FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Bakalářská práce

Malý mobilní robot se systémem pro analýzu obrazu

Studijní program: B0613A140005 Informační technologie
Studijní obor: Aplikovaná informatika
Autor práce: **Jakub Šikola**
Vedoucí práce: doc. Ing. Josef Chaloupka Ph.D.
Liberec 2023

FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Zadání bakalářské práce

Malý mobilní robot se systémem pro analýzu obrazu

Jméno a příjmení:	**Jakub Šikola**
Osobní číslo:	M21000139
Studijní program:	B0613A140005 Informační technologie
Studijní obor (specializace):	Aplikovaná informatika
Zadávající katedra:	Ústav informačních technologií a elektroniky
Akademický rok:	2023/2024

Zásady pro vypracování:

1. Seznamte se s počítačem Raspberry Pi 4, mikropočítači řady Arduino a s problematikou zpracování a rozpoznávání obrazu.
2. Navrhněte a vytvořte malého pásového robota osazeného kamerou.
3. Vytvořte v PC desktopový program pro řízení vytvořeného robota přes bluetooth.
4. Součástí programu bude i systém pro řízení robota v prostoru, který bude využívat algoritmy z oblasti zpracování a rozpoznávání obrazu.

Rozsah grafických prací:	Dle potřeby dokumentace
Rozsah pracovní zprávy:	30–40 stran
Forma zpracování práce:	Tištěná/elektronická
Jazyk práce:	Čeština

Seznam odborné literatury:

[1] Šonka, M., Hlaváč, V., Boyle, R.: Image processing, analysis, and machine vision. Fourth Edition. Australia: Cengage Learning, ISBN 978-1-133-59369-0, 2015.

[2] Gonzales, R. C., Woods, R: Digital image processing. Global edition. New York: Pearson, ISBN 978-1-292-22304-9, 2017.

[3] Hlaváč, V., Sedláček, M.: Zpracování signálů a obrazů. 2. přeprac. vyd. Praha: ČVUT, 255 s. ISBN 978-80-01-03110-0, 2007.

Vedoucí práce:	doc. Ing. Josef Chaloupka Ph.D. Ústav informačních technologií a elektroniky
Datum zadání práce:	20. října 2023
Předpokládaný termín odevzdání:	14. května 2024

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

prof. Ing. Ondřej Novák, CSc.
vedoucí ústavu

V Liberci 20. října 2023

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval/a samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Jsem si vědom/a toho, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom/a povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má bakalářská práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom/a následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

13.05.2024 Jakub Šikola

Abstrakt

Tato práce se zabývá stavbou pásového robota osazeného kamerou společně s mikropočítači Arduino a Jetson Nano. Na začátku jsou popsány použité algoritmy z oblasti zpracování obrazu a rozpoznávání objektů a také řídící a pohybové komponenty robota. Dále je popsána konstrukce robota a schéma jeho zapojení. Schéma řeší napájení všech komponent a komunikaci mezi nimi.

V další části je popsán řídící systém robota. Jednou z částí systému je desktopová aplikace, která slouží pro komunikaci s robotem. Druhou částí je program běžící na platformě Jetson Nano, který se stará o získávání a zpracování obrazu. V této části práce je i porovnán výkon Raspberry Pi 4 oproti Jetson Nano při úloze rozpoznávání objektů. Poslední částí řídícího systému je program běžící na mikropočítači Arduino Uno, který zajišťuje ovládání motorů a zpracování dat z dotykových senzorů.

Abstract

This thesis focuses on the construction of a tracked robot equipped with a camera, alongside Arduino and Jetson Nano microcontrollers. Initially, the algorithms used in image processing and object recognition are described, as well as the control and motion components of the robot. Furthermore, the robot's construction and its wiring diagram are outlined. The wiring diagram addresses the power supply for all components and the communication between them.

In the next section, the robot's control system is described. One part of the system is a desktop application used for communication with the robot. The second part consists of a program running on the Jetson Nano platform, responsible for image acquisition and processing. This section also compares the performance of the Raspberry Pi 4 to the Jetson Nano in image recognition tasks. The final part of the control system is a program running on the Arduino Uno microcontroller, which controls the motors and processes data from touch sensors.

Poděkování

Rád bych poděkoval doc. Ing. Josefu Chaloupkovi Ph.D. za konzultace a užitečné připomínky při tvorbě bakalářské práce. Dále patří mé díky Ing. Martinu Rozkovci, Ph.D. za zapůjčení počítače Jetson Nano.

Obsah

Seznam obrázků

Úvod

Mobilní roboti nabízejí široké možnosti využití. Ve firmách je lze naleznout nejčastěji ve skladech, kde jsou využíváni pro manipulaci se zbožím a materiálem, avšak mohou být integrováni i do montážních linek. Jejich přínosem je zvýšená produktivita, konzistence a dlouhodobé snížení nákladů. Využití těchto robotů lze najít i v zemědělství, kde pomáhají s monitorováním plodin nebo s jejich sázením. Dále se využívají ve zdravotnictví, ve kterém asistují při péči o pacienty nebo se starají o distribuci léků. Výzkumníci zase využívají roboty pro průzkum nebezpečných prostředí, sběr dat nebo monitorování životního prostředí.

V dnešní době se již moderní technologie stávají součástí každodenních životů. V domácnosti roboti pomáhají hlavně s úklidem, a to například ve formě vysávání nebo čištění oken. Na zahradě zase lze využít robotické sekačky. Tím ale využití robotů v domácnosti nekončí. Existují například roboti pomáhající lidem s omezenou mobilitou, kteří jim umožňují přiblížit se k normálnímu každodennímu životu.

Cílem této práce je navrhnout a sestrojit robota ovládaného za pomocí mikropočítačů Arduino a Jetson Nano. Robot bude osazen kamerou, díky které bude získávat obraz, který se bude dále zpracovávat. V rámci práce bude vytvořena desktopová aplikace vytvořená pomocí frameworku PyQt5, která bude uživateli umožňovat ovládání robota pomocí grafického rozhraní. Tato aplikace bude s robotem komunikovat pomocí bluetooth.

Jednou z funkcí robota bude automatické vyhledávání objektů pomocí algoritmu YOLO. Ten poběží na počítači Jetson Nano a jeho výsledky budou zpracovávány za pomoci Arduina. V rámci práce bude sestrojena i nabíjecí stanice, do které robot samostatně zajede. K vyhledání této stanice budou použity algoritmy z oblasti zpracování obrazu.

1 Využité metody pro zpracování obrazu

1.1 Morfologické operace

Morfologické operace jsou nelineární operace, které mění tvar objektů v binárním obraze (obraz obsahující pixely pouze se dvěma různými hodnotami). Jsou to jedny ze základních technik v oblasti zpracování obrazu. Morfologické operace spolu úzce souvisí se segmentací obrazu a to tak, že se často aplikují buď na vstupní data segmentace nebo na její výsledky. Další využití nalézají morfologické operace při odstraňování šumu, zvýrazňování hran nebo filtraci obrazu. [1]

Výsledky těchto operací vždy závisí jak na vstupním obraze, tak na strukturním elementu. Strukturní element je malý obrazový vzor, jehož velikost a tvar určuje rozsah úprav vstupního obrazu.

1.1.1 Dilatace

Dilatace je první ze dvou základních morfologických operací, při které se střed strukturního elementu postupně položí nad každý pixel vstupního obrazu. Pokud se alespoň jeden pixel obrazu pod některým jedničkovým pixelem strukturního elementu rovná jedna, pak je pixel ve výsledném obrazu nastaven také na jedna. Dilatace se využívá k eliminaci izolovaných děr, zvětšení objektů na obraze a vyhlazení jejich hran. To může být užitečné například při segmentaci obrazu. [2]

1.1.2 Eroze

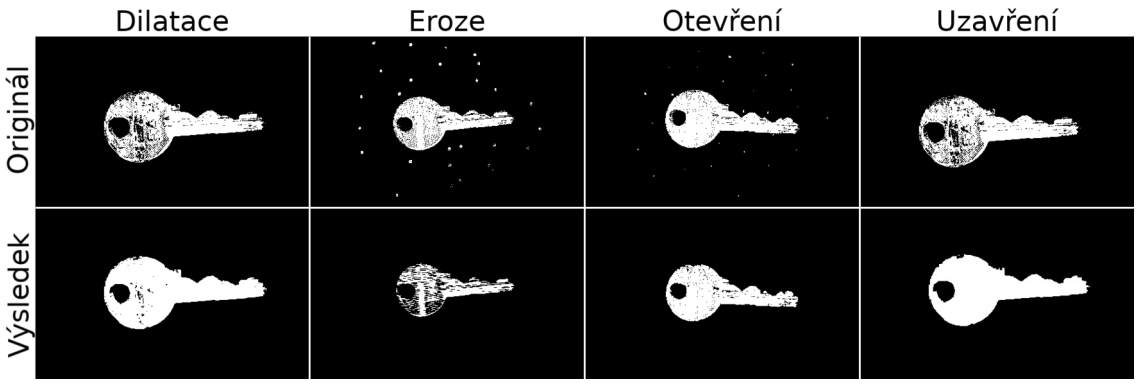
Druhá základní morfologická operace se nazývá eroze. Při ní se podobně jako u dilatace posouvá střed strukturního elementu přes všechny pixely vstupního obrazu. Aby se výsledný pixel rovnal jedna, musí pod každým jedničkovým pixelem strukturního elementu ležet pixel obrazu rovnající se jedna.

Eroze zmenšuje obrysy a maže malé izolované objekty. Pokud od původního obrazu odečteme jeho erodovanou verzi, dostaneme obrysy objektů nacházejících se na původním obrazu. Dále se eroze využívá například pro odstranění šumu. [2]

1.1.3 Otevření a uzavření

Otevření a uzavření využívají faktu že dilatace a eroze nejsou navzájem inverzní operace. To znamená, že pokud na obraz nejprve aplikujeme erozi a poté dilataci nedostaneme zpátky původní obraz. Při použití stejného strukturního elementu nemění již opakované otevření nebo uzavření vstupní obraz. Obrazy, na které již bylo aplikováno otevření nebo uzavření pomocí strukturního elementu B, jsou nazývanými otevřenými popřípadě uzavřenými vzhledem k B. [3]

Otevření je operace při které nejprve provedeme erozi a následně dilataci stejným strukturním elementem. Tím se odstraní efekt eroze, která sice odstraní malé nežádoucí objekty ale zároveň zmenší i objekty žádoucí. Uzavření je naopak dilatace následovaná erozí. Tato transformace spojí objekty nacházející se blízko sebe a vyplní díry velikostně srovnatelné se strukturním elementem. [2]



Obrázek 1: Ukázka morfologických operací

1.2 Segmentace obrazu

Segmentace obrazu je skupina metod, při které se obraz dělí na regiony, které spolu logicky souvisí. Toto rozdělení je možné provádět na základě různých vlastností, jako je například barva, jas, textura nebo lze využít kombinaci těchto vlastností.

1.2.1 Prahování obrazu

Prahování obrazu je nejjednodušší metoda segmentace obrazu. Obvykle se aplikuje na monochromatické obrázky a funguje na principu rozdílného jasu objektu a jasu pozadí. Prahování je časově a výpočetně velmi nenáročná metoda, proto se stále využívá pro některé jednoduché aplikace. Pro dosažení uspokojivého výsledku při prahování je důležité, aby objekty a pozadí v obraze měly dostatečný kontrast a aby se jednotlivé objekty na obraze nepřekrývaly.

Při prahování se obraz prochází pixel po pixelu a pro každý pixel se určuje, zda je jeho jas vyšší než stanovená hranice. Tato hranice může být určena jako konstanta pro celý obraz (globální prahování) nebo ji lze volit dynamicky (adaptivní prahování). Globální prahování většinou nepřináší příliš dobré výsledky, vzhledem k rozdílnosti úrovní jasu způsobené například nerovnoměrným osvětlením. Adaptivní prahování se provádí rozdělením obrazu na regiony pro které se různými metodami spočítá práh. [4]

V některý případech, kdy je známa určitá vlastnost obrazu, je spočtení optimálního prahu jednoduché. Například u tištěného textu je možné určit poměrové zastoupení 1/p písma oproti bílým částem stránky. Díky tomu ze zvolit práh T, tak aby 1/p pixelů mělo hodnotu jasu nižší, než je T a zbytek pixelů mělo vyšší hodnotu než T. [4]

Ve většině případů však nelze tyto vlastnosti předem určit, a proto se využívají metody, které analyzují histogram obrazu a z něho určují optimální práh. Obrazy, ve kterých je dostatečný kontrast pozadí a popředí mají bimodální histogram (histogram obsahující dvě lokální maxima). Mezi těmito dvěma vrcholy lze najít minimum, které může udávat optimální hodnotu prahu. [3]

Pokud není histogram bimodální je možné využít metody, které histogram upraví. Jednou z těchto metod je vynechání pixelů s vysokou hodnotou gradientu při tvorbě histogramu. Tím je docíleno prohloubení údolí na histogramu a jednodušší určení prahu. Tyto metody je možné použít buď pro celý obraz nebo pro jeho části. [3]

1.2.2 Segmentace na základě regionů

Další z metod segmentace obrazu je regionální metoda, při které se hledají regiony pixelů s podobnou vlastností (například jas pixelu nebo jeho barva). Cílem tohoto přístupu je vytvořit co nejrozlehlejší homogenní oblasti. Obecně platí, že oproti metodě hran jsou regionální metody méně ovlivnitelné šumem. Pro stejný obraz poskytují hranové a regionální metody odlišné výsledky, jejichž kombinací je často možné dostat optimální výsledky. [3]

Ve zkoumané vlastnosti obrazu se povoluje jistá flexibilita. Pokud požadujeme, aby byly pixely velmi podobné, mohlo by dojít k rozdělení obrazu na příliš mnoho částí, které samostatně netvoří celek. Při příliš volných podmínkách podobnosti vzniknou segmenty, obsahující objekty, které spolu nemusí zcela souviset. [4]

Jedním z přístupů k regionální segmentaci je spojování regionů. V obraze jsou nejprve náhodně rozmístěny seed pixely. Poté se k těmto pixelům iterativně přidávají sousední pixely s podobnými vlastnostmi. Růst regionu se zastaví v moment, kdy se v okolí již nenachází žádný podobný pixel. Pro každý seed pixel v obraze se musí tato iterace provádět zároveň, jinak by došlo ke zvýhodnění regionů, které se vytvořili jako první. [4]

Druhým přístupem je rozdělování regionů. Na počátku se obraz považuje za jeden region, který se dále iterativně dělí na menší regiony. Metody spojování a rozdělování navzájem netvoří stejný výsledek. Pokud na obraz aplikujeme rozdělování i spojování zároveň, může vzniknout metoda kombinující výhody obou přístupů. [3]

1.2.3 Metoda detekce hran

Pro segmentaci fungující na principu detekce hran existuje řadu algoritmů. Oproti regionální metodě hledají v obraze oblasti, které jsou navzájem nějak odlišné. Cílem je najít uzavřené hranice ohraničující jednotlivé objekty. Metoda obvykle sestává ze dvou fází.

V první fázi se obraz přemění na takzvanou mapu hran. Mapa hran je zobrazení, které zobrazuje oblasti v obraze, ve kterých dochází k prudké změně intenzity. Pro vytvoření takové mapy existuje několik algoritmů, například Cannyho detektor hran.

Cannyho detektor hran je jedním z nejúčinnějších algoritmů pro detekci hran v obrazech. V prvním kroku je obraz převeden do černobílé podoby. Poté se v obraze pomocí Gaussouva filtru zredukuje šum. Následuje výpočet gradientu, který určuje míru intenzity změny oproti okolním pixelům. Toho lze dosáhnout Sobelovým operátorem. Ten vypočte sílu intenzity změny v horizontálním i vertikálním směru. Dále se z gradientu vyberou pouze lokální maxima, čímž dosáhneme odstranění nepodstatných hran. V posledním kroku se na gradient aplikuje prahování s hysterézou. To znamená, že se určí dva prahy T1 a T2 (T1 > T2). Pokud má pixel hodnotu menší než T2 je nulován, pokud má vyšší hodnotu než T1 je považován za silnou hranu a pokud je v rozmezí mezi T1 a T2 je považován za hranu pouze pokud se v jeho okolí nachází silná hrana. [5]

Ve druhé fázi se v mapě hran hledají jednotlivé obrysy objektů. V mapě se najde jeden bod, který by mohl ležet na hranici objektu. Toho lze dosáhnout nalezením bodu s nejvyšší intenzitou. Od tohoto bodu se dále sleduje vždy nejintenzivnější pixel v okolí, dokud nedojde k návratu do původního bodu. Tím vznikne uzavřená hranice ohraničující objekt. [4]

Obrázek 2: Mapa hran vytvořená Cannyho detektorem hran

1.3 Detekce objektů

Detekce objektů je oblast počítačového vidění, která se zabývá automatickým lokalizováním objektů v obraze a určením jejich tříd. Využití detekce objektů lze nalézt například v bezpečnostních systémech, automobilovém průmyslu nebo robotice.

Existují dva hlavní přístupy pro detekci objektů v obraze. Prvním z nich jsou two-stage detektory, které nejprve identifikují oblasti zájmu a následně pomocí neuronových sítí určí třídy jednotlivých nalezených objektů. Takovéto detekce jsou výpočetně náročné, avšak dosahují vyšší přesnosti. Druhým přístupem jsou single-stage detektory. Tyto detektory používají jedinou neuronovou síť, která zároveň vyhledává objekty a určuje jejich třídu. Díky tomu jsou výrazně rychlejší oproti two-stage detektorům. To umožňuje využívat tyto algoritmy v reálném čase i na méně výkonném hardwaru.

Oba tyto přístupy ke svému fungování využívají konvoluční neuronové sítě (CNN). CNN využívají tři hlavní typy vrstev: konvoluční vrstvy, pooling vrstvy a plně propojené vrstvy. Tento typ neuronových sítí se používají k detekci objektů z důvodu snížení počtu vstupních parametrů sítě. [6]

Konvoluční vrstvy fungují na principu malých (typicky 3x3) detektorů, které se postupně posouvají přes celý obraz a aplikují na něj skalární součin s vahami nacházejícími se v detektoru. Krok o který se okénko posouvá se nazývá stride. Výsledky jednotlivých součinů jsou ukládány do pole nazvaného aktivační mapa. Na tuto mapu se dále aplikuje nelineární aktivační funkce, například ReLU (Rectified Linear Unit). Dále se tato mapa používá jako vstup do pooling vrstvy. Podobně jako u konvoluční vrsty se u pooling vrstvy přesouvá malé okénko přes celá vstupní data. Oproti konvolučním vrstvám se ale neaplikuje jako operace skalární součin ale pouze agregační funkce. Existují dva typy pooling vrstev, jeden aplikující max funkci a jeden počítající průměrnou hodnotu. CNN pak obsahují několik těchto kombinací konvoluční a pooling vrstvy. Výsledky těchto vrstev dále vstupují do plně propojené vrstvy. U plně propojených vrstev je každý výstupní uzel přímo propojen se všemi vstupními uzly. Tato vrstva provádí klasifikaci vzorů získaných v předchozích vrstvách a vypočítává pravděpodobnost nalezeného objektu. [7]

1.3.1 Faster R-CNN

Faster R-CNN (rychlejší regionální konvoluční neuronová síť) je nástupce R-CNN, který byl průlomovým způsobem využití konvolučních neuronových sítí pro detekci objektů. Faster R-CNN se řádí do kategorie two-stage detektorů.

Oproti svým předchůdcům nevyužívá algoritmu selektivního vyhledávání pro určení lokací objektů. Tento algoritmus byl často místem největšího zdržení u předchozích metod detekce. Namísto toho používá Faster R-CNN takzvaný RPN (region proposal network). RPN vezme na vstupu obraz a vygeneruje množinu obdélníkových návrhů lokací objektů společně se skórem určujícím pravděpodobnost, že se daný objekt v lokaci nachází. Na vstupu do RPN se nachází několik klasických konvolučních vrstev, která vytvoří mapu příznaků. Pro každý výsledek v mapě příznaků generuje RPN sadu takzvaných anchor boxů. Tyto anchor boxy se většinou generují se třemi různými velikostmi a třemi poměry stran (dohromady 9 boxů). [8]

Výsledky z RPN jsou dále využity jako vstup do další neuronové sítě, která obsahuje dvě plně propojené vrstvy. První vrstva určí zda a s jakou pravděpodobností v boxu nachází objekt. Druhá vrstva pak objekt klasifikuje a upřesní velikost a pozici ohraničujících rámečků. [9]

1.3.2 YOLO

YOLO (you only look once) je jeden z prvních a do dneška nejpopulárnější single-stage detektor. Prní verze algoritmu byla poprvé představena v roce 2015 ve článku, který napsali pánové Joseph Redmon, Santosh Divvala, Ross Girshick a Ali Farhadi. Jejich přístup využíval jedinou neuronovou síť, jak pro určení lokace objektů tak k určení jejich tříd. Díky tomu bylo dosaženo rychlosti, která byla v oblasti rozpoznávání objektů revoluční. Zároveň si však YOLO udržuje vysokou míru přesnosti.

V současné době existuje 9 verzí tohoto algoritmu. Každá nová verze přináší oproti svým předchůdcům zrychlení a zvýšení přesnosti detekce. Pro tuto práci byla vybrána verze 7. Ta je oproti předchozím verzím optimalizovaná snížením počtu vstupních

parametrů, bez snížení přesnosti inference. YOLO je široce využíváno například v: autonomních vozech, zabezpečovacích systémech nebo pro medicínské aplikace. [10]

YOLO rozděluje vstupní obraz na mřížku buněk. Každá tato buňka je zodpovědná za předpověď daného množství ohraničujících rámečků. V každém rámečku se pro každou třídu detekovaných objektů určí s jakou pravděpodobností se v rámečku nachází. Dále se určí souřadnice středu objektu a jeho velikost. K tomu jsou využity konvoluční vrstvy sítě. Tímto způsobem je možné, že je pro jeden objekt vytvořeno více rámečků. Pro tyto případy se používá algoritmus NMS (non-max supression), který odstraní redundantní detekce. NMS nejprve seřadí detekce podle jejich skóre pravděpodobnosti. Poté se postupně pro každou detekci odstraní všechny rámečky, které mají s rámečkem této detekce vyšší hodnotu IoU (intersection over union) než je daný práh. IoU se počítá jako podíl plochy průniku dvou rámečků a plochy jejich sjednocení. [11]

Jedním z důvodů vysoké rychlosti tohoto algoritmu, je jeho páteřní síť nazvaná DarkNet. Ta obsahuje 24 konvolučních vrstev, následována dvěma plně propojenými vrstvami. Každá vrstva kombinuje 3x3 konvoluce s 1x1 konvolucemi pro redukci kanálů. [11]

Pro použití YOLO je potřeba stáhnout nebo natrénovat YOLO model, což je obvykle soubor s příponou pt. Díky tomuto modelu můžeme v jazyce Python provádět inferenci. Pro účely robota byl využit model yolov7-tiny, trénovaný na datasetu COCO obsahující 80 různých tříd. Z těch bylo vybráno několik tříd objektů, které lze běžně naleznout v domácnosti. Třídy jsou následující: person, bottle, cup, banana, apple, plant.

2 Řídící a pohonné komponenty

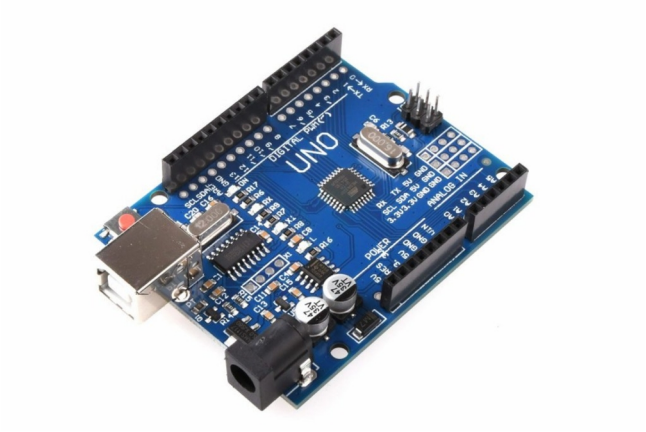
2.1 Arduino uno

Arduino Uno je mikropočítač od společnosti Atmel disponující 8-bitovým procesorem ATmega328P. Na desce se nachází 16 digitálních I/O pinů, z nichž lze 6 použít jako PWM výstup. Dále máme k dispozici 6 analogových vstupů, vestavěnou LED a tlačítko pro restartování. [12]

Mikroprocesor AT-mega328p pracuje na maximální frekvenci 20MHz (16MHz u Arduino Uno). Obsahuje 32kB flash paměti pro ukládání programu a 2kB SRAM. Pro převod analogových signálů se na procesoru nachází 10bitový AD převodník se šesti kanály. To znamená, že je možné přiřadit 1024 hodnot pro signály v rozmezí 0-5V na šesti různých pinech. Na procesoru se dále nacházejí dva 8bitové a jeden 16bitový čítač/časovač pro realizaci časových funkcí, generování PWM nebo čítaní impulzů. Ke komunikaci s jinými zařízeními je možné využít sběrnice USART (Universal Synchronous/Asynchronous Receiver/Transmitter), SPI (Serial Peripheral Interface), nebo I2C (Inter-integrated circuit). Pin 2 a 3 lze využít jako zdroj přerušení, což umožňuje okamžitě reagovat na asynchronní události.

K napájení desky lze přivést zdroj napětí v rozsahu 7-12V na externí napájecí konektor nebo na pin označený jako VIN. Napájení je možné provést i pomocí vestavěného USB, přes který se i Arduino programuje. Digitální vstupy a výstupy pak pracují na 5V logické úrovni. Každý výstup snese proudové zatížení 40mA (doporučuje se 20mA), avšak celkové zatížení všech pinů dohromady nesmí překročit 200mA. [12]

Programy pro Arduino se tvoří v jazyce C++. Tvorbu nám usnadňuje velká komunita, která vytvořila množství užitečných knihoven. Program pro Arduino se skládá ze dvou částí. V první části, která proběhne pouze jednou se program inicializuje. Druhou částí je nekonečný cyklus, ve kterém se odehrává hlavní část programu. Pro tuto práci bylo Uno využito k vyhodnocování dat z Raspberry Pi (později z Jetson Nano) a komunikaci s motory.



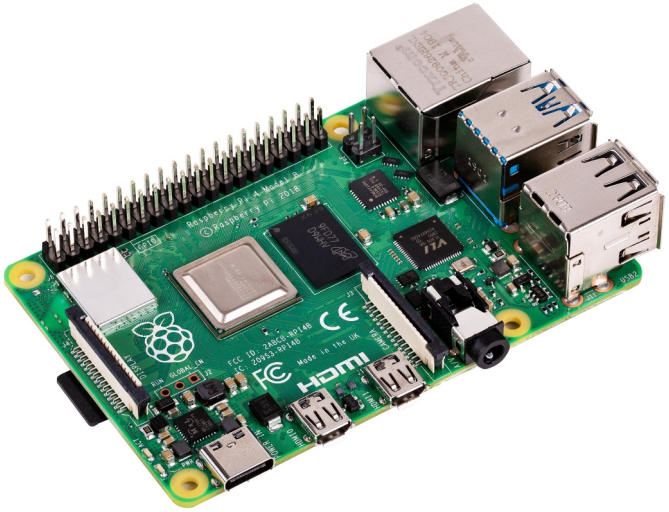
Obrázek 3: Arduino Uno

2.2 Raspberry Pi 4 model B

Raspberry Pi 4 je výkonný jednodeskový počítač od společnosti Raspberry Pi Foundation. Srdcem této desky je 64-bitový ARM procesor Broadcom BCM2711. Tento procesor obsahuje čtyři jádra a pracuje na frekvenci 1,8GHz. Model umožňuje výběr mezi 2, 4 a 8GB paměti. Díky tomu je Raspberry Pi schopné zvládnout i náročnější úlohy. Desku lze napájet napětím 5V a to buď pomocí usb konektoru nebo přímo přes 5V pin. V obou případech je potřeba zdroj podporující alespoň 2,5A proud. [13]

Co se týče konektivity, nachází se na desce 40 I/O pinů, které pracují na 3.3V logice. Dále se na desce nachází 4 USB konektory, 1 HDMI konektor, stereo výstup, LAN, DSI (Display Serial Interface) a CSI (Camera Serial Interface) konektor. Raspberry Pi plně podporuje 2,4GHz i 5GHz wifi a bluetooth 5.0. [13]

Tento počítač je hlavně využíván s operačním systémem Raspberry Pi OS (dříve Raspbian), který se instaluje na micro SD kartu. Raspberry Pi OS je založený na linuxové distribuci Debian a je zdarma. V základu má nainstalovaný interpret pro Python a další užitečné nástroje jak pro správu zařízení, tak pro psaní kódu.



Obrázek 4: Raspberry Pi 4

2.3 Jetson Nano

Jetson Nano je jednodeskový počítač od firmy NVIDIA, který je navržen pro práci s neuronovými sítěmi. Procesorovým výkon je srovnatelný s Raspberry Pi 4, avšak oproti Raspberry obsahuje navíc 128 jádrový grafický procesor s architekturou CUDA. Díky tomu dosahuje mnohem vyšších rychlostí při úlohách, jako je zpracování obrazu či rozpoznávání objektů.

Počítač obsahuje 4GB RAM, ze kterých je pouze přibližně 2,5GB využitelných uživatelem a zbytek si zabírá operační systém. Ten je podobně jako u Raspberry založen na linuxové distribuci Ubuntu. V systému je předinstalovaný interpret pro Python 3.6 s nainstalovanými knihovnami, optimalizovanými pro výpočty na grafickém jádře. Příkladem těchto knihoven jsou OpenCV, TensorRT nebo PyTorch. [14]

Na desce se nachází gigabitový ethernetový port, 4x USB konektor, 26x GPIO pin, HDMI a display port a 2x CSI konektor. Dále můžeme s deskou komunikovat přes rozhraní jako je UART, SPI nebo I2C. Stejně jako Raspberry Pi pracují GPIO piny na

3,3V logice a je napájena 5V napětím. To usnadňuje případnou výměnu při aplikacích, ve kterých je výkon Raspberry Pi nedostačující. [14]

Obrázek 5: Jetson Nano

2.4 Ax-12+

AX-12+ jsou výkonné servomotory od firmy Robotis. Oproti běžným servomotorům se neovládají zvenku pomocí počítače, ale ovládá je vestavěný mikrokontrolér, kterému pouze posíláme pokyny. Díky tomu může o sobě motor sám poskytnout informace, jako je rychlost, pozice motoru, teplota a napětí. Motory se dají díky jejich unikátním ID, řetězově propojovat, což velmi usnadňuje zapojení.

Motory jsou schopné komunikovat rychlostí až 1Mbps. I přes malé rozměry 32x50x40mm dosahují točivého momentu 1.5Nm při rychlosti 60 otáček za minutu. Motory mají dva režimy, režim servo a režim nekonečného otáčení. V režimu servo mají rozlišení 0,29°. Napájet je lze napětím 9-12V, přičemž při 12V odebírají proud přibližně 1,5A.

Obrázek 6: Motory Ax12+

3 Hardwarový návrh robota

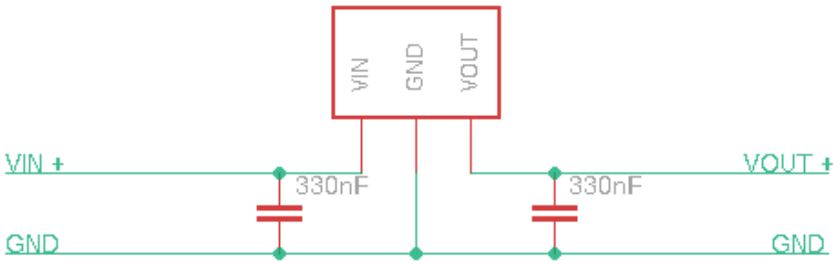
3.1 Schéma zapojení

V této části práce je popsáno schéma zapojení všech komponent. Schéma se dá rozdělit do dvou částí. První z nich obsahuje způsob napájení všech komponent. Druhá pak řeší jejich komunikaci. Kromě toho do schématu patří ještě čtyři spínače připojené k Arduinu, sloužící jako senzory dotyku a kamera připojená pomocí CSI (camera serial interface) konektoru k Jetson Nano.

3.1.1 Napájení komponent

K dodání energie do robota slouží baterie PAP 20 A1 od firmy Parkside, který se normálně využívá pro napájení akumulátorového nářadí. Napětí tohoto akumulátoru je 20V, což je pro napájení všech komponentů robota příliš. Proto je využit step-down měnič, který toto napětí snižuje. K nastavení jeho výstupního napájení lze využít trimr, který se na něm nachází. Jako hodnota výstupního napětí bylo zvoleno 11V. Tato hodnota je doporučena výrobcem pro motory AX12+ a zároveň to je bezpečná hladina napětí pro Arduino Uno.

Jak Raspberry Pi tak Jetson Nano však striktně vyžadují pro své napájení napětí 5V. Pro tuto napěťovou hladinu byl původně zvolen stabilizátor L7805CV, poskytující proud 2A. Tento stabilizátor je sice velmi levný ale při vyšším zatížení Raspberry Pi byl na jeho výstupu pokles napětí o přibližně 0.2V. To způsobovalo chybovost některých funkcí, jako je například sériová komunikace. Proto byla součástka vyměněna za lineární stabilizátor LM1085IT-5.0, který je schopný poskytnou proud až 3A. Pro vyhlazení napětí jsou ke stabilizátoru ještě paralelně připojeny dva kondenzátory o hodnotě 330nF.



Obrázek 7: Zapojení stabilizátoru napětí

Použité indikační LED mají maximální proudový odběr 20mA, proto je možné je připojit přímo k Arduinu v sérii s 200Ω rezistory. Pro zapnutí a vypnutí robota slouží kolébkový spínač zapojený v sérii hned za akumulátorem, na kterém se nachází LED pro indikaci stavu robota.

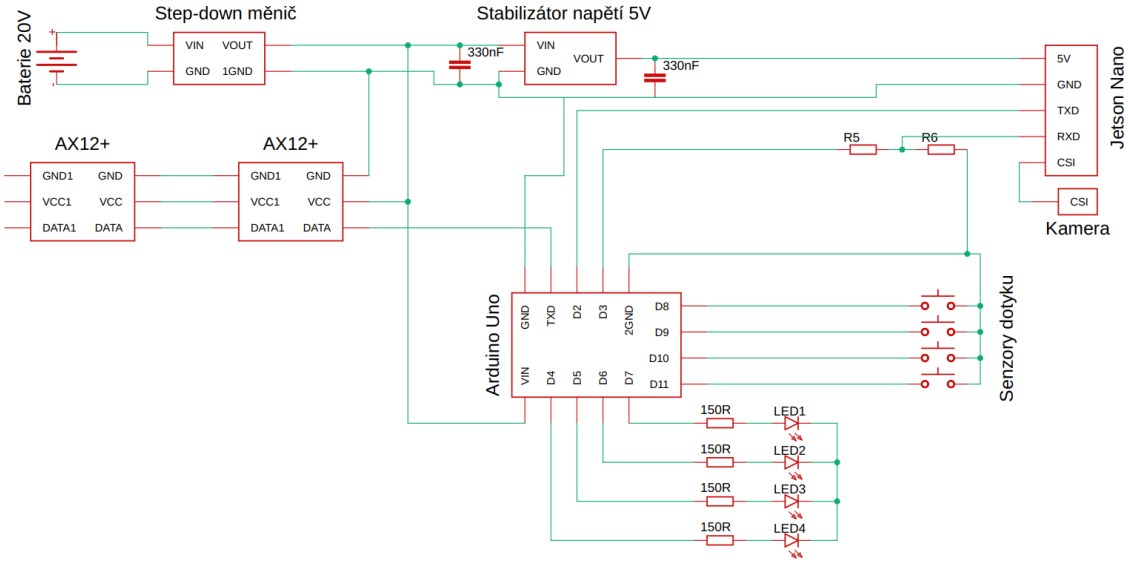
3.1.2 Komunikace komponent

Komunikace mezi Arduinem a Jetson Nano byla provedena pomocí rozhraní UART. Vzhledem k tomu, že Arduino detekuje logickou 1 už při 2.6V není, při komunikaci ve směru z Jetson Nano do Arduina, problém v rozdílných logických úrovních mezi oběma zařízeními. V opačném směru, by při přímém zapojení 5V na Rx pin u Jetson Nano, došlo k poškození tohoto pinu nebo dokonce celé desky. Proto je mezi Tx pinenm Arduina a Rx pinem Jetsonu připojen dělič napětí. Dělič napětí se skládá ze dvou rezistorů zapojených v sérii s předem vypočtenými hodnotami podle vzorce (3.1).

$$V_{Výstupní} = V_{Vstupní} \cdot \frac{R2}{R1+R2} \tag{3.1}$$

Pro použití mezi Arduinem a Raspberry Pi byly zvoleny hodnoty 1000R a 2000R. Tím se hodnota 5V sníží na 3.3V. Vzhledem k tomu, že Arduino Uno obsahuje pouze jednu sériovou linku, byly využity pro toto propojení klasické digitální piny, na kterých lze pomocí knihovny sériovou komunikaci provádět.

Mezi Arduinem a motory probíhá komunikace také pomocí UART s tím rozdílem, že je tato komunikace pouze ve směru z Arduina do motorů. Na motorech AX12+ se nachází dva konektory, umožňující zřetězené zapojení více motorů. Díky tomu lze oba motory ovládat pouze jedním Tx kabelem z Arduina.



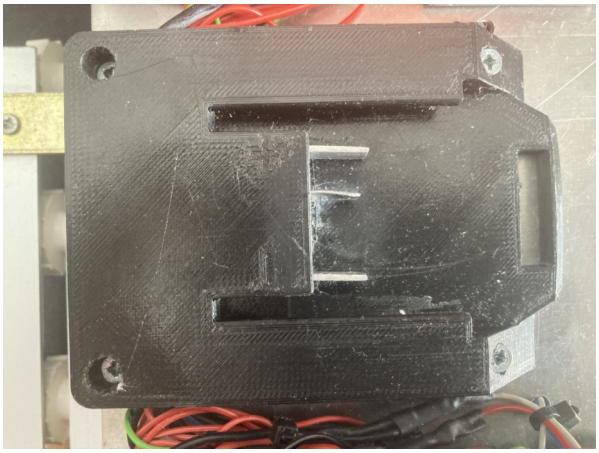
Obrázek 8: Celkové schéma zapojení

3.2 Konstrukce robota

Konstrukce robota se skládá ze dvou hliníkových plechů, dvou motorů AX12+, několika hliníkových profilů a řady dílů vytisknutých na 3D tiskárně. Pro tisk byl použit plast PLA v černé a oranžové barvě. Dále jsou na robotovi umístěny mikropočítače Arduino Uno a Jetson Nano pro řízení. Pro většinu komponent byly využity šrouby M3 různých délek. Pro připevnění kol byly použity šrouby M5x40mm a pro upevnění kamery šrouby M2.

Základ robota tvoří dvoumilimetrový hliníkový plechu o rozměrech 233x140 mm, který slouží jako podvozek. V zadní části podvozku jsou pomocí úhelníků připevněny oba motory. Mezi motory se nachází vytištěný díl obsahující čtyři indikační LED a kolébkový spínač. Jak spínač, tak LED jsou v dílu uchyceny zacvaknutím do otvorů.

Akumulátor je umístěn před motory, kde je zasazen do adaptéru. Adaptér byl navrhnut tak, aby umožňoval vložení tří kusů hliníkového plechu, které se při vložení baterie zasunou do kontaktů baterie. K plechům jsou poté šrouby připevněny kabely, čímž je poskytnuto elektrické spojení mezi baterií a zbytkem robota. Na baterii se nachází mechanizmus, který se při zasunutí baterie do adaptéru zacvakne a zabrání tak nechtěnému vysunutí baterie.

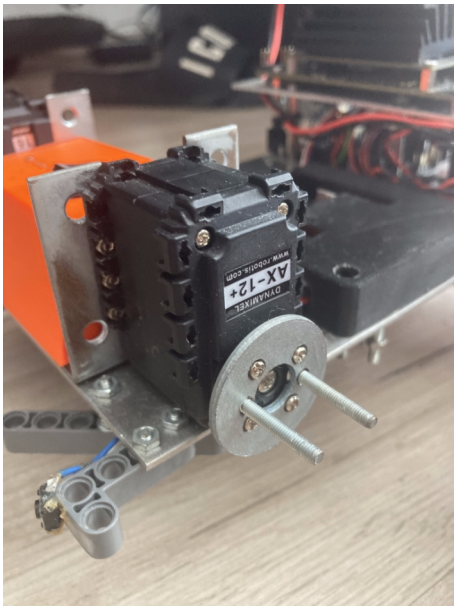


Obrázek 9: Adaptér na akumulátor

V přední části robota se nachází Arduino připevněné na distančních sloupcích o délce 5 mm. Vedle Arduina je pomocí stahovacích pásků přichycen step-down měnič. Nad Arduinem se nachází menší hliníkový plech o rozměrem 60x140mm. Ten stojí na 30mm distančních sloupcích, které jsou zašroubovány do distančních sloupků držící Arduino. Na tomto plechu se nachází počítač Jetson Nano.

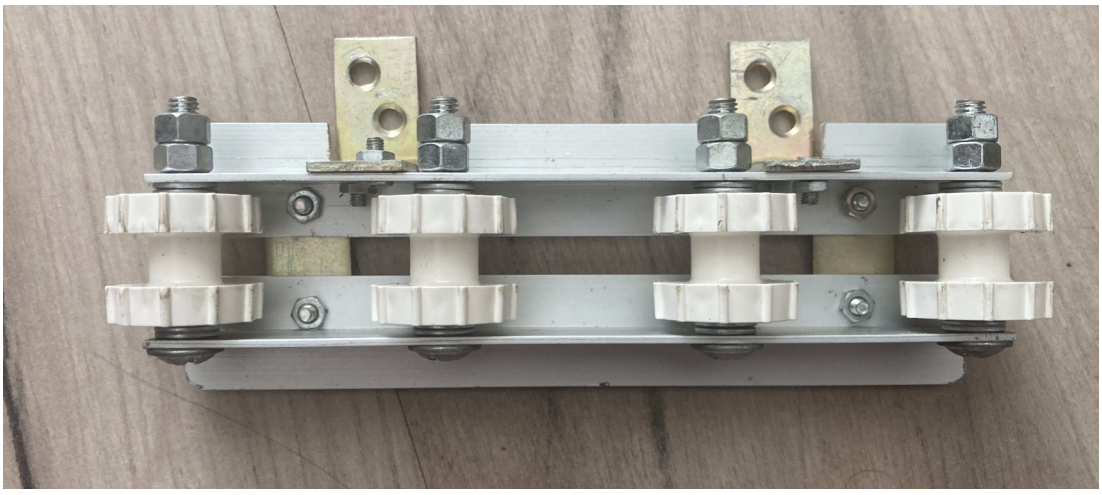
Dále se v přední části nachází přilepený adaptér pro kameru vytisknutý na tiskárně. Tento díl má tvar L, který spolu tenkostí stěn dílu poskytuje malou pevnost. Pro budoucí vylepšení by bylo vhodné přidat dílu diagonální podpory, které by tento problém vyřešily.

K pohonu robota slouží pásy a kola ze sady Lego education. K napojení těchto kol k motorům byly využity podložky o průměru 30mm. Ty jsou z jedné strany přišroubovány k motorům a z druhé strany vyčnívají šrouby, které lze zasunout do kol. Díry pro tyto šrouby jsou vyvrtány tak, aby při zasunutí šroubu byla jeho hlavička zarovnána s povrchem podložky. Jednou z výzev při konstrukci bylo napojení těchto podložek na motory tak, aby nebylo kolo vyosené vůči ose motoru. Po několika neúspěšných pokusech, při kterých díry v podložce neseděli přesně na díry motoru, byl na 3D tiskárně vyroben pomocný díl. Tento díl obsahuje malé díry vyznačující přesnou polohu pro umístění vrtáku.



Obrázek 10: Adaptér pro připojení lego kol

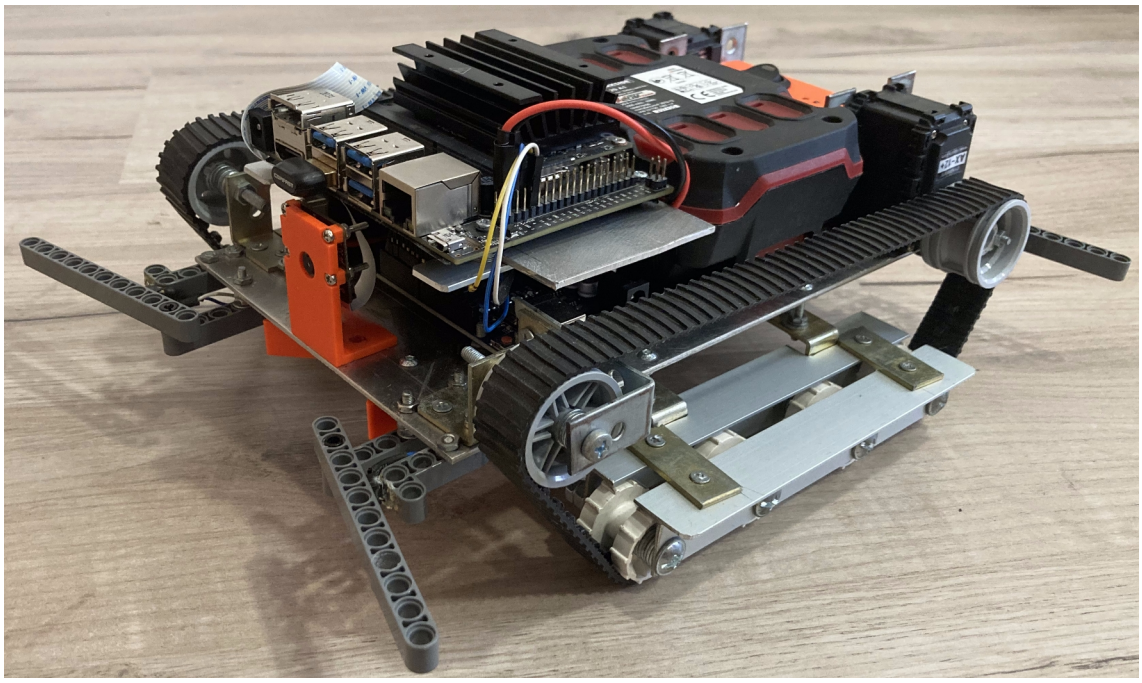
V přední části robota se na obou stranách nachází další kolo držené třemi úhelníky. Tyto úhelníky jsou postaveny tak, aby umožnily rovnoběžné vyrovnání pomocí podložek s kolem připevněným k motoru. Na spodku podvozku je připevněna lišta vyrobená ze dvou hliníkových T profilů. K této liště jsou upevněny čtyři kola, které umožňují dotyk pásu se zemí. Kola jsou připevněny pomocí šroubů a dvou matek, které jsou zalepeny lepidlem na závity, aby se zamezilo jejich uvolňování. Lišta je spojena s podvozkem dlouhými šrouby, které dovolují posunout celou lištu směrem nahoru a dolu. Díky tomu je možné libovolně upravovat napnutí pásů.



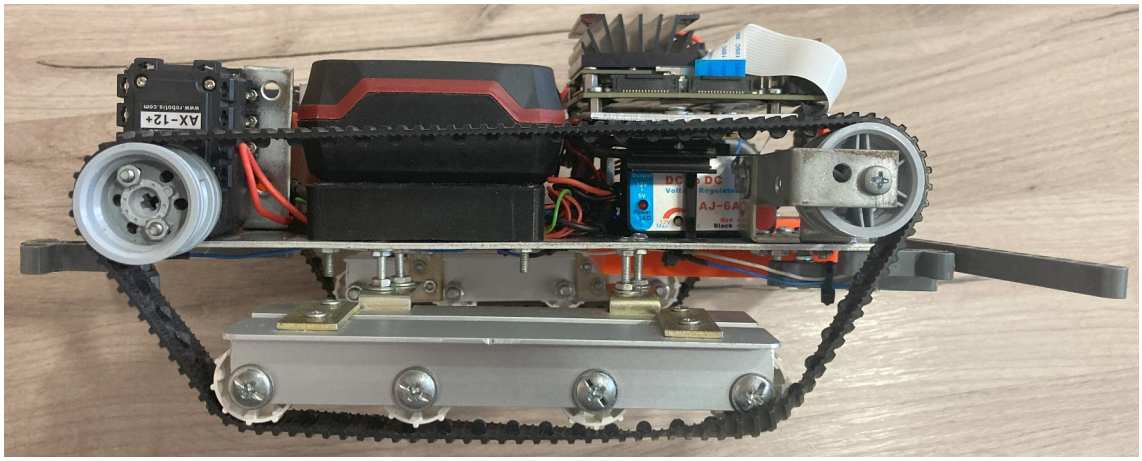
Obrázek 11: Lišta pro upevnění kol

Ke každému rohu podvozku jsou uchyceny dva lego díly. Na jednom z nich je přilepen klasický 6x6 mm spínač. Na druhém z nich je připevněn další lego díl, který se může volně otáčet. Tyto díly jsou umístěny tak, aby přesahovaly šířku robota. Tím se zajistí, že pokud robot narazí, sepne tento díl daný spínač, čímž řídící logice poskytne informace o nárazu.

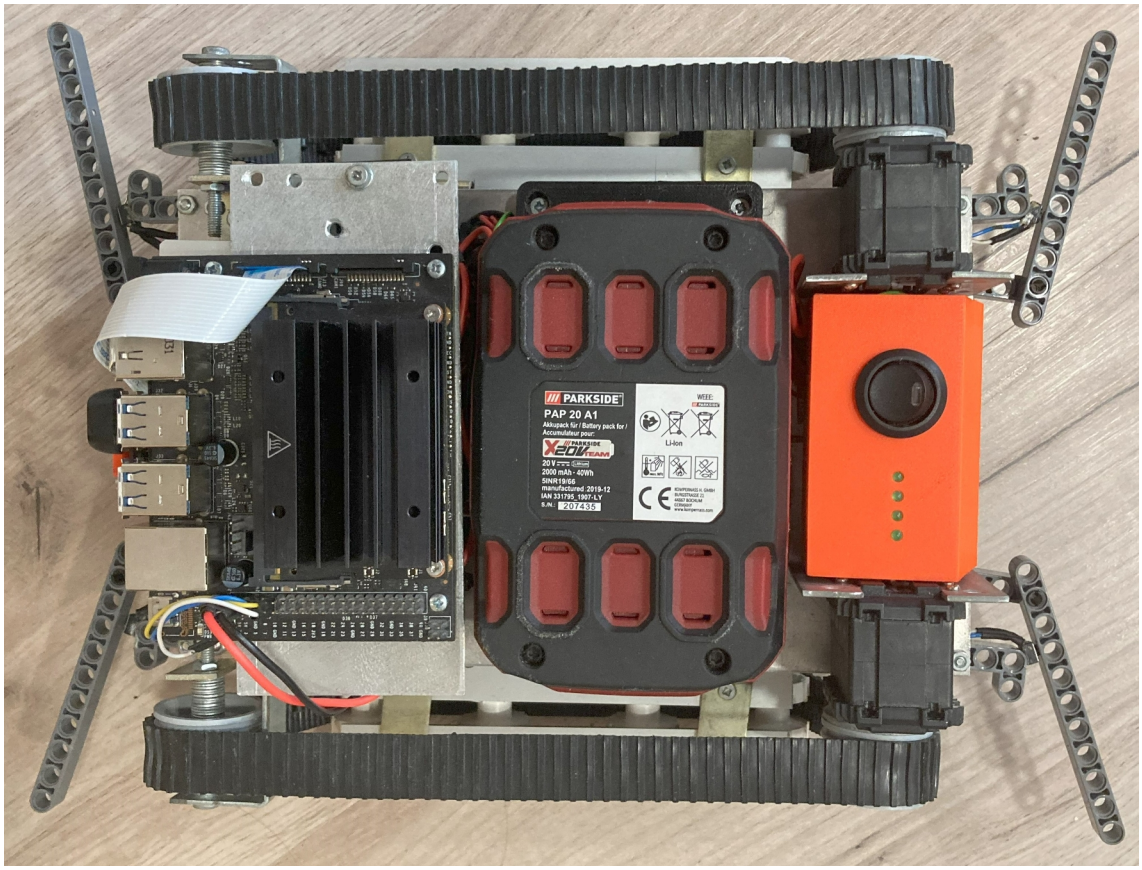
Poslední komponentou, která se na robotovi nachází je konektor vespodu podvozku. Tento konektor byl vytištěn na 3D tiskárně. Je zde proto, aby robotovi zajistil propojení s nabíjecí stanicí. Konektor je vymodelován do tvaru V a obsahuje tři měděné plechu sloužící jako kontakty při vjetí robota do stanice. Aby se zaručilo přesné zasunutí kontaktů do sebe, je tento díl vymodelován do tvaru V, čímž robota při vjíždění přesně srovná.



Obrázek 12: Celková konstrukce robota



Obrázek 13: Celková konstrukce robota 2

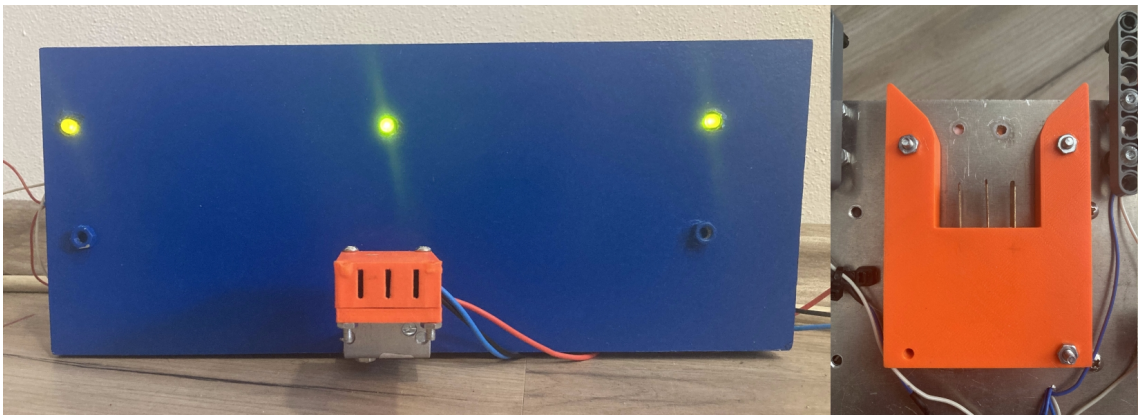


Obrázek 14: Celková konstrukce robota 3

3.2.1 Nabíjecí stanice

Základem nabíjecí stanice je dřevěná deska o rozměrech 280x150x15 mm. Na desce jsou vyvrtány tří díry, ve kterých jsou vlepeny zelené LED. Dvě LED se nachází na okraji a třetí přesně ve středu desky. Ve středu desky se na spodním okraji nachází konektor vyrobený na 3D tiskárně, upevněný dvěma úhelníky. Konektor je navržen tak, aby při parkování vjel do jeho protikusu na spodku robota.

V konektoru se nachází šest měděných plechů, které spolu dohromady tvoří tři kontakty. Pro každý kontakt je jeden rovný plech a druhý zahnutý do tvaru luku směrem k prvnímu plechu. Tím je zaručen kvalitní pružinový kontakt mezi nabíječkou a robotem.



Obrázek 15: Nabíjecí stanice a konektor, který se do ní zapojuje

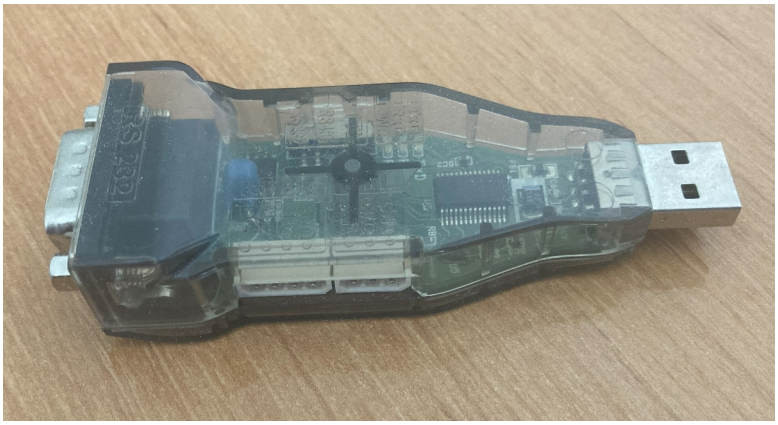
4 Softwarové část robota

Software, kterým lze robota řídit, sestává ze tří částí. První z nich je desktopový program, který slouží k ovládání robota z uživatelského rozhraní počítače. Druhá je aplikace běžící na platformě Jetson Nano, specializovaná na zpracování obrazu a zajištění vstupních dat pro navigaci a detekci objektů. Třetí část je software implementovaný v mikrokontroléru Arduino, který ovládá pohybový systém robota a reaguje na podněty ze senzorů. Tyto tři části vzájemně komunikují a umožňují tak funkčnost robota.

4.1 Arduino program

Program běžící na Arduinu má za hlavní cíl vyhodnocovat informace z Jetson Nano a podle nich řídit motory a indikační LED. Prvním krokem pro ovládání motorů je zjištění jejich unikátních ID a také toho, na jaký baud rate jsou nastaveny.

K tomu lze využít program Dynamixel Wizard, ke kterému motory připojíme pomocí zařízení USB2Dynamixel. Toto zařízení funguje jako rozhraní mezi USB sběrnicí a sériovou komunikací motorů. K zařízení se připojí jeden ze dvou tří-pinových výstupů motoru. Zároveň je nutné motory připojit k napájecímu napětí, jinak by je Dynamixel Wizard nebyl schopný detekovat.



Obrázek 16: Zařízení USB2Dynamixel

V prostředí Dynamixel Wizard je pak možné si navolit komunikační port, na kterém jsou motory připojeny. Pro tento port lze provést sken určitého rozsahu ID, které mohou

být od 0 do 255, čímž se dozvíme jaké ID je přiřazeno připojenému motoru. Zároveň lze v programu nastavit řadu parametrů jako například: baud rate, omezení točivého momentu, maximální teplota motoru nebo režim motoru. Režim může být buď výchozí, kdy se motor chová jako klasické servo, nebo režim endless turn, kdy se motor otáčí dokola. Pro komunikaci s Arduinem byl zvolen baud rate 1Mbs, při kterém byla komunikace spolehlivá a nebyl tak důvod volit pomalejší rychlost. Ostatní parametry byly ponechány v jejich výchozím stavu.

Dalším krokem je zajištění správného formátu dat posílaných do motorů. Motory AX12+ jsou ovládané protokolem vytvořeným společností Robotis. Komunikace probíhá na principů krátkých zpráv. Na začátku zprávy se nacházejí dva 0xFF byty, oznamující začátek přenosu. Následuje byte určující, pro jaké ID motoru je zpráva určena. Po ID se pošle délka, určující počet parametrů funkce, která je po motoru vyžadována. Poté se přenese ID instrukce a všechny její parametry. Na konci paketu se ještě nachází byte pro kontrolní součet.

Pro implementaci tohoto protokolu existuje Arduino knihovna s názvem AX12A. Ta dělá použití protokolu jednoduchým tím, že dovoluje pouze zavolat funkci s parametry a knihovna se postará o zaslání správného formátu dat do motorů. Pro použití knihovny AX12A ji je nejprve potřeba inicializovat funkcí begin(), která jako parametry přijímá baud rate, sériový port Arduina a číslo pinu, který určuje směr komunikace mezi oběma zařízeními. Motory využívají jediný vodič ke komunikaci, proto je potřeba speciální součástka jako například 74LVC2G241, aby byla umožněno posílání dat v obou směrech. Pro tuto práci byla využita komunikace poloduplexní, tedy pouze se posílají data do motorů bez zpětné vazby.

Dále je potřeba oba motory nastavit do režimu endless turn pomocí funkce setEndless(), která jako parametry přijímá ID motoru a ON/OFF hodnotu. Motory se poté ovládají funkcí turn() s parametry ID, směr otáčení a rychlost. Rychlost je dána jako číslo mezi 0 a 1023.

Kromě komunikace s motory bylo potřeba vyřešit i komunikaci s Jetson Nano. Vzhledem k tomu, že Arduino Uno má pouze jeden sériový port musela být použita

knihovna SoftwareSerial. Pro její použití je potřeba si vytvořit proměnou typu SoftwareSerial, v jejímž konstruktoru určíme digitální piny, které budou sloužit jako Rx a Tx pin. Poté lze využívat stejné funkce jako při práci s hardwarovým serial portem.

Pro přenos dat mezi Arduinem a Jetson nano byl zvolen baud rate 115200 bps, u kterého se ale ukázalo, že je komunikace chybová. To je způsobeno použitím softwarové sériové linky, která nezvládá hlavně příjem dat při vyšších rychlostech. Maximální rychlost, při které byla komunikace bezproblémová, vyšla po několika testech jako 19200 bps. Posledním úkolem inicializační části programu je nastavit piny pro tlačítka do režimu INPUT_PULLUP a piny pro LED do režimu OUTPUT.

Ve smyčce programu se pokaždé, když přijde příkaz, tento příkaz přečte a následně zpracuje. Jako formát komunikace byl zvolen podobný přístup jako u gcode, který využívají například CNC stroje nebo 3D tiskárny. Na začátku každého příkazu se nachází znak "C" následovaný číslem příkazu, případně další číselné parametry. Před každým parametrem je znak označující, o jaký parametr se jedná.

Pokud by byl robot například v režimu vyhledávání a Jetson Nano byl chtěl Arduinu zaslat informaci a poloze objektu a jeho velikosti, vypadal by příkaz následovně: "C20 X120 H60 W40", kde 120 je x-ová souřadnice středu objektu, 60 je výška objektu a 40 jeho šířka. V tabulce níže jsou uvedeny veškeré příkazy, které robot přijímá společně s jejich možnými parametry.

Příkaz	**Funkce**	**Parametry**
C01	Pohyb robota	F (dopředu), B (dozadu), L (doleva). R (doprava)
C02	Nastavení rychlosti	S (nová rychlost)
C20	Zaslání souřadnic objektu	X (x-ová souřadnice), Y (y-ová souřadnice), H (výška objektu), W (šířka objektu)
C30	Zaslání velikosti obrazu	W (šířka obrazu), H (výška obrazu)
C40	Zaslání souřadnic LED	X, Y, Z (x-ové souřadnice jednotlivých LED)
C50	Zapnout indikační LED	L (číslo LED)

C51	Vypnout indikační LED	L (číslo LED)
C52	Blikat indikační LED	L (číslo LED), P (perioda blikání)
C60	Zapnout manuální režim	žádné
C61	Zapnout režimu parkování	žádné
C62	Zapnout režim vyhledávání	žádné

Výchozím stavem je při zapnutí robota je stav připojování, ve kterém Arduino pouze bliká indikačními LED. Po připojení bluetooth k Jetson Nano dojde k přepnutí do režimu manuální kontroly, při kterém Arduino očekává příkazy k přepnutí do dalšího režimu nebo k pohybu robota. Tyto příkazy mají jediný parametr určující směr, který následuje číslo jedna nebo nula podle toho, zda se v tomto směru požaduje začít nebo ukončit pohyb.

4.1.1 Režim parkování

V režimu parkování se nejprve očekává zpráva určující velikost vstupního obrazu. To je poté důležité pro určení polohy všech LED na nabíjecí stanici. O jejich x-ové souřadnice poté Arduino v cyklu přijímá informace, podle kterých řídí motory.

Pokud při startu parkování Jetson Nano nenalezne žádnou LED začne Arduino otáčet robotem dokud nedojde k jejich nalezení. Při jejich nalezení srovná program robota tak, aby obě krajní LED měli stejnou vzdálenost od okraje. Pokud je v této fázi prostřední LED ve středu obrazu, rozjede se robot dopředu a případně se podle potřeby otočí tak, aby krajních LED udržovaly stejnou vzdálenost od obou krajů.

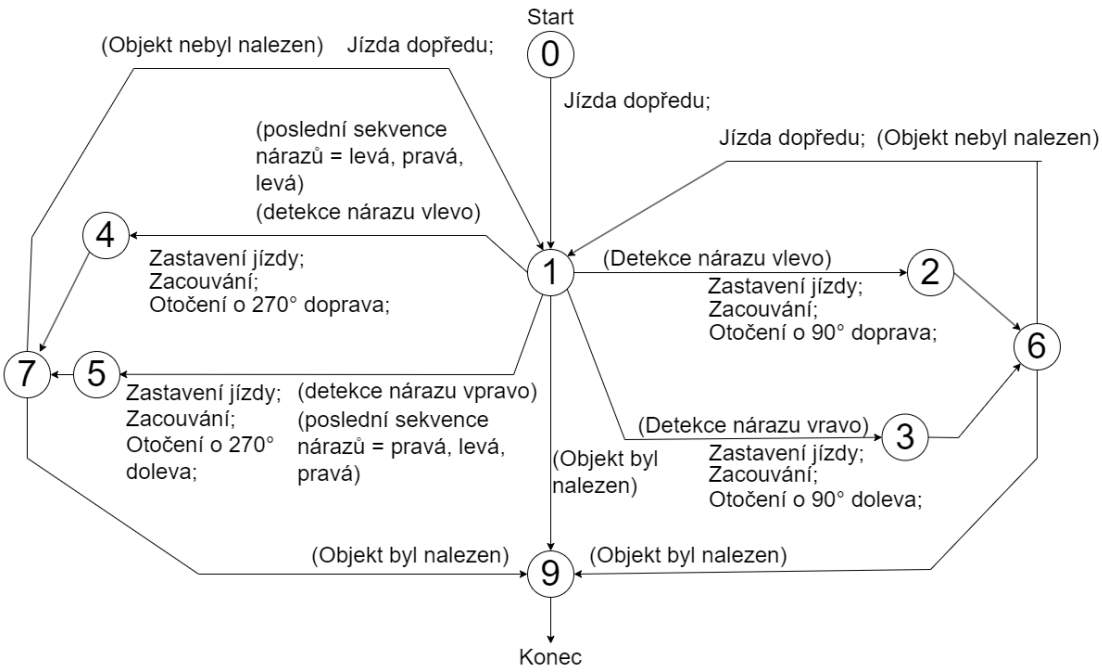
Pokud se prostřední LED nenachází s jistou tolerancí ve středu mezi krajními LED, lze tím indikovat to, že je robot posunut oproti nabíjecí stanici doprava nebo doleva. Z odchylky LED od středu obrazu lze vypočítat o jakou vzdálenost se robot musí posunout, aby se nacházel přibližně před nabíjecí stanicí. Tento výpočet byl určen experimentem, při kterém byla zkoušena úspěšnost zaparkování při různých startovacích polohách.

Pro provedení posunutí se robot otočí podle potřeby doleva nebo doprava tak, aby směřoval mimo nabíjecí stanici. Poté ujede dopředu vypočítanou vzdálenost a otočí se zpátky. Pokud by při některých startovacích pozicích nebylo provedeno toto posunutí, najel by robot na nabíjecí stanici pod takovým úhlem, který by nedovoloval správné vsunutí nabíjecích konektorů do sebe. Při úspěšném zajetí se jeden z předních senzorů aktivuje o základnu stanice, čímž se parkování ukončí.

4.1.2 Režim vyhledávání

Ve režimu vyhledávání očekává Arduino příkaz obsahující celkové rozměry obrazu, ze kterého může vypočítat x-ovou hodnotu jeho středu. Dále očekává příkazy s parametry X, W a H určující x-ovou souřadnici středu objektu a jeho šířku s výškou. Pokud Jetson Nano nenalezne ihned po přechodu do tohoto režimu hledaný objekt, indikuje to zasláním x-ové souřadnice jako -1.

V tomto případě se robot nejprve otočí o 360 stupňů a když ani poté není objekt spatřen, spustí se stavový automat zajišťující pohyb robota v prostoru. Tento automat nechá jet robota dopředu do okamžiku, kdy jeden z předních senzorů detekuje náraz. Při nárazu robot krátce zacouvá a poté se otočí přibližně o 90 stupňů. Strana na kterou se otočí je udána podle strany, na které přišel náraz. Pokud přišel náraz na pravé straně otáčí se robot nalevo a naopak. Pokud by strany tří po sobě jdoucích nárazu byly v sekvenci pravá, levá, pravá nebo levá, pravá, levá, lze tím indikovat, že se robot zasekl a pokračoval by dále ve stejných pohybech tam a zpátky. Proto při detekci této sekvence nárazu zacouvá robot o dvojnásobek vzdálenosti a poté se otočí o 270 stupňů. Níže je uveden diagram tohoto automatu. V závorkách jsou uvedeny podmínky pro přechod mezi stavy a bez závorek jsou označeny akce provedené při přechodu.



Obrázek 17: Diagram stavového automatu pro řízení pohybu robota

Pokud v kterékoli fázi pohybu robota dojde k nalezení hledaného objektu, vyrovnává se robot tak, aby byl objekt na středu. Po vyrovnání se robot rozjede dopředu a zastaví se, pokud poměr velikosti objektu a velikosti obrazu dosáhne určité úrovně. Po zastavení rozsvítí LED pro indikaci nalezení objektu.

4.2 Jetson Nano program

Program na Jetson Nano se stará o příjem obrazu z kamery a jejich následné zpracování, podle toho v jakém režimu se robot nachází. Aby došlo ke spuštění programu po startu Jetson Nano bez zásahu uživatele, je potřeba přidat do souboru /.bashrc přidat řádek "python3 /cesta/k/programu.py &".

Kvůli nekompatibilitě knihovny PyBluez s operačním systémem běžícím na Jetson Nano byla pro bluetooth komunikaci použita knihovna socket. S tou se nejprve vytvoří objekt typu socket, u kterého se specifikuje jeho typ (pro bluetooth komunikaci AF_BLUETOOTH) a komunikační protokol, který má používat (BTPROTO_RFCOMM). Tím se vytvoří bluetooth server, který dále čeká na připojení prvního klienta. K přijímání a odesílaní dat slouží funkce recv() a send(). Pro odesílání

je nejprve nutné data zakódovat zvoleným kódováním, například: utf-8. Stejným kódováním se pak data dekódují při příjmu.

Při manuálním režimu Jetson Nano pouze přijímá zprávy z řídícího programu, konvertuje je do podoby příkazů pro Arduino a následně je odešle. Pro ostatní režimy je potřeba získat data z kamery. K tomu lze využít knihovnu OpenCV a její objekt typu VideoCapture. Při inicializování této třídy je možné jí jako argument předat string obsahující konfiguraci kamery. V konfiguraci lze nastavit parametry jako id kamery, velikost snímku, počet snímků za vteřinu nebo převrácení obrazu. Jednotlivé snímky jsou získané funkcí read.

```
"nvarguscamerasrc sensor-id=0 ! "
"video/x-raw(memory:NVMM), width=(int)1920, height=(int)1080, framerate=(fraction)30/1 ! "
"nvvidconv flip-method=0 ! "
"video/x-raw, width=(int)1920, height=(int)1080, format=(string)BGRx ! "
"videoconvert ! "
"video/x-raw, format=(string)BGR ! appsink"
```

Obrázek 18: Konfigurační string pro kameru

4.2.1 Režim parkování

Úkolem Jetson Nano při parkování je lokace LED, které jsou umístěné na nabíjecí stanici. Do Arduina pak posílá informace o jejich x-ových souřadnicích, kde jsou dále zpracovávány. K určení polohy LED využívá program operace prahování společně s morfologickými operacemi. Pro prahování i morfologické operace byly použity funkce knihovny OpenCV.

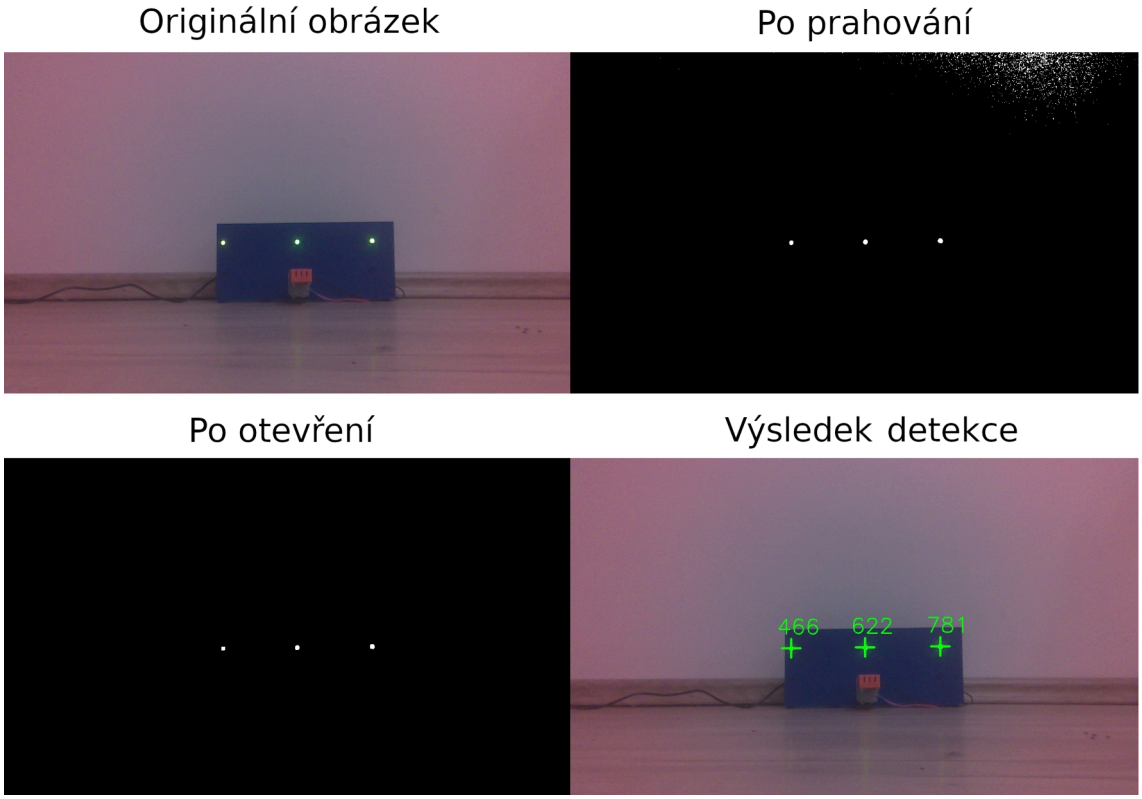
Před započetím parkování odešle Jetson Nano do Arduina informaci o celkové šířce obrazu, ve kterém budou souřadnice počítány. Pro parkování byl zvolen obraz o velikosti 1920x1080 pixelů.

Pro výpočet souřadnic se snímek z kamery nejprve převede na monochromatický obraz, ze kterého se dále pomocí prahování převede do binární podoby, ve které jsou označeny světlé oblasti. Použité LED jsou dostatečně výrazné na to, aby mohl být zvolen práh s hodnotou 200, což zajišťuje, že výsledný binární obraz neobsahuje

mnoho šumu. Pro odstranění zbytku šumu je použita operace otevření se strukturním elementem 5x5. Práh i velikost strukturního elementu byl zvolen tak, aby bylo dosaženo co nejpřesnějších výsledků i při různých světelných podmínkách.

K rozeznání jednotlivých LED na obraze je využita funkce label() z knihovny SciPy. Tato funkce přiřazuje každé skupině spojených pixelů jedinečnou hodnotu. Pro každou tuto hodnotu lze pak vypočítat těžiště. Pro x-ovou souřadnici se sečtou x-ové hodnoty souřadnic všech pixelů s danou hodnotou. Těžiště se poté vypočítá vydělením tohoto součtu s celkovým počtem pixelů dané hodnoty. Pro y-ovou souřadnici je postup obdobný.

Tím vznikne sada tří souřadnic, jejichž x-ové složky posílá program do Arduina. Před odesláním souřadnic do Arduina je nejprve nutné je seřadit. Pokud je některá ze tří LED nenalezena odešle program její souřadnici jako hodnotu -1.



Obrázek 19: Průběh detekce LED

4.2.2 Režim vyhledávání

Při vyhledávání objektů se program na Jetson Nano stará o pořízení vstupního obrazu a určování polohy hledaného objektu. K tomu byl využit algoritmus YOLOv7 pro jeho vysokou rychlost, potřebnou při detekci objektů v reálné čase.

Verze 7 YOLO algoritmu byla zvolena z důvodu kompatibility s python verzí 3.6.9, která je na Jetson Nano ve výchozím stavu nainstalována. Tato verze již přichází s mnoha předinstalovanými balíčky, které jsou přímo optimalizované pro práci s grafickým jádrem, které se na desce nachází. Z tohoto důvodu není produktivní tuto verzi pythonu měnit.

Pro zprovoznění YOLO algoritmu na platformě Jetson Nano je potřeba doinstalovat některé baličky. Mezi ně patří: PyYAML, tqdm, cython, libjpeg, matplotlib, gfortran, libopenblas, liblapack, scipy, typing-extensions, seaborn, torch a torchvision.

Dále je potřeba stáhnout GitHub repozitář yolov7 od autora WongKinYiu, který obsahuje další pomocné funkce pro běh algoritmu. Z tohoto repozitáře se překopírují složky utils a models do složky, ve kterém se nachází program, který chceme využívat pro detekci.

V programu je nejprve nutné načíst YOLO model pomocí funkce attemptLoad(), u které je potřeba zadat cestu k pt souboru obsahující váhy modelu. Funkce pak vrací Pytorch model, který je možný použít k inferenci. Pro model je pak potřeba nastavit velikost vstupního obrazu. Pro účely robota byla zvolena velikost 480x320 pixelů. Dále je nutné určit pravděpodobnostní práh, který určuje, jak moc si musí být algoritmus jistý, že se daný objekt v obraze nachází. Posledním vstupním parametrem modelu je IoU práh. Ten je potřeba v poslední fázi detekce, při které dochází k aplikaci non-max suppression. IoU práh určuje maximální míru překrytí objektů, při které nebudou objekty sloučeny do jednoho.

Po úspěšném načtení modelu je možné zahájit detekci. Pro tu je nejprve nutné konvertovat vstupní obraz na numpy pole. Toto pole se pak transponuje tak, aby v něm bylo uspořádání rozměrů: kanál, výška, šířka. Jednotlivé hodnoty jsou pak dělením normalizovány tak, aby se v poli nacházely čísla od 0 do 1. Takto upravené vstupní data se konvertují na Pytorch tenzor. Funkcí unsqueeze() se přidá další rozměr na začátek tenzoru, což je často požadovaný formát pro vstup do modelů v PyTorch.

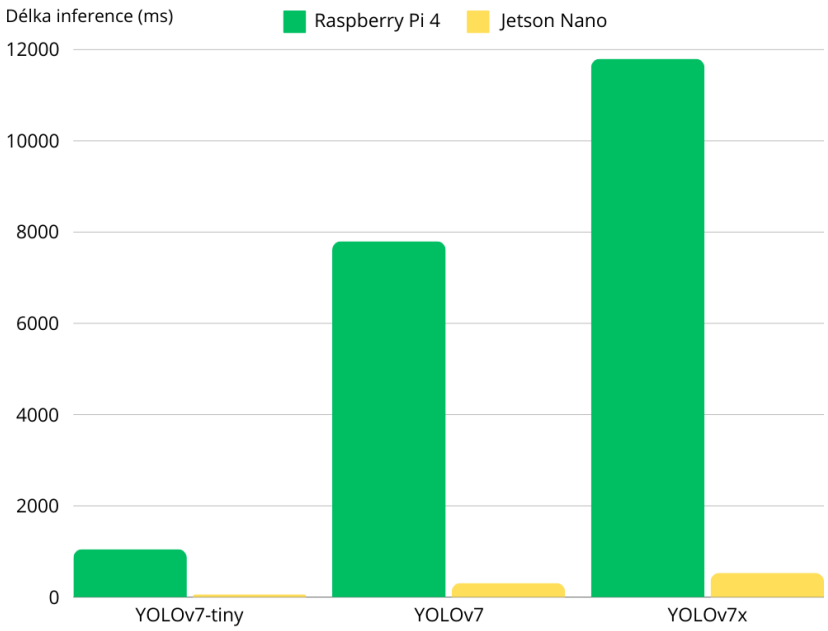
Poté lze provést samotnou inferenci následovanou algoritmem non-max suppression. Tím je získán seznam objektů v obraze společně s jejich rozměry a souřadnicemi. Z tohoto seznamu je vybrán právě hledaný objekt a informace o něm jsou poslány do Arduina.

Obrázek 20: Výsledek YOLO algoritmu

4.2.3 Srovnání rychlosti detekce u Raspberry Pi a Jetson Nano

V původním návrhu robota měla být prováděna detekce objektů na mikropočítači Raspberry Pi 4. Pro detekci se použila velikosti vstupního obrazu 480x320 pixelů a model yolov7-tiny, který je nejmenším možným modelem. Přestože algoritmus YOLO nabízí vysoké rychlosti, zvládalo Raspberry Pi detekovat objekty pouze jednou za vteřinu. Oproti tomu Jetson Nano při stejné konfiguraci dosahuje rychlosti přes patnáct snímku za vteřinu.

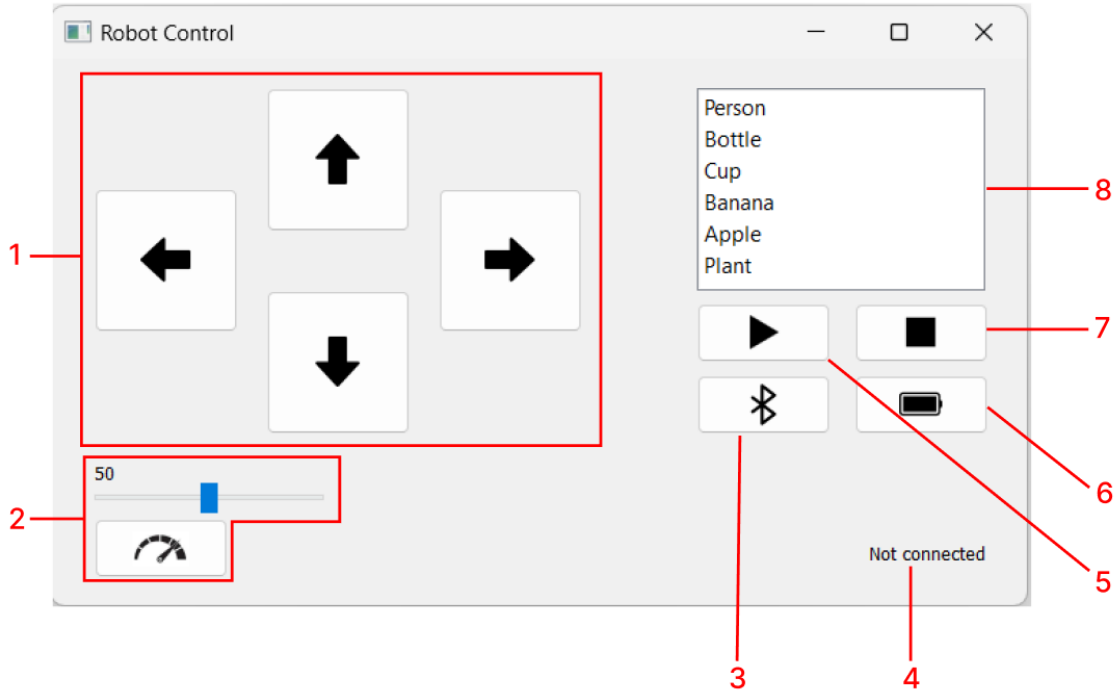
Při použití většího modelu yolov7 se již u obou počítačů několikanásobně prodlužuje doba inference a to konkrétně na 7800 ms u Raspberry Pi a 310 ms u Jetson Nano. U Jetsonu ještě dochází u větších modelů k problému s velikostí RAM. Pro načtení modelu do paměti je nutné povolit swapování na SD kartu, čímž se snižuje její životnost.



Obrázek 21: Délka inference na Jetson Nano a Raspberry Pi

4.3 Desktopový program

Desktopový program má za úkol poskytnout spojení mezi počítačem a robotem. Zároveň umožňuje uživateli používat veškeré funkce robota pomocí grafického rozhraní. Pro tvorbu uživatelského rozhraní byl zvolen framework PyQt5. Pro tvorbu rozložení uživatelského rozhraní lze buď využít funkce tohoto frameworku nebo nástroje Qt Designer. S jeho použitím lze jednotlivé prvky aplikace, jako například tlačítka, umisťovat stylem drag and drop do vytvořených oken. Poté co je v Qt designeru vytvořeno rozložení jednotlivých prvků aplikace, lze z něj exportovat XML soubor s příponou ui. Tento soubor se dále konvertuje na python skript v příkazové řádce pomocí příkazu "pyuic5 input_file.ui -o output_file.py". Po spuštění skriptu se zobrazí navržené okno se všemi prvky, které byly přidány v Qt designeru. Na obrázku (21) je zobrazeno navržené grafické rozhraní.



Obrázek 22: Grafické rozhraní desktopové aplikace

1. Tlačítka pro manuální kontrolu — Pokud se robot nachází v režimu parkování nebo vyhledávání a některé z těchto tlačítek je zmáčknuto, dojde k ukončení tohoto režimu a přepnutí robota do manuálního režimu. Stejnou funkcionalitu jako tyto tlačítka mají i klávesy WSAD, pro řízení robota pomocí klávesnice.

2. Nastavení rychlosti. Posuvník umožňuje zvolit číslo v rozmezí 0 až 100, které určuje rychlost, jakou se robot pohybuje při manuálním řízení. Po stisku tlačítka "set speed" se toto nastavení odešle do robota.

3. Tlačítko pro připojení. Po stisku tohoto tlačítka se otevře nové okno se seznamem dostupných bluetooth zařízení. Pokud nedojde k nalezení robota, je možné obnovit vyhledávání tlačítkem "refresh". Pokud k nalezení dojde zvolí uživatel robota ze seznamu zařízení a tlačítkem "OK" se k němu připojí.

4. Popis stavu robota. Text označuje, ve kterém stavu se robot právě nachází. Stavy mohou být následující:
    - not connected – Robot čeká na bluetooth připojení.
    - Connected – K robotovi se úspěšně připojil bluetooth klient.
    - Manual – Robot se nachází v manuálním módu.
    - Parking – Robot zahájil parkování.
    - Parked – Robot úspěšně vjel do nabíjecí stanice.
    - Locating – Robot vyhledává objekt.
    - Found – Robot úspěšně nalezl objekt.

5. Tlačítko zahajující vyhledávání.
6. Tlačítko zahajující parkování.
7. Tlačítko pro zastavení vyhledávání nebo parkování
8. Seznam objektů pro vyhledávání. V tomto seznamu se nachází seznam všech podporovaných objektů, které může robot vyhledávat.

Aby jednotlivé prvky grafického rozhraní prováděli nějakou funkcionalitu, je jim potřeba přiřazovat posluchače událostí. Pro různé typy prvků existují v PyQt5 různé posluchače. Například pro tlačítko může být posluchač pro jeho stisk nebo uvolnění. Posluchače se v PyQt5 přidělují prvkům pomocí funkce connect(), která jako parametry přijímá název funkce, která se spustí, pokud dojde k dané události.

Pro připojení aplikace k bluetooth serveru, který běží na robotovi slouží knihovna pyBluez. Pomocí této knihovny může aplikace vypsat dostupná zařízení, nacházející se v okolí. Uživatel si pak může vybrat zařízení, se kterým chce navázat spojení, a následně se s ním spojit pomocí Bluetooth socketu. Po úspěšném navázání spojení může aplikace komunikovat se serverem na robotovi, posílat mu instrukce a přijímat zpětné informace.

Závěr

V rámci bakalářské práce byl zkonstruován pásový robot osazený kamerou. Pro řízení byly vybrány mikropočítače Arduino Uno a Jetson Nano. K pohonu robota jsou použity motory AX12+ a soustava kol a pásů ze stavebnice lego education. V rámci konstrukce bylo vytvořeno schéma zapojení, které zajišťuje napájení veškerých komponent a komunikaci mezi nimi.

Dále byl vytvořen systém pro řízení robota. Díky němu je možné s robotem pohybovat, nechat ho automaticky zaparkovat do nabíjecí stanice nebo mu určit objekt k vyhledání. Systém pro řízení robota je složen ze tří částí.

První z nich je program vytvořený pomocí frameworku PyQt5 který s robotem komunikuje pomocí bluetooth. Ten umožňuje uživateli ovládat robota z desktopového prostředí. V programu se nachází prostředky pro připojení k robotovi, k jeho manuálnímu řízení, seznam objektů, které lze vyhledávat a tlačítko pro zaparkování.

Druhou částí je program běžící na Jetson Nano. Ten se stará o získávání dat z kamery a jejich zpracování. Při parkování využívá tento program algoritmy pro zpracování obrazu například: prahování nebo morfologické operace k určení polohy LED, které se na nabíjecí stanici nachází. Pro vyhledávání objektů byl zvolen algoritmus YOLO, který je nejrychlejším dostupným řešením. V rámci práce byla také porovnána rychlost tohoto algoritmu na platformách Jetson Nano a Raspberry Pi. Díky integrovanému grafickému jádru byl Jetson až 15x rychlejší oproti Raspberry Pi, které dosahovalo rychlostí detekce pouze jeden snímek za vteřinu.

Třetí částí řídícího systému je program vytvořený pro Arduino, který vyhodnocuje výsledky z Jetson Nano a senzorů. V rámci parkování se stará o to, aby robot úspěšně vjel do stanice téměř z jakékoli startovní polohy. Při vyhledávání objektů řídí robota pomocí stavového automatu tak, aby se nezasekl.

Použitá literatura

[1] CHHIKARA, Prateek. Understanding Morphological Image Processing and Its Operations. online. 2022. In: *Medium*. Dostupné z: https://towardsdatascience.com/understanding-morphological-image-processing-and-its-operations-7bcf1ed11756. [citováno 2024-04-29].

[2] A.S, Moravské přístroje. *Základní principy strojového vidění – 8. díl - ElektroPrůmysl.cz*. Webové sídlo. 2024. Dostupné z: https://www.elektroprumysl.cz/automatizace/zakladni-principy-strojoveho-videni-8-dil. [citováno 2024-04-29].

[3] SONKA, Milan; Václav HLAVÁČ a Roger BOYLE. *Image processing, analysis, and machine vision*. 3. ed. Toronto: Thompson Learning, 2008. ISBN 978-0-495-24438-7.

[4] DOUGHERTY, Geoff. *Digital image processing for medical applications*. Cambridge, UK ; New York: Cambridge University Press, 2009. ISBN 978-0-521-86085-7.

[5] SAMINA. What is Canny edge detection? online. In: *Educative*. Dostupné z: https://www.educative.io/answers/what-is-canny-edge-detection. [citováno 2024-04-30].

[6] *What are Convolutional Neural Networks? | IBM*. Webové sídlo. 2024. Dostupné z: https://www.ibm.com/topics/convolutional-neural-networks. [citováno 2024-04-30].

[7] PILÁT, Martin. *Martin Pilát*. Webové sídlo. Dostupné z: https://martinpilat.com/cs/prirodou-inspirovane-algoritmy/neuronove-site-konvolucni-site-zpracovani-obrazu. [citováno 2024-04-30].

[8] ANANTH, Shilpa. Faster R-CNN for object detection. online. 2020. In: *Medium*. Dostupné z: https://towardsdatascience.com/faster-r-cnn-for-object-detection-a-technical-summary-474c5b857b46. [citováno 2024-04-30].

[9] GAD, Ahmed Fawzy. Faster R-CNN Explained for Object Detection Tasks. online. 2020. In: *Paperspace Blog*. Dostupné z: https://blog.paperspace.com/faster-r-cnn-explained-object-detection/. [citováno 2024-04-30].

[10] DEMARCQ. What is YOLO? An In-Depth Introduction to Object Detection in Computer Vision. online. In: *ikomia*. Dostupné z: https://www.ikomia.ai/blog/what-is-yolo-introduction-object-detection-computer-vision. [citováno 2024-05-08].

[11] KUNDU, Rohit. *YOLO Algorithm for Object Detection Explained [+Examples]*. Webové sídlo. Dostupné z: https://www.v7labs.com/blog/yolo-object-detection, https://www.v7labs.com/blog/yolo-object-detection. [citováno 2024-04-30].

[12] *Arduino Docs | Arduino Documentation*. Webové sídlo. Dostupné z: https://docs.arduino.cc/. [citováno 2024-04-29].

[13] *Raspberry Pi Documentation - Microcontrollers*. Webové sídlo. Dostupné z: https://www.raspberrypi.com/documentation/microcontrollers/. [citováno 2024-04-30].

[14] NVIDIA Jetson Nano. online. In: *NVIDIA*. Dostupné z: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/product-development/. [citováno 2024-04-30].