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

Bakalářská práce

Robot s kamerkou

Plzeň 2023 Zdeněk Vavřička

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Zdeněk VAVŘIČKA**
Osobní číslo:	**A19B0220P**
Studijní program:	**B0613A140015 Informatika a výpočetní technika**
Specializace:	**Výpočetní technika**
Téma práce:	**Robot s kamerkou**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Prostudujte problematiku zpracování obrazu s důrazem na rozpoznávání čáry a jednoduchých základních objektů.
2. Prostudujte možnosti platformy ESP32-cam a vytvořte na ni API pro tyto funkce.
3. Pomocí tohoto API na robotu implementujte základní funkce sledování čáry či vykrývání zadaného prostoru.
4. Ověřte funkci a definujte případná omezení.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce

Vedoucí bakalářské práce: **Ing. Tomáš Mainzer, Ph.D.**
Katedra informatiky a výpočetní techniky

Datum zadání bakalářské práce: **3. října 2022**
Termín odevzdání bakalářské práce: **4. května 2023**

L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 25. října 2022

Prohlášení

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

V Plzni dne 22. června 2023

Zdeněk Vavřička

Poděkování

Rád bych poděkoval Ing. Tomáši Mainzerovi, Ph.D. za cenné rady, věcné připomínky a vstřícnost při konzultacích a vypracování bakalářské práce.

Abstract

This bachelor thesis is going to address the construction of a robot with camera, that is able of movement on a predefined line.

The first part of this thesis concerns itself with finding suitable components and methods of image processing.

The following practical part presents the robot assembly itself. Here we test chosen components and methods of image processing presented in the first part. The results will be evaluated and described from the point of view of advantages and possible disadvantages of the chosen solution.

Abstrakt

Bakalářská práce se bude zabývat návrhem a konstrukcí robota s kamerou, který se bude pohybovat po předem definované dráze.

V první části bakalářské práce se zabýváme výběrem vhodných komponentů a metod pro zpracování obrazu.

V navazující praktické části práce bude provedeno samotné sestavení robota. Otestujeme zde vybrané komponenty a metody pro zpracování obrazu představené v první části. Výsledky práce vyhodnotíme a popíšeme zde pozitiva, případná negativa a omezení navrhnutého řešení.

Obsah

1 Úvod

Slovo robot se poprvé objevilo v divadelní hře R.U.R. (Rossum's Universal Robots) od Karla Čapka ve 20. letech 20. století, kde byli tímto slovem označováni umělí dělníci. Rozvoj robotiky zaznamenáváme v druhé polovině 20. století, zejména ve Spojených státech amerických. Za přispění inženýrů George Devola a Josepha Engelbergera, kteří sestavili prvního robota UNIMATE pro závody General Motors. Od té doby se roboti stali součástí velkého množství činností.

Využívají se prakticky ve všech oborech, například v průmyslu, zdravotnictví, dopravě, domácnosti, zemědělství, službách a výzkumu. Šetří nám především lidskou sílu tam, kde by byl člověk vystaven těžké monotónní práci v nebezpečném nebo škodlivém prostředí. Dalšími jejich přednostmi jsou spolehlivost, přesnost a poměrně kvalitně vykonaná práce, která usnadňuje nebo nahrazuje lidskou činnost. V dnešní době i přes veškeré výhody musí roboti stále spolupracovat s člověkem, především co se týče definování úkonů, které má robot vykonávat, či v oblasti bezpečnosti, kdy není vždy vhodné spoléhat na autonomní řízení.

Tato práce se zabývá návrhem a realizací robota s kamerou, který bude schopen se pohybovat po určené dráze. Jeho řízení bude naprogramováno tak, aby se mohl pohybovat zcela autonomně.

Pro realizaci robota s kamerou v bakalářské práci bude použita deska ESP32-cam, která byla předem stanovena v zadání práce. Bude třeba vybrat kameru z několika možných modelů pro realizaci práce. Pro realizaci modelu robota bude zapotřebí znalostí z oborů informatiky, elektrotechniky, robotiky a matematiky.

Po sestavení modelu robota může být robot využit jako výukový model nebo může sloužit dalším studentům jako základ jejich vědeckých prací.

První část bakalářské práce se bude zabývat metodami zpracování obrazu a popisem hardwarových částí robota, které budou použity pro sestavení robota. V druhé praktické části této práce bude realizováno sestavení samotného robota, implementace vybraných metody zpracování obrazu a autonomní řízení robota. Následně bude otestován vybraný hardware a vyhodnoceni problémy s realizací robota.

2 Existující řešení

Existuje mnoho řešení pro realizaci sledování čáry. Většina těchto řešení by se dala rozdělit do dvou hlavních skupin. První skupinou jsou řešení, která pro tyto úkoly využívají kameru a druhou skupinou jsou řešení, která kameru nevyužívají nebo využívají pro jiné účely, např. sledování okolního prostoru nebo detekci objektů. Sledování čáry je tedy řešeno za pomoci jiného hardwaru. Nejčastěji infračervenými senzory.

2.1 Sledování čáry s kamerou

Nyní si představíme několik řešení sledování čáry pomocí kamery. První řešení [24], které si nyní představíme, využívá pro sledování čáry kameru. Autor tohoto řešení vybral desku a kameru Raspberry Pi. Programovací jazyk využívaný autorem je Python. Pro konstrukci robota byla deska Raspberry Pi nejspíše vybrána, protože pro metody zpracování obrazu autor využívá knihovny OpenCV.

Robot se dokáže pohybovat zcela autonomně po definované čáře a je schopen reagovat na řídící značky, jež určují, kterou cestou se má robot vydat.

Autor při realizaci projektu narazil na několik problémů při implementaci sledování čáry. Těmito problémy byly především nespojitost detekované čáry či detekování malých objektů v obraze, které byly chybně vyhodnoceny jako další linie. Výhodou tohoto řešení je možnost implementování detekce řídících značek pro lepší možnost navádění robota.

Dalším řešením [34], které si zde představíme, je robot s využitím smartphonu s operačním systémem Android pro rozpoznávání čáry a tvarů. Tato realizace využívá pro sledování čáry kameru telefonu a pro zpracování obrazu využívá také knihovny OpenCV.

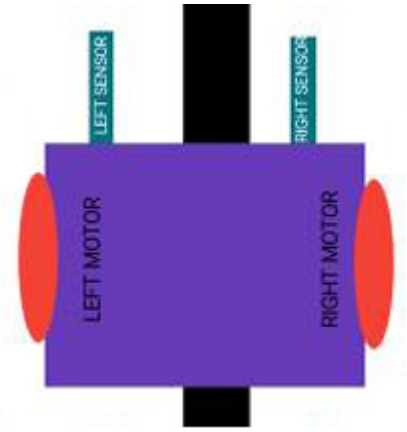
Robot je schopen se autonomně pohybovat po definované dráze a rozeznat určité tvary, například trojúhelník. Autor tohoto řešení vytvořil také demoaplikaci pro možnost nastavení různých metod pro zpracování obrazu a detekci čáry, aby bylo možné porovnat jejich efektivitu.

Jak sám autor práce popisuje, je nevýhodou tohoto řešení problém s nedostatečným rozpoznáváním barev za zhoršených světelných podmínek. Výhodou tohoto přístupu je jednoduchost realizace díky využití smartphonu, jenž disponuje poměrně velkým výpočetním výkonem a implementací značného množství senzorů.

2.2 Sledování čáry bez kamery

Představíme si realizaci robota s infračervenými senzory [9]. Infračervený senzor pracuje tak, že má dvě diody, jednu pro příjem a druhou infračervenou pro vyslání světelného signálu. Pokud tedy senzor přijme odražený signál od překážky nebo se změní jeho intezita, senzor detekuje překážku.

Tento robot pro sledování čáry využívá dva infračervené senzory umístěné tak, že sledovaná čára se nachází mezi těmito senzory, viz obr. 2.1. Pokud tedy robot detekuje jedním z infračervených senzorů sledovanou čáru například levým senzorem, pak je nutné, aby robot provedl korekci dráhy. Je tedy nutné, aby se robot otočil do leva a opět ani jeden ze senzorů nezaznamenal sledovanou čáru. Pokud však robot zaznamená čáru oběma senzory, můžeme tento stav chápat jako stop stav, tedy jako hranici, za kterou daný robot nemůže. Takto je možné realizovat pohyb robota ve vyznačeném prostoru.



Obrázek 2.1: Znázornění robota s IR senzory [9]

Toto řešení je velmi jednoduché a funkční, jestliže se bude robot pohybovat po uzavřené linii, která se nebude větvit. Kdyby se robot pohyboval po linii, která se větví, nastalo by nepředvídatelné chování, například by se robot zastavil na jednom místě a vyhodnotil by situaci jako případ zastavení.

Dále si představíme řešení [26], které jako předchozí řešení využívá k detekci čáry infračervené senzory. Toto řešení navíc disponuje webkamerou pro detekci překážek. Tento robot využívá desky Arduiono Uno spolu se zařízením NI Myrio pro připojení k počítači pomocí wifi. Softwarem pro naprogramování robota a zpracování obrazu je LabView a jeho jazyk G. Řešení disponuje čtyřmi infračervenými senzory na přední straně robota. Kamera je umístěna na vrchní straně robota a směřuje ve směru jízdy, aby mohla sledovat případné překážky v cestě robota. Informace z infračervenných senzorů

jsou zpracovány přímo robotem. Obraz je odesílán na počítač, kde probíhá jeho zpracování. Po zpracování obrazu se vyhodnotí zdali se nalézá překážka v cestě robota nebo nikoliv a odešlou se informace zpět robotu.

Řešení je velice spolehlivé s malou šancí minout definovanou čáru. Tato realizace oproti předchozí pracuje i s detekcí překážek v cestě robota. Na druhou stranu určitou slabinou tohoto řešení je jeho závislost na wifi připojení a zpracování informací na vzdáleném počítači. Pokud by například zkolabovala síť, robot není schopen detekovat překážky.

2.3 Shrnutí

Výše zmíněné přístupy mají své výhody a nevýhody, které si nyní shrneme. Nejdříve bude shrunuto sledování čáry a následně konstrukce robota.

Přístupy, které využívají kameru zcela pro sledování čáry nebo pouze pro detekci objektů využívají především knihovny OpenCV, která obsahuje velké množství metod pro zpracování obrazu. Tato knihovna je velmi náročná na paměť a některé metody v ní uvedeny jsou náročné ohledně výpočetního výkonu. Proto pro realizaci výše zmíněných řešení byli autory zvoleny buď zařízení s dostatečnou pamětí a výpočetním výkonem, nebo docházelo ke zpracování obrazu na jiném zařízení, kam byla data z kamery přeposílána například pomocí Wifi. Tento postup nemůže být v této práci zvolen jelikož je určené zařízení, které pro realizaci musí být využito. Zároveň by se mohlo zpomalit zpracování obrazu v závislosti problémů spojených s přenosem a případnou ztrátou dat.

Dále jsou uvedena řešení, která využívají infračervené senzory pro sledování čáry, tento přístup řešení je velice spolehlivý a výpočetně nenáročné. Zároveň by se správným umístěním více senzorů dalo dobře reagovat na křižovatky na trase a reagovat na ostré zatáčky až do 90°. Bohužel tohoto přístupu, nemůže být využito v této práci jelikož je zadáním omezeno, že se detekce čáry má provádět zapomoci kamery a metod zpracování obrazu.

Nyní porovnáme realizace konstrukcí robota. Většina výše zmíněných realizací má stejnou nebo obdobnou konstrukci robota založenou na dvou pohonných jednotkách tedy pohyblivých kolech, které jdou ovládat nezávisle a třetím kole, které je všesměrové. Toto řešení umožňuje lepší ovladatelnost robota v zatáčkách. Využití této realizace podvozku a pohonu robota by mohlo být užitečné při realizaci této práce.

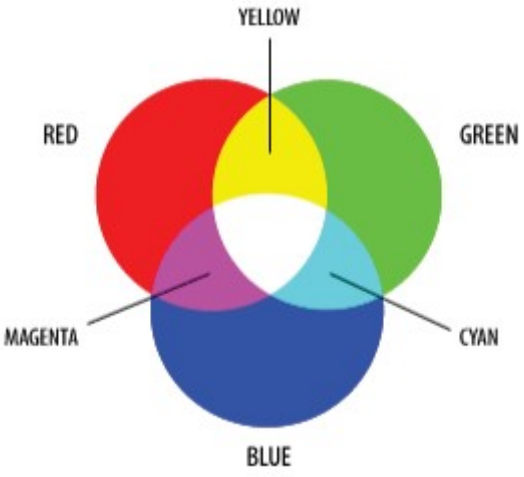
Dále u všech realizací disponující kamerou byla kamera vždy umístěna na vrchu robota. Buď kamera směřuje přímo před robota nebo je umístěna přímo nad sledovanou čárou ať již po určitým úhlem nebo kolmo na čáru.

3 Barevné modely

V této části bakalářské práce se seznámíme se základními barevnými modely. Tyto modely se využívají pro reprezentaci rastrových obrázků na digitálních zařízeních. Každý model je charakterizovaný určitými parametry například sytostí, jasem, světlostí a svítivostí. Dále se využívají základní barvy a jejich skládání, aby se dosáhlo požadovaného efektu. Barevné modely můžeme podle míchání barev rozdělit do dvou skupin. První skupina se nazývá aditivní. V této skupině se pracuje při míchání se světelnými zdroji barev tzn. jestliže jsou jednotlivé složky barevného modelu větší tím bude výsledná barva světlejší. Druhá skupina se označuje jako subtraktivní, u které při míchání barevný model pracuje s odrazem světla. Volba barevného modelu může také zásadně ovlivnit výsledek segmentace obrazu [7].

3.1 RGB

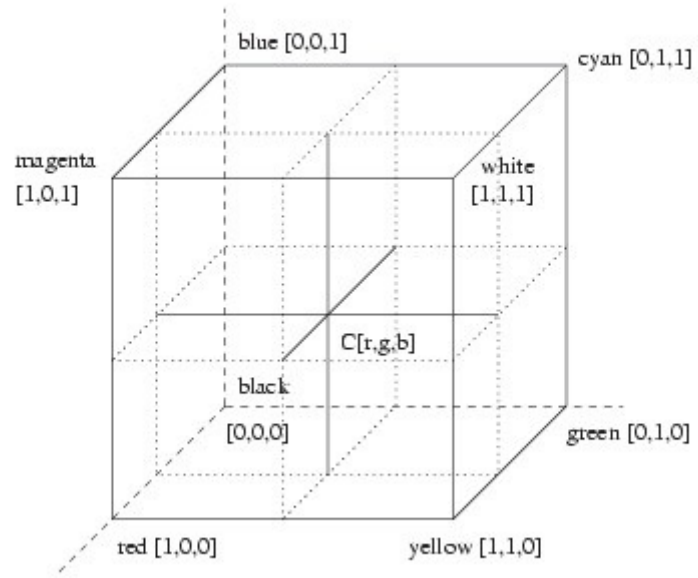
Barevný model RGB patří do první skupiny, tedy do aditivního míchání barev. Využívá se například při reprezentaci barev v monitorech nebo televizích.



Obrázek 3.1: RGB skládání barev [7]

Model RGB je představován váženým součtem tří základních barev. Červené (Red), zelené (Green) a modré (Blue). Pro základní vyjádření podílu

dané barvy se nejčastěji používá interval $\langle 0, 1\rangle$. Tento model si můžeme pro lepší pochopení představit jako jednotkovou krychli, viz obr. 3.2



Obrázek 3.2: RGB model krychle [7]

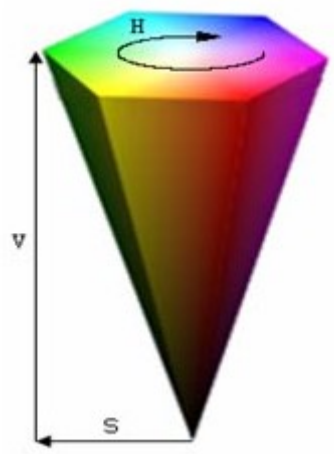
V počátku soustavy souřadnic $(0, 0, 0)$ nalezneme černou barvu a v protilehlém vrcholu krychle $(1, 1, 1)$ bílou barvu. Na jednotlivých osách soustavy souřadnic jsou jednotlivé barvy, červená $(1, 0, 0)$, zelená $(0, 1, 0)$ a modrá $(0, 0, 1)$.

V počítačové grafice je interval $\langle 0, 1\rangle$ převeden na celočíselný interval $\langle 0, 255\rangle$, poté je každý jednotlivý barevný bod reprezentován 24 bity. Jelikož je barevný model reprezentován 3 složkami po 8 bitech, můžeme takto zobrazit až $256^3 = 16777216$ barevných kombinací neboli odstínů. Takto vyjádřené barvy nazýváme pravé barvy neboli true colors. Samozřejmě je možné z důvodu paměťových nároků reprezentovat jednotlivé složky na méně bitech například čtyřech nebo v kombinaci 5:6:5, ale již není možné vyjádřit tolik odstínů jako v případě true colors. Vyjádření RGB v kombinaci 5:6:5, tedy 5 bitů pro červenou a modrou barvu a 6 bitů pro zelenou barvu se často využívá z důvodu, že lidské oko je více citlivé na změny zelené barvy, nežli na změny červené a modré barvy. Je tedy možné prezentovat pozorovateli obrázek s podobnou kvalitou, jako kdybychom využili 8 bitů pro každou barevnou složku, ale obrázek nebude tak paměťově naročný [7].

3.2 HSV

Tento model vznikl z důvodu neschopnosti lidí si představit, jaká barva vznikne smícháním jiných barev. Model HSV řeší tuto problematiku tím, že zavádí jinou techniku pro míchání barev. V tomto modelu se barvy míchají intuitivně.

Výsledná barva v modelu HSV je složena ze 3 hodnot. Těmito hodnotami jsou barevný tón (Hue), sytost (Saturation) a jas (Value). Tento model je geometricky reprezentován jako šestiboký jehlan, viz obr. 3.3.



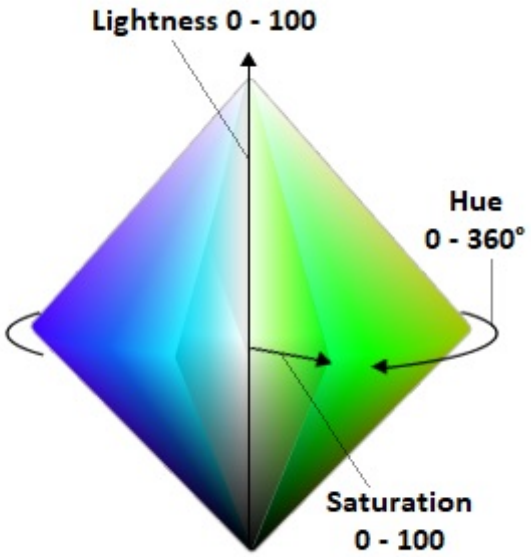
Obrázek 3.3: Model HSV [13]

Barevný tón je zde reprezentován jako úhel H, který může nabývat hodnot $\langle 0°, 360° \rangle$. Jas v geometrické reprezentaci je udáván jako výška jehlanu a určuje množství bezbarvého světla ve výsledné barvě. Jasová složka může nabývat hodnot z rozsahu $\langle 0, 1 \rangle$. Sytost je určena relativní vzdáleností bodu od osy jehlanu a vyjadřuje, jak budou zastoupeny různé barevné příměsi ve výsledné barvě. Sytost je možné brát jako čistotu barvy. Sytost nabývá hodnot z rozsahu $\langle 0, 1 \rangle$.

Nevýhodou tohoto barevného modelu je jeho jehlanovitý tvar. V tomto modelu při změně barevného tónu, který označujeme jako H, se musí bod S, jímž je označena sytost, pohybovat po dráze, která je definovaná obvodem šestiúhelníku. Přirozené by bylo, kdyby se bod S pohyboval po kružnici, ale v tomto modelu se pohybuje po obvodu šestiúhelníku. Další zápornou vlastností tohoto modelu je také jeho nesymetrie z pohledu jasu [7].

3.3 HSL

Barevný model HSL doplňuje a vylepšuje nedostatky modelu HSV 3.2. Barva je zde zastoupena barevným tónem (Hue), sytostí (Saturation) a světlostí/jasem (Lightness). Geometrický model můžeme znázornit jako dva podstavami spojené kužely, viz obr. 3.4 [7].



Obrázek 3.4: Model HSL [19]

Barevný tón jako u předchozího barevného modelu může nabývat hodnot $\langle 0°, 360° \rangle$. Světlost v modelu HSL udává výšku spojených kuželů a velikost achromatické složky, jde o zastoupení bílé a černé barvy, případně jejich kombinaci v odstínech šedi. Spojení kuželů odpovídá 0,5 světlosti. Saturace je relativní vzdálenost bodu od osy kuželů, podobně tomu bylo i u modelu HSV s tím rozdílem, že pokud je bod na plášti jehlanu, má hodnotu 1.

3.4 YCrCb

Model YCrCb se řadí do druhé skupiny míchání barev, zatímco v první skupině dochází k aditivnímu míchání barev, u této druhé skupiny dochází k subtraktivnímu míchání barev. Barva je zde popsána tříprvkovým vektorem $\begin{bmatrix} Y, & C_r, & C_b \end{bmatrix}^T$. Y zde představuje jas a Cr, Cb jsou barevné složky. Složky Cr a Cb jsou chroma parametry, které představují sytost červené a modré barvy. Model YCrCb umožnuje jednoduchou formou oddělit černobílý obraz.

Značné využití má model YCrCb v přístrojích jako, jsou videokamery a fotoaparáty založené na digitální technologii [7].

3.5 Shrnutí

V této kapitole je uveden pouze výčet z několika existujích barevných modelů. Zde uvedené modely patří k nejznámějších a některé nám mohou pomoci při realizaci práce. Důležité je také uvést, že barevné modely jsou mezi sebou navzájem převoditelné, a tudíž není nutné pracovat pouze s daným model, pokud zařízení podporuje jiné barevné modely. Převod jako takový může být náročný, a proto může zpomalovat zpracování informací zařízením. Převody mezi barevnými modely jsou blíže popsány v tomto článku [25].

4 Metody zpracování obrazu

Metody zpracování obrazu neboli segmentace obrazu je souhrnný proces, jenž je často složen z několika metod. Vstupní obraz se musí ve většině případů upravit. Úprava se týká většinou všudypřítomného šumu a velkého počtu dalších drobných detailů, které jsou chápány též jako informační šum. Po úpravě vstupního obrazu se musí provést samotná segmentace, úprava totiž není vždy vyhovující. Jako nežádoucí se mohou objevit malé segmenty objektu, které je třeba spojit, nebo naopak rozdělit do takového tvaru, dokud nebudeme mít dostačující výsledky.

4.1 Prahování

Prahování jako takové má velice jednoduchou implementaci a není moc časově náročné. Tato metoda pracuje především s úrovní jasu, ale může pracovat i s jinými paremetry obrazové reprezentace bodu v obraze. Výstupem prahování je binární obraz o stejné velikosti jako vstupní obraz, nad kterým bylo provedeno prahování. Binární obraz se skládá ze dvou barev a to bílé a černé. V RGB modelu se jedná o hodnoty $[0, 0, 0]$ a $[255, 255, 255]$. Pro jednoduchost tyto hodnoty budou substituovány jako hodnoty 0 pro bílou barvu a 1 pro černou barvu. Je také nutné definovat obraz jako takový. Obraz může být definován jako dvoudimenzionální funkce. Tato funkce bude označena například $f(x, y)$. Parametry funkce x a y udávají souřadnice jednotlivých pixelů v obrazu. Horní levý roh obrazu je počátkem soustavy souřadnic. Jedná se o pixel se souřadnicemi $[0, 0]$. Vysvětlování principu prahování omezíme pouze na prahování podle úrovně jasu. Prahování probíhá tak, že nejprve bude definována úroveň prahu. Označme ji například T. Následně jsou procházeny jednotlivé pixely a porovnávány hodnoty jasu obrazu s hodnotou prahu T, který byl stanoven. Pokud je úroveň jasu vyšší nežli stanovený práh, tak do výsledného obrazu, který označme $g(x, y)$, bude uložena na pozici $[x, y]$ hodnota 1, pokud je úroveň jasu nižší nežli práh T, bude uložena do výsledného obrazu hodnota 0. Takto pokračujeme, dokud nebude zpracovaný celý obraz [7, 28].

$$g(x, y) = \begin{cases} 1 & \text{pro } f(x, y) > T \\ 0 & \text{jinak} \end{cases} \tag{4.1}$$

Hodnota prahu může být zvolena náhodně, ale je lepší vycházet z obrazu, který bude prahován, aby se docílilo lepších výsledků.

(a) Výchozí obraz (b) Výsledný obraz

Obrázek 4.1: Prahování

4.1.1 Dvojité prahování

U dvojitého prahování se definují dva prahy na rozdíl od základního prahování, kde stačí práh jeden. Výsledná oblast, která má hodnotu 1, se nachází mezi hodnotami prahů T_1 a T_2. Algoritmus prahování je stejný jako u prahování s jedním prahem až na to, že hodnota jasu musí být v intervalu $\langle T_1, T_2 \rangle$, viz rovnice 4.2 [7].

$$g(x,y) = \begin{cases} 1 & \text{pro } f(x,y) \in \langle T_1, T_2 \rangle \\ 0 & \text{jinak} \end{cases} \tag{4.2}$$

4.1.2 Poloprahování

Tato modifikace prahování vychází ze základního prahování, ale lze vycházet i z dvojitého prahování. Hlavním rozdílem je výsledný obraz, který není binární. Část vstupního obrazu, která má vyšší hodnotu jasu nežli je stanovený práh, je tak uložena i do výsledného obrazu. Ve výstupním obrazu se tedy budou nácházet části vstupního obrazu obklopené černou barvou [7].

$$g(x,y) = \begin{cases} f(x,y) & \text{pro } f(x,y) > T \\ 0 & \text{jinak} \end{cases} \tag{4.3}$$

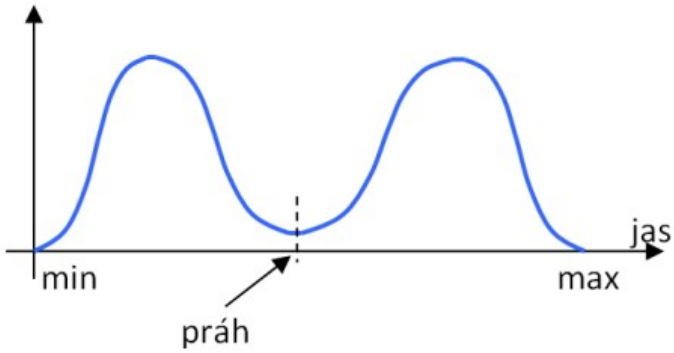
4.1.3 Adaptivní prahování

Dalším typem prahování je adaptivní prahování. Tento typ prahování se liší od základního tím, že se obraz rozdělí do menších oblastí, pro které se zvolí různé úrovně prahu. Je tedy obraz prahován po oblastech. Množina prahů se značí D [7].

$$g(x,y) = \begin{cases} 1 & \text{pro } f(x,y) > D \\ 0 & \text{jinak} \end{cases} \tag{4.4}$$

4.2 Automatické určování prahu

U prahování by bylo nejlepší určovat hodnotu vhodného prahu automaticky, aby se mohlo pracovat s více rozdílnými obrázky. K nalezení vhodného prahu se nejčastěji využívá analýzy histogramu. Podle podoby histogramu se liší metody pro určení prahu. Musí se určit zda histogram obsahuje jeden nebo více dominantních vrcholů. Jednou z možností pro analýzu histogramu je vytvoření bimodálního histogramu z obrazu. Podoba bimodálního histrogramu, viz obr. 4.2.



Obrázek 4.2: Vizualizace bimodalního histogramu [15]

U bimodálního histogramu lze určit hodnotu prahu jako lokální minimum mezi dvěma dostatečně vzdálenými lokálními maximy [15, 28].

4.3 Konvoluce obrazu

Konvoluce je způsob, jak aplikovat filtry či jiné operátory na vstupní obraz. K provedení konvoluce je nutné mít definované konvoluční jádro. Konvoluční jádro si můžeme představit jako čtvercovou matici n. Za n se nejčastěji volí

liché číslo, abychom byli schopni střed matice namapovat na jednotlivé pixely. Toto jádro posouváme po jednotlivých pixelech obrazu tak, že střed konvolučního jádra se překrývá s pixelem, pro který počítáme výslednou hodnotu. Konvoluci lze zjednodušeně chápat také jako vážený průměr určitého okolí daného pixelu. Velikost okolí, s kterým ve výpočtu pracujeme, je shodná s velikostí konvolučního jádra. Konvoluce se provádí nejprve ve směru osy x a následně ve směru osy y.

Uveďme si nyní příklad konvoluce. Pokud při konvoluci využijeme jako konvoluční jádro matici A, viz 4.5, docílíme ve výsledném obrazu zvýraznění hran objektu ze vstupního obrazu, viz obr. 4.3.

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{4.5}$$

Časová náročnost výpočtu roste přímo úměrně s velikostí konvolučního jádra. Konvolucí můžeme dosáhnout třeba zvýraznění hran, zostření obrazu nebo rozmazání obrazu, například Gausovým filtrem [28].

(a) Výchozí obraz (b) Výsledný obraz

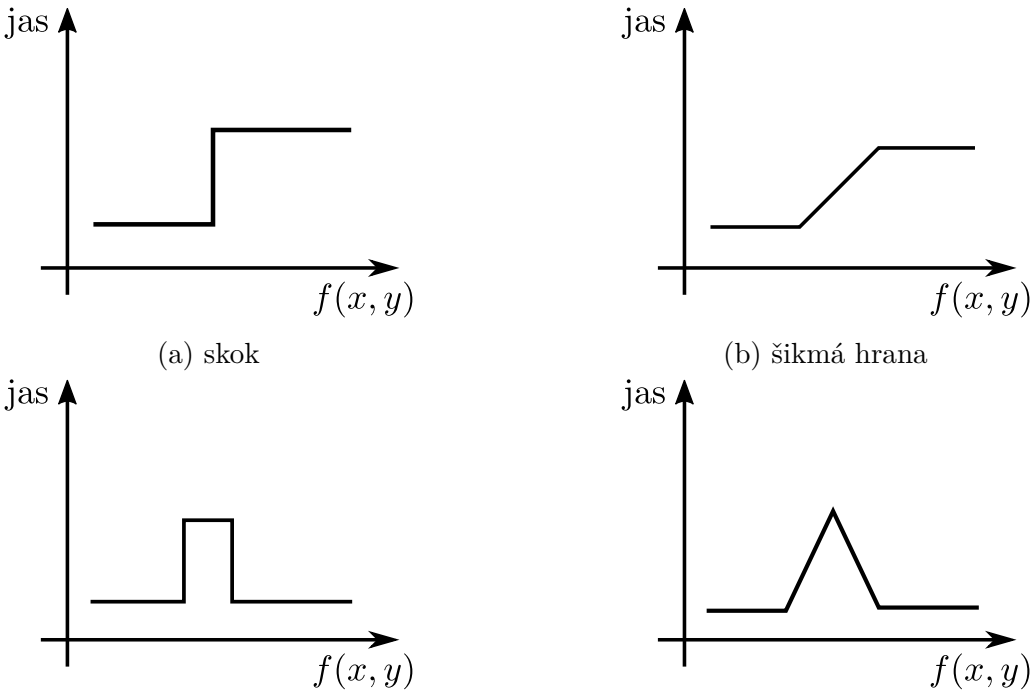
Obrázek 4.3: Prahování

4.4 Segmentace založená na detekci hran

Tato metoda je jedna z důležitých částí zpracování obrazu. Místa v obrazu, kde dochází k prudké změně sledované veličiny, si definujeme jako hrany obrazu. Nejčastěji je touto sledovanou veličinou jas. Volba jasu není bezdůvodná, protože lidské oko je citlivé na změny jasu. Pokud budeme jas

v obraze reprezentovat pomocí funkce, tak bychom měli pozorovat skokové změny v průběhu funkce, které představují hrany mezi objekty.

Obraz obsahuje různé typy hran. Ideální hranou je skok (step), ale reálně dochází k pomalé změně jasu, tím pádem se hrana projeví jako šikmá hrana (ramp). Pokud dojde k výše uvedeným změnám v obraze velmi blízko sebe, vznikají ještě další dva typy hran a to linie (line) a střecha (roof). Tyto průběhy jsou zobrazeny na obrázku 4.4. Všechny tyto průběhy mohou a také jsou často ovlivněny šumem. Proto je často nutné nejdříve vstupní obraz filtrovat a tím odstranit šum [7].



(a) skok (b) šikmá hrana

(c) linie (d) střecha

Obrázek 4.4: Průběhy jasu v obraze

4.4.1 Detekce hrany první derivací

U metod založených na první derivaci se využívá detekce rozdílů sousedních pixelů. Jedním možným postupem je vypočítat první derivaci pro sloupce a řádky obrazu zvlášť. Gradient se vypočítává z nejbližších sousedních pixelů ve vertikálním a horizontálním směru. Gradient vypočteme pomocí vzorce 4.6.

$$G(x,y) = \sqrt{G_R(x,y)^2 + G_S(x,y)^2} \tag{4.6}$$

Výsledný gradientní obraz je $G(x, y)$, gradient pro detekci hran v horizontálním směru je $G_R(x, y)$ a $G_S(x, y)$ je gradient ve vertikálním směru.

Pro určení gradientu obrazu můžeme využít konvoluce za pomoci hranových detektorů/operátorů. Hranové detektory jsou v konvoluci využity jako konvoluční jádro 4.3 [7]. Zde je uvedeno několik hranových operátorů [28]:

Sobelův operátor

$$G_R(x,y) = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad G_S(x,y) = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{4.7}$$

Prewittův operátor

$$G_R(x,y) = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad G_S(x,y) = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \tag{4.8}$$

Robinsonův operátor

$$G_R(x,y) = \begin{bmatrix} -1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & 1 & 1 \end{bmatrix} \qquad G_S(x,y) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{bmatrix} \tag{4.9}$$

Kirshův operátor

$$G_R(x,y) = \begin{bmatrix} -5 & 3 & 3 \\ -5 & 0 & 3 \\ -5 & 3 & 3 \end{bmatrix} \qquad G_S(x,y) = \begin{bmatrix} 3 & 3 & 3 \\ 3 & 0 & 3 \\ -5 & -5 & -5 \end{bmatrix} \tag{4.10}$$

4.4.2 Detekce hrany druhou derivací

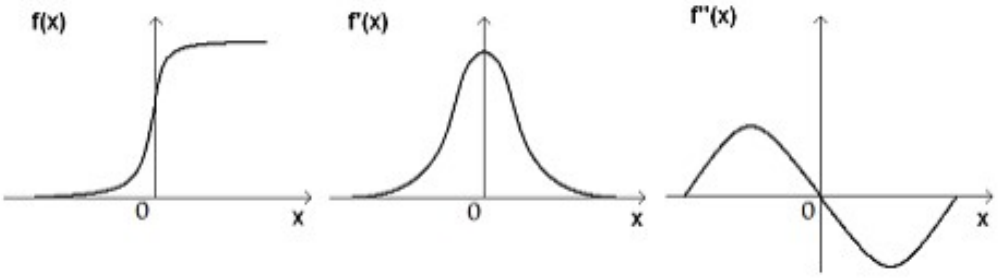
Metody založené na detekci hrany za pomoci druhé derivace hledají průchod nulou, nikoliv již velikost gradientu. Průchod nulou je na nalezení jednodušší nežli zjišťovat gradient.

Ve druhé derivaci se využívá funkce Laplacian. Tato funkce má výhodu, že je stejná ve všech směrech, není tedy závislá na rotaci.

$$\Delta^2 f(j,k) = \frac{\delta^2 f(j,k)}{\delta^2 j} + \frac{\delta^2 f(j,k)}{\delta^2 k} \tag{4.11}$$

Laplacián má nevýhodu ve dvojité odezvě na některé hrany v obraze. Funkci Laplacian, můžeme nahradit Laplaceovými operátory [28]:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 2 & -1 & 2 \\ -1 & -4 & -1 \\ 2 & -1 & 2 \end{bmatrix} \quad \begin{bmatrix} -1 & 2 & -1 \\ 2 & -4 & 2 \\ -1 & 2 & -1 \end{bmatrix} \tag{4.12}$$



Obrázek 4.5: Průběh signálu v první a druhé derivaci [7]

Nevýhodami druhé derivace jsou velké vyhlazování obrazu, ztráta ostrých rohů a vytváření uzavřených smyček hran [7].

4.5 Houghova transformace

Houghova transformace je výpočetně mnohem náročnější než předešlé uvedené metody zpracování obrazu, ale tuto nevýhodu vyvažuje svou větší přesností při popisu hran. Aby bylo možné využít Houghovu transformaci, je důležité znát analytický popis tvaru hledaného objektu, který chceme detekovat. Například analytický popis přímky, kružnice nebo trojúhelníka. Pro použití Houghovy transformace se nejvíce hodí prahovaný (binární) obraz, ve kterém byly detekovány hrany. U této metody dochází k transformaci kartézského souřadnicového systému do polárního pomocí vzorce:

$$\rho = x * \cos\Theta + y * \cos\Theta \tag{4.13}$$

kde x a y jsou souřadnice bodu např. A náležícího přímce v kartézské soustavě souřadnic. Řecké písmeno ρ zde udává nejkratší vzálenost bodu A od počátku soustavy souřadnic a Θ je úhel mezi osou x a přímkou spojující počátek s bodem A. Díky této vzdálenosti a úhlu můžeme zobrazit přímku jako bod v prostoru, který má na jedné ose hodnoty ρ a na druhé ose hodnoty úhlu Θ. Tento prostor nazveme Hough space. Pokud se v Kartézské soustavě bude protínat několik přímek v tom samém bodě, vznikne nám v Hough space množina bodů, které budou tvořit sinusoidu. Ve výsledku dojde v Hough space k propnutí několika sinusoid v jednom bodě. Tento bod nám udává možný výskyt přímky v Kartézské soustavě souřadnic.

Výhodou této metody je její malá citlivost na porušená data a šum, jelikož čím více sinusoid se protne v určitém bodě tím je větší pravděpodobnost výskytu přímky v obraze [7].

4.6 Skeletonizace

Tato metoda je určena k vytvoření skeletu, který je reprezentován grafem dané oblasti nebo objektu. Skelet je tvořen množinou středů kružnic, které se nacházejí uvnitř objektu, u něhož hledáme kostru. Zároveň se dané kružnice musí dotýkat hranice daného objektu, alespoň ve dvou bodech. Tento skelet je možné vytvořit pomocí operací eroze a dilatace z oblasti matematické morfologie. Podrobněji je tento proces popsán zde [14].

Můžeme zde popsat jak lze jednoduchým způsobem realizovat skeletonizaci v digitálním obraze. Nejprve než bude uveden zmiňovaný příklad je nutné definovat určité předpoklady pro tento proces. Prvním předpokladem je práce s prahovaným obrazem, kde tvar je prahován bílou barvou. Dalším předpokladem je logická matice/maska:

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{4.14}$$

kdy u této matice kontrolujeme zda jsou okolní prvky středu matice pravdivé, tak bude pravdivý i prvek na středu masky. Pokud jsou okolní pixely bílé bude bílý i obklopený pixel a zaneseme tuto informaci do nového obrazu.

S těmito předpoklady budeme postupovat při procházení obrazu jako u konvoluce. Tento proces opakujeme vždy nad nově vytvořeným obrazem do té doby nežli nebude docházet k žádné změně v tomto obraze.

4.7 Odstranění šumu

V této kapitole je několikrát zmíněna informace, že je nutné odstranit šum. Odstranění šumu je velmi obsáhlá kapitola, která by zabrala značnou část této práce. Proto zde budou uvedeny názvy několika nejpoužívanějších algoritmů a odkaz, kde je možné se o těchto algoritmech dozvědět více. Mezi nejpoužívanější algoritmy se například řadí Gaussovo vyhlazování, Anizotropní filtrování a Wienerův filtr. Zde je uveden odkaz na zmiňovanou práci zabívající se odtraněním šumu z obrazu: [35].

5 Možnosti implementace

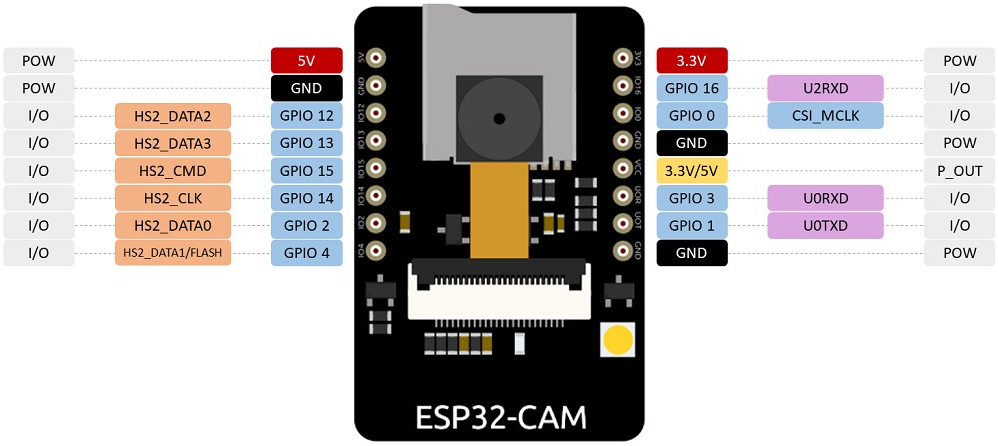
V této kapitole jsou popsány možnosti HW implementace. Nejprve je popsána samotná vývojová deska ESP-32 cam, dále jednotlivý zástupci kamer a kamerových modulů, které lze připojit k desce ESP-32 cam.

5.1 Vývojová deska ESP-32 cam

Vývojová deska **ESP-32 cam** [30] vychází z desky ESP32-S. Svými rozměry se řadí mezi menší vývojové desky. Její rozměry jsou 27×40,5mm. Tato vývojová deska může být široce využita v různých IoT aplikacích, například pro bezdrátové ovládání, sledování, načítání QR kódů a další IoT aplikace. Disponuje 32 bitovým procesorem. Procesor může být přetaktován až na 160 MHz. S výpočetním výkonem až 600 DMIPS. Vývojová deska ESP-32 cam má také zabudovaný Wi-Fi modul podporující standardy 802.11b/g/n. Spolu s Wi-Fi připojením podporuje zapezbečení WPA, WPA2, WPA2-Enterprise a WPS.

Deska má dále zabudovanou 520KB SRAM paměti a také podporuje Pulzní šířkovou modulaci (PWM), SPI a I2C sběrnici spolu s ADC a DAC převodníky. Mezi oficiálně výrobcem podporované kamery patří kamera OV2640 a kamera OV7670. Vývojová deska má zabudovanou LED diodu, která slouží jako baterka. Dalším vybavením je slot pro připojení paměťové karty, přesněji Micro SD karty o maximální velikosti 4GB, která rozšiřuje vnitřní paměť zařízení. Dále disponuje modulem pro Bluetooth komunikaci na verzi 4.2. Doporučené napájecí napětí je 5V, ale taktéž dokáže na několika pinech pracovat s napětím 3,3V. Podle dostupné dokumentace je schopna pracovat v rozmezí teplot od -20°C do 85°C. Mezi podporované formáty patří `JPEG` a `BMP`. Podporuje obraz v odstínech šedi.

Vývojová deska má 16 externích pinů a také několik interních pinů, které můžeme rozdělit do dvou skupin. Popis externích pinů je zobrazen na obrázku 5.1. V první skupině jsou interní piny, ke kterým se připojují kamery, a ve druhé skupině interní piny, kterými se přistupuje k paměťové mikro SD kartě. První skupina je popsána v tabulce 5.1 a druhá skupina je popsána v tabulce 5.2. Vývojová deska ESP-32 cam je zobrazena na obrázku 5.2



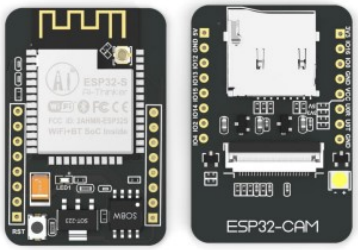
Obrázek 5.1: Rozložení pinů ESP32-cam [32]

název pinu	typ pinu	název pinu	typ pinu
Y0	———	XCLK	GPIO0
Y1	———	DOVDD	VCC
Y2/D0	GPIO5	DVDD	VCC
Y3/D1	GPIO18	HREF	GPIO23
Y4/D2	GPIO19	PWDN	———
Y5/D3	GPIO21	VSYNC	GPIO25
Y6/D4	GPIO36	RESET	———
Y7/D5	GPIO39	SIO_C	GPIO26
Y8/D6	GPIO34	SIO_D	GPIO27
Y9/D7	GPIO35	AVDD	VCC
PCLK	GPIO22	AGND	GND
DGND	GND	NC/POWER	GPIO32

Tabulka 5.1: Interní piny pro připojení kamery k desce ESP32-cam[20, 32]

název pinu	typ pinu
CLK	GPIO14
CMD	GPIO15
DATA0	GPIO2
DATA1/FlashLight	GPIO4
DATA2	GPIO12
DATA3	GPIO13

Tabulka 5.2: Interní piny pro komunikaci s paměťovou kartou na desce ESP32-cam[20]



Obrázek 5.2: Vývojová deska ESP32-cam [30]

5.2 Kamery

Typů kamer je dnes na trhu velké množství. Liší se nejen rozměry, ale také způsobem připojení, rozlišením a dalšími různými parametry. Některé z těchto typů kamer si zde přiblížíme.

Prvním typem kamer jsou kamery s analagovým rozhraním. U analogového rozhraní se využívá jednoho ze dvou nejvíce rozšířených standardů kódování analogového signálu, kterými jsou `NTSC` nebo `PAL`. Tyto standardy se především využívají pro televizní vysílání. Větší kamery, například bezpečnostní, které tyto formáty také využívají, jsou často připojovány za pomoci tří vodičů, případně dvou, kdy jeden kabel obsahuje dva vodiče. Dva vodiče nebo jeden společný kabel se dvěma vodiči slouží pro připojení napájení a zemnícího vodiče. Zbývající samostatný vodič je využit pro přenos analogových dat. Pro tento účel se nejčastěji využívá koaxiálního kabelu [8, 29].

Dalším možným typem kamer jsou kamery, které využívají k připojení konektor USB (Universal serial bus) a jsou primárně určeny pro připojení k počítači. Komunikace zařízení s kamerou probíhá po sériové sběrnici. K dnes nejpoužívanějším typům konektorů patří typ A a C verze 3.0 nebo 3.1.

Jedním z dalších možných typů kamer jsou kamery, které využívají pro připojení konektor RJ45. Konektor a vodič tzv. kroucená dvojlinka se nejvíce využívá v oboru počítačových sítí a telekomunikací. Tento typ připojení se nejčastěji využívá pro zapojení počítačových sítí, ale může se také využít pro zapojení bezpečnostních kamer. Toto zapojení je výrazně lepší oproti čistě analogovému rozhraní. Bezpečnostní kamery buď již mají zabudovaný konektor pro ethernetové připojení, nebo se využívá externí adaptér, který má analogové rozhraní pro připojení kamery.

Dalším typem kamer jsou kamery s DVP (digital video port) připojením. Jedná se o paralelní port a je možné přenášet několik bitů současně. Často jsou tyto kamery připojeny k desce plošných spojů za pomoci FCC/FPC kabelů a konektorů, pokud nejsou již přímo připájeny na desku plošných spojů [33].

Posledním z představovaných typů kamer v této kapitole jsou kamery s připojením MIPI/CSI. Jedná se o sériové rozhraní, které je rychlejší a podporuje větší rozlišení nežli DVP rozhraní. Připojení tohoto typu se nejčastěji dnes využívá ve fotoaparátech a mobilních zařízeních. Tento typ připojení například využívá kamera určena pro Raspberry Pi. MIPI/CSI rozhraní potřebuje také pro správné fungování dodatečný hardware. V tomto ohledu je lepší DVP port [16].

Z předešlého výčtu nejpoužívanějších typů kamer a jejich připojení je patrné, že pro připojení k desce ESP-32 cam 5.1 je nejvhodnější využít kamer s DVP připojením.

5.2.1 Sledované parametry

U kamery, která je jednou z nejdůležitější částí návrhu budeme, sledovat čtyři důležité parametry: výstupní formát dat, napájecí napětí, pozorovací úhel a způsob připojení. Tyto paramtery budeme porovnávat s dříve popsanou deskou ESP-32 cam.

Kamery, které budeme porovnávat, mají napájecí napětí 3,3V nebo 5V. Všechny modely lze připojit k desce ESP-32 cam. Rozdíly mezi kamerami jsou ve velikosti, rozlišení, pozorovacím úhlu a připojení k mikrokontroléru.

5.2.2 ArduCam-Mini OV5642

Kamerový modul ArduCam-Mini OV5642 [2, 31] rozšiřuje možnosti různých desek mikrokontrolérů, například Arduino, mikrokontrolérů STM32 nebo minipočítače Raspberry Pi. Jedná se o modul, nikoliv o samostatnou kameru. Rozměry kamerového modulu, jsou 34×24 mm. Kamerový modul dokáže pracovat jak s napětím 3,3V, tak i s napětím 5V. Maximální rozlišení kamerového modulu je 2592×1944 pixelů. Výstupní formát dat z kamerového modulu:

- `YUV 422/420`
- `YCbCr 422`
- `RGB 565/555/444`
- `CCIR 656`
- `komprimovaná 8 bitová RGB data`
- `8/10 bit raw RGB data`
- `JPEG`

Data ve formátech `YCbCr 422`, `RGB 565/555/444` a `8/10 bit raw RGB data` by se dala využít pro realizaci práce. Jedná se o nekomprimované formáty dat. Odpadá zde proces dekomprese dat při zpracování obrazu.

Obrázek 5.3: Kamerový modul ArduCam-Mini OV5642 [2]

Pozorovací úhel snímače obrazu připojeného k modulu je 24°.

Kamerový modul se připojuje k mikrokontroléru za pomoci 8 pin konektorů.

- `CS` - Chip Select, výběr zařízení na sběrnici SPI
- `MOSI` - výstupní datová linka sběrnice SPI
- `MISO` - vstupní datová linka sběrnice SPI
- `SCK` - hodinová linka sběrnice SPI
- `GND` - uzemnění systému
- `VCC` - napájecí napětí od 3,3 V do 5 V
- `SDA` - výstupní datová linka sběrnice I2C
- `SCL` - hodinová linka sběrnice I2C

Kamerový modul je možné nastavit pomocí I2C sběrnice. Jde tedy ovlivnit parametry jako vyvážení, jas, kontrast a sytost.

Podoba kamerového modulu je zachycena na obrázku 5.3

5.2.3 ArduCam-Mini OV2640

Kamerový modul ArduCam-Mini OV2640 [1] jako předešlý kamerový modul 5.2.2 rozšiřuje možnosti různých desek mikrokontrolérů a minipočítače Raspberry Pi. Jedná se jako v předchozím případě o kamerový modul, nikoliv o samostatnou kameru. Kamerový modul se liší od přechozího kamerového modulu pouze maximálním rozlišením, které je 1600×1200 pixelů. Rozměry a napájecí napětí jsou stejné jako u kamerového modulu ArduCam-Mini OV5642. Tento kamerový modul nepodporuje na rozdíl od předchozího kamerového modulu výstupní formáty dat `CCIR 656` a `RGB 444`. Pozorovací úhel snímače obrazu připojeného k modulu je 25°.

Kamerový modul ArduCam-Mini OV2640 se připojuje k mikrokontroléru za pomoci stejných 8 pin konektorů jako kamerový modul ArduCam-Mini OV5642. Kamerový modul je taktéž možné nastavit pomocí I2C sběrnice a ovlivnit stejné parametry jako u předchozího kamerového modulu.

Vzhled kamerového modulu je v podstatě identický jako u kamerového modulu zobrazeného na obrázku 5.3.

5.2.4 Kamera NT99141

Kamera NT99141 [3] je nejmenší z modelů kamer, které jsou představeny, a to s rozměry 20,87×12,5 mm a má také jiné maximální rozlišení kamery, které je 1280×720 pixelů. Kamera je založena na technologii `CMOS`. Napájecí napětí kamery je 3,3V.

Výstupní formát dat kamery:

- `YCbCr 422`
- `Raw RGB data`
- `RGB 565/555/444`
- `CCIR 656`
- `JPEG`

Pozorovací úhel kamery je 68°.

Kamera podporuje funkce automatického ovládání Auto-Exposure Control a Auto-White Balance. Zároveň lze také ovlivnit i parametry jako redukce šumu, gama, sytost barev, protože kamera také disponuje korekcí vad.

Kamera se připojuje k mikrokontroléru za pomoci 24 pinového FCC/FPC konektoru.

- `NC`
- `VSYNC`
- `XCLK`
- `D2`
- `AGND`
- `PWDN`
- `D8`
- `D5`
- `SIO_D`
- `HREF`
- `DGND`
- `D3`
- `AVDD`
- `DVDD`
- `D7`
- `D4`
- `SIO_C`
- `DOVDD`
- `PCLK`
- `D1`
- `RESET`
- `D9`
- `D6`
- `D0`

Kamera je zachycena na obrázku 5.4.

Obrázek 5.4: Kamera NT99141 [3]

5.2.5 Kamera OV7670

Kamera OV7670 [5] má rozměry 21×12,5 mm a je založena na technologii `CMOS`. Maximální rozlišení kamery je 640×480 pixelů. Nepodporovaným výstupním formátem dat jsou formáty `CCIR 656`, `JPEG`, které byly podporovány kamerou NT99141 5.2.4. Kamera OV7670 navíc podporuje formáty `YUV 422`, `GRB 422`.

Pozorovací úhel kamery je 45°.

Kamera OV7670 na rozdíl od kamery NT99141 také navíc podporuje funkce Automatic gain control, Automatic band filter a Automatic blacklever calibration. Tato kamera se připojuje jako předchozí kamera k mikrokontroléru za pomoci 24 pinového FCC/FPC konektoru. Vzhled kamery je v podstatě identický jako u kamery zobrazené na obrázku 5.4.

5.2.6 Kamera OV2640

Kamera OV2640 [4, 22] má stejné rozměry jako předešlá kamera OV7670 5.2.5. Maximální rozlišení kamery je 1600×1200 pixelů. Tato kamera je doporučena výrobcem k připojení k desce ESP32-cam. Kamera OV2640 na rozdíl od předchozí kamery nepodporuje formáty `RGB 444` a `GRB 422`. U této kamery je výstupní formát `YUV 422` rozšířen o variantu `YUV 420`. Kamera podporuje `komprimovaná 8 bitová RGB data` a `8/10 bit raw RGB data`.

Pozorovací úhel kamery je 52°.

Kamera pouze podporuje funkce automatického ovládání Auto-Exposure Control a Auto-White Balance. Zároveň kamera na rozdíl od kamery OV7670 disponuje redukcí šumu a korekcí vad. Kamera se připojuje k mikrokontroléru stejným způsobem jako u předchozích dvou uvedených kamer. Tedy za pomoci 24 pinového FCC/FPC konektoru.

Vzhled kamery je v podstatě identický jako u kamery zobrazené na obrázku 5.4.

5.2.7 Shrnutí

Výše popsané kamery nebo kamerové moduly jsou jen vybranou částí z různých možností, které jsou kompatibilní s deskou ESP-32 cam. Existuje mnoho dalších kamer či kamerových modulů, které by se daly využít k jiným projektům. Pro námi zvolený projekt je tento výčet dvou kamerových modulů a tří kamer dostačující. Jako výslednou kameru pro využití v této práci jsem zvolil kameru OV2640 z důvodu možnosti připojení přímo na integrované piny na desce ESP-32 cam. Dalším důvodem pro zvolení byly výstupní formáty dat, které jsou nekomprimované a můžeme s nimi rovnou pracovat. Zároveň má kamera dostačující pozorovací úhel. U většiny zbývajících kamer byl pozorovací úhel malý a nedostačující pro detekci čar a jednotlivých objektů v prostoru. Nespornou výhodou je, že kamera OV2640 je primárně výrobcem určena pro připojení k desce ESP-32.

6 Návrh a konstrukce robota

Návrh robota lze rozdělit do tří částí. První částí je výběr vhodných komponentů pro stavbu robota, jako je pohon, ovládání pohonných jednotek a další příslušenství. Následující částí návrhu je celkové zapojení jednotlivých komponentů do funkčního celku. Poslední částí celého návrhu je návrh kostry robota neboli z jakého materiálu bude kostra tvořena, jaké metody výroby využijeme a především jak samotný robot bude vypadat to je vytvoření digitálního modelu jednotlivých částí robota.

6.1 Základní parametry robota

Nejprve než započneme se samotným výběrem komponentů a návrhem robota, musíme si stanovit několik základních specifikací. První specifikací, kterou si stanovíme, je typ pohonu robota.

Můžeme vybrat ze tří možností pohonu. První možností je pohon pomocí pásů, které zajišťují lepší průchodnost terénem, ale hlavním úkolem robota je sledovat čáru, nikoliv se pohybovat v nepřístupném terénu. O této variantě pohonu nebudeme dále uvažovat. Druhým typem pohonu je pohon všech kol, kdy by na robotu byla poháněna jak přední, tak zadní kola. Tento pohon by umožňoval rychlejší pohyb, ale na druhou stranu by hůře reagoval na změny dráhy. Třetí možností je pohon pouze dvou předních kol a nahrazení zadní nápravy, například otočným kolečkem. Z dříve uvedeného výčtu je patrné, že pro naše řešení je vyhovující třetí druh pohonu, který nám umožní lepší pohybové vlastnosti, především otáčení robota.

Další specifikací, kterou si musíme určit je, jak bude robot napájen. Zdali bude odděleno napájení desky ESP32-cam od napájení motorů či nikoliv. Výhodou odděleného napájení by bylo snadné napájení rozdílným napětím. Nevýhodou tohoto řešení je jeho prostorová náročnost. Na druhou stranu napájení z jednoho zdroje má jednu hlavní nevýhodu a tou je napájení nedostatečným proudem mikrokontrolér a samotné motory, pokud zdroj není schopen poskytovat dostatečný proud. Další nevýhodou je nutnost přidání regulátoru napětí tak, aby mohlo být dosaženo napájení více urovněmi napětí. Z důvodu úspory místa byla zvolena varianta napájení z jednoho zdroje i přes jeho nevýhody.

Nyní byly uvedeny hlavní specifikace, které se týkají výběru komponentů.

Poslední specifikací, kterou je nutno zvážit, je umístění kamery. Tato specifikace již spadá do návrhu kostry robota. Pro kameru můžeme zvolit

více míst umístění. Za prvé kamera může být umístěna přímo nad čárou v dostatečné vzdálenosti tak, aby zabírala pouze sledovanou čáru a malé okolí této čáry. Dalším způsobem umístění kamery je mít kameru nakloněnou tak, aby snímala i částečně prostor před robotem. Toto umístění by bylo vhodné, kdybychom realizovali vyhýbání se objektům v prostoru. Jak již ale bylo zmíněno dříve v této práci, je realizováno sledování čáry. Pro realizaci návrhu je lepší první varianta umístění.

6.2 Komponenty

Při výběru komponentů se zohlední dříve definované parametry viz 6.1.

6.2.1 Motory

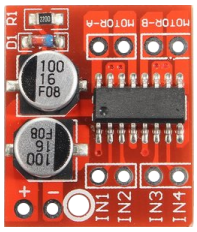
Motory, které byly zvoleny pro realizaci robota, jsou DC motory s převodovkou [18]. Převod motorů je v poměru 48:1. Dané motory mají pracovní napětí od 3V do 6V a proud od 100mA do 120mA.

Obrázek 6.1: DC motor s převodovkou [18]

6.2.2 Ovládání motorů

Pro ovládání motorů bylo rozhodováno mezi dvěma motor drivery. Jeden disponuje H-můstkem L298N a druhý můstkem MX1508 viz obr. 6.2.

Nakonec byl vybrán druhý zmiňovaný můstek z důvodu menších rozměrů, které jsou 24,7×21×7mm, a možností napájení již od 2V. Tento můstek může být napájen napětím od 2V do 10V a stálý maximální proud je 1,5A. Tento můstek je schopen špičkově pracovat s proudem 2,5A [27].



Obrázek 6.2: Dual DC motor driver MX1508 [27]

6.2.3 Kola

Výběr kol pro motory byl velice jednoduchý, protože k vybraným motorům jsou často dodávány i příslušná kola. Poslední kolo, které zbývá vybrat, je zadní všesměrové kolo. Nejprve bylo vybráno kolo, které je primárně určeno pro nábytek. Kolo bylo využito spolu s ložiskem, aby bylo docíleno lepší otáčení daného kola. Tento předpoklad byl nesprávný, jelikož pro otočení daného kola v reakci na změnu pohybu robota bylo nutné vyvinou dostatečně velkou sílu v novém směru pohybu robota, kterou robot dokázal vyvinou pouze tehdy, když motory pracovaly na plný výkon. Reakce na změnu směru ovšem nebyla okamžitá a docházelo k vybočování robota z definované dráhy. Došlo k nahrazení tohoto návrhu a bylo využito všesměrové kuličky, která je uvedena na obrázku 6.3. Ta zajišťuje plynulý pohyb a okamžitou reakci na změnu pohybu.

Obrázek 6.3: Všesměrová kulička [10]

6.2.4 Napájení

Napájení robota je řešeno pomocí napájecího modulu nepájivého pole [23]. Tento modul poskytuje maximální proud 700mA a napájecí napětí buď to 3,3V nebo 5V. Jelikož deska vyžaduje napájení 5V a pro napájení motorů je lepší využít vyššího napětí, než je minimální doporučené, proto motory jsou napájené také 5V.

Další součástí napájení je samotný napájecí zdroj. Nejdříve byla využívána pro napájení powerbanka s výstupním napětím 5V a proudem až 1A s

kapacitou 4000mAh. Po otestování se ukázalo, že v napájecím modulu dochází k poklesu napětí a modul není schopen tento úbytek kompenzovat. Proto bylo učiněno rozhodnutí vyměnit zdroj napájení za dvě Li-ion baterie typu 18650 [12]. Tyto baterie mají jmenovité napětí 3,7V se stálým proudem 520mA a kapacitou 2600mAh. Baterie jsou umístěny v bateriovém držáku, kde jsou zapojeny v sérii, takže poskytují ve výsledku napájecí napětí 7,4V. S tímto vstupním napětím je již modul schopen kompenzovat úbytek napětí a poskytovat dostatečné napájecí napětí.

6.3 Zapojení

Nyní bude popsáno zapojení vybraných komponentů. Pro ovládání motorů jsou využívány piny 12,13,14,15 na desce ESP32-cam. Tyto piny umožňují ovládání pomocí pulzní šířkové modulace, je možné motorům nastavit různý výkon podle síly PWM signálu. Pro napájení desky je využit 5V napájecí pin, který je připojen k napájecímu modulu. Dále je z napájecího modulu přímo napájen 5V i H-můstek, který ovládá motory. Napájecí modul je napájen dvěma Li-ion bateriemi pomocí napájecího konektoru. Diagram zapojení je uveden na obrázku 6.4.

6.4 Návrh konstrukce robota

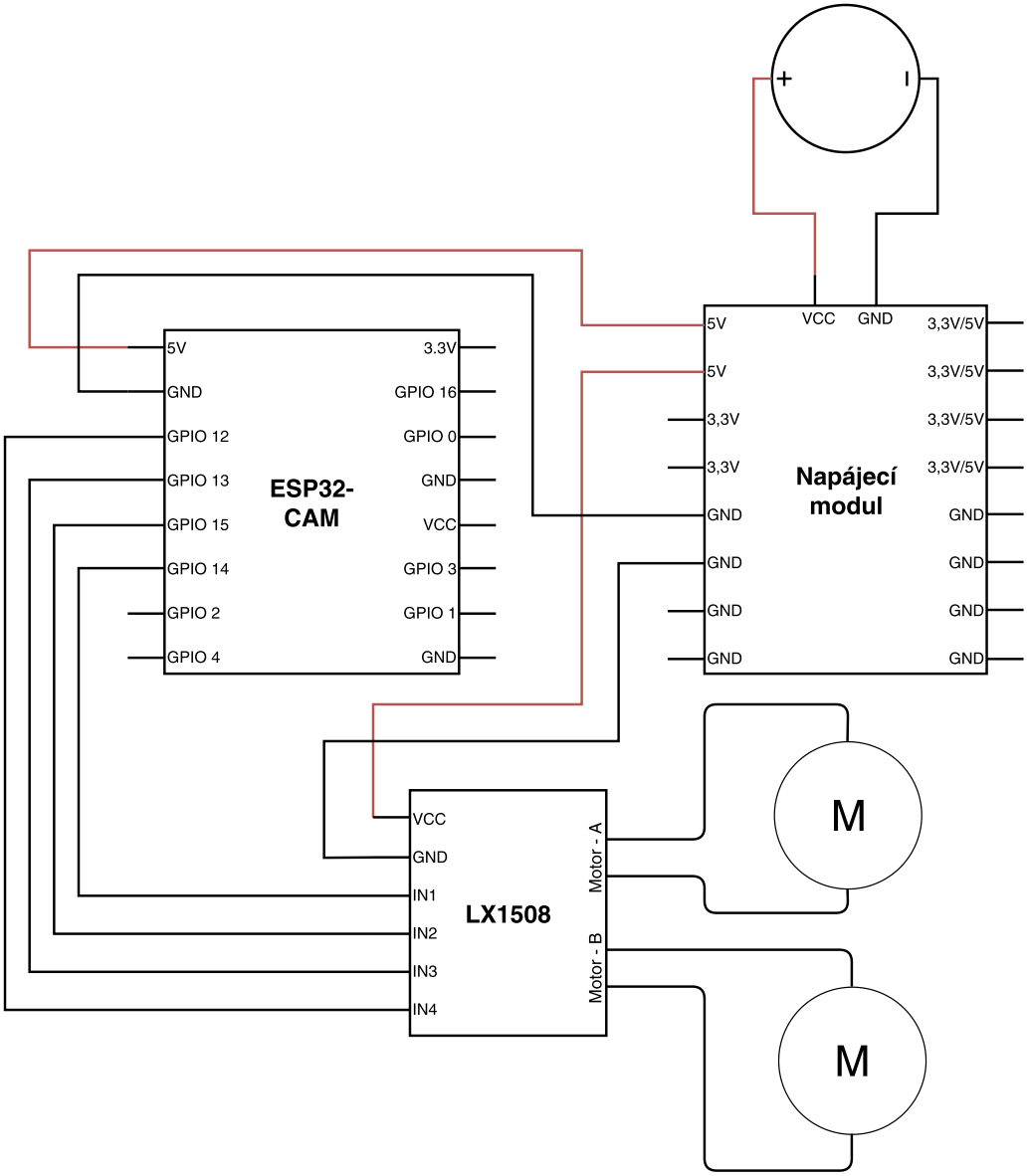
Při návrhu robota se zohlední dříve definovaný parametr umístění kamery.

Samotný návrh byl inspirován z již existující stavebnice robota LAFVIN Smart Robot Car Kit 4WD [17]. Návrh byl nakonec rozdělen do několika samostatných částí pro lepší prototypování a úpravy těchto částí v budoucnu.

- Podvozek robota
- Uchycení motoru
- Uchycení baterie
- Distanční sloupek
- Vrchní deska robota
- Uchycení napájecího modulu
- Uchycení H-můstku
- Rameno kamery
- Uchycení kamery

Návrh výše uvedených částí byl proveden v programu Fusion 360 od společnosti Autodesk, Inc. za využití studentské licence.

Modely jednotlivých částí jsou součástí příloh bakalářské práce.



Obrázek 6.4: Diagram zapojení

Nyní je důležité zvolit výrobní metodu pro jednotlivé díly konstrukce. Z důvodu dostupnosti a rychlého prototypování byla zvolena možnost 3D tisku. Výroba jednotlivých dílů byla provedena na tiskárně Ender 3 Pro.

Výrobní metodou byl omezen i výběr materiálů možných pro výrobu jednotlivých dílů konstrukce robota. Při výběru materiálů byly zohledněny tyto parametry: dostupnost materiálu, teplota tavení materiálu, pevnost materiálu a náročnost tisku. Ve výsledku byly vybrány tři materiály, které jsou uvedeny v tab. 6.1. Tyto materiály jsou nejvíce vyhovující pro výrobu jednotlivých částí robota.

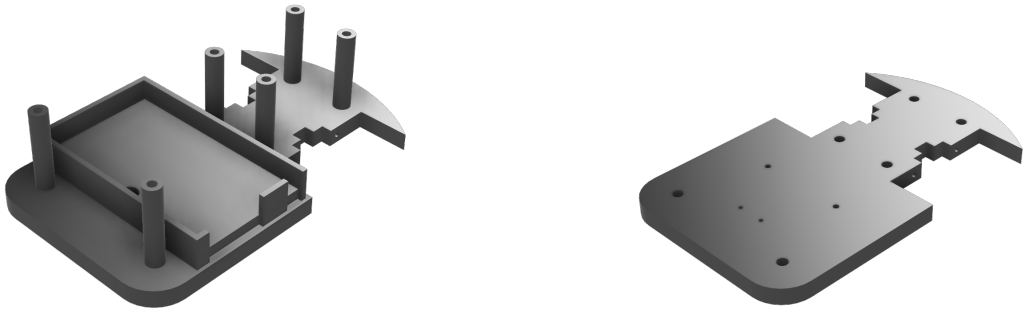
Materiál	Teplota tepelné deformace [°C]	Mez pevnosti v tahu [MPa]	Deformace tisku velkých modelů
PLA	55	57	ne
PETG	68	46	ne
ASA/ABS	93	40	ano

Tabulka 6.1: Vlastnosti filamentů[6]

Nejlepším materiálem pro výrobu jednotlivých částí se stal materiál PETG (Polyethylene terephthalate glycol), který zajišťuje lepší teplotní odolnost na rozdíl od materiálu PLA (Polylactic acid). Navíc tento materiál netrpí na deformaci tištěného modelu jako je tomu u materiálu ASA/ABS (Acrylonitrile styrene acrylate/Acrylonitrile butadiene styrene).

6.4.1 Podvozek robota

Tato část byla jednou z nejtěžších pro návrh, protože doznala největších změn v průběhu prototypování. Nejprve měly být na této části robota pevně umístěny úchyty pro motory spolu s dilatačními sloupky a úchytem baterie. Nakonec byly tyto části odděleny od podvozku a nahrazeny jednotlivými částmi, které jsou popsány dále. Na obrázku 6.5 je vidět prvotní návrh podvozku robota tak i finální verze tohoto konstrukčního prvku.



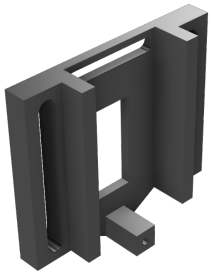
(a) Prvotní návrh (b) Konečný návrh

Obrázek 6.5: Model podvozku robota

6.4.2 Uchycení motoru

Uchycení motoru bylo odděleno od podvozku z důvodu možnosti nastavení pozice motorů a úpravě světlé výšky robota. Prvotní návrh úchytu motoru

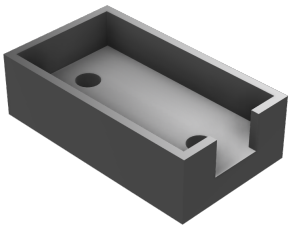
byl převzat z podvozku robota, ale následně byl upraven kvůli lepšímu uchycení motoru k danému úchytu a lepší stabilitě motoru. Výsledná verze uchycení motoru se nachází na obrázku 6.6.



Obrázek 6.6: Model uchycení motoru

6.4.3 Uchycení baterie

Uchycení baterie bylo odděleno od podvozku robota z důvodu použití různých zdrojů napájení během vývoje. Toto oddělení umožňuje v budoucnu využití jiného zdroje napájení, nežli je tomu nyní. Výsledný model uchycení baterie se nachází na obr. 6.7.



Obrázek 6.7: Model uchycení baterie

6.4.4 Distanční sloupek

Distanční sloupky byly odděleny od podvozku robota z důvodu urychlení tisku podvozku a také z důvodu nedostatečné pevnosti při uchycení vrchní desky robota pomocí šroubů. Model distančního sloupku je zobrazen na obr. 6.8.

Obrázek 6.8: Model distančního sloupku

6.4.5 Vrchní deska robota

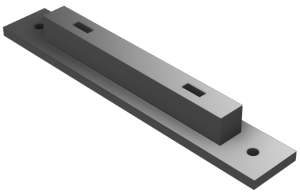
Vrchní deska v prvotní fázi návrhu obsahovala uchycení napájecího modulu a H-můstku. Tyto části byly odděleny. Důvody oddělení jsou popsány níže u popisu jednotlivých částí. Vrchní deska disponuje otvory pro uchycení zmíněných částí včetně ramene pro kameru. Na obrázku 6.9 je vidět prvotní i finální návrh vrchní desky robota.

(a) Prvotní návrh (b) Konečný návrh

Obrázek 6.9: Model vrchní desky robota

6.4.6 Uchycení napájecího modulu

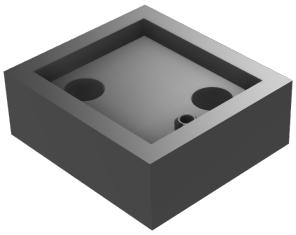
Uchycení napájecího modulu bylo upraveno v poslední fázi návrhu z důvodu co nejlepšího uchycení a možnosti nahrazení napájecího modulu. Do uchycení napájecího modulu je nutné vlepit do připravených otvorů dvě dvoupinové dutinkové lišty s roztečí pinů 2,54 mm. Uchycení napájecího modulu se nachází na obr. 6.10.



Obrázek 6.10: Model uchycení napájecího modulu

6.4.7 Uchycení H-můstku

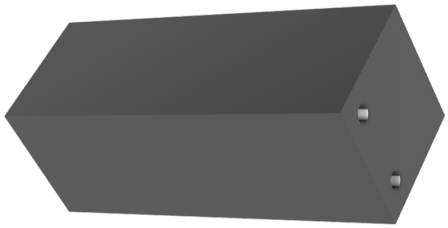
Oddělení této části bylo provedeno až v poslední fázi návrhu, z důvodu možného nahrazení vybraného H-můstku jinou alternativou. Proto nemusí být vyráběna celá vrchní deska robota, případný uživatel upraví pouze tuto část. Uchycení H-můstku se nachází na obr. 6.11.



Obrázek 6.11: Model uchycení H-můstku

6.4.8 Rameno kamery

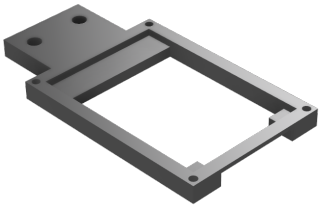
Rameno kamery v průběhu vývoje doznalo změn. Bylo nutné prodloužit rameno z důvodu vzdálenosti kamery od sledované čáry, a tím došlo k lepšímu snímání této čáry. Výsledný model ramene pro uchycení kamery se nachází na obr. 6.12.



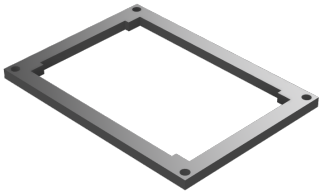
Obrázek 6.12: Model ramene kamery

6.4.9 Uchycení kamery

Uchycení kamery se skládá ze dvou částí. Ze samotného uchycení kamery a krytu, kterým se kamera zajistí proti případnému pohybu při jízdě robota.



(a) Uchycení kamery



(b) Kryt

Obrázek 6.13: Model uchycení a krytu kamery

6.4.10 Seznam součástek pro sestrojení robota

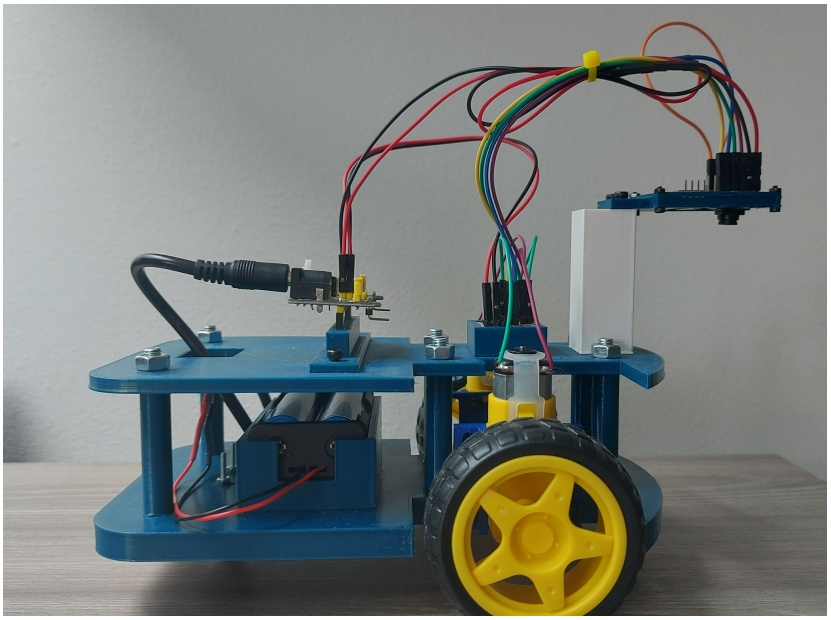
V této podkapitole je uveden seznam 6.2 jednotlivých dílů a spojovacího materiálu nutného k sestavení robota.

název	ks
Modul ESP32-cam	1
DC-motory s převodovkou	2
H-můstek LX1508	1
Napájecí modul nepájivého pole	1
Bateriový box pro 2× Li-ion baterie 18650	1
Li-ion baterie 18650 3,7V	2
Napájecí modul nepájivého pole	1
Dutinková lišta 2,54 mm 2 piny	2
Všesměrová kulička	1
Podvozek robota	1
Uchycení motoru	2
Distanční sloupek	6
Vrchní deska robota	1
Uchycení napájecího modulu	1
Uchycení H-můstku	1
Rameno kamery	1
Uchycení kamery	1
Šroub DIN 933 M6×60	6
Šestihranná matice DIN 934 M6	6
Šroub DIN 912 M4×14	2
Šroub DIN 912 M4×10	4
Šroub DIN 912 M3×8	4
Šroub DIN 912 M3×12	4
Šroub DIN 912 M2×12	4
Šestihranná matice DIN 934 M2	4

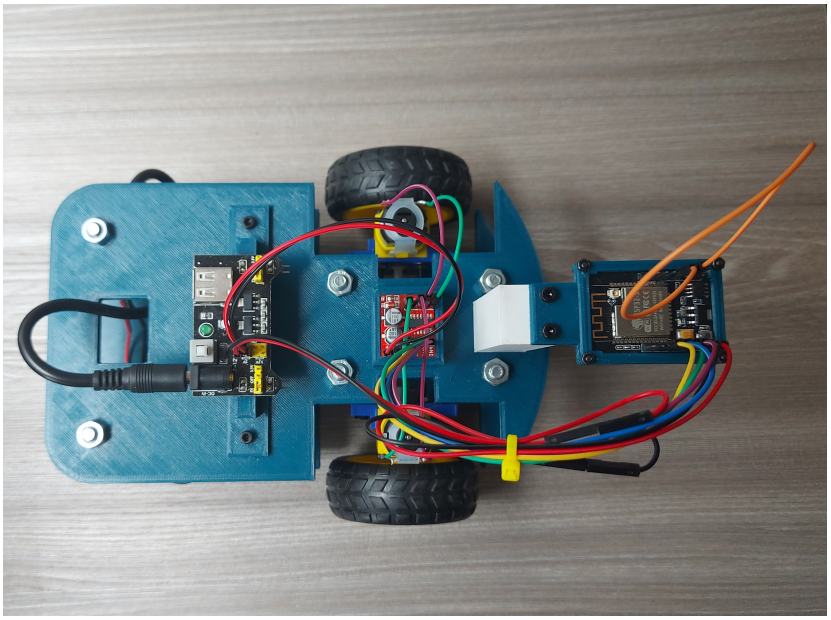
Tabulka 6.2: Seznam jednotlivých dílů

6.5 Výsledná podoba robota

Na níže uvedených obrazcích 6.14a a 6.14b, je uvedena výsledná podoba robota.



(a) Pohled z prava



(b) Pohled z hora

Obrázek 6.14: Výsledná podoba robota

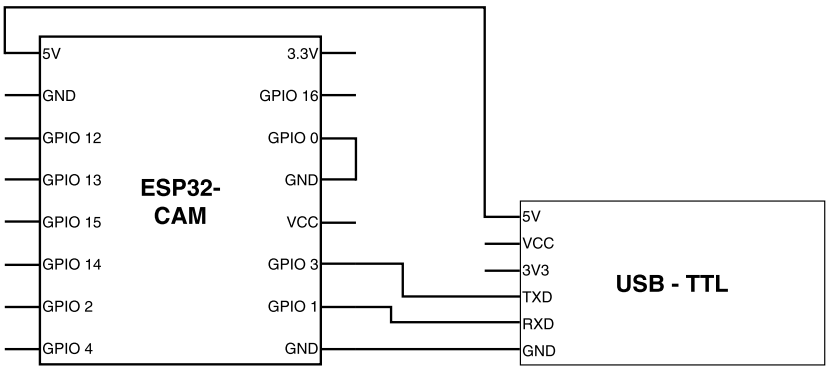
7 Softwarová implementace

Tato kapitola se zabývá popisem řešení softwarové implementace. Výsledné řešení lze rozdělit do několika částí: snímání obrazu, zpracování obrazu, zobrazení výsledků, ovládání motorů a řízení. Následně části snímání obrazu a zpracování obrazu budou tvořit výsledné API. Nejprve bude vysvětleno programování mikrokontroléru ESP32-cam a následně budou popsány jednotlivé části řešení.

7.1 Programování ESP32-cam

7.1.1 Hardware pro programování ESP32-cam

Dříve než proběhne samotné psaní programu, je nutné nalézt způsob jak propojit desku ESP32-cam spolu s počítačem. Pro propojení byla zvolena varianta USB - TTL UART převodníku, který slouží pro přenos dat mezi počítačem a deskou ESP32-cam. Pro připojení je nutné propojit jednotlivé piny pro napájení a přenos dat. Piny pro přenos dat jsou označeny TXD a RXD. Napájecí piny jsou označeny 5V, 3V3 a GND. Je doporučeno napájet desku napětím 5V, protože při napájení 3,3V není zaručeno správné fungování při nahrávání programu. Pro nahrání programu je také nutné propojit pin GND s pinem GPIO 0, jinak nelze program nelze nahrát do flash paměti mikrokontroléru. Zapojení pro nahrání programu na desku ESP32-cam je znázorněno na obrázku 7.1.



Obrázek 7.1: Diagram zapojení pro programování ESP32-cam

7.1.2 Software pro programování ESP32-cam

Program byl naprogramován ve vývojovém prostředí Visual Studio Code s rozšířením Platformio. Program je napsán pod Arduino frameworkem pro desku ESP32-cam. Dokumentaci tohoto frameworku lze nalézt zde [11].

7.2 Snímání obrazu

Pro snímání obrazu byla vybrána kamera OV2640, která byla blíže představenena v podkapitole 5.2.6. Pokud chceme kameru využít je nutné ji inicializovat a získávat z ní data. Za tímto účelem je využita knihovna `esp_camera.h`.

Implementace snímání obrazu je realizována v souborech `cam.h` a `cam.cpp`

Nastavení kamery je realizováno funkcí `camera_init(pixformat_t pixel_format)`, která přijímá jako parametr nastavení výstupního formátu kamery. Tato funkce nastaví i další parametry kamery například piny kamery, rozlišení, počet alokovaných bufferů kamery a další důležité parametry. Pro tuto práci bylo zvoleno nejnižší možné rozlišení kamery, tedy 160×120 pixelů. Velikost obrazového pole je v tomto případě 13×9 cm. Výstupní formáty kamery jsou omezeny pouze na formáty RGB565 a GRAYSCALE z důvodu snažší implementace funkcí pro zpracování obrazu.

K **získání obrazu z kamery** slouží funkce `camera_capture(img_t *img)`, která přijmá jako parametr ukazatel na proměnnou datového typu `img_t`, který je popsán v podkapitole 7.3.1. Uživatel tímto parametrem definuje případné oříznutí výstupního obrazu při převodu výstupních dat kamery do struktury `img_t`, která bude popsána v podkapitole 7.3.1.

7.3 Zpracování obrazu

7.3.1 Ukládání dat z kamery

Nejdříve než dojde ke zpracování obrazu je nutné jej nejprve uložit. Pro **uložení obrazu** byla vytvořena v struktura `img_t`, která obsahuje tyto parametry.

- `uint8_t *buf` - buffer pro uložení dat z kamery
- `size_t len` - velikost bufferu v bytech
- `size_t width` - šířka obrazu
- `size_t height` - výška obrazu
- `pixformat_t format` - formát obrazu

Nastavení a alokování proměnné pro uložení obrazu probíhá pomocí přetížené funkce `setup_image()`, která jako parametry přijmá buď formát a velikost obrazu z výčtového typu `FRAME_SIZE` nebo šířku, výšku a formát obrazu. K případnému uvolnění alokované paměti slouží funkce `free_img()`

Výčtový typ `FRAME_SIZE` obsahuje definici několika rozlišení, které jsou menší než nastavené rozlišení kamery. Toto řešení bylo zvoleno z důvodu zmenšení obrazu.

Pro dříve zmíňovaný převod dat z kamery do proměnné datového typu `img_t` je definována funkce `convert_camera_fb2img(camera_fb_t *fb, img_t *img)`, která jako parametry přijímá výstupní data kamery a ukazatel cílovou proměnnou obrazu.

Funkce `shrink_width_resolution(img_t *img_src,img_t *img_res, int factor)`, sloužila v práci pro zmenšení šířky obrazu, kdy docházelo k zpracování jenom části obrazu. Tato funkce byla implementována pro urychlení zpracování. Jelikož došlo k přepracování práce s daty z kamery již tato funkce není nutná. Tato funkce přejímá jako parametry ukazatele na proměnné obrazů. Jednoho, který má být zmenšen a druhého výstupního. Výstupní obraz musí být již definován předem. Posledním paramtetrem, který funkce přijmá je faktor zmenšení, který udává o kolik se vstupní obraz zmenší.

Další definovanou funkcí je `RGB565_2_GRAYSCALE_img(img_t *img_src, img_t *img_res)`. Tato funkce převádí obraz ve formátu RGB565 do formátu GRAYSCALE. Funkce jako parametry přijímá ukazatel na zdrojový obraz a na výstupní obraz. Výstupní obraz musí být také předem nastaven. Toto nastavení dopředu, urychlí samotné zpracování, jelikož se předejde při každém volání funkce k alokaci a uvolňování paměti.

Poslední definovanou funkcí je funkce `copy_img(img_t *img_src,img_t *img_res)`. Za pomoci této funkce je možné zkopírovat uložený obraz. Je nutné, aby přejímané parametry byly předem nastaveny. Důvod tohoto přednastavení je popsán u předchozí funkce.

Implementace dříve zmíněných datových struktur a funkcí je obsažena v souborech `img.h` a `img.cpp`.

7.3.2 Metody zpracování obrazu

Metody pro zpracování obrazu jsou implementovány ve dvou souborech a to `image_processing.h` a `image_processing.cpp`. Jedná se o metody dříve popsané v kapitole 4.

- `threshold`
- `double_threshold`
- `semi_threshold`
- `adaptiv_threshold`
- `convolution`
- `hough_transform`
- `skeletonization`

Jenotlivé funkce a parametry, které tyto funkce přijímají jsou blíže popsány v okomentovaném kódu, ale všechny funkce až na `hough_transform` upravují obraz, který je předán jako parametr.

Funkce `hough_transform(img_t *img, int threshold, Point_Array *point_array)`, přijímá jako parametry uložený obraz, práh pro nalezení přímky, a ukazatel na pole bodů. Pro výše zmíněnou funkci byly definované dva datové typy. Jedním je datový typ `Point`, který představuje bod se souřadnicemi $[x, y]$. Druhým datovým typem je `Point_Array`, který představuje pole bodů. Pro oba tyto datové typy jsou také definované funkce `free_Point` a `free_Point_Array`, které slouží pro uvolnění alokované paměti.

Jsou zde, ale také definované další funkce. Jednou z těchto funkcí je funkce `setup_image_processing(FRAMESIZE frame_size, pixformat_t pixel_format)`, která slouží pro nastavení a přípravu zpracování dat z kamery. Tato funkce přijímá jako parametry rozlišení a formát obrazu. Funkce vrací ukazatel na alokovanou proměnnou datového typu `img_t`, do které se mohou ukládat data z kamery.

Další definovanou funkcí, je přetížená funkce pro automatické určování prahu `get_threshold()`, která vrací vypočtený práh z dat obrazu. Jako parametry tato funkce přijímá ukazatel na uložená data, šířku a výšku obrazu, nebo ukazatel na uložený obraz. Výpočet prahu je realizován za pomoci bimodálního histogramu, jak bylo popsáno v podkapitole 4.2.

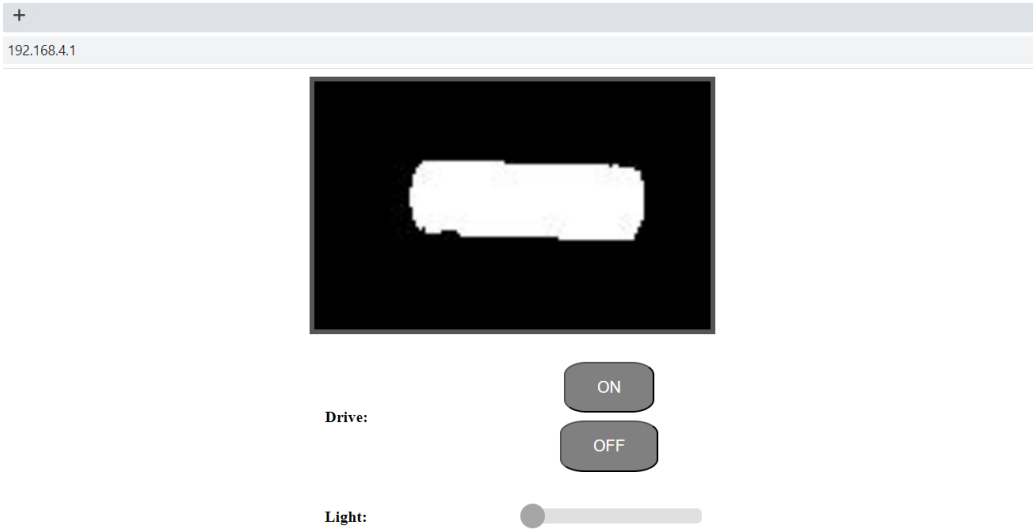
Poslední zde definovanou funkcí je funkce `find_colors(img_t *img)`, která přijmá jako parametr ukazatel na uložený obraz. Tato funkce nalezne v předaném obraze jednu ze čtyř barev a vrátí nalezenou barvu s největším výskytem. Jedná se o červenou, zelenou, modrou a černou barvu. Tyto barvy jsou definované ve výčtovém typu `COLORS`.

Dále jsou v souboru `image_processing.h` definované různé operátory, pro provedení konvoluce.

7.4 Zobrazení výsledků

Tato část se zabývá zobrazením výsledků zpracování obrazu a pro základní bezdrátové ovládání robota. Pro řešení této části bylo zvoleno vytvoření Ac-

cess Pointu (AP) a HTTP serveru, který je spuštěn na ESP32-cam. Nejdříve je nutné se připojit k AP, kdy název sítě je **WIFI-CAR** a heslo je **12345678**. Poté je nunté se připojit na HTTP server, kdy IP adresa serveru je 192.168.4.1. Po připojení k serveru se zobrazí jednoduché menu viz obr. 7.2.



Obrázek 7.2: Web server

Zde je okno pro zobrazení výsledku zpracování obrazu. Jsou zde uvedena dvě tlačítka `ON` a `OFF`, která slouží k zapnutí/vypnutí jízdy robota. Poslední částí je slider pro nastavení intenzity přísvitu zabudovanou LED diodou.

Jelikož síťování není hlavním předmětem této práce bylo využito již existujícího řešení [21]. Úpravy řešení byly provedeny pouze ve vzhledu a ovládání jízdy robota.

Implementace je uvedena v souborech `HTTP_server.h`, `HTTP_server.cpp` a bylo využito knihoven `WiFi.h`, `AsyncTCP.h`, `ESPAsyncWebServer.h`.

Uživatel může využít HTTP serveru. Stačí tento server inicializovat funkcí `setup_wifi_server()` a výsledek zpracování obrazu odeslat pomocí funkce `send_camera_picture(img_t *img)`, která jako parametr přijmá ukazatel na proměnou datového typu `img_t`.

Tato část práce, ale slouží především pro snazší debugování programu.

7.5 Ovládání motorů

Pro implementaci řízení je nutné mít možnost ovládat motory robota. Jak již bylo popsáno výše 6.2.2 ovládání motorů probíhá pomocí můstku MX1508. Je možné ovládat rychlost motorů pomocí pulzní šířkové modulace (PWM). Za tímto účelem byla využita knihovna `driver/ledc.h`, která umožňuje nastavení pinů schopných PWM.

Implementace ovládání motorů je obsažena v souborech `motor_control.h` a `motor_control.cpp`.

Nastavení pinů probíhá pomocí funkce `motors_init()`, ve které dojde k nastavení frekvence a rozlišení PWM a nastavení jednotlivých pinů. Pro tento účel byly vybrány GPIO piny 12, 13, 14 a 15. Frekvence je definována na 1 kHz a rozlišení na 8 bitů.

Ovládání motorů je implementováno pro pohodlí uživatele pomocí těchto funkcí, kdy některé jako parametry přijímají rychlost, type motoru a směr otáčení motorů a některé pouze rychlost nebo rychlost spolu s typem motoru.

- `motor_run(MOTOR_TYPE motor, MOTOR_DIRECTION direction, int speed)`
- `motor_forward(MOTOR_TYPE motor, int speed)`
- `motor_back(MOTOR_TYPE motor, int speed)`
- `motor_stop(MOTOR_TYPE motor)`
- `move_forward(int speed)`
- `move_back(int speed)`
- `rotate_right(int speed)`
- `rotate_left(int speed)`
- `stop()`

Rychlost motorů je nastavitelná v rozmezí od 0, kdy se jedná o nulovou rychlost až do 255, kdy se jedná o plnou výkon motorů. Pro ulehčení využívání uživatelem byl vytvořen výčet `MOTOR_SPEED`, který obsahuje definované nejběžnější rychlosti. Dále byl také vytvořen výčet `MOTOR_TYPE`, který obsahuje definici jednotlivých motorů.

Poslední dvě zde implementované funkce jsou `set_movement_enabled()` a `get_movement_enabled()`, které upravují proměnnou `movement_enabled`, která zapíná a vypíná jízdu robota.

7.6 Řízení

Nejprve bude popsána jízda a detekce zatáček/křižovatek. Implementace těchto částí je uvedena v souborech `driving.h` a `driving.cpp`

Nežli dojde k popsání dříve uvedených částí je nutné uvést funkci, která slouží pro prvotní nastavení řízení, motorů a zpracování obrazu. Touto funkcí je funkce `setup_driving(bool debug)`, která jako parametr přijmá pravdivostní hodnotu, která slouží pro zapnutí debugovacího nástroje, kterým je HTTP server. Tento server byl popsán v kapitole 7.4.

7.6.1 Jízda robota, detkce zatáček/křižovatek

Jízda robota je realizována pomocí dříve zmíněného ovládání motorů, které je popsáno v kapitole 7.5. Bohužel, při testování jízdy robota došlo k některým zjištěním, které jsou blíže popsány v podkapitole 8.1.2. Z těchto důvodů bylo realizováno ovládání motorů pomocí časovače, aby bylo možné dané motory spustit na určitou dobu a nedošlo k blokování dalších procesů. Pro využití časovače je nutné použít knihovnu `esp_timer.h`. K tomuto účelu slouží funkce `motor_timer()`, která je volána každou 1 milisekundu.

Aby robot mohl sledovat předem stanovenou dráhu, tak se u navádění pracuje jak s barevným tak černobílým obrazem. Obrazy mají rozlišení 125×75 pixelů. Jedná se o nestandartní rozlišení. Toto rozlišení bylo zvoleno v závislosti na přibližné šířce čáry v obraze, která je 25 pixelů. Šířka čáry v obraze byla zjištěna testováním. Skutečná šířka čáry je 2 cm a velikost obrazového pole je při tomto rozlišení 10×6 cm. Díky tomuto rozlišení je možné přesněji detekovat čáru. Detkce čáry bude popsána později v části Detekce čáry. Při detekci čáry a zatáček/křižovatek je využito černobílého obrazu, naopak při detekci barevných příkazů v obraze je využito snímaného obrazu ve formátu RGB565, který je následně pro potřeby detekce převeden do formátu HSL. Tento převod je realizován především pro snazší detekci předem stanovených barev. Tato detkce je blíže popsána v kapitole 7.3;

Detekce barevných příkazů

Detekce příkazů probíhá v celém obraze, jelikož je velikost zorného pole kamery pouze 10×6 cm. Při této detekci se zvýšuje šance na zaznamenání daného příkazu. Pro detekci barev slouží dříve popsaná funkce `find_colors`, která vrací nejvíce přítomnou barvu v obraze. Na tomto základě jsou pak definovány dva barevné příkazy, které jsou vykonány v závislosti na vrácené hodnotě této funkce. Pro barevné příkazy byla zvolena modrá a červená barva. Modrá barva realizuje příkaz pro započnutí jízdy a červená barva

naopak příkaz pro zastavení. Doporučený tvar příkazu je čtverec o velikosti stran 2 cm, ale může se jednat i o jiný tvar. Nejlepší umístění příkazu je přímo na sledované čáře, jelikož je nejmenší pravděpodobnost, že by robot daný příkaz nezaznamenal.

Detekce čáry

Detekce čáry v obraze probíhá z černobílého obrazu tak, že se obraz rozdělí do pěti oblastí o rozměrech 25×75 pixelů. V každé oblasti se provede součet jasu a je vypočtena pravděpodobnost výskytu čáry v této oblasti. Následně se vybere blok s největší pravděpodobností, ale zároveň musí blok splňovat pro uznání minimální hranici pravděpodobnosti výskytu čáry. Pro tuto implementaci slouží funkce `sum_brightness(img_t *img, int sector, DETECTION_TYPE type)` a `drive_straight()`.

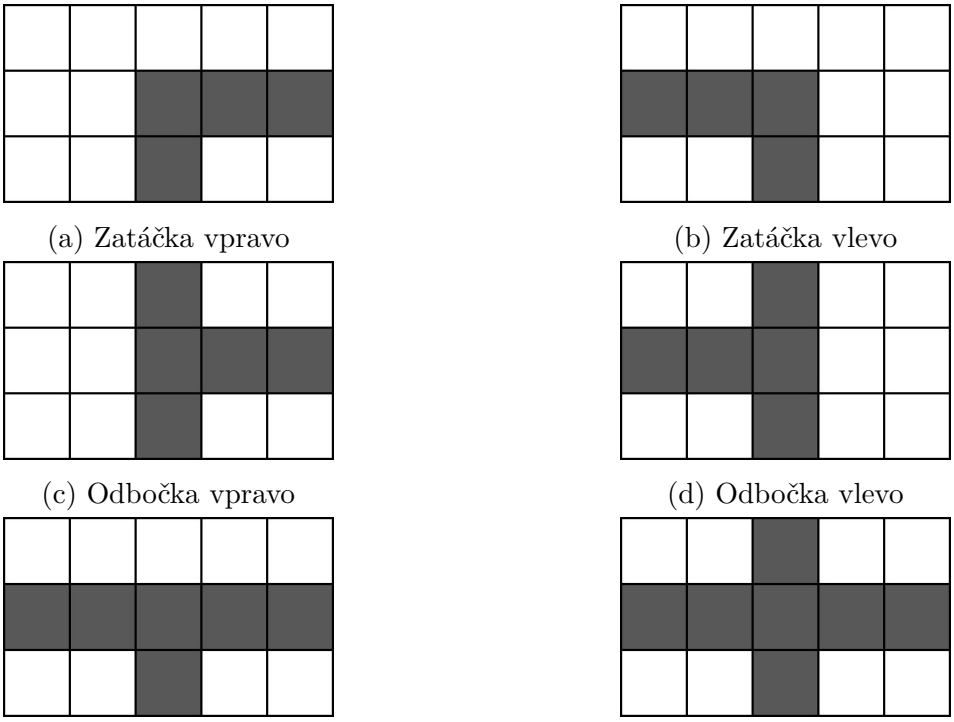
Funkce `sum_birghtness(img_t *img, int sector, DETECTION_TYPE type)`, slouží pro součet jasu v daném bloku, přijímá tedy jako parametry černobílý obraz, číslo sektoru a typ detekce. Číslo sektoru je uvedeno ve výčtovém typu `SECTOR` a detekční typ ve výčtovém typu `DETECTION_TYPE`. Detekční typ je zde pro určení velikosti bloků, jelikož tato funkce je využita i pro součet jasů při detekci křižovatek, která bude popsána později v části Detekce zatáček/křižovatek.

Pro výsledné upravení výkonu motorů slouží funkce `drive_straight()`, kde pomocí funkce `calc_probability(double value,double max_value)` je vypočtena pravděpodobnost výskytu čáry. Tato funkce přijímá jako parametry zjištěnou hodnotu jasu v dané oblasti a maximální možnou hodnotu jasu v téže oblasti a vrací vypočtenou pravděpodobnost. Následně z vypočtených pravděpodobností je nalezena nejvýšší hodnota pomocí funkce `find_max(int *values, size_t len)`, která jako parametry přijímá pole hodnot a jeho délku a vratí index, na kterém se nachází nejvyšší hodnota. Následně je podle nalezeného indexu určena odchylka PID regulátoru a vypočtena akční veličina, která slouží k úpravě výkonu motorů. Blíže je PID regulátor posán v podkapitole 7.6.2. Pokud dojde k vyjetí robota mimo definovanou dráhu robot zastaví.

Detekce zatáček/křižovatek

Detekce zatáček/křižovatek funguje na stejném principu jako detekce čáry a je využito i stejných metod pro určení součtu jasu v dané oblasti a určení pravděpodobnosti výskytu čáry. Reálný rozdíl, ale je v dělení obrazu, který je v tomto případě rozdělen do 15 oblastí o velikosti 25×25 pixelů. Dojde k rozdělení obrazu do matice 5×3. Následně jsou v takto rozděleném obraze hle-

dány zatáčky a křižovatky, které jsou definovány pomocí pravdivostních map. Pro toto vyhledání slouží funkce `find_path_object(img_t *img)`, která vrací nalezený objekt, který je definován ve výčtovém typu `PATH_OBJECTS`. Tato metoda jako parametr přijmá černobílý obraz. Pokud je nalezen nějaký z definovaných objektů dojde k provedení dané zatáčky/křižovatky. Tyto zatáčky a křižovatky musí mít úhel 90°, jinak nemusí být detekovány. Pro tento účel slouží funkce `make_turn(PATH_OBJECTS object)`, které provede dané otočení robota. Toto řešení podporuje několik předem definovaných typů křižovatek, jelikož není možné pokrýt všechny existující typy. Podporované typy křižovatek a zatáček jsou vidět na obrázku 7.3;



(a) Zatáčka vpravo

(b) Zatáčka vlevo

(c) Odbočka vpravo

(d) Odbočka vlevo

(e) Křižovatka ve tvaru T

(f) Křižovatka ve tvaru kříže

Obrázek 7.3: Typy podporovaných zatáček a křižovatek.

Jelikož není možné ovlivnit na křižovatkách ve tvaru kříže a T kam robot odbočí pomocí grafických příkazů. Bylo tedy předem stanoveno, že na těchto křižovatkách robot odbočí doprava.

Funkcí, která zastřešuje všechny tyto jednotlivé úkony a realizuje tak jízdu robota po čáře je funkce `drive_on_path(bool detect_colors)`. Tato funkce přijmá jako parametr pravdivostní hodnotu, která určuje zdali bude robot reagovat na barevné příkazy.

7.6.2 PID regulátor

Pohon robota zajišťují dva motory, kterými lze pomocí regulace otáček ovlivňovat do jisté míry směr jízdy. Aby bylo možné tuto možnost realizovat je nutné definovat určité odchylky. Pro tento účel je obraz snímaný kamerou rozdělen do pěti částí pro které jsou definované dané odchylky, které obsahuje množina e. Pomocí těchto definovaných odchylek a PID regulátoru s diskrétním krokem, je možné ve výsledku upravovat jednotlivé otáčky motorů. PID regulátor obsahuje, proporcionální, integrační a derivační složku. Tyto složky jsou definovány určitými vztahy. Proporcionální složka je definována vztahem:

$$P(k) = e(k) \tag{7.1}$$

integrační složka vztahem:

$$I(k) = I(k-1) + e(k) \tag{7.2}$$

a derivační složka vztahem:

$$D(k) = e(k) - e(k-1) \tag{7.3}$$

kde $k \in N_0$ je pořadí diskrétního kroku a $e \in \{-6; -4; 0; 4; 6\}$ je daná odchylka. Kladné hodnoty odchylky znamenají, že robot se nachazí vlevo od sledované čáry. Za pomoci dříve definovaných vztahů je nutné vypočíst akční veličinu podle vztahu:

$$u(k) = K_p * P(k) + K_i * I(k) + K_d * D(k) \tag{7.4}$$

kde K_p je zesílení proporcionální složky, K_i zesílení integrační složky a K_d zesílení derivační složky [36].

Jednotlivé zesilovací členy, byly stanoveny experimentálně při testování. Zesílení proporcionální a derivační složky bylo nastaveno na hodnotu 7.5. Co se týče integrační složky ta je nastavena na hodnotu 0.1, ale součástí výpočtu je pouze tehdy, když je čára poblíž středu, tedy pokud jsou hodnoty odchylky -4, 0 a 4. Toto nastavení pro ověření navrženého řešení je dostačující, ale není vyloučeno, že existuje i lepší nastavení jednotlivých složek zesílení.

Pro realizaci regulátoru bylo definováno několik funkcí. První z nich je funkce `calc_pid(double error)`, která jako parametr přijímá dříve definované hodnoty z množiny odchylek e. Tato funkce vrací vypočtenou akční veličinu. Další funkcí, která se týká regulátoru je funkce `reset_pid()`, která slouží pro resetování PID regulátoru.

7.7 Program

Výsledný program se ve výsledku skládá z těchto zdrojových souborů:

- `cam.h`
- `cam.cpp`
- `img.h`
- `img.cpp`
- `image_processing.h`
- `image_processing.cpp`
- `motor_control.h`
- `motor_control.cpp`
- `HTTP_server.h`
- `HTTP_server.cpp`
- `driving.h`
- `driving.cpp`
- `return_values.h`
- `return_values.cpp`

Soubor `main.cpp` je vstupním bodem programu. Tento soubor obsahuje funkci `setup()` a funkci `loop()`. V tomto souboru uživatel naimportuje jeden ze souborů představující API. Nasledně ve funkci `setup()` zavolá inicializační funkci daného API. Uživatel v té samé funkci zavolá funkci pro řízení, nebo ve funkci `loop()` realizuje zpracování obrazu jak bylo popsáno výše v kapitole o snímání obrazu 7.2 a kapitole o zpracování obrazu 7.3.

Jsou zde také soubory `return_values.h` a `return_values.cpp`, které definují návratové chybové kódy. Zároveň je zde také definovaná funkce pro výpis chybového hlášení a daného kódu.

7.8 API

Cílem práce bylo vytvořit API pro platformu EPS32-cam, které implementuje dříve definované metody zpracování obrazu 4. Výsledná práce ve výsledku realizuje dvě API. Jedno API, které implementuje metody zpracování obrazu a je nazváno zpracování obrazu. Druhé API, které nazveme řízení implementuje základní funkce řízení.

K realizování více API došlo z důvodu umožnit případným uživatelům buď to samostatnou práci pouze s metodami zpracování obrazu, či možnost implementovat již realizované řízení na základě metod zpracování obrazu.

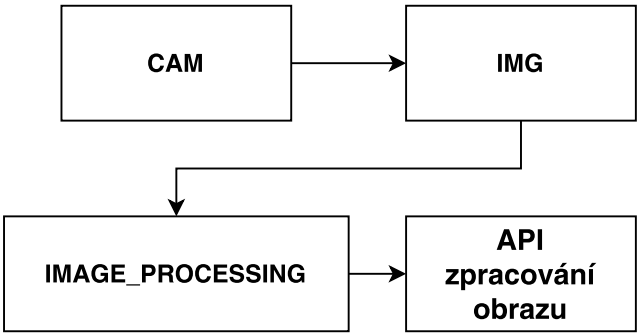
7.8.1 API - zpracování obrazu

API - zpracování obrazu se skládá z několika zdrojových souborů, kterými jsou

- `cam.h`
- `cam.cpp`
- `img.h`
- `img.cpp`
- `image_processing.h`
- `image_processing.cpp`

Soubor, který bude budoucí uživatel implementovat a představuje API pro zpracování obrazu je soubor `image_processing.h`.

Struktura tohoto API je znázorněna na obr. 7.4.



Obrázek 7.4: Diagram API - zpracování obrazu

API pro zpracování obrazu umožňuje práci s dříve zmíněnými metodami zpracování obrazu, které jsou uvedeny v kapitole 4.

Pokud tedy uživatel chce využít tohoto API pro zpracování obrazu musí nejdříve zavolat funkci `setup_image_processing(FRAMESIZE frame_size, pixformat_t pixel_format)`, která přejímá jako parametr rozlišení obrazu výstupní formát dat. Funkce vrací ukazatel na alokovanou proměnnou představující uložený obraz z kamery a zároveň inicializuje kameru.

Následně již uživateli stačí volat v funkci `camera_capture(img_t *img)`, která získá data z kamery a ukládá je do proměnné, na kterou byl předaný ukazatel jako parametr. Dále již může využít jednu z definovaných funkcí pro zpracování obrazu:

- threshold
- double-threshold
- semi-threshold
- adaptiv-threshold
- convolution
- hough transform
- skeletonization

Blíže jsou jednotlivé metody popsány v okomentovaném kódu.

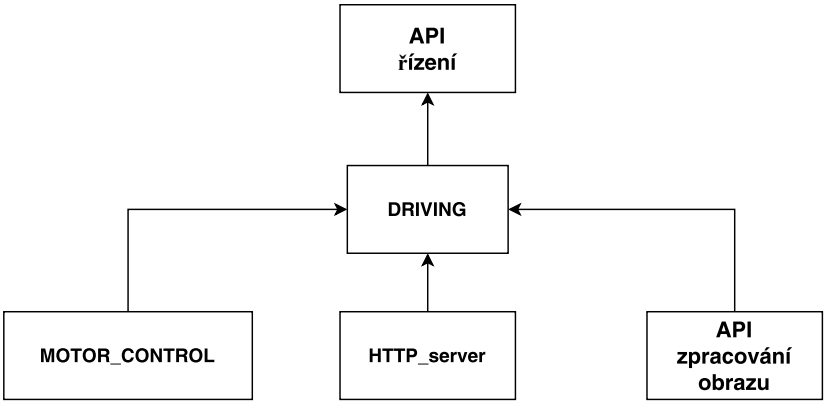
7.8.2 API - řízení

API - řízení se skládá z několika zdrojových souborů a API pro zpracování obrazu, kterými jsou

- `motor_control.h`
- `motor_control.cpp`
- `HTTP_server.h`
- `HTTP_server.cpp`
- `driving.h`
- `driving.cpp`

Soubor, který bude budoucí uživatel implementovat a představuje API pro řízení robota je soubor `driving.h`.

Struktura tohoto API je znázorněna na obr. 7.4.



Obrázek 7.5: Diagram API - řízení

API pro řízení robota poskytuje budoucímu uživateli výchozí bod, pro řízení robota založené na metodách zpracování obrazu.

Uživatel musí nejdříve zavolat funkci `setup_driving(bool debug)`, která inicializuje vše potřebné pro ovládání robota například kameru a motory. Tato funcke také přijímá jako parametr pravdivostní hodnotu, která určuje zdali bude zapnutý HTTP server, který slouží pro vizuální kontrolu a snazší ovládání robota.

Následně již uživateli stačí zavolat definovanou funkci realizujících řízení. Jedná se o funkci `drive_on_path(bool detect_colors)`, která realizuje ježdění po čáře a dokáže reagovat na křižovatky. Tato funkce přijímají jako parametr pravdivostní hodnotu pomocí, které je ovlivňena reakce robota na barevné příkazy v rovných úsecích sledované čáry.

8 Zhodnocení vlastností

Tato kapitola se zabývá zhodnocením vlastností realizovaného řešení a osvětlením odhalených problémů při této realizaci.

8.1 Ověření funkčnosti

Ověření funkčnosti můžeme rozdělit do dvou částí. První se týká jednotlivých metod zpracování obrazu. Druhé se věnuje ověření funkčnosti řízení.

8.1.1 Funkčnost metod zpracování obrazu

Co se týká výsledků metod zpracování obrazu byl výsledek jednotlivých metod kontrolován vizuálně a také byl změřen přibližný čas běhu tohoto zpracování v milisekundách pomocí funkce `millis()`. Aby se docílilo přesnějších výsledků bylo provedeno 40 samostatných měření a následně z těchto jednotlivých měření byla vypočtena průměrná hodnota. Následně pro lepší prezentaci výsledků byl z průměrných hodnot vypočten počet snímků za sekundu. Nežli budou uvedeny naměřené hodnoty je nutné uvést, že průměrný počet snímků za sekundu u zachycení a uložení obrazu z kamery je přibližne 15 fps. Ať již se jedná o snímání barevného či černobílého obrazu. V tabulce 8.1 je uveden průměrný počet snímků za sekundu pro jednotlivé metody a jejich kombinace, jelikož například pro skeletonizaci je vyžadován prahovaný obraz.

Metoda zpracování obrazu	FPS
Prahování	15
Prahování (automatické určení prahu)	16
Dvojité prahování	15
Poloprahování	15
Adaptivní prahování	15
Konvoluce (Sobelův operátor)	16
Prahování + Konvoluce	16
Prahování + Konvoluce + Houghova transformace	0.24
Prahování + Skeletonizace	15

Tabulka 8.1: Průměrný počet snímků za sekundu v závislosti na metodě zpracování obrazu

Jak je možné pozorovat v naměřených datech, tak většina metod má minimální vliv na výkon až na houghovu transformaci. To je způsobeno její náročností, jelikož od ostatních metod, které upravují přímo obraz a pracují na základě dat v obraze. Tak u Houghovy transformace, dochází k nárustu zpracovaných dat, především při mapování dat ze souřadného systému obrazu do Houghova prostoru.

Při testování prahování byl ověřen předpoklad, že u prahování je častým problémem vliv okolního osvětlení viz obr. 8.1.

(a) Vliv vnějšího osvětlení

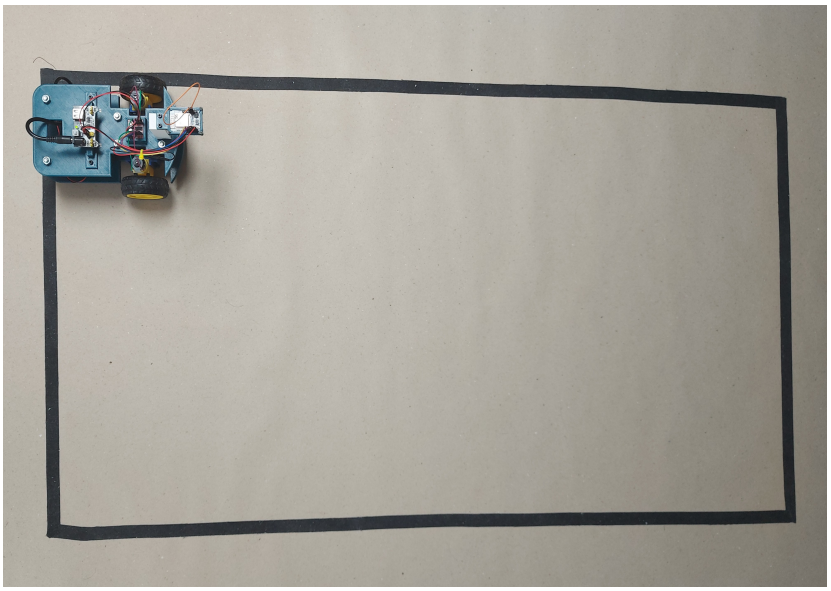
(b) Vliv stínění robotem

Obrázek 8.1: Problémy prahování

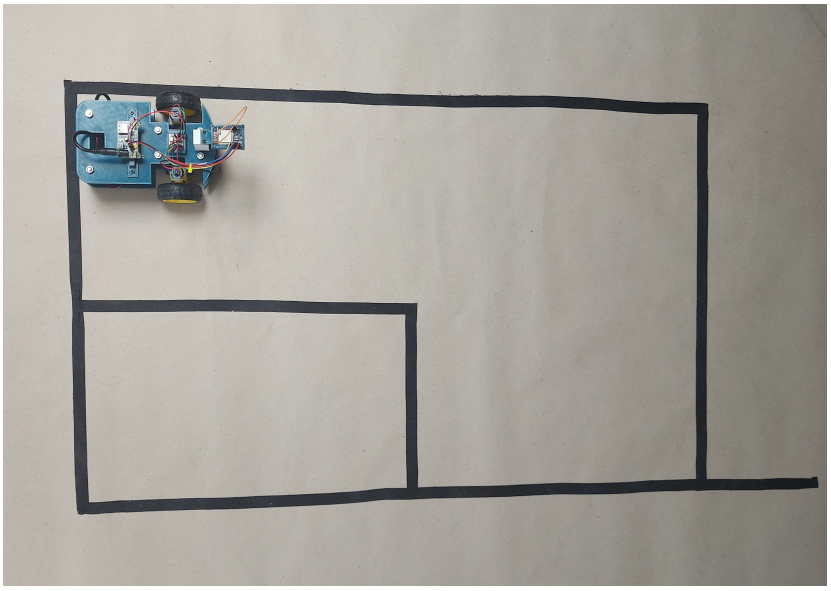
V tomto případě je problém ve vrhání stínu ramenem robota, případně samotným robotem. Stejně tak při velmi silném osvětlení scény, není skoro možné hledanou čáru vůbec rozeznat. Ani automatické určování prahu není zcela efektivním řešením tohoto problému.

8.1.2 Funkčnost řízení

Funkčnost realizovaného řízení robota byla testovánána na dvou vytvořených testovacích drahách viz obr. 8.2 Tyto testovací dráhy byly vytvořeny z hnědého balícího papíru a černé pásky.



(a) 1.testovací dráha



(b) 2.testovací dráha

Obrázek 8.2: Testovací dráhy

Při testování bylo zjištěno, že pro rychlou a plynulou jízdu robota po čáře je průměrný počet snímků za sekundy přílíš nízký. Jedná se o dříve

zmíněnou hodnotu 15 FPS. Při rychlé jízdě, která je přibližně 0,1 $m \times s^{-1}$ docházelo k vyjetí robota mimo čáru a k nedetekování zatáčky/křižovatky, z tohoto důvodu bylo přistoupeno k ovládání motorů pomocí timeru. Díky, kterému lze nastavit dobu po kterou mají motory běžet nezávisle na době zpracování obrazu. Nynější rychlost jízdy robota se nyní pohybuje okolo 0,3 $m \times s^{-1}$. Další problém byl také v nastavení rychlosti motorů, kdy při určité nízké hodnotě výkonu vybrané motory neměli dostatečný výkon pro rozjetí. Doporučoval bych zvážit změnu využitých motorů pro realizaci práce.

Pro detekci čáry byla nejprve zvažována metoda prahování. Kvůli dříve popsaným problémům této metody bylo od této metody upuštěno a nyní je navádění robota čistě závislé na hodnotách jasu v obraze. Tento přístup umožnil lepší detekci i za mírně zhoršených podmínek. Při velkém okolním osvětlení je možné, že stín robota ovlivní detekci zatáček/křižovatek.

Co se týče detekce barev je tímto jevem, také ovlivněna. Proto občas dojde k falešné detekci určité barvy. Jelikož je pomocí barev umožněno ovládat robota dojde k nežádoucímu vykonání určité činosti. Tento problém také nastává při zapnutí robota, jelikož dochází k automatickému vyrovnání jasu kamerou a zaznamenaný obraz může být tímto ovlivněn. Volba podkladu dráhy tedy hnědý balící papír nebyl nejlepší volbou, protože často dochází k chybné detekci příkazu pro jízdu i když se tento příkaz v obraze nenachází. Děje se tak po zapnutí robota. Řešením tohoto problému je změna tohoto podkladu za bílý papír. Barevné příkazy byly vytvořeny z barevných papírů.

Dalším zjištěným problémem v tomto řešení je možné nenalezení čáry v obraze, když se čára nachází mezi jednotlivými bloky, tento problém byl vyřešen využitím poslední známe pozice v obraze. Vhodnějším řešením by bylo definovat jednotlivé bloky tak, že se z poloviny překrývají a došlo by tedy ke spolehlivější detekci čáry.

8.2 Shrnutí

Realizované řešení je vcelku spolehlivé, ale není zcela ideální. Pokud porovnáme podobná řešení, která jsou popsána v kapitole 2. Tak tento přístup realizace je v tomto ohledu unikátní. Jelikož využívá informace ze zpracovaného obrazu obdobně jako je tomu při detekci čáry pomocí IR senzorů. Kdy obraz je rozdělen do oblastí a na základě pravděpodobnosti je rozhodováno zda se v dané oblasti nachází čára. Kombinací jednotlivých bloků, jsou definovány tvary zatáček/křižovatek.

U tohoto řešení jsou dva hlavní problémy, které souvisí se zvolenou deskou ESP32-cam a kamerou OV2640. První problém vidím v pomalém získání ob-

razu z kamery, který značně omezuje rychlost jízdy robota. Druhým problémem je velká absence pokročilých knihoven pro zpracování obrazu pro desku ESP32-cam. Tyto knihovny by umožňovali lepší práci s obrazem. Proto bych doporučoval pro podobné řešení zvolit výkonější HW například Raspberry Pi, kde je možné lépe pracovat s obrazem, díky větší podpoře knihoven.

9 Závěr

Cílem této práce bylo prostudovat možnosti platformy ESP32-cam a metody pro zpracování obrazu, které kladou důraz na rozpoznání čáry a jednoduchých základních objektů. Následně pro tyto metody vytvořit API, za pomoci kterého by bylo implementováno základní sledování čáry.

Ve výsledné práci byly zmíněny i základní barevné formáty, ze kterých byly využity formáty RGB a HSL. Následně byly implementovány metody zpracování obrazu: prahování, konvoluce, Houghova transformace a skeletonizace. Tyto metody byly otestovány a došlo k určitým negativním zjištěním ohledně časové náročnosti Houghovy transformace. Jelikož tyto metody tvoří výše zmiňované API, byl tento bod zadání splněn. Byla také realizována detekce barev v obraze, díky kterým bylo možné implementovat barevné příkazy pro řízení robota. Ve výsledku po konzultaci s vedoucím práce bylo vytvořeno ještě druhé API, které rošiřuje předchozí API o funkce jízdy a detekci křižovatek.

Co se týče hardwarové realizace, zde bylo nejnáročnější navrhnout samotnou kostru robota, která prošla mnohými iteracemi vývoje, aby bylo dosaženo co nejlepšího výsledného modelu. Všechny výsledné modely byly vytvořeny v programu Fusion 360 a vytištěny na 3D tiskárně Ender 3 Pro. Výběr vhodné kamery po porovnání jednotlivých typů byl vcelku očividný a byla vybrána kamera OV2640.

V kapitole o zhodnocení výsledků byly zmíněny jednotlivé zjištěné problémy, které tuto práci doprovázely. Po zhodnocení těchto zjištěních bylo doporučeno zvolit výkonnější a více podporovanou platformu, například Raspberry Pi. Dalším vylepšením této práce by mohla být úprava programu na objektově orientovaný. Tato změna by mohla zpřehlednit výsledný kód programu a usnadnit práci s některými definovanými datovými typy, které by mohly být specifikovány vlastní třídou.

Literatura

[1] Arducam.com. *ArduCAM-M-2MP Camera Shield* [online]. [cit. 2022-12-11]. Dostupné z: https://www.arducam.com/downloads/shields/ArduCAM_Mini_2MP_Camera_Shield_Hardware_Application_Note.pdf.

[2] Arducam.com. *ArduCAM-M-5MP Camera Shield* [online]. [cit. 2022-12-11]. Dostupné z: https://www.arducam.com/downloads/shields/ArduCAM_Mini_5MP_Camera_Shield_DS.pdf.

[3] Arducam.com. *Arducam 1MP NT99141 1/4" HD Color CMOS Sensor OEM Camera Module UC99141-A* [online]. [cit. 2022-12-11]. Dostupné z: https://www.arducam.com/product/1-4-hd-nt99141-cmos-sensor-standalone-camera-uc99141-a/.

[4] Arducam.com. *OV2640 – Specs, Datasheets, Cameras, Features, Alternatives* [online]. [cit. 2022-12-11]. Dostupné z: https://www.arducam.com/ov2640/#ov2640-specs.

[5] Arducam.com. *Arducam OV7670 Camera Module, VGA Mini CCM Compact Camera Modules Compatible with Arduino ARM FPGA, with DVP 24 Pin Interface* [online]. [cit. 2022-12-11]. Dostupné z: https://www.arducam.com/product/arducam_ov7670_camera_module_vga_mini_ccm_compact_camera_modules_m0030/.

[6] a.s., P. R. *Prusament* [online]. [cit. 2022-04-05]. Dostupné z: https://www.prusa3d.com/cs/kategorie/prusament/?page=2.

[7] Baletka, T. Pokročilá segmentace obrazu pro 3D zobrazení, 2012. [cit. 2022-12-11] Diplomová práce, Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií. Ústav radioelektroniky.

[8] Baltz, A. *NTSC vs PAL: What are they and which one do I use?* [online]. [cit. 2023-01-05]. Dostupné z: https://learn.corel.com/ntsc-vs-pal-what-are-they-and-which-to-use/.

[9] CIRCUIT DIGEST. *Building an easy Line Follower Robot using Arduino Uno* [online]. [cit. 2023-01-11]. Dostupné z: https://circuitdigest.com/microcontroller-projects/arduino-uno-line-follower-robot.

[10] Corporation, P. *Pololu Ball Caster with 3/4 Metal Ball* [online]. [cit. 2022-04-05]. Dostupné z: https://www.pololu.com/product/955.

[11] DOCS.ESPRESSIF.COM. *Arduino-ESP32 2.0.6 documentation* [online]. [cit. 2022-04-22]. Dostupné z: https://docs.espressif.com/projects/arduino-esp32/en/latest/index.html.

[12] GENERAL ELECTRONICS TECHNOLOGY CO., L. *Li-ion Cylindrical Battery Specification* [online]. [cit. 2022-04-05]. Dostupné z: https://www.laskakit.cz/user/related_files/geb_18650_3_7v_2600mah_-_datasheet.pdf.

[13] GERLA, V. – HOZMAN, J. – POP, M. *HSV, HSL* [online]. 2004. [cit. 2023-01-14]. Dostupné z: http://webzam.fbmi.cvut.cz/hozman/Help/Pages/23c.htm.

[14] GONZALEZ, R. C. – WOODS, R. E. *Digital Image Processing (3rd Edition).* Pearson, 2007. ISBN 0-131-68728-X.

[15] KRŇOUL, Z. *Zpracovani digitalizovaného (ZDO) - Segmentace* [online]. [cit. 2022-12-11]. Západočeská univerzita v Plzni, Fakulta Kybernetiky. Dostupné z: https://www.kky.zcu.cz/uploads/courses/zdo/prezentace/ZDO_segmentace_I.pdf.

[16] KUMAR, P. *What is a MIPI Camera? How does MIPI Camera Work?* [online]. [cit. 2023-01-05]. Dostupné z: https://www.e-consystems.com/blog/camera/technology/what-is-a-mipi-camera-how-does-mipi-camera-work/.

[17] LAFVIN. *LAFVIN 4WD Smart Robot Car Kit V2 for Arduino Robot STEM /Graphical Programming Robot Car* [online]. [cit. 2022-04-05]. Dostupné z: https://lafvintech.com/products/lafvin-4wd-multi-robot-car-kit-upgraded-v2-0-for-arduino-robot-stem-graphical-programming-robot-car.

[18] LASKAKIT. *TT motor s převodovkou - plastové převody* [online]. [cit. 2022-04-05]. Dostupné z: https://www.laskakit.cz/tt-motor-s-prevodovkou-plastove-prevody/.

[19] MICROSOFT. *Color* [online]. 2022. [cit. 2023-01-14]. Dostupné z: https://learn.microsoft.com/en-us/windows/win32/uxguide/vis-color.

[20] MISCHIANTI, R. *ESP32-CAM: pinout, specs and Arduino IDE configuration – 1* [online]. 2022. [cit. 2022-12-11]. Dostupné z: https://www.mischianti.org/2021/08/30/esp32-cam-pinout-specs-and-arduino-ide-configuration-1/.

[21] NANDANWAR, U. *Camera Car* [online]. September 2021. [cit. 2022-05-03]. Dostupné z: https://github.com/un0038998/CameraCarWithPanTiltControl.

[22] OmniVision Technologies. *OV2640 Color CMOS UXGA CAMERACHIP* [online]. [cit. 2022-12-11]. Dostupné z: https://www.uctronics.com/download/OV2640_DS.pdf.

[23] Opencircuit.shop. *YwRobot Breadboard Power Supply MB-V2* [online]. [cit. 2022-04-05]. Dostupné z: https://cdn.bluecommerce.shop/media/1/8ff1620_breadboard-power-supply.pdf.

[24] Out of the BOTS. *Line Follower using Computer Vision* [online]. 2017. [cit. 2023-01-14]. Dostupné z: https://gist.github.com/flyboy74/3088832dac862b67addcf1611301fec5.

[25] PM, N. – Chezian, D. R. Various colour spaces and color space conversion algorithms. *Journal of Global Research in Computer Science*. January 2013, 4, 1, s. 5. ISSN 2229-371X. Dostupné z: https://www.rroij.com/open-access/pdfdownload.php?aid=37895.

[26] Saad, W. H. M. et al. Line Follower Mobile Robot for Surveillance Camera Monitoring System. *Journal of Telecommunication, Electronic and Computer Engineering*. July 2018, 10, 2-7, s. 5. ISSN 2180-1834. Dostupné z: https://jtec.utem.edu.my/jtec/article/view/4409/3269.

[27] Saetern, C. *MX1508* [online]. September 2019. [cit. 2022-04-05]. Dostupné z: https://github.com/Saeterncj/MX1508.

[28] Straka, S. Segmentace obrazu, 2009. [cit. 2022-12-11] Diplomová práce, Masarykova univerzita, Fakulta informatiky.

[29] Tech Cube. *CCTV Cameras Explained* [online]. [cit. 2023-01-05]. Dostupné z: https://www.techcube.co.uk/blog/cctv-cameras-explained/.

[30] UCTRONICS Store. *ESP32-CAM Module* [online]. [cit. 2022-12-11]. Dostupné z: https://www.uctronics.com/download/esp32-cam-board-specs.pdf.

[31] UCTRONICS Store. *OV5642* [online]. [cit. 2022-12-11]. Dostupné z: https://www.uctronics.com/download/cam_module/OV5642DS.pdf.

[32] VOXCAFE s.r.o. *ESP32-Cam AI-Thinker Pinout* [online]. [cit. 2022-12-11]. Dostupné z: https://www.voxcafe.cz/mindblog/clanky/arduino/esp32-cam-ai-thinker-pinout.html.

[33] Wannartek Camera Module. *Camera Modul Interface Introduction* [online]. [cit. 2023-01-05]. Dostupné z: https://www.wannatek.com/camera-module-interface/.

[34] Zahradník, L. Robot pro sledování čáry pro android, 2020. [cit. 2023-01-11] Bakalářská práce, Západočeská univerzita v Plzni, Katedra informatiky a výpočetní techniky.

[35] Čišecký, R. Metody pro odstranění šumu z digitálních obrazů, 2012. [cit. 2023-01-16] Diplomová práce, Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií. Ústav telekomunikací.

[36] Špicar, K. Mobilní robot Micromouse, 2022. [cit. 2023-05-25] Bakalářská práce, Univerzita Pardubice. Fakulta elektrotechniky a informatiky.

A Popis adresářové struktury příloh

- `/Aplikace_a_knihovny`
  - `/Aplikace`: Program pro ESP32-cam včetně metadat vývojového prostředí platform.io. Vlastní zdrojové kódy se nacházejí v `/src`.
  - `/AsyncTCP, /ESPAsyncWebServer`: Knihovny pro HTTP webserver.
- `/Text_prace`
  - `/BP.pdf`: Zkompilovaná práce.
  - `/src`: Adresář se zdrojovými soubory pro LaTeX
  - `/img`: Adresář obsahující grafiku práce
  - `/pdf`: Adresář obsahující zadání práce
- `/Vysledky`
  - `/Modely`: Vytvořené 3D modely jednotlivých částí robota.
  - `/Videa`: Videa jízdy robota.
- `/Readme.txt`: Popis adresářové struktury příloh