měla být uvedena v dokumentaci k modelu Raspberry Pi Zero na straně 107 v tabulce 6-35 [31], avšak v této tabulce se nachází chyba a frekvence zde nejsou popsány, chyby v dokumentaci lze najít na této webové stránce [32]). Nastavení provedeme funkcí *pwmSetClock(int DIVISOR)*, hodnota musí být v rozmezí 2 - 4095. Nastavenou hodnotou je podělena základní frekvence.

Funkce *pwmSetRange(int RANGE)* je poslední funkce potřebná pro nastavení PWM signálu, funkce nastavuje hodnotu RANGE, která představuje vzorkování

PWM signálu (nastavíme-li hodnotu RANGE na 10, bude mít výstupní PWM signál 10 vzorků). Hodnota RANGE má také vliv na nastavení celkové frekvence signálu a smí se pohybovat v rozmezí 2 - 4096. Výsledná frekvence signálu je vypočtená podle vzorce 8.1.

$$f = \frac{19,2 * 10^6}{DIVISOR * RANGE} \qquad [Hz] \tag{8.1}$$

Pro zapnutí PWM signálu a také nastavení střídy signálu je potřeba zavolat funkci *pwmWrite(int PIN, int VALUE)*, funkce má dva parametry, PIN a VALUE. Parametr PIN představuje pin, na kterém se má PWM signál zapnout, tento pin musí být shodný s pinem, který jsme nastavili do módu PWM, parametr VALUE představuje hodnotu střídy a musí být v rozmezí 0 - RANGE. Parametr VALUE představuje kolik vzorků PWM signálu bude nastaveno do log. 1 (3,3 V).

8.5.2 Konstruktory a destruktory třídy PWM

Třída má celkem tři konstruktory, základní konstruktor, který pouze vytvoří objekt a nastaví hodnoty signálu PWM na základní hodnoty. Poté konstruktor, který dovoluje nastavení frekvence a střídy a poslední konstruktor dovolující nastavení veškerých parametrů, včetně pinu a módu PWM.

Třída obsahuje také destruktor. Při destrukci objektu dojde k vypnutí signálu PWM a nastavení příslušného pinu jako digitální výstup s hodnotou log. 0.

8.5.3 Privátní metody třídy PWM

Třída obsahuje tři privátní metody, všechny metody jsou typu „void“ a nemají tak žádnou návratovou hodnotu. Metoda *InitializePWM()* volá výše uvedené funkce pro nastavení PWM signálu, funkcím jsou předávány parametry nastavení, které byly inicializovány pomocí konstruktorů.

Metoda *ComputeWrite(int aDuty)* má jeden parametr. Metoda přepočítává uživatelem požadovanou střídu signálu (v %), na hodnotu, kterou lze zapsat do funkce *pwmWrite*. Po přepočtu funkce volá metodu *SetWriteValue()*.

Metoda *SetWriteValue(int aValue)* nastavuje privátní člen metody *WriteValue*. Tento člen je využíván pro funkci zapnutí PWM signálu *pwmWrite*, popsanou výše.

8.5.4 Veřejné metody třídy PWM

Mimo konstruktory a destruktory obsahuje třída také tři veřejné metody. Metody jsou využívány pro zapnutí, vypnutí, či nastavení střídy PWM signálu - změnou střídy je měněna také rychlost autíčka.

Metoda *Start(int const aDuty)* pouze zapíná PWM signál na pin a požadovanou střídu, nastavené při vytváření objektu třídy - v případě volání metody bez parametru. Při volání metody s parametrem lze nastavovat střídu PWM signálu, nebude signál zapnut se střídou určenou konstruktorem třídy.

Další metoda *Stop()* nemá žádný parametr a slouží pro zastavení PWM signálu - nastaví střídu signálu na 0 %.

8.6 Třída CAR

Pro snadné ovládání celého autíčka byla vytvořena třída CAR. Pomocí této třídy lze ovládat rychlost, rozsvěcení a zhasínání světel. Ve třídě je také vytvořen objekt třídy PWM, není proto potřeba v programu pracovat s třídou PWM samostatně. Třída značně zlehčuje ovládání všech zásadních funkcí autíčka. Zdrojový kód třídy se nachází na přiloženém CD.

8.6.1 Konstruktor třídy CAR

Třída obsahuje jeden konstruktor. Jedná se o výchozí konstruktor třídy. Při vytvoření objektu dojde k vytvoření objektu třídy PWM a nastavení PWM signálu na potřebnou frekvenci. Zároveň dojde k nastavení GPIO pinů ovládajících LED diody na autíčku jako výstupní.

8.6.2 Privátní metody třídy CAR

V této třídě se nachází jedna privátní metoda, která je volána při vytvoření objektu. Metoda *SetModes()* nastavuje příslušné GPIO piny jako výstupní. Pomocí této inicializace pinů je následně umožněno jejich ovládání a rozsvěcování, či zhasínání připojených LED diod.

8.6.3 Veřejné metody třídy CAR

Veřejné metody zpřístupňují snadné ovládání rychlosti a světel autonomního autíčka. Pomocí metody *SetSpeed(unsigned int aSpeed)* lze nastavovat rychlost autíčka. Parametr aSpeed je předáván objektu třídy PWM jako střída PWM signálu. Metoda *Stop()* nastaví hodnotu střídy na nulu a zastaví autíčko.

Pomocí metod *FrontLightsOn()* a *FrontLightsOff()* lze rozsvěcovat, nebo zhasínat přední světla. Metody využívají funkce z WiringPi.h knihovny *digitalWrite(pin, value)*. Funkce umožňuje nastavit příslušný GPIO pin na hodnotu HIGH (log. 1), nebo LOW (log. 0). GPIO pin musí být před použitím funkce nastaven jako výstupní.

Stejně fungují také metody *RearLightOn()*, *RearLightsOff()*, *LeftSignalOn()*, *LeftSignalOn()*, *RightSignalOn()* a *RightSignalOff()*, které ale ovládají zadní světla a levé a pravé blinkry autíčka.

Posledními metodami třídy jsou metody *LeftSignalBlink()* a *RightSignalBlink()*. Jak již název metod napovídá jedná se o blikání směrových světel autíčka. Metody tak rozsvěcují a zhasínají příslušné LED diody s intervalem 0,5 sekundy.

Závěr

Cílem práce bylo sestavit autonomní autíčko pro model dopravní situace, který by odpovídal reálné situaci. Autonomní autíčko je sestavené pro autodráhu Carrera 143, ale při správném napájecím napětí U = +15 V je autíčko schopné fungovat i na jiné autodráze. Autíčko pro rozpoznávání okolí využívá kamerového modulu a „mini“ počítače Raspberry Pi Zero, který provádí veškeré výpočty. Autíčko bylo sestaveno jako jedno kompaktní řešení s veškerou elektronikou skrytou uvnitř.

Pro autonomní autíčko bylo nezbytné navrhnout vlastní desku plošných spojů, která řeší ovládání elektromotorku a LED diod autíčka (viz kapitola 5). Deska plošných spojů je navržena tak, aby dosahovala stejných rozměrů jako Raspberry Pi Zero a bylo možné tyto jednotky mezi sebou propojit pomocí GPIO pinů.

Jelikož došlo k přidání ovládací elektroniky, byl vytvořen vlastní 3D model (viz kapitola 6). Model je dostatečně velký, aby mohl skrýt veškerou navrženou elektroniku. V této kapitole se také nachází popis úpravy autodráhy, aby vyhovovala navrženým algoritmům a snadnému přenosu.

Kapitola 7 se věnuje popisu SW vybavení, se kterým se v rámci práce pracovalo. Vhodné nastavení operačního systému a instalace správných knihoven je nezbytné pro správnou funkčnost.

Navržené algoritmy popsané v kapitole 8 byly vyvinuty s ohledem na omezený výpočetní výkon „mini“ počítače Raspberry Pi Zero (viz podkapitola 4.2). Zpracování obrazu se děje jen na částech obrazu, kde je to nezbytně nutné, nebo kde se předpokládá výskyt hledaného objektu. Algoritmy dosahují rychlosti zpracování 40 milisekund pro nalezení semaforu a 30 milisekund pro nalezení cesty. Tyto rychlosti jsou dostačující pro samostatnou jízdu autonomního modelu vozidla. K vylepšení detekce může přispět například zpomalení jízdy. Zpomalení ale není možné s nynějším HW vybavením autíčka. Díky návrhu vlastní elektroniky a 3D modelu se autíčko stalo čtyřikrát těžší než původní (viz podkapitola 6.1). Větší váha má za příčinu, že využitý elektromotorek z původního modelu není dostatečně výkonný (nedostatečný kroutící moment), aby bylo umožněno rozjetí při nižších otáčkách.

Navržený model autíčka popsaný v této práci naráží na jistá HW omezení. Jedním z nejvýraznějších je limitovaný zorný úhel kamery (69,1°). Zorný úhel není dostačující, aby byla snímána cesta bezprostředně před vozidlem (viz podkapitola 6.1) a také nelze vidět průběh cesty, a tedy ani překážky, při průjezdu zatáčkou. Jak je popsáno v podkapitole 4.2.4, je možné vybrat i jiné kamerové moduly, ale celý model by se stal větším.

Výsledný model autonomního autíčka je schopen rozpoznat cestu před sebou a na základě jejího tvaru upravit svou jízdu. V případě zatáčky autíčko zpomalí a zapne příslušné blinkry. Stojí-li autíčku v cestě jiné autíčko, či jakákoliv jiná překážka, je

tato skutečnost vyhodnocena a jízda je přerušena. Zároveň se rozsvítí zadní brzdová světla. Autonomní autíčko je rovněž schopno správného vyhodnocení stavu semaforu. Na červené, nebo oranžové a červené autíčko zastaví, jinak projede. Autíčko je také vybaveno mechanismem, který rozpozná výpadek napájení a při výpadku delším jak 60 sekund dojde k jeho vypnutí. Zamezí se tak poškození dat a možnému zamezení opětovného spuštění. Ke spuštění autíčka a všech jeho funkcí dojde po přiložení napětí a nastartování operačního systému. U autíčka je také počítáno s možností úpravy operačního systému, nebo algoritmů. K jejich úpravě je nezbytné zapnout autíčko s odpojenou kamerou, v takovém případě nedojde k vypnutí celého systému za 60 sekund (v případě, že není položeno na autodráze).

Literatura

[1] *Tesla [online]. 2020 [cit. 2020-05-31]. Dostupné z:* https://www.tesla.com/autopilot

[2] *KOČÍ, Michal. Stabilizátory napětí[online]. 01.10.2018 [cit. 2020-05-31]. Dostupné z:* https://www.kondik.cz/stabilizator

[3] *Buck Converter: Basics, Working, Design and Operation [online]. 26.04.2019 [cit. 2020-05-31]. Dostupné z:* https://components101.com/articles/buck-converter-basics-working-design-and-operation

[4] *Reverse Voltage Protection [online]. [cit. 2020-05-31]. Dostupné z:* https://www.sunpower-uk.com/glossary/what-is-reverse-voltage-protection/

[5] *Supercapacitor vs Battery - Ultracapacitor Pros & Cons [online]. 09.10.2018 [cit. 2020-05-31]. Dostupné z:* https://www.arrow.com/en/research-and-events/articles/supercapacitor-vs-battery-ultracapacitor-pros-and-cons

[6] *VOJÁČEK, Antonín.Jak se nabíjejí Lithiové akumulátory ? [online]. 20.05.2008 [cit. 2020-05-31]. Dostupné z:* https://vyvoj.hw.cz/teorie-a-praxe/dokumentace/jak-se-nabijeji-lithiove-akumulatory.html

[7] *RICHTER, Miloslav. Zpracování vícerozměrných signálů. Dostupné také z:* http://midas.uamt.feec.vutbr.cz/ZVS/Exercise02/content_cz.php

[8] *Lekce 2 - Úvod do počítačové grafiky - Základy optiky, barevné modely: Barevné modely [online]. [cit. 2020-05-31].Dostupné z:* https://www.itnetwork.cz/grafika/uvod/uvod-do-pocitacove-grafiky-optika-modely

[9] *JANÁKOVÁ, Ilona. Předzpracování obrazu. Dostupné také z:* http://midas.uamt.feec.vutbr.cz/POV/Lectures/05_Predzpracovaniobrazu.pdf

[10] *Morphology [online]. [cit. 2020-05-31]. Dostupné z:* homepages.inf.ed.ac.uk/rbf/HIPR2/morops.htm

[11] *Machester code. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2019 [cit. 2019-12-30]. Dostupné z:* https://en.wikipedia.org/wiki/Manchester_code#/media/File:Manchester_encoding_both_conventions.svg

[12] *Manuál 20040028. Carrera-toys [online]. [cit. 2019-12-30]. Dostupné z:* https://www.carrera-toys.com/en/3186/manuals

[13] *NIEHUES, Ing. Peter. Carrera Protokolldecodierer mit Arduino [online]. 2013 [cit. 2019-12-30]. Dostupné z:* http://www.wasserstoffe.de/carrera-hacks/protocol-decode/index.html

[14] *Atmel 8-bit AVR Microcontroller with 2/4/8K Bytes In-System Programmable Flash [online]. s. 234 [cit. 2020-05-31]. Dostupné z:* https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2586-AVR-8-bit-Microcontroller-ATtiny25-ATtiny45-ATtiny85_Datasheet.pdf

[15] *Introduction to Arduino [online]. [cit. 2019-12-30]. Dostupné z:* https://www.arduino.cc/en/Guide/Introduction#

[16] *Arduino Official Store [online]. [cit. 2019-12-30]. Dostupné z:* https://store.arduino.cc

[17] *When to Use Arduino vs Raspberry Pi [online]. 22.07.2019 [cit. 2019-12-30]. Dostupné z:* https://roboticsbackend.com/when-to-use-arduino-vs-raspberry-pi/

[18] *Arduino-shop.cz [online]. [cit. 2019-12-30]. Dostupné z:* https://arduino-shop.cz

[19] *Raspberrypi.org [online]. [cit. 2019-12-30]. Dostupné z:* https://www.raspberrypi.org

[20] *RPishop.cz [online]. [cit. 2019-12-30]. Dostupné z:* https://rpishop.cz

[21] *Waveshare: share awesome hardware [online]. [cit. 2020-05-31]. Dostupné z:* https://www.waveshare.com

[22] *RPi Low-level peripherals: Power pins [online]. 28.06.2019 [cit. 2020-05-31]. Dostupné z:* https://elinux.org/RPi_Low-level_peripherals#Power_pins

[23] *ČERNÝ, Michal.Superkondenzátory místo akumulátorů? [online]. 06.05.2015 [cit. 2020-05-31]. Dostupné z:* http://robodoupe.cz/2015/superkondenzatory-misto-akumulatoru-4/

[24] *OpenCV [online]. 2020 [cit. 2020-05-31]. Dostupné z:* https://opencv.org/

[25] *RaspiCam: C++ API for using Raspberry camera with/without OpenCv[online]. [cit. 2020-05-31]. Dostupné z:* https://www.uco.es/investiga/grupos/ava/node/40

[26] *HENDERSON, Gordon. Wiring Pi [online]. [cit. 2019-12-30]. Dostupné z:* http://wiringpi.com/

[27] *Databáze operačních systémů Raspbian [online]. [cit. 2020-05-31]. Dostupné z:* http://downloads.raspberrypi.org/raspbian/images/raspbian-2019-09-30/

[28] *Raspberry Pi Documentation [online]. [cit. 2020-05-31]. Dostupné z:* https://www.raspberrypi.org/documentation

[29] *Raspberry Pi Zero USB/Ethernet Gadget Tutorial [online]. [cit. 2020-05-31]. Dostupné z:* https://www.circuitbasics.com/raspberry-pi-zero-ethernet-gadget/

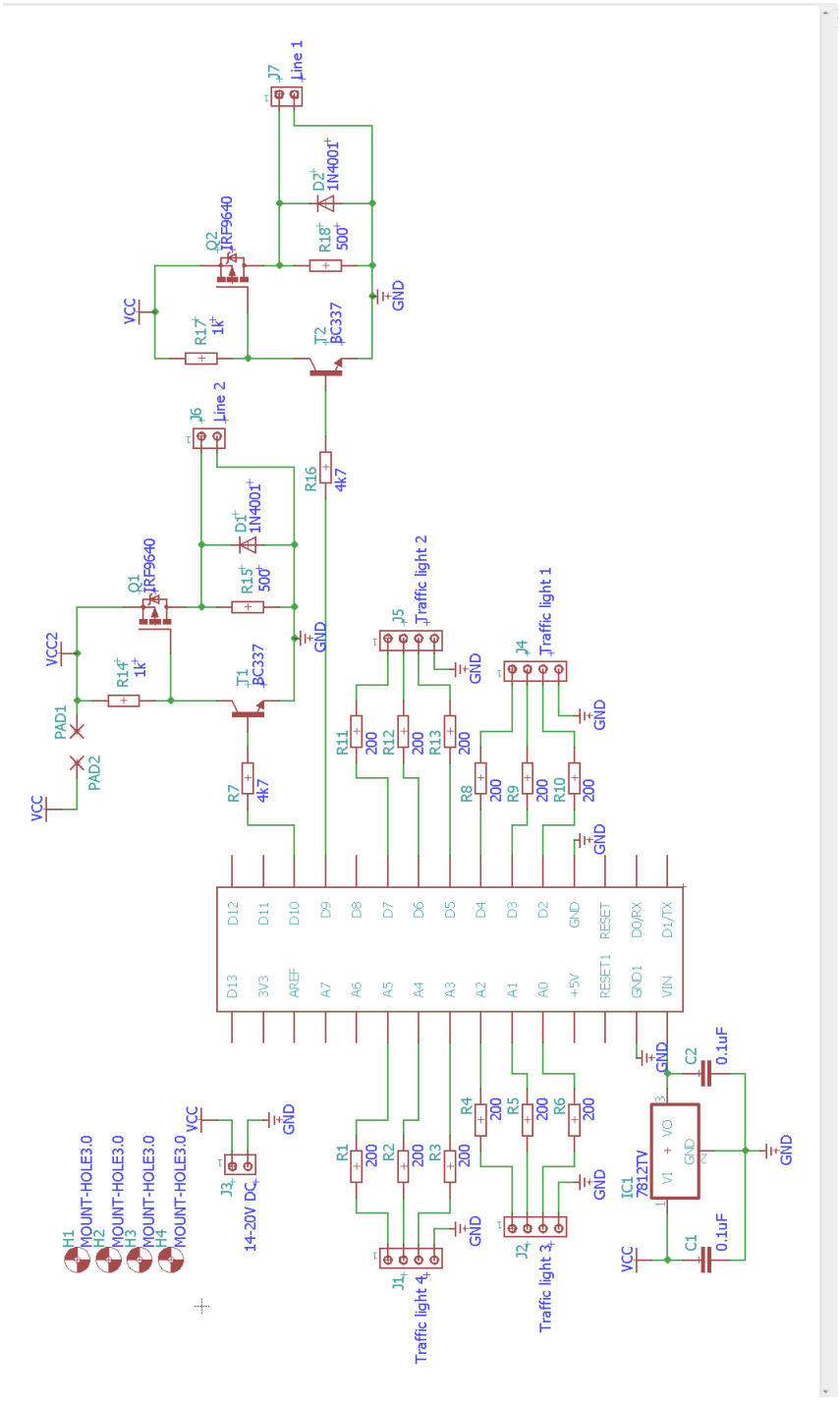
[30] *KLEČKA, Jan. BPRP - Robotika a počítačové vidění: Kamerový modul pro Raspberry Pi, knihovna OpenCV [online]. 2018 [cit. 2020-05-31]. Dostupné z: https://sites.google.com/a/vutbr.cz/bprp/cviceni/2018/09 Cvičení v rámci předmětu. Vysoké učení technické v Brně*

[31] *BCM2835 ARM Peripherals [online]. s. 205 [cit. 2019-12-30]. Dostupné z:* https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2835/BCM2835-ARM-Peripherals.pdf

[32] *BCM2835 datasheet errata [online]. [cit. 2019-12-30]. Dostupné z:* https://elinux.org/BCM2835_datasheet_errata

Seznam příloh

A Schéma zapojení autodráhy s mikrokontrolérem Arduino



Obr. A.1: Schéma zapojení autodráhy s mikrokontrolérem Arduino

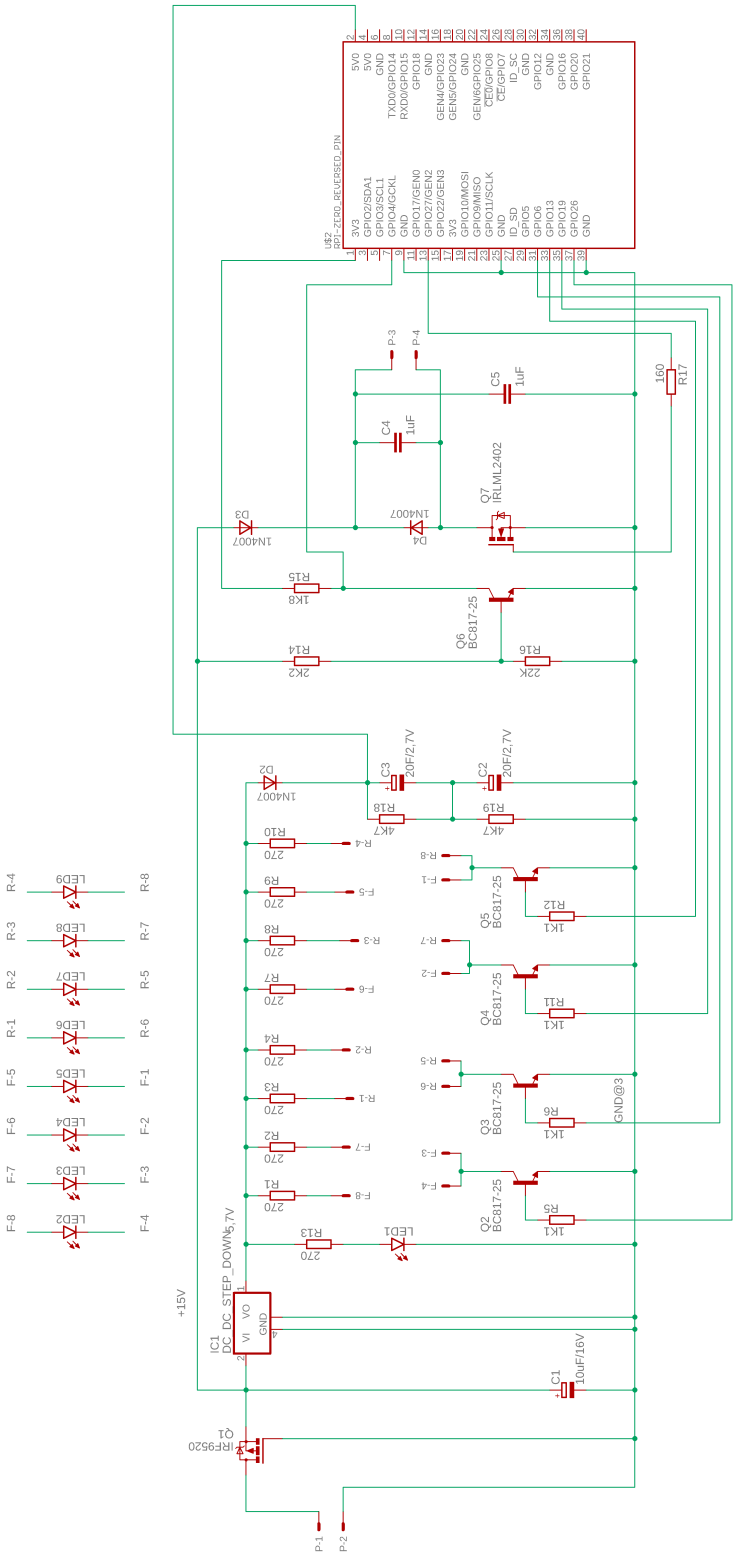
B Periferní obvody pro Raspberry Pi Zero

B.1 Seznam použitých součástek

Označení	Název	Hodnota	Typ
R1 - R4	Rezistor	270 Ω	-
R5, R6	Rezistor	1k1 Ω	-
R7 - R10	Rezistor	270 Ω	-
R11, R12	Rezistor	1k1 Ω	-
R13	Rezistor	270 Ω	-
R14	Rezistor	2k2 Ω	-
R15	Rezistor	1k8 Ω	-
R16	Rezistor	22k Ω	-
R17	Rezistor	160 Ω	-
R18, R19	Rezistor	4k7 Ω	-
C1	Kondenzátor	10 μF/16 V	Elektrolytický
C2, C3	Kondenzátor	20 F/2,7 V	Elektrolytický
C4, C5	Kondenzátor	1 μF/16 V	Keramický
Q1	P-MOS Tranzistor	-	IRF9520
Q2 - Q6	NPN Tranzistor	-	BC817-25
Q7	N-MOS Tranzistor	-	IRLML2402
D2 - D4	Dioda	-	1N4007
LED2, LED3	Bílá LED	-	R3528W-W5-1F
LED4, LED5	Oranžová LED	-	L-424EDT
LED6, LED7	Červená LED	-	L-3N4SRD
LED8, LED9	Oranžová LED	-	L-424EDT
IC1	DC/DC měnič	-	MH-MINI-360

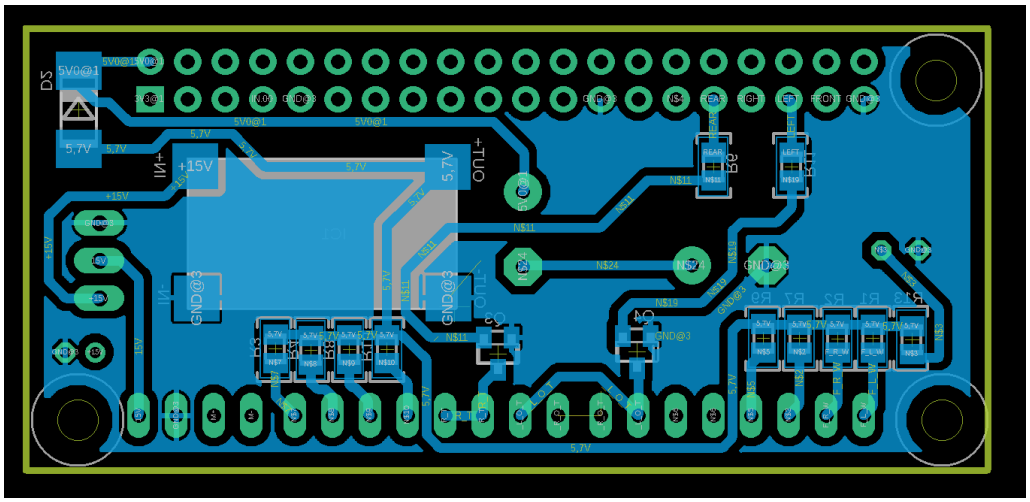
Tab. B.1: Seznam použitých součástek

B.2 Schéma elektrického obvodu



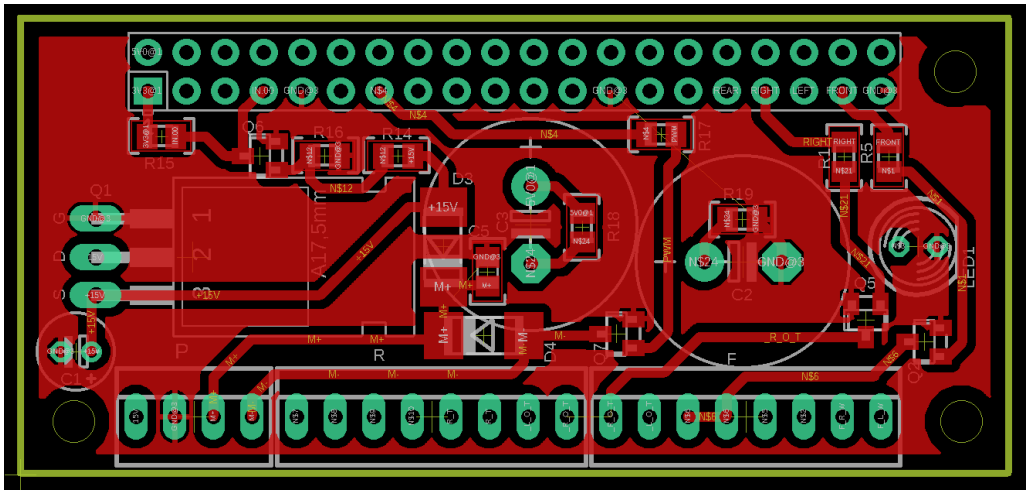
Obr. B.1: Elektrické schéma zapojení periferních obvodu pro Raspberry Pi Zero

B.3 Deska plošného spoje - spodní vrstva



Obr. B.2: Deska plošného spoje periferních obvodu spodní vrstva

B.4 Deska plošného spoje - horní vrstva



Obr. B.3: Deska plošného spoje periferních obvodu horní vrstva

C Obsah přiloženého CD

- / ..... kořenový adresář přiloženého CD
  - 3D_model ..... soubory 3D modelu
    - Soubory_pro_tisk ..... soubory pro tisk na 3D tiskárně
      - Auticko.stl
      - Auticko_dvere.stl
      - Auticko_strecha.stl
      - Drzak_kamera.stl
      - Krabicka_arduino_telo.stl
      - Krabicka_arduino_viko.stl
    - Auticko_3D_model.f3d ..... soubor pro editaci v programu Fusion 360
    - Krabicka_arduino.f3d ..... soubor pro editaci v programu Fusion 360
  - Navrzena_DPS
    - GerberFiles_soubory_pro_vyrobu ..... soubory pro výrobu DPS
      - copper_bottom.gbr
      - copper_top.gbr
      - profile.gbr
      - silkscreen_bottom.gbr
      - silkscreen_top.gbr
      - soldermask_bottom.gbr
      - soldermask_top.gbr
      - solderpaste_bottom.gbr
      - solderpaste_top.gbr
      - gerber_job.gbrjob
    - DPS_periferni_desky.brd
    - Schema_periferni_desky.sch
  - Zdrojove_soubory ..... zdrojové soubory C/C++
    - functions.cpp
    - main.cpp
    - Car.hpp
    - functions.hpp
    - PWM.hpp
  - DP_Dominik_Schneiderka.pdf ..... elektronická verze diplomové práce