Pokročilý kamerový systém pro zemědělskou techniku

Lukáš Darebníček

Bakalářská práce
2024

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2023/2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Lukáš Darebníček**
Osobní číslo:	**A21047**
Studijní program:	**B0613A140020 Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Pokročilý kamerový systém pro zemědělskou techniku**

Zásady pro vypracování

1. Pro svou aplikaci vyberte vhodnou kameru a hardwarovou platformu.
2. V rozsahu potřebném ke zpracování práce se seznamte s algoritmy zpracování obrazu.
3. Z vybraných komponent sestavte kamerový systém tak, aby byl připraven k instalaci do zemědělského vozidla.
4. K sestavenému kamerovému systému vytvořte vhodný firmware.
5. Doplňte programové vybavení kamerového systému o rozšířenou realitu umožňující odhadovat vzdálenost objektu za vozidlem.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. CARROLL, Brandon. *Bezdrátové sítě Cisco : autorizovaný výukový průvodce* [online]. Brno: Computer Press, 2011. ISBN 9788025128848.
2. MATOUŠEK, Radomil. *Práce s inteligentními displeji LCD* [online]. Praha: BEN – technická literatura, 2006. ISBN 8073001217.
3. MIRON, Douglas. *Small antenna design* [online]. Burlington, MA: Newnes/Elsevier, 2006 [cit. 2023-11-09]. ISBN 9780080498140. Dostupné z: https://vufind.katalog.k.utb.cz/Record/73878/Description#tabnav
4. SCHMALSTIEG, Dieter a Tobias HÖLLERER. *Augmented reality : principles and practice*. Boston: Addison-Wesley, 2016. ISBN 9780321883575 0-321-88357-8.
5. FROZE, Roger. *Augmented reality for beginners : principles and practices for augmented reality and virtual computers*. CreateSpace Independent Publishing Platform, 2016. ISBN 9781539919377.

Vedoucí bakalářské práce: **Ing. Martin Pospíšilík, Ph.D.**
Ústav elektroniky a měření

Datum zadání bakalářské práce: **5. listopadu 2023**
Termín odevzdání bakalářské práce: **13. května 2024**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
DĚKAN
L.S.

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 5. ledna 2024

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.
- Prohlašuji, že při tvorbě této práce jsem použil/a nástroj generativního modelu AI Chat GPT 3.5; https://chat.openai.com za účelem úpravy textu a návrhem vhodných slov, pomoci s programováním a perplexity.ai; https://www.perplexity.ai za účelem formátování textu a ověření nalezených informací. Po použití tohoto nástroje jsem provedl/a kontrolu obsahu a přebírám za něj plnou zodpovědnost. Při používání nástrojů AI je důležité také rozlišovat, zda jejich využití ovlivnilo samotný obsah předkládané práce.

Ve Zlíně, dne 06. 05. 2024 Lukáš Darebníček, v.r.

ABSTRAKT

Tato práce se zaměřuje na výběr vhodných komponent pro konstrukci parkovací kamery určené pro zemědělskou techniku. Klíčovou součástí systému je mikropočítač s dostatečným výkonem pro zpracování obrazu a měření vzdálenosti k překážce za vozidlem v reálném čase. Dále jsou důležité komponenty, jako je kamera a ultrazvukový senzor, které společně umožňují detekci překážek a bezpečné parkování. Analytický pohled na programy pro zpracování obrazu nám poskytne hlubší pochopení fungování celého systému. Postupné programování a testování jednotlivých částí kódu nám pak umožní lépe porozumět problematice a získat komplexní přehled o realizaci daného úkolu.

Klíčová slova: domácí parkovací kamera, zemědělská technika, mikropočítač, zpracování obrazu, měření vzdálenosti, kamera, ultrazvukový senzor, detekce překážek, bezpečné parkování.

ABSTRACT

This thesis focuses on selecting suitable components for constructing a home parking camera for agricultural machinery. A vital system component is a microcontroller with sufficient power for real-time image processing and distance measurement behind the vehicle. Other essential components include the camera and ultrasonic sensor, enabling obstacle detection and safe parking together. An analytical examination of image processing programs will provide a deeper understanding of the entire system's functionality. Sequential programming and testing of individual code segments will allow us to grasp the issues better and gain a comprehensive overview of task implementation.

Keywords: home parking camera, agricultural machinery, microcontroller, image processing, distance measurement, camera, ultrasonic sensor, obstacle detection, safe parking.

Rád bych vyjádřil upřímné poděkování vedoucímu mé bakalářské práce, panu Martinu Pospíšilíkovi, za jeho cenné rady, trpělivost a odborné vedení během celého procesu tvorby této práce. Dále děkuji své přítelkyni, rodině a přátelům za jejich neustálou podporu a pochopení během mého studia.

Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Obsah

ÚVOD

V dnešní době se moderní technologie stávají nedílnou součástí každodenního života, a to i v oblasti zemědělství a pracovních procesů s ním spojených. S narůstající digitalizací a automatizací se objevují nové možnosti využití technologií, které mohou významně usnadnit a zefektivnit práci v zemědělství. Jedním z takových aplikovaných technologických řešení je vývoj parkovací kamery určené speciálně pro zemědělskou techniku.

Současné parkovací kamery nejsou primárně navrženy pro potřeby zemědělců a často jim chybí klíčové prvky, jako je měření vzdálenosti a rozšiřitelnost. Přenos signálu je často realizován pomocí vodičů, které by se mohly při použití v zemědělství poškodit kvůli velkému množství pohyblivých součástí strojů. Tato skutečnost znamená, že zemědělci nemají k dispozici adekvátní nástroje pro bezpečné a efektivní parkování svých strojů. Proto jsem se rozhodl iniciovat tento projekt s cílem vyvinout a integrovat domácí parkovací kameru speciálně navrženou pro potřeby zemědělců.

Cílem této práce je navrhnout a realizovat systém, který umožní řidičům zemědělské techniky snadné a bezpečné parkování při zpětném pohybu. To zahrnuje výběr a integraci komponent nezbytných pro konstrukci domácí parkovací kamery, která bude schopna splnit specifické požadavky zemědělských operací. Věřím, že tento projekt může přinést významné vylepšení v oblasti bezpečnosti a efektivity práce na poli a poskytnout zemědělcům praktický nástroj pro každodenní použití.

I. TEORETICKÁ ČÁST

1 VÝBĚR VHODNÝCH HARDWAROVÝCH KOMPONENT

Tato kapitola je zaměřena na popis a výběr vhodných hardwarových komponent pro zemědělskou kameru. Budou detailněji rozebrány vlastnosti dostupných mikropočítačů, kamer a ultrazvukových senzorů. Budou porovnány vlastnosti těchto komponent, a následně vybrána ta nejvhodnější varianta s ohledem na výkon, cenu, kompatibilitu a možnosti rozšiřitelnosti.

1.1 Popis jednotlivých komponent z obecného hlediska

Kapitola se zaměřuje na rozbor jednotlivých klíčových komponentů z pohledu jejich funkce, role a vzájemného propojení v rámci celého systému. Tato analýza poskytne ucelený přehled o tom, jak jednotlivé komponenty přispívají k fungování systému.

1.1.1 Mikropočítač

Samotný pojem „mikropočítač“ se postupem času měnil. Dříve, v 60. až 80. letech 20. století, byl pojem mikropočítač jakýsi protiklad „velkých počítačů“. [1] Pod označením „velký počítač“ si můžeme představit velké a výkonné zařízení, které mělo velkou kapacitu a byly navrženy pro zpracování většího obsahu dat. V dnešní době si pod pojmem mikropočítač také představíme mnohonásobně menší zařízení, než je stolní počítač.

Lze tedy tvrdit, že mikropočítač je jednotka, která se by se mohla označit za „malý stolní počítač“. Mikropočítače jsou nejčastěji založeny na Von Neumannově architektuře, která spojuje paměť pro data a paměť pro instrukce. [2] Obsahují mikroprocesor, aritmetickologickou jednotkou s registry, paměti, řadiče a vstupně/výstupní porty, tzv. I/O porty. Monitor může, ale také nemusí být přímo integrovaný na desce mikropočítače. Pokud není monitor integrovaný, připojuje se k moderním mikropočítačům nejčastěji pomocí HDMI a DisplayPortu. Klávesnici a myš lze připojit pomocí rozhraní USB nebo bezdrátovým připojením Bluetooth.

Tímto popisem velmi často označujeme zařízení jako Raspberry Pi, Arduino nebo BeagleBone, která jsou schopna mít vlastní operační systém a provádět složitější operace. Tento popis je velmi podobný popisu mikrokontroleru, ale mikrokontroler je jednotka, která řídí jednoduchá elektrická zařízení, jako jsou například domácí spotřebiče. Mikrokontrolery bývají často menší, levnější a energeticky méně náročnější než mikropočítače, a často jsou navrženy pro specifické účely. Nejsou tedy tolik univerzální, jako mikropočítače.

1.1.1.1 Architektury mikropočítačů

Mezi základní architektury mikropočítačů můžeme zařadit Harvardskou a Von Neumannovu architekturu. Jak bylo již zmíněno, Von Neumannova architektura [3] sjednocuje paměť pro data a pro instrukce řídící jednotky. Z toho vyplývá, že řídící jednotka čte potřebná data a instrukce přes jednu sběrnici. Jedná se o sekvenční (postupné) zpracování dat. Mezi hlavní výhody patří jednoduchá architektura (stačí nám pouze jedna datová sběrnice), není nutno rozlišovat instrukce pro přístup k paměti dat a paměti programu. Mezi hlavní nevýhody patří omezená propustnost dat a instrukcí, protože jsme omezeni tím, že můžeme v jednom okamžiku pouze číst nebo zapisovat. [4] Chybou programu je možné při ukládání dat přepsat i část programu. Von Neumannova architektura se nejčastěji používá u univerzálních počítačů.

Harvardská architektura [3] oproti Von Neumannově odděluje paměť pro data a paměť pro instrukce. Řídící jednotka má oddělené sběrnice, což znamená, že se zde uplatňuje paralelní (souběžné) provádění operací. Díky tomu může mít každá z pamětí jiné velikosti, délku slova, časování, technologii a typ. Také sběrnice mohou mít jiné šířky. Tato konfigurace nám umožňuje vyšší rychlost při vykonávání instrukcí, protože je řídící jednotka schopna číst instrukce a data zároveň. Složitější architektura a implementace spolu s vyššími náklady kvůli potřebě oddělených sběrnic a pamětí jsou hlavními nevýhodami této architektury. Nejčastěji se Harvardská architektura používá u jednoúčelových mikrokontrolérů a specializovaných DSP (Digitální Signálové Procesory), obvykle v audio nebo také video technice.

1.1.1.2 Mikroprocesor

Mikroprocesor nebo také CPU je centrální jednotka provádějící většinu výpočtů a řízení systému. Samotný mikroprocesor se skládá ze dvou základních částí: řadič a aritmetickologická jednotka. [5] Řadič řídí funkce mikropočítače, generuje řídící impulzy, dekóduje instrukce a v případě potřeby je modifikuje (při skoku programu) a vysílá do ostatních částí mikropočítače číslicové řídící signály potřebné pro provádění instrukcí. V samotném řadiči se nachází ještě registr instrukce, který uchovává operační kód právě vykonávané instrukce po dobu jejího vykonávání, programový čítač obsahující adresu instrukce, jež se bude vykonávat. Jestliže má dojít ke skokové instrukci, signály z řadiče neinkrementují čítač, ale dojde k přenesení nové adresy do programového čítače. Dále je součástí i registr adresy instrukce, který uchovává adresu právě vykonávané instrukce. ALU

zpracovává data přicházející do procesoru. [3] Provádí aritmetické (sčítání a odčítání) a logické (AND, OR, NOT) operace. Na základě signálů výběru operace provede příslušnou operaci s jedním či více operandy (vstupními proměnnými). Přivedení operandů a jejich uložení zajišťuje podle dekódované instrukce řadič. Alu používá střadač (akumulátor) jako hlavní pracovní registr k ukládání jednoho zpracovávaného operandu a také k uložení výsledku. V registru příznaků se nastavují příznakové bity podle výsledku ukončené operace a mohou signalizovat například přetečení registru, nulový výsledek nebo záporný výsledek.

1.1.1.3 Paměti

Počítačová paměť je zařízení, které slouží k ukládání dat, se kterými počítač pracuje. Tyto paměti lze rozdělit na tři základní skupiny [6]:

1) Registry: jsou malá paměťová místa na čipu procesoru, která jsou využívána pro krátkodobé ukládání dat, se kterými procesor právě pracuje. Mají nejmenší kapacitu dat (jednotky bytů).
2) Vnitřní (interní nebo také operační) paměti: tyto paměti jsou často vloženy do základní desky. Jsou na ně vkládána data z právě spuštěných programů, s nimiž procesor pracuje v danou chvíli. Mají větší kapacitu dat (100 kB až 100 MB).
3) Vnější (externí) paměti: tyto paměti jsou velmi často realizovány pomocí výměnných médií v podobě disků, které využívají zápis pomocí magnetické nebo optické technologie. Slouží k dlouhodobému uchovávání dat. Tento typ paměti má největší kapacitu dat (řádově od 500 MB až 2 TB).

Mezi základní parametry lze zařadit například přístupovou dobu (doba, za kterou paměť zpřístupní danou informaci), kapacitu (neboli objem dat, jež můžeme vložit do paměti), přenosovou rychlost (množství dat, které lze z paměti přečíst anebo i zapsat za jednotku času). Také je lze rozdělit podle toho, jak udržují informace, a to na statické a dynamické paměti (statické paměti uchovávají danou informaci po celou dobu, kdy je paměť připojena ke zdroji, kdežto dynamické paměti ztrácejí informaci z paměti i když jsou připojeny ke zdroji napájení. Je nutné je tedy v určitých intervalech oživovat, aby danou informaci neztratily) a pak na energicky závislé a nezávislé (energicky závislé paměti ztratí všechny uložené informace po odpojení zdroje elektrické energie a energicky nezávislé paměti neztratí žádné informace, i když se odpojí zdroj elektrického napájení).

V poslední řadě je ještě třeba zmínit paměti typu ROM, RAM a CASHE, se kterými mikropočítače často pracují. [6]

1) ROM (read only memory) je paměť, v níž se informace uchovávají i po vypnutí počítače. Tato paměť se velmi často využívá pro uložení BIOSu.

   ROM je již výrobcem nadefinována a slouží uživateli pouze pro čtení. Je realizována pomocí elektrického odporu či pojistek, které se vhodně zapojí a po připojení k elektrické síti se na každé „buňce“ ukáže logická 1 anebo 0.

   PROM (programovatelná ROM) umožňuje uživateli jednou zapsat určité množství informace a poté se z ní stane klasická ROM paměť sloužící pouze pro čtení.

   EPROM (vymazatelná PROM) – na tuto paměť je možné opakovatelně zapisovat informace, které se zde uchovávají pomocí elektrického náboje. Odstraňování dat se provádí pomocí ultrafialového záření.

   EEPROM (elektrická EPROM) – jedná se o EPROM, kterou lze mazat pomocí elektrických impulzů.
   Flash-EPROM – jedná se o nejrychlejší přepisovatelný typ. Tato paměť se po odpojení napětí vymaže a je programovatelná přímo z počítače. Počet cyklů (přepisů) je největší a pohybuje se okolo 1000 cyklů.
2) Paměť typu RAM (random access memory) nejčastěji operuje samotný procesor. Je rychlejší než ROM paměť a také má větší kapacitu a dělí se na statickou (SRAM) dynamickou (DRAM) paměť. SRAM je tvořena bistabilním klopným obvodem, díky němuž má nízkou přístupovou dobu. Tyto obvody není potřeba obnovovat, protože klopný obvod se postupem času nemění. DRAM je paměť tvořená kondenzátory, které nám reprezentují logickou 1 a 0. Tyto kondenzátory je nutné opakovaně dobíjet (provádět refresh), jelikož se bez elektrického napětí postupem času vybíjí.
3) Paměť CACHE je typ paměti, jenž urychluje přístup k datům, která se často používají nebo mají tendenci být používány opakovaně. L1 cache je paměť umístěná nejblíže k jádru procesoru a velmi často bývá integrována do jádra procesoru. Má nejmenší kapacitu (řádově v jednotkách až stovkách kilobajtů), avšak má nejkratší přístupovou dobu. Jsou v ní velmi často uloženy instrukce či data, která jsou právě používána nebo mají větší tendenci být používána procesorem. L2 cache bývá od jádra procesoru dál než L1 cache. Má větší paměť (pro představu v řádech stovek až tisíců kilobajtů) a je pomalejší. Tato paměť je umístěna strategicky mezi operační pamětí a jednotkami procesoru, aby omezovala snižování rychlosti procesoru rychlostí operační paměti. L3 cache je nejdál od procesoru a je často sdílena mezi všemi jádry

procesoru. Má největší kapacitu ze všech cache pamětí (jednotky až stovky megabajtů), zato je z uvedených CACHE pamětí nejpomalejší.

1.1.1.4 Potřebné I/O rozhraní

USB porty (Universal Serial Bus) slouží k připojení různých periferních zařízení, jako jsou klávesnice, myši, tiskárny, USB flash disky, externí disky a další. HDMI port (High-Definition Multimedia Interface) slouží k připojení monitorů, televizorů nebo jiných zobrazovacích zařízení, které podporují digitální video a zvukové signály. Díky HDMI rozhraním lze přenášet vysoké rozlišení obrazu a kvalitní zvukový výstup, což je ideální pro multimediální aplikace a sledování obsahu ve vysokém rozlišení. Ethernet port (RJ-45) umožňuje připojení mikropočítače k síti Ethernet pro přenos dat a přístup k internetu. Tato rozhraní poskytují stabilní a spolehlivé propojení s lokální sítí nebo internetem, což je důležité pro komunikaci a přenos dat mezi zařízeními. GPIO porty (General Purpose Input/Output) poskytují obecné digitální vstupy a výstupy, které mohou být použity pro různé účely. Porty umožňují ovládání LED, čtení stavu tlačítek, propojení se senzory a další interakce s okolním prostředím. Napájecí konektory, jež jsou nejčastěji realizovány přes již zmíněný USB, dodávají potřebnou dávku energie pro správnou práci mikropočítače. Je nutné brát na zřetel doporučené vstupní napětí a proud určené výrobcem a podle těchto kritérií vhodně volit zdroj. Sloty pro paměťové karty, jako jsou SD karty, poskytují možnost rozšíření paměti mikropočítače a ukládání dat. Sloty umožňují snadné a flexibilní rozšíření úložné kapacity zařízení pomocí paměťových karet různých typů a kapacit.

1.1.2 Kamera

Kamera je z obecného hlediska elektronické zařízení sloužící k zachycení obrazu, jejž je následně převeden do digitální podoby. Kamery se skládají z několika částí, jako je objektiv, fotosenzor, procesor a další.

1.1.2.1 Objektiv

Objektiv je klíčovou součástí fotografické kamery, umožňuje vstup světla do vnitřních částí kamery a následné vytvoření obrazu na fotosenzoru. Jeho primární funkcí je ohýbání vstupujícího světla a zaostření obrazu. Objektiv bývá často složen z jedné nebo více čoček či skel, jež umožňují regulaci světla a zaostření. Jednotlivé části objektivu mohou mít různé ohniskové vzdálenosti, což ovlivňuje úhel záběru a hloubku ostrosti. Jako primitivní objektiv

může sloužit jednoduchá čočka nebo dokonce pouhý otvor v neprůsvitném materiálu. V praxi se však často používá soustava několika čoček, čímž dojde k lepšímu potlačení optických vad a zajištění lepší kvality obrazu. Tato optická soustava může být schopna i změny své ohniskové vzdálenosti, což se nazývá zoomování. Součástí objektivu je také clona regulující množství světla vstupujícího do kamery. Clona umožňuje kontrolu expozice snímku a přizpůsobení se různým světelným podmínkám. Některé objektivy mohou mít integrovanou závěrku, která reguluje dobu expozice snímku. Objektivy se dále dělí podle své ohniskové vzdálenosti, což ovlivňuje jejich úhel záběru. Normální objektiv má úhel záběru přibližně 50° [7], což odpovídá přirozenému úhlu vnímání lidského oka. Širokoúhlý objektiv má kratší ohniskovou vzdálenost a umožňuje širší záběr, zatímco teleobjektiv má užší úhel záběru a umožňuje fotografování vzdálených objektů.

1.1.2.2 Fotosenzor

Fotosenzor je tvořen velkým množstvím malých buněk nazývaných pixely, které jsou citlivé na světlo. Každý z těchto pixelů zachytí světlo, jež dopadá do jeho oblasti a tento světelný signál následně převede na elektrický signál. Jednotlivé pixely mají své čtecí obvody, které naměří množství elektrického náboje vygenerovaného světlem a poté jej převedou na digitální signál [8]. Fotosenzory je možné rozdělit na dva hlavní typy: CCD (Charge-Coupled Device) a CMOS (Complementary Metal-Oxide-Semiconductor).

U systému CCD je signál přesouván přes matici fotodiod do výstupního zesilovače, což umožňuje vyšší citlivost na světlo a rychlejší přenos signálu. Nevýhodou je vyšší cena a ovlivnění náboje okolních pixelů v případě velmi světlých objektů, čímž je snížena kvalita výsledného obrazu. Naopak u CMOS systému jsou buňky seřazeny do sloupců a řádků, což umožňuje nižší spotřebu energie a levnější výrobu senzorů.

1.1.3 Ultrazvukový senzor

Ultrazvukový senzor, také nazývaný ultrazvukový snímač, je zařízení schopné měřit vzdálenost pomocí ultrazvukových signálů vysílaných do okolí. Měření se provádí ve dvou fázích, kdy senzor vyšle ultrazvukovou vlnu daným směrem. Vyslaný signál se šíří prostředím a při styku s objektem se vrací zpět k senzoru. Mezi vysláním dané vlny a jejím přijetím uběhne doba, kterou senzor spočítá. Na základě změřeného času poté určuje vzdálenost překážky od sebe. Generované ultrazvukové vlny jsou obvykle v rozsahu 20 kHz až 200 kHz, což je mimo dosah lidského sluchu [9]. Citlivost ultrazvukového senzoru je

velmi ovlivněna povrchem objektu, od kterého se signál odráží, a úhlem odrazu, vlhkostí a teplotou okolního prostředí. Například lépe zjistí vzdálenost objektu s hladkým povrchem, jenž je kolmý ke směru vysílání vln, v teplotně stabilním prostředí s nižší vlhkostí vzduchu, než objektu s nerovným či houbovitým povrchem při extrémních teplotách s vysokou vlhkostí vzduchu.

1.2 Mikropočítač Raspberry Pi

Tato podkapitola se zabývá pouze vybranými modely Raspberry Pi a Arduino, které by mohly být použity ke konstrukci kamerového systému. V tabulce jsou uvedeny základní sledované parametry ovlivňující faktory při rozhodování. Ceny včetně DPH byly odečteny z internetového obchodu rpishop.cz dne 29.02.2024.

1.2.1 Historické pozadí

Historicky sahá původ Raspberry Pi až do roku 2006, kdy Eben Upton, Rob Mullins, Jack Lang a Alan Mycroft spustili projekt, ve kterém se snažili vyřešit nedostatek technických vědomostí studentů ve Spojeném království [10]. Začali se zabývat návrhem malého a cenově dostupného počítače. Z prvopočátku se jim nedařilo cíle dosáhnout, ale poté, co vznikla Raspberry Pi Foundation ve spolupráci s Petem Lomasem a Davidem Brabenem, se podařilo vytvořit model Raspberry Pi 1 model B, jenž vyšel na trh roku 2012. Stal se natolik populární, že v dnešní době je na trhu mnoho druhů a verzí Raspberry Pi, které mají různé vlastnosti, vylepšení a funkce. Díky své variabilitě má tento mikropočítač veliké využití ve vzdělávání, průmyslu, domácí automatizaci, robotice a dokonce i vědeckých experimentech.

1.2.2 Hardware

Hardware je postaven na platformě ARM (architektura procesorů s nízkou spotřebou energie) [11], díky níž má dostatečně velký výpočetní výkon pro běžné úlohy. Mezi základní prvky hardwarové konfigurace patří procesor, RAM paměť, grafický výstup, porty pro připojení periferií, slot pro microSD kartu a také GPIO. V důsledku toho, že se jedná o otevřený design, má mnoho možností rozšíření a přídavných modulů, což z něj činí ideální nástroj pro edukaci, vědecké experimenty a automatizaci.

1.2.3 Rozdělení a výběr modelů Raspberry pi

Po delším průzkumu byl vybrán model Raspberry Pi 4 z těchto důvodů: je vybaven čtyřjádrovým procesorem s dostatečně velkým výkonem pro zpracovávání obrazu a měření

vzdálenosti a spolu s 2 GB paměti RAM je možné dosáhnout lepších výsledků. Cenově je také výhodnější než Raspberry Pi 2 model B.

Tabulka 1: Tabulka rozdělení modelů Raspberry Pi

Model	Výkon	Konektory	RAM	Ethernet	Bezdrátové připojení	Rozměry v cm	Cena
Raspberry Pi Zero	1 GHz jednojádrový ARM11	Mini HDMI, Micro USB	512 MB	Ne	Ne	6,5 x 3,0	280 Kč
Raspberry Pi Zero W	1 GHz jednojádrový ARM11	Mini HDMI, Micro USB	512 MB	Ne	Wi-Fi, Bluetooth	6,5 x 3,0	440 Kč
Raspberry Pi 1 Model B+	700 MHz jednojádrový ARM11	HDMI, USB, RCA	512 MB	Ano	Ne	8,5 x 5,6	770 Kč
Raspberry Pi 2 Model B	900 MHz čtyřjádrový ARM Cortex-A7	HDMI, USB, Ethernet	1 GB	Ano	Ne	8,5 x 5,5	1400 Kč
Raspberry Pi 3 Model B	1.2 GHz čtyřjádrový ARM Cortex-A53	HDMI, USB	1 GB	Ano	Wi-Fi, Bluetooth	8.5 x 5.6	940 Kč
Raspberry Pi 4 Model B	1,5 GHz čtyřjádrový ARM Cortex-A72	2x HDMI, 2x USB	2/4/8 GB	Ano	Wi-Fi, Bluetooth	6,5 x 4,0	1200–2010 Kč

1.3 Mikropočítač Arduino

Kapitola přináší historické pozadí a stručný přehled architektury a modelů mikropočítačů Arduino. Dále kapitola popisuje základní hardwarovou konfiguraci Arduino desek, včetně mikrokontroléru, paměti a I/O rozhraní. Poté následuje přehled dostupných modelů Arduino, včetně jejich parametrů jako MCU, flash paměť, SRAM, počet I/O pinů a cena. Toto rozdělení umožňuje čtenáři lépe porozumět vlastnostem a možnostem jednotlivých modelů Arduino pro jejich projekty.

1.3.1 Historické pozadí

Mikropočítač Arduino vznikl v italském městě Ivrea v roce 2003, když skupina inženýrů, včetně Massima Banziho, Hernanda Barragana a Davida Cuartiellesa, chtěla vytvořit jednoduchý a cenově dostupný nástroj pro studenty, umělce, designéry a nadšence v oblasti elektroniky [12]. Cílem bylo poskytnout platformu, která by umožňovala snadné a rychlé prototypování interaktivních zařízení a otevřené sdílení jejich návrhů a kódů. První Arduino deska byla uvedena na trh v roce 2005 a okamžitě získala popularitu mezi nadšenci z celého

světa. Od té doby bylo vydáno mnoho dalších verzí a variant nabízejících různé vlastnosti a možnosti pro různé projekty a aplikace.

1.3.2 Hardware

Arduino je postaven na platformě mikrokontroléru ATMega, který poskytuje dostatečný výkon a flexibilitu pro mnoho různých aplikací. Základními prvky hardwarové konfigurace Arduino desky jsou mikrokontrolér, paměť pro ukládání programu a dat, analogové a digitální vstupy a výstupy, USB konektor pro programování a komunikaci s počítačem a napájecí konektor pro připojení k napájecímu zdroji. Desky Arduino mají také pinové hlavičky umožňující snadné připojení různých periferií a rozšíření, jako jsou senzory, motory a displeje. [12]

1.3.3 Rozdělení a výběr modelů Arduino

Z nabízených dostupných modelů Arduino nebyl vybrán žádný model, který by vyhovoval aplikaci, protože modely Arduino uvedené v tabulce nemají dostatečný výkon pro zpracovávání obrazu. Pokud by byla aplikace postavena na platformě Arduino, pravděpodobně by nebylo dosaženo vysoké kvality výsledného obrazu.

Tabulka 2: Tabulka rozdělení modelů Arduino

Model	MCU	Flash paměť	SRAM	I/O piny	Ethernet	Bezdrátové připojení	Rozměry v cm	Cena
Arduino Uno Rev3	ATmega328P 16 MHz	32 KiB	2 KiB	14 digitálních, 6 analogových	Ne	Ne	6,8 x 5,3	700 Kč
Arduino Leonardo	ATmega32U4 16 MHz	32 KiB	2,5 KiB	20 digitálních, 7 analogových	Ne	Ne	6,8 x 5,3	670 Kč
Arduino Mega 2560 Rev3	ATmega2560 16 MHz	256 KiB	8 KiB	54 digitálních, 16 analogových	Ne	Ne	10,2 x 5,3	1250 Kč
Arduino Due	SAM3X8E 84 MHz	512 KiB	96 KiB	54 digitálních, 12 analogových	Ne	Ne	10,2 x 5,3	1230 Kč
Arduino Nano 33 IoT	SAMD21G18A 48 MHz	256 KiB	32 KiB	22 digitálních, 8 analogových	Ne	Wi-Fi, Bluetooth	4,5 x 1,8	630 Kč
Arduino Portenta H7	STM32H747XI 480 MHz	26 MB	1 MB	128 digitálních, 16 analogových	Ano	Wi-Fi, Bluetooth	10,5 x 5,3	2800 Kč

1.4 Výběr ultrazvukového senzoru

Při výběru ultrazvukového senzoru bylo rozhodováno mezi modelem HC-SR05 a US-100. Senzor HC-SR05 je senzor měřící vzdálenost od 2 cm do 400 cm s přesností asi ±0,3 cm. Napájecí napětí je uváděno výrobcem 5 VDC. Efektivní úhel je do 15°. US-100 senzor má stejný efektivní úhel (<15°) a přesnost měření (±0,3 cm), ale má odlišné napájecí napětí pohybující se mezi 3,3 a 5 VDC. Také má větší dosah měření, a to až 450 cm. Výhoda senzoru US-100 spočívá v měřící vzdálenosti a napěťové kompatibilitě s platformou Raspberry Pi, která rovněž používá napětí 3,3 VDC. Jeho hlavní nevýhodou je ovšem vysoká cena, která se pohybuje okolo 140 Kč za jeden senzor. Náklady senzoru HC-SR05 se pohybují okolo 40 Kč za jeden senzor, což je v porovnání s předchozím senzorem téměř o 72 % levnější. Jedinou nevýhodou senzoru HC-SR05 je, že pracuje na 5 VDC, a tedy je potřeba odporového děliče ke snížení výstupního signálu na 3,3 VDC. Maximální vzdálenost měření by se mohla taktéž považovat za nevýhodu, jelikož 50 cm je poměrně veliký rozdíl, ale na aplikaci kamerového systému to nemá tak velký vliv, protože měření vzdálenosti je pouze podpůrná funkce. Závěrem byl tedy zvolen sensor HC-SR05, který je ideálnějším kandidátem pro zhotovení aplikace.

1.5 Výběr kamery

Při výběru kamery se bralo v ohled rozlišení, rozměry kamery a kompatibilita s operačním systémem. Ze všech možností byla vybrána kamera SriHome SH001 s rozlišením videa Full HD (1920 x 1080) s 30 fps, které je dostačující pro snímání okolí zemědělské techniky. Také má široký snímací úhel 90°, díky čemuž je schopna snímat dostatečně velký prostor za vozidlem. Kamera podporuje operační systémy Windows, Mac OS, ale především Linux a Ubuntu. Další předností je dlouhý USB kabel o délce 1,5 m, což umožňuje umístění kamery na vhodné místo k dosažení co nejlepšího možného výsledku záběru. Se svými rozměry (šířka 12 cm, výška 5 cm a hloubka 4 cm) je možné tuto kameru umístit na téměř jakýkoliv typ přípojného vozidla. Cena této kamery se pohybuje okolo 200 Kč.

2 ALGORITMY ZPRACOVÁNÍ OBRAZU

Tato kapitola je zaměřena na popis algoritmů, jež je možné později využít na zpracovávání obrazu z kamery.

2.1 Co jsou to algoritmy zpracování obrazu

Algoritmy zpracování obrazu jsou matematické postupy, které se využívají k analýze, manipulaci a zlepšení digitálních obrazových dat. Tyto algoritmy umožňují detekovat vzory, extrahovat informace a provádět různé úpravy na obrazech, aby se zlepšila jejich kvalita, získaly užitečné informace nebo byly vhodné pro další zpracování.

2.2 Dělení algoritmů zpracování obrazu

Kapitola představuje matematické postupy využívané pro analýzu a úpravy digitálních obrazových dat. Po vysvětlení podstaty algoritmů zpracování obrazu se zabývá jejich dělením a aplikacemi. Základními technikami jsou filtrování a konvoluce, kde se filtrování používá k odstranění šumu a vyhlazení obrazu a konvoluce k transformaci obrazu pomocí masky nebo jádra. Segmentace obrazu dělí obraz na segmenty na základě jejich vlastností, jako je barva nebo jas, a extrakce rysů identifikuje klíčové informace, jako jsou hrany nebo tvary, čímž usnadňuje analýzu obrazu.

2.2.1 Filtrování a konvoluce

Filtrování a konvoluce jsou základními technikami zpracování obrazu. Filtrování je technika zpracování obrazu, která je použita za účelem odstranění šumu nebo vyhlazení obrazu. Při filtrování se aplikuje matematická operace na každý dílčí pixel obrazu. Jako příklad lze uvést Gaussovy filtry [13], které se používají pro vyhlazení obrazu a redukci detailů. Tyto filtry aplikují Gaussovu funkci na obrazová data, což má za následek rozostření obrazu, ovšem efektivně odstraní jemný šum a zjemní ostré hrany v obrazech. Gaussovy filtry počítají vážený průměr hodnot pixelů v okolí každého pixelu, kde váhy odpovídají Gaussově funkci.

Konvoluce je matematická operace, při které se jeden signál nebo obraz transformuje pomocí jiného signálu, který se nazývá maska nebo jádro. Jádro je obvykle malá matice o velikosti 3x3 nebo 5x5 složená z číselných hodnot, jež se postupně přemisťuje po obrazu a pro každou pozici provede konvoluční operace. Pro každý pixel obrazu se provede násobení hodnot

pixelu s odpovídajícími prvky jádra a výsledné součiny se poté sečtou. Tato suma nahradí původní hodnotu pixelu. Proces se opakuje pro všechny pixely obrazu. Jako příklad konvoluce lze uvést Sobelův operátor sloužící pro detekci hran v obrazech. Tento operátor identifikuje změny intenzity pixelů v obraze a tím umožňuje vytvoření obrysů objektů pomocí dvou masek. Jedna maska slouží pro detekci horizontálních a druhá pro detekci vertikálních hran. Z kombinace výsledků poté získáme celkový obrys objektů.

2.2.2 Segmentace obrazu

Segmentace obrazu je proces rozdělení obrazu na jednotlivé segmenty nebo regiony na základě jejich vlastností, jako jsou barva, jas, textura nebo geometrické vlastnosti. Cílem segmentace je identifikovat a izolovat jednotlivé objekty nebo části obrazu, což umožňuje lepší analýzu a porozumění obsahu obrazu. Existuje několik metod segmentace jako například prahování a detekce hran [14].

Prahování je jednoduchá segmentace, která rozděluje obraz na základě intenzity pixelů a prahovací hodnoty. Často se využívá pro detekci objektů s kontrastními barvami.

Detekce hran identifikuje hrany mezi různými regiony v obrazech na základě změn intenzity pixelů.

II. PRAKTICKÁ ČÁST

3 SESTAVENÍ HARDWAROVÝCH KOMPONENT

Kapitola je zaměřena na postup sestavení kamerového systému z již vybraných komponent. Důležitým úkolem je snížení napětí ultrazvukového senzoru pro kompatibilitu s Raspberry Pi, čehož se dosáhne přidáním odporů mezi výstupními piny senzoru a vstupními piny GPIO. Kapitola detailně popisuje postup redukce napětí pomocí rezistorů a zapojení senzoru k Raspberry Pi s využitím odporů. Zároveň je popsán proces pájení odpovídajících konektorů ke sníženým rezistorům. Obrázky ilustrují výsledné zapojení a použití odporů k dosažení požadovaného poklesu napětí a kompatibility s Raspberry Pi. Také je v této kapitole popsáno zapojení dalších periferií, jako je klávesnice, myš a monitor.

3.1 Snížení napětí ultrazvukového senzoru

Aby bylo možné správně měřit vzdálenost pomocí ultrazvukového senzoru, musíme výstupní signál snížit o určité napětí, aby bylo kompatibilní s napětím Raspberry Pi, které používá napětí o velikosti 3,3 VDC. Je potřeba omezit napětí na pinu ECHO (5 VDC) u senzoru, jež budeme přikládat na GPIO porty Raspberry Pi. Redukce napětí se provádí přidáním vhodných odporů mezi výstupní piny senzoru a vstupní piny GPIO. Pro aplikaci byly zvoleny tři rezistory, každý o velikosti 10 kΩ. Ke snížení napětí byla použita rovnice:

$$V_{out} = V_{in} * \left(\frac{R_2}{R_1 + R_2}\right)$$

V_{out} je napětí, které je potřebné jako výstupní napětí na vstupu GPIO portu, V_{in} je napětí vstupující do senzoru a v závorce jsou uvedeny odpory, díky kterým se napětí sníží o požadovanou hodnotu. Po dosazení do první rovnice vznikne tento vztah.

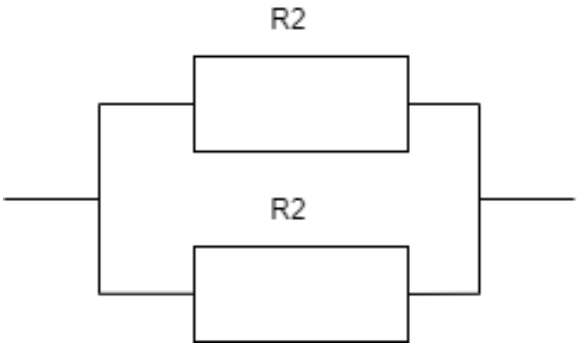
$$3{,}3 = 5 * \left(\frac{10\ k\Omega}{R_1 + \ 10\ k\Omega}\right)$$

$$3{,}3 * (R_1 + 10\ k\Omega) = 50\ k\Omega$$

$$3{,}3R_1 = 50\ k\Omega - 33\ k\Omega$$

$$R_1 \approx 5{,}15\ k\Omega$$

Jelikož byly použity tři rezistory o odporu 10 kΩ, ale výsledek vyšel okolo 5 kΩ, je zapotřebí ještě vhodné zapojení dvou rezistorů k tomu, aby bylo dosaženo požadovaného snížení napětí.



Obrázek 1: Paralelní zapojení dvou rezistorů

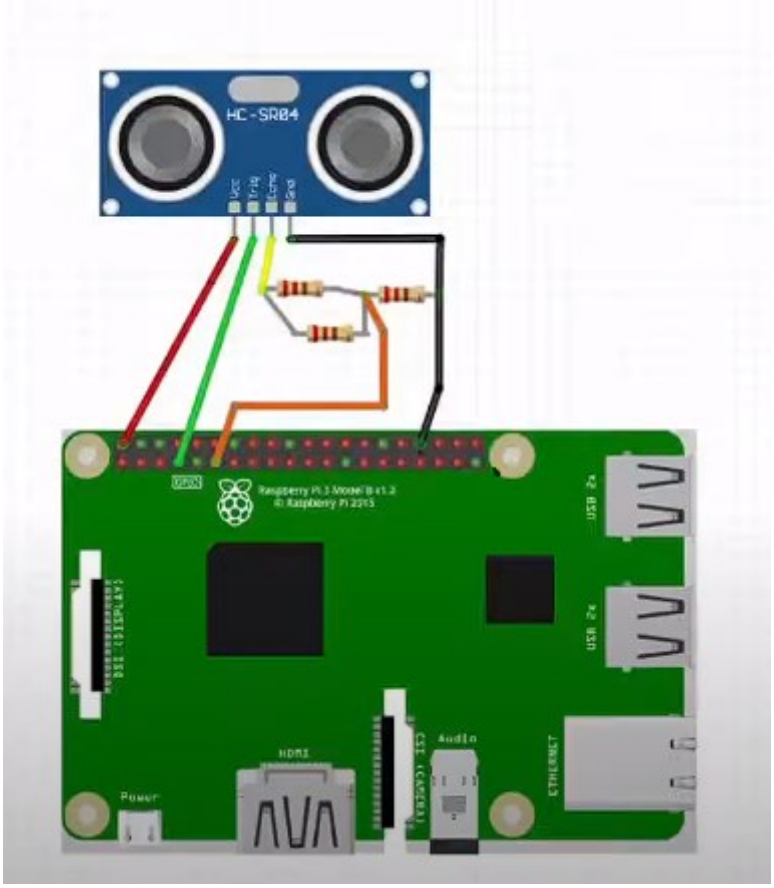
Pokud jsou dva rezistory zapojeny paralelně, jako je uvedeno na obrázku 1, odpor se sníží podle následujícího vzorce:

$$\frac{1}{R} = \frac{1}{R_2} + \frac{1}{R_2}$$

Odpor R je výsledná hodnota odporu a odpory R_2 jsou odpory s hodnotou 10 kΩ. Po dosazení do vzorce bude rovnice vypadat následovně:

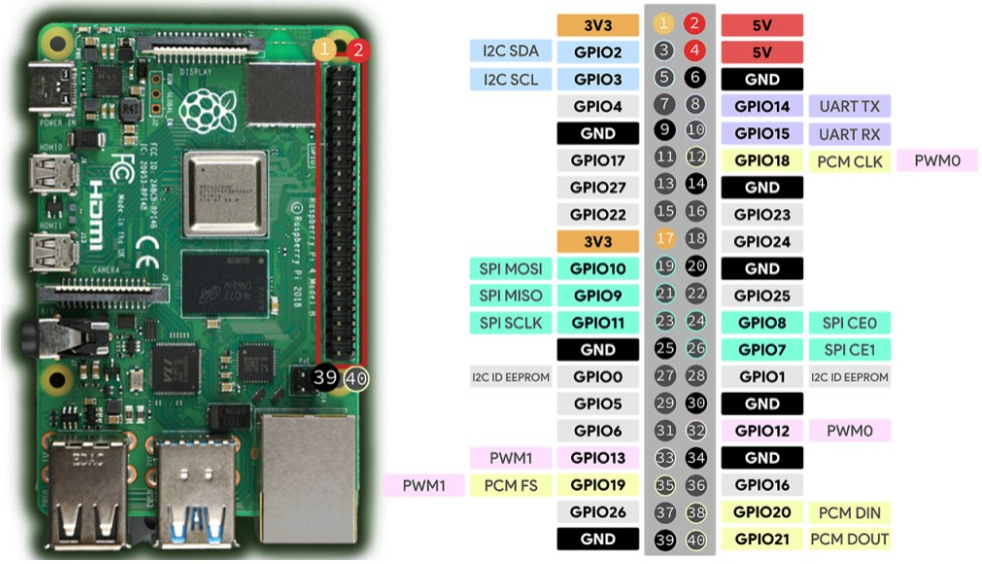
$$\frac{1}{R} = \frac{1}{10\ k\Omega} + \frac{1}{10\ k\Omega}$$
$$\frac{1}{R} = \frac{1}{5\ k\Omega}$$
$$R = 5\ k\Omega$$

Z výpočtů bylo dokázáno, že je možné použít tři odpory s hodnotou 10 kΩ, pokud dva odpory budou zapojeny paralelně, díky čemuž bude výsledný odpor 5 kΩ.



Obrázek 2: Zapojení ultrazvukového senzoru k Raspberry Pi přes GPIO porty pomocí odporů [15]

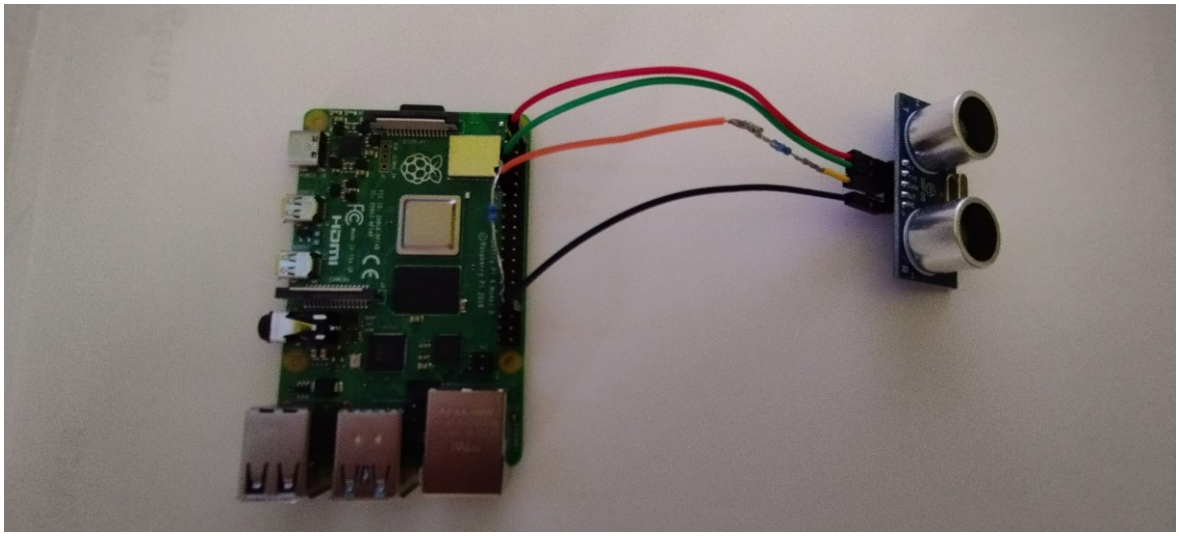
Na obrázku 2 lze vidět použití a zapojení ultrazvukového senzoru za pomocí odporů, jež sníží napětí z 5 VDC na 3,3 VDC. Červený vodič je přiveden na druhý pin GPIO portů (jedná se o napájecí pin 5 VDC) a první pin senzoru označený jako VCC. Zelený vodič je přiveden na 7 pin GPIO (jedná se o volný GPIO4 pin) a na druhý pin senzoru označený TRIG. Oranžový je přiveden na 11 pin GPIO (Volný GPIO17 pin) a jeho konec, označený žlutě, je přiveden na pin ECHO na senzoru. Černý vodič označovaný jako GND jak na senzoru, tak i na GPIO pinech, je zapojen do GPIO rozhraní na pin 34 (GND) a na senzoru do GND pinu. Na obrázku 3 je uvedeno celé rozhraní GPIO Raspberry Pi destičky, které bylo použito pro zapojení obvodu.



Obrázek 3: Raspberry Pi pinout.

3.2 Realizace zapojení s odpory

K aplikaci je zapotřebí mít ultrazvukový senzor a rezistory zapojený bez pomoci nepájivého pole. Rezistory byly napájeny k DuPont kabelům. Podle obrázku 2 byly propojeny červený a zelený DuPont kabel. Druhá polovina zapojení byla provedena pájením pomocí mikropájky. Aby bylo zachováno schéma zapojení, bylo zapotřebí tří DuPont kabelů (oranžový, žlutý a černý). Paralelně řazené rezistory byly zkráceny a nožičky byly zatočeny do spirály pro snadnější pájení. Žlutý DuPont kabel byl zkrácen přibližně na desetinu své původní délky a byla odstraněna izolace v délce asi 1 cm. Oranžový kabel byl zkrácen v polovině a také byla odstraněna izolace o stejné délce. Zamotané nožičky rezistorů i obnažené konce DuPont kabelů byly namočeny do pájecí kapaliny, aby došlo k odstranění mastnoty a případných nečistot. Rozžhavený konec pájky byl následně přiložen na jednu stranu rezistorů a konec oranžového kabelu. Po dostatečném rozehřátí byl přiložen cín, který se roztavil a spojil oba konce dohromady. Obdobný proces byl proveden i na druhé straně rezistorů se žlutým kabelem. Následně byly koncovky z oranžového a černého kabelu svlečeny. Ke každému konektoru byl připájen jeden konec rezistoru a následně navlečeny ochranné koncovky konektorů. Obrázek níže ukazuje výsledek zapojení ultrazvukového senzoru.



Obrázek 4: Zapojení mikropočítače Raspberry Pi 4 s ultrazvukového senzorem HC-SR05

3.3 Zapojení periferií

Připojení myši a klávesnice je možné provést pomocí USB portů na Raspberry Pi 4. USB konektory myši a klávesnice jsou zasunuty do volných USB portů na zařízení. Po správném připojení byly myš a klávesnice automaticky rozpoznány a připraveny k použití.

Pro připojení monitoru k Raspberry Pi 4 je nezbytné využít HDMI port. HDMI konektor monitoru je propojen s HDMI portem na Raspberry Pi 4. Tímto způsobem je zajištěn přenos digitálního video signálu z mikropočítače na monitor, což umožňuje zobrazení obsahu na obrazovce.

Zdroj zatím není nutné zapojovat do napájecího konektoru typu USB-C, jelikož do mikropočítače zatím nebyl zaveden operační systém. Je ovšem nutné dávat si pozor na výstupní napětí a proud zvoleného zdroje. Pokud nemáme k dispozici originální napájecí zdroj od výrobce Raspberry Pi, je možné jej nahradit zdrojem s výstupním napětím 5 VDC a výstupním proudem 3 A.

4 STAŽENÍ A INSTALACE SOFTWARU RASPBIAN

Po sestavení kamerového systému je potřeba do něj nahrát operační systém Raspbian. Raspbian je operační systém odvozený z Debianu, který se používá jak pro platformy Raspberry Pi, tak i pro osobní počítače. Ačkoliv kamerový systém nepoužívá klávesnici, myš ani monitor, bylo původně uvažováno o takzvané „Headless“ metodě, díky níž je možné ovládat mikropočítač přes lokální síť wifi. Nýbrž pro snadnější komunikaci a plynulejší tetování byla vybrána metoda s připojením klávesnice, myši a monitoru. Pro pozdější použití je potřeba mít stažený software RealVNC-Viewer umožňující připojení přes lokální síť, aby nebylo nutné drátové propojení kamery a zobrazovacího zařízení, čímž se dosáhne pohodlného připojení uživatele z mobilního zařízení.

4.1 Stažení potřebných softwarů ke správnému zavedení OS a běhu Raspberry Pi

Kapitola se zaměřuje na stažení a instalaci potřebného softwaru pro správné zavedení operačního systému a běh Raspberry Pi. Obsahuje několik podkapitol, které popisují postup stažení a instalace konkrétních softwarových nástrojů, jako je Balena Etcher pro vytváření bootovacích médií, PuTTY pro vzdálený přístup k Raspberry Pi přes terminál a RealVNC Viewer pro možnost připojení k Raspberry Pi bez monitoru. Každá podkapitola podrobně popisuje postup stažení, instalace a potřebné kroky pro správnou konfiguraci softwaru.

4.1.1 Stažení Balena Etcher

Aby byl softwarový balíček balenaEtcher nainstalován, je třeba stáhnout aplikaci z oficiální webové stránky. Nejprve je nutné kliknout na tlačítko "Download Etcher" na této webové stránce: https://etcher.balena.io. Níže na stránce bude nutné kliknout na tlačítko "Etcher for Windows", vybrat umístění pro stažení balíčku a potvrdit stiskem tlačítka "Uložit". Je nezbytné zvolit vhodnou verzi aplikace pro daný operační systém – v tomto případě "Etcher for Windows (x86|x64) (Installer)". Po stažení a spuštění souboru "balenaEtcher-Setup-1.18.11.exe" bude zobrazen dialogový box, v nemž je třeba kliknout na tlačítko "Souhlasím", aby byl software nainstalován do počítače.

4.1.2 Stažení PuTTY

Pomocí PuTTY se lze v případě chyby či absence monitoru, myši a klávesnice, připojit k Raspberry Pi 4. Samotné připojení se provádí přes terminál, je ovšem nutno znát IP adresu

zařízení. PuTTY je bezplatný open-source software umožňující vzdálený přístup k počítačům přes různé síťové protokoly, jako jsou například SSH, Telnet a Raw. Odkaz na stažení je rozkliknut: https://www.chiark.greenend.org.uk/~sgtatham/putty/ . Na stránce je kliknuto na „Download it here“ a vybrána verze pro daný operační systém. V tomto případě je to verze PuTTY 64-bit x86 pro Windows. Poté je zvoleno, kde se má instalační program stáhnout. Po stažení je otevřen instalační program, následně je kliknuto na tlačítko „Next“ a v nově zobrazeném okně je zvolena cesta, kam se má PuTTY instalovat. Po zvolení vhodného místa uložení se pokračuje na další stránku, kde je stisknuto tlačítko „Install“. Po dokončení instalace je stisknuto tlačítko „Finish" a instalace je dokončena.

4.1.3 Stažení RealVNC-Viewer

Pokud by bylo časem nutné se připojit k mikropočítači bez použití monitoru, myši a klávesnice, může být využita „Headless“ metoda. Tato metoda umožní připojení k Raspberry Pi 4 přes lokální síť pomocí VNC-Viewer. Stránka, odkud je možné stáhnout požadovaný software, je otevřena na adrese: https://www.realvnc.com/en/connect/download/viewer/. Po kliknutí na žluté tlačítko „Download RealVNC Viewer“ bude opět zvoleno, kam se má instalační program uložit, a volba bude potvrzena. Po rozkliknutí instalátoru bude zvolen jazyk a poté potvrzena volba. Následně bude stisknuto tlačítko „Next“. Smluvní podmínky budou potvrzeny. Pokud bude žádáno, bude možné si nechat vložit zástupce na plochu. V posledním kroku bude kliknuto na tlačítko „Install“.

4.2 Stažení a zavedení Operačního systému Raspbian na kartu SD

Pro stáhnutí image softwaru Raspbian ze stránky https://www.raspberrypi.com/software/ je třeba kliknout na možnost "See all download options". Před stažením je nutné vybrat z jednotlivých operačních systémů (OS). Existují tři hlavní verze Raspberry Pi OS, které se liší v poskytovaných funkcích a velikosti stahování:

- Raspberry Pi OS Lite: Tato verze je bez grafického uživatelského rozhraní a je vhodná pro vývojáře, kteří staví vlastní rozhraní nebo provozují serverové aplikace. Velikost stahování je menší než 500 MB a je ideální pro zařízení bez grafického uživatelského rozhraní.
- Raspberry Pi OS with desktop: Tato verze obsahuje desktopové prostředí Pixel s LXDE a prohlížečem Chromium, což je vhodné pro běžné domácí uživatele a

studenty. Velikost stahování je kolem 1.1 GB a zahrnuje prohlížeč, uživatelské rozhraní a základní nástroje.

- Raspberry Pi OS with desktop and recommended software: Tato verze je určena pro pokročilé uživatele, jako jsou vývojáři softwaru, testovači nebo tvůrci obsahu. Obsahuje širokou škálu balíčků jako LibreOffice, openssh-server, VLC atd., což umožňuje okamžité použití mnoha aplikací. Velikost stahování přesahuje 2.7 GB a je vhodná pro uživatele potřebující rozsáhlý software.

Po výběru požadované verze Raspberry Pi OS s desktopem a doporučeným softwarem je kliknuto na tlačítko "Download" a vyčká se, až bude image softwaru stažen na počítač. Po dokončení stahování je nutno kliknout pravým tlačítkem myši na stažený soubor raspios-bookworm-armhf-full.img.xz a vybrat možnost "Rozbalit zde" nebo "Extrahovat sem". Po dokončení extrakce bude mít soubor raspios-bookworm-armhf-full.img. (Před názvem je uvedeno datum vydání aktuální verze image, proto nebyl název uveden celý.)

Po dokončení extrahování image je připojena flash paměť k počítači pomocí redukce na flash paměti. Poté je spuštěn software Balena Etcher, který slouží k zápisu obrazu operačního systému na flash paměť. Při výběru image je kliknuto na možnost "Flash from file", kde je vybrána stažená image operačního systému. Poté je kliknuto na možnost "Select target", kde je určena cílová flash paměť, na kterou bude OS zaveden. Nakonec je stisknuto tlačítko "Flash!" a spuštěno klonování image na flash disk. Tato akce trvá i několik minut, než je operační systém zaveden na flash disk.

Po dokončení klonování operačního systému na flash disk je možné jej vyjmout z počítače a vložit do slotu pro paměťovou kartu na Raspberry Pi, který se nachází na spodní straně mikropočítače, naproti portům USB a ethernetu. Poté je zavedeno napájení. Po prvním spuštění je nutno počkat delší chvíli, než se operační systém zavede a správně nabootuje. Pokud nenastala někde při instalaci a stahování chyba, měla by se na monitoru objevit domovská obrazovka. Pro audiovizuální návod lze shlédnout video na platformě YouTube. [16]

4.3 První boot a nastavení OS

Poté, co se operační systém poprvé nastartuje, se na ploše objeví okno s názvem „Welcome to Raspberry Pi“, které provede uživatele prvním a základním nastavením. Na prvním okně

je stisknuto tlačítko OK, jež zobrazí další okno, kde může uživatel nastavit zemi, jazyk a časovou zónu. Pro lepší práci byla zvolena země Česká republika a jazyk čeština. Časová zóna se poté sama nastavila na Prahu. Po tomto nastavení je kliknuto na tlačítko Next. Následně je v dalším okně založen nový uživatelský účet, pod kterým se může uživatel přihlašovat do systému. Opět je stisknuto tlačítko Next. Další okno obsahuje možnost bezdrátového připojení k síti. Po chvíli čekání se objeví dostupné sítě Wi-Fi. Z výběru dostupných sítí je vybrána síť s nejlepším signálem. K připojení je potřeba zadat heslo, pod kterým se běžně uživatel k síti připojuje. Po připojení k síti se operační systém zeptá, zda chce uživatel provést update systému. Na tuto výzvu je odkliknuto tlačítko Next, které zkontroluje dostupnost updatu. Je dostupná novější verze, která je po vyhledávání následně stažena. Po dokončení stahování je mikropočítač restartován, aby byly nové změny správně aplikovány.

Po dokončení restartu je uživatelem otevřen terminál, který se nachází v levém horním rohu obrazovky. Do otevřeného okna je vložen příkaz: sudo raspi-config a je stisknuta klávesa enter. Otevře se obecné nastavení, ve kterém se uživatel orientuje pomocí šipek na klávesnici. Vybere se možnost Interfacing Options a pomocí enteru se tato možnost potvrdí. V nově zobrazeném okně je pomocí šipek vybrána možnost VNC a opět potvrzen výběr enterem. Poté je označena možnost Yes a potvrzena. Nastavení se na malou chvilku ztratí kvůli přenastavení konfigurace, ale následně se objeví okno potvrzující úspěšné povolení VNC. Jako reakce na toto zobrazení je stisknuto OK. Po dokončení nastavení VNC je vybráno tlačítko Finish a následně je systém restartován, aby byly změny správně aplikovány.

4.4 Otestování funkčnosti ultrazvukového senzoru

Pro snadnější pochopení je možné zhlédnout video postup zveřejněný na platformě YouTube, který uživatele provede postupem, jak měřit vzdálenost. [17] V popisku videa je odkaz na tuto stránku, kde autor uvádí mimo jiné i kód k měření vzdálenosti použitý pro ověření funkčnosti zapojení ultrazvukového senzoru.

V průzkumníku je vytvořena nová složka, v níž se budou nacházet dílčí kódy, které nejprve otestují funkčnost každého komponentu. Je vytvořena složka s názvem Projekt a v ní je

vytvořen soubor mereni.py. Tento soubor je otevřen pomocí programu „mu“ s ikonkou červeného hada.

Mu Editor je textový editor a integrované vývojové prostředí (IDE) speciálně navržené pro programování v jazyce Python. Tento editor je známý svou snadnou použitelností a je určený pro začátečníky, což ho činí ideální volbou pro uživatele Raspberry Pi. Mu Editor nabízí různé režimy, z nichž jeden je Python 3, který umožňuje psát a spouštět Python kód přímo v editoru. Editor také poskytuje užitečné funkce, jako je zvýrazňování chyb, kontrola kódu bez spuštění programu a jednoduché spouštění a zastavení kódu.

Po otevření tohoto souboru je vložen kód do prostředí a spuštěn. Výstup je vypisován v konzoli přímo v IDE. Níže je uveden příklad výstupu zkopírovaný přímo z konzole při přibližování a oddalování objektu od ultrazvukového senzoru:

Distance : 59.63 cm
Distance : 49.22 cm
Distance : 41.89 cm
Distance : 32.74 cm
Distance : 24.80 cm
Distance : 30.51 cm
Distance : 162.56 cm
Distance : 162.12 cm
Distance : 163.45 cm
Distance : 162.16 cm

Tohoto výstupu bylo dosaženo pomocí níže uvedeného kódu.

Kód 1: Měření vzdálenosti

```
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)

# Define GPIO to use on Pi
GPIO_TRIGGER = 4
GPIO_ECHO    = 17

# Set pins as output and input
GPIO.setup(GPIO_TRIGGER,GPIO.OUT)
GPIO.setup(GPIO_ECHO,GPIO.IN)

# Set trigger to False (Low)
GPIO.output(GPIO_TRIGGER, False)
time.sleep(0.1)

# Measurement Function
def measureDistance():
  GPIO.output(GPIO_TRIGGER, True)
  # Let the trigger be on for 10us
  time.sleep(0.000001)
  GPIO.output(GPIO_TRIGGER, False)

  while GPIO.input(GPIO_ECHO)==0:
    start = time.time()

  while GPIO.input(GPIO_ECHO)==1:
    stop = time.time()

  time_diff = stop-start

  # 34600 cm/s is the speed of sound at 25 Deg C (77F).
  distance = (time_diff * 34600 )/2
  return distance

try:
  while True:
    distance = measureDistance()
    print("Distance : {0:.2f} cm".format(distance))
    # wait for half second
    time.sleep(.5)
except KeyboardInterrupt:
  # User pressed CTRL-C
  # Reset GPIO settings
  GPIO.cleanup()

```

V první části kódu importujeme knihovnu - time, RPi.GPIO a nastavujeme mód GPIO na BCM. Knihovna time poskytuje funkce pro manipulaci s časem v Pythonu, obsahuje funkce pro zpoždění, měření času a práci s časovými údaji. Knihovna RPi.GPIO je knihovna pro práci s GPIO piny na Raspberry Pi. Umožňuje nastavování pinů jako vstupní nebo výstupní, čtení hodnot z pinů. Díky tomuto je mikropočítač schopen komunikovat s periferiemi. Funkce GPIO.setmode(GPIO.BCM) je součástí již zmiňované knihovny RPi.GPIO, která slouží k nastavení režimu použití GPIO pinů. GPIO.BCM je pinové číslování odpovídající číslům GPIO (Broadcom SOC channel). Tento mód je častější, protože poskytuje přehlednější číslování pinů.

V sekci pod knihovnami se nachází část, kde se definuje do proměnné číslo portu, které odpovídá zapojení na GPIO portech. Fyzicky jsou Trigger a Echo pin na ultrazvukovém snímači zapojeny do 7 a 11 pinu na Raspberry Pi. Podle obrázku 3 tyto fyzické piny odpovídají pinům GPIO4 a GPIO17. Další řádky nastavují, zda se jedná o pin výstupní (GPIO_TRIGGER) nebo vstupní (GPIO_ECHO). Tyto dva řádky fungují následovně: zavolá se funkce knihovny a za tečkou je uvedena funkce. V závorce je následně uvedený pin a za čárkou je jeho funkce.

Následně se nastaví TRIGGER na hodnotu false, tedy nízkou úroveň, čímž je deaktivován spouštěcí signál. Díky tomu je zabráněno nežádoucímu spuštění měření senzorem. Následuje krátká časová prodleva trvající 0.1 sekundy pro stabilizaci signálu před měřením.

Část kódu measureDistance() slouží k měření vzdálenosti pomocí ultrazvukového senzoru. Nejprve se nastaví TRUGGER pin na true, vysokou úroveň signálu, čímž je vyslán signál od senzoru do prostoru směrem k měřenému objektu. Tento signál se vysílá po dobu 10 mikrosekund a po uplynutí této doby se nastaví TRIGGER na false, díky čemuž se signál zastaví. V této chvíli program čeká na odraz odeslaného signálu od objektu. Toho je dosaženo tak, že pokud je ECHO pin nastaven na false, tedy ve stavu LOW, uloží se čas do proměnné start. Následně program čeká, až bude ECHO na hodnotě true, tedy ve stavu HIGH. Když se tak stane, uloží se čas do proměnné stop. Následuje výpočet rozdílu času start a stop. Tento rozdíl je vynásoben rychlostí šíření vlny ve vzduchu při teplotě 25 °C a tento výsledek je vydělen dvěma, protože čas je doba, která uplynula od odeslání k objektu a jeho návratu. Ultrazvuková vlna urazila tedy dvojnásobnou vzdálenost.

Poslední část kódu vytváří nekonečnou smyčku, jež stále dokola volá definovanou funkci measureDistance() a ukládá výslednou hodnotu do proměnné distance. Následně je tato proměnná vypsána do konzole pomocí funkce print. Formát je zaokrouhlen na dvě desetinná místa pomocí této části printu: {0:.2f}. Následně program čeká polovinu sekundy než pokračuje v měření. Pokud v průběhu měření uživatel stiskne kombinaci kláves CTRL + C, provedou se operace na vyčištění a resetování nastavení GPIO, které se provádějí pro uvolnění GPIO pinů a jejich opětovné použití.

4.5 Otestování funkčnosti USB kamery

Pro otestování kamery bylo provedeno několik kroků. Nejprve je otevřen terminál jako v předchozích případech a vloží se do něj příkaz „lsusb“, který zobrazí veškerá dostupná USB zařízení aktuálně připojená k mikropočítači. Níže na obrázku je ukázka vypsané kamery. Jedná se o třetí řádek „Smartonix, Inc. webcam“. Následně je stažena knihovna pro obsluhu webové kamery pomocí příkazu: sudo apt isntall fswebcam. Po úspěšné instalaci balíčku je použit příkaz „ls“, pomocí kterého je zobrazeno místo, kde se terminál otevřel. Následně je zvolena složka, do níž bude vytvořen výstup první fotografie a videa. V tomto případě byla vybrána složka Obrázky, jež je k tomuto účelu vytvořena. Je použit příkaz „cd Obrázky“, pomocí kterého se terminál přemístí do této složky. V této složce je spuštěn příkaz: fswebcam -r 1280x720 --no-banner test.jpg. Jako první je napsán název programu, jejž bude obsluhovat kameru, následně je uvedeno rozlišení, v tomto případě 1280 na 720 pixelů. Příkaz --no-banner odstraňuje záběrový banner, který se obvykle zobrazí na snímku, tedy nebude obsahovat žádný textový popisek. Poslední argument je název a typ souboru. Na obrázku číslo 6 je ukázka fotografie zachycené z USB kamery.

Následně je zadán příkaz: ffmpeg -f v4l2 -video_size 1280x720 -i /dev/video0 -t 10 -an video.avi, který vytvoří video s příponou .avi v kvalitě 1280 na 720 pixelů. Specifikuje, jaká kamera má použita a po jak dlouhý časový úsek, v tomto případě pouze pět sekund. Jedná se pouze o video, nemá tedy zvuk.

Následně byl vytvořen ve složce Projekt soubor kamerka.py, do níž byl vytvořen kód pro ovládání kamery. Tento kód pouze zobrazí výstup z kamery bez měření vzdálenosti. Byla využita knihovna cv2, která je součástí knihovny OpenCV [18] napsaná v jazyce C a C++, ale lze ji použít i pro Python. Umožňuje základní manipulaci s obrazem a videem a provádět úpravy barev, kreslení do obrázků a jiné. V tomto souboru byl následně napsán níže uvedený kód:

Kód 2: Zobrazení výstupu USB kamery

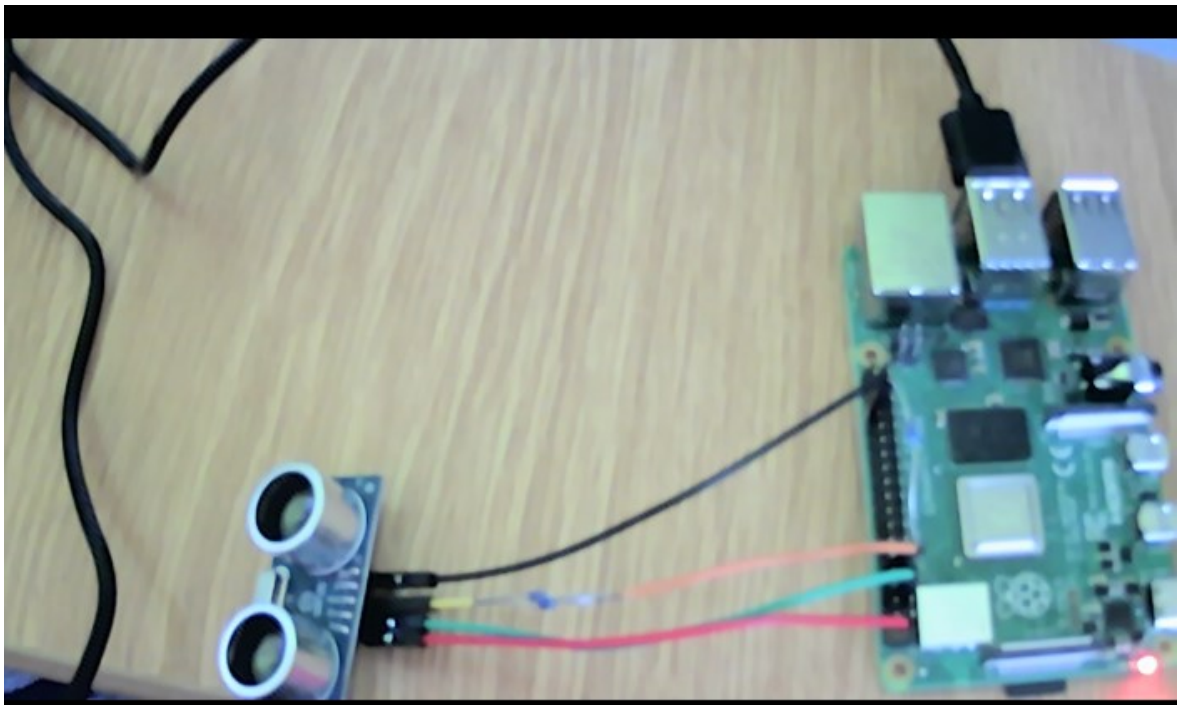
```
 1. import cv2
 2.
 3. def main():
 4.     cap = cv2.VideoCapture(0)
 5.     if not cap.isOpened():
 6.         print("Nelze otevrit USB kameru.")
 7.         return
 8.     while True:
 9.         ret, frame = cap.read()
10.         if not ret:
11.             print("Chyba pri cteni snimku z kamery.")
12.             break
13.         cv2.imshow('USB Kamera', frame)
14.         if cv2.waitKey(1) == ord('q'):
15.             break
16.
17.     cap.release()
18.     cv2.destroyAllWindows()
19. if __name__ == "__main__":
20.     main()
21.
```

Nejprve je importována knihovna cv2, se kterou se v této části pracuje. Následně je definována hlavní funkce programu. Uvnitř je vytvořen objekt „cap“ sloužící k přístupu USB kamery. Číslo v závorce, v tomto případě 0, označuje kameru, již má program použít. Nula označuje první nalezenou kameru. V projektu je použita pouze jedna kamera, takže se vždy užije tato kamera.

Následně proběhne kontrola, zdali byla kamera úspěšně otevřena. Jestliže se kamera neotevře správně nebo není dostupná, program vypíše zprávu o chybě a ukončí se. Pokud je kamera otevřena správně a vše proběhlo bez problémů, program se dostane do nekonečné smyčky While, která neustále čte výstup z kamery pomocí metody cap.read(). V proměnné ret je uložena hodnota True anebo False podle toho, zda byl snímek přečten úspěšně či nikoliv. Do proměnné frame se nahraje samotný snímek z kamery. Pokud je proměnná ret False, cyklus se přeruší a do konzole se vypíše zpráva o chybě čtení snímku z kamery. Pokud je ret proměnná True, aktuální snímek se zobrazí na obrazovce pomocí metody imshow(). Jestliže uživatel kdykoliv během této smyčky zmáčkne klávesu „q“, smyčka se ukončí a program skončí. Ještě před ukončením se uvolní veškeré využívané zdroje, to znamená, že se zavře přístup ke kameře a následně se zavřou všechna okna. [19]

```
Soubor  Upravit  Karty  Nápověda
lukas@raspberrypi:~ $ lsusb
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 003: ID 1d5b:0104 Smartronix, Inc. webcam
Bus 001 Device 002: ID 2109:3431 VIA Labs, Inc. Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
lukas@raspberrypi:~ $
```

Obrázek 5: Připojená USB kamera



Obrázek 6: Ukázka zachycení fotografie

5 ROZŠÍŘENÁ REALITA: INTEGRACE MĚŘENÍ VZDÁLENOSTI A SNÍMÁNÍ Z KAMERY

Kapitola se bude zabývat integrací dvou důležitých částí kódu: měřením vzdálenosti pomocí ultrazvukového senzoru a zachycováním obrazu z USB kamery. Tato integrace umožní vytvoření aplikace schopné zobrazit v reálném čase snímek z kamery a současně na něm zobrazovat měřenou vzdálenost od překážky.

Během této kapitoly budou části kódu propojeny tak, aby měřená vzdálenost byla zobrazena jako textový popisek přímo na snímku z kamery. To umožní uživateli snadnou vizualizaci vzdálenosti od objektů zachycených na obrazovce.

Dále bude v kapitole vysvětleno, jakým způsobem jsou výsledky měření vzdálenosti interpretovány a jaká je jejich vizuální reprezentace na snímku z kamery. Taktéž budou popsány barvy použité k zobrazení vzdálenosti v závislosti na hodnotě vzdálenosti samotné.

5.1 Spojení kódu pro měření a snímání výstupu z kamery

V následující části kódu byla provedena integrace obou předchozích částí tak, aby se vypočítaná vzdálenost z ultrazvukového senzoru vykreslovala přímo na výstupní obraz z kamery. Tento proces využívá knihovnu OpenCV, která umožňuje vykreslování textu a dalších grafických prvků přímo do obrazu. Cílem této práce je implementovat takzvanou "rozšířenou realitu", kdy je k obrazu z kamery přidána informace o vzdálenosti získaná senzorem, což může být užitečné při navigaci nebo při vizualizaci vzdálenosti překážek. Níže je uvedený tento kód s vysvětlením jeho funkce.

Kód 3: Zobrazení výstupu USB kamery s měřením vzdálenosti

```
import cv2
import RPi.GPIO as GPIO
import time

# Inicializace GPIO
GPIO.setmode(GPIO.BCM)

# Definice GPIO
GPIO_TRIGGER = 4
GPIO_ECHO = 17

# Nastaveni pinu jako vystup a vstup
GPIO.setup(GPIO_TRIGGER, GPIO.OUT)
GPIO.setup(GPIO_ECHO, GPIO.IN)

# Funkce pro mereni vzdalenosti
def measureDistance():
    GPIO.output(GPIO_TRIGGER, True)
    time.sleep(0.00001)
    GPIO.output(GPIO_TRIGGER, False)

    pulse_start = time.time()
    pulse_end = time.time()

    while GPIO.input(GPIO_ECHO) == 0:
        pulse_start = time.time()

    while GPIO.input(GPIO_ECHO) == 1:
        pulse_end = time.time()

    pulse_duration = pulse_end - pulse_start


    distance = (pulse_duration * 34300) / 2

    return distance

def main():
    # Vytvoreni objektu pro pristup ke kamere
    cap = cv2.VideoCapture(0)

    # Kontrola, zda se kamera otevrela spravne
    if not cap.isOpened():
        print("Nelze otevrit USB kameru.")
        return

    # Neustala smycka z kamery, mereni vzdalenosti a zobrazeni vystupu kamery
    while True:
        ret, frame = cap.read()
        if not ret:
            print("Chyba pri zachiceni snimku z kamery.")
            break

        # Mereni vzdalenosti
        distance = measureDistance()
        distance_text = f"Vzdalenost: {distance:.2f} cm"

        # ZobrazenÃi snimku se vzdalenosti
        if distance > 100:
            color = (0, 255, 0)  # Zelena pro vzdalenost mensi nez 100 cm
        elif distance > 50:
            color = (0, 165, 255)  # Oranzova pro vzdalenost mezi 50 a 100 cm
        else:
            color = (0, 0, 255)  # Cervena pro vzdalenost mereni mensi nez 50 cm

        # ZobrazenÃ textu s odpovědnou barvou
        cv2.putText(frame, distance_text, (10, 50), cv2.FONT_HERSHEY_SIMPLEX, 1.5, color,
2)
```

```
68.
69.         # Zobrazeni snimku
70.         cv2.imshow('USB Kamera', frame)
71.
72.         # Pro ukonceni programu staci stisknout klávesu ‚q‘
73.         if cv2.waitKey(1) & 0xFF == ord('q'):
74.             break
75.
76.     # Uvolneni prostredku
77.     cap.release()
78.     cv2.destroyAllWindows()
79.
80. if __name__ == "__main__":
81.     main()
82.
```

Po importování důležitých knihoven, které zahrnují OpenCV pro zpracování obrazu a RPi.GPIO pro ovládání GPIO pinů na Raspberry Pi, následuje definice funkce measureDistance(), jež je odpovědná za měření vzdálenosti pomocí ultrazvukového senzoru. Poté následuje definice funkce main() sloužící jako vstupní bod programu. Uvnitř této funkce je vytvořen objekt cap pro přístup k USB kameře pomocí funkce cv2.VideoCapture(0), kde číslo 0 označuje index zařízení kamery. Obdobně jako bylo popsáno výše. Po tomto následuje kontrola kamery. Pokud by nastala chyba, program se ukončí s uvedenou chybou v terminálu.

Pokud je ovšem vše v pořádku, následuje nekonečná smyčka while True, která zajišťuje neustálé čtení obrazu z kamery a měření vzdálenosti. Během každé iterace smyčky je nejprve zkontrolováno, zda byl snímek z kamery úspěšně načten pomocí cap.read(). V případě, že není načten snímek (ret je False), program vypíše chybové hlášení a přeruší smyčku. Následuje měření vzdálenosti pomocí funkce measureDistance(). Výsledek měření je formátován do textového řetězce distance_text, jenž obsahuje informaci o vzdálenosti v centimetrech s přesností na dvě desetinná místa.

Následuje zobrazení snímku z kamery s přidaným textem o vzdálenosti. Barva textu je určena podle hodnoty vzdálenosti: zelená pro vzdálenosti větší než 100 cm, oranžová pr vzdálenosti mezi 50 a 100 cm a červená pro vzdálenosti menší než 50 cm. Barevná odlišnost textu zobrazeného na videu byla zvolena z důvodu snadnějšího rozpoznávání potencionálního nebezpečí.

Nakonec je snímek zobrazen pomocí funkce cv2.imshow(). Jestliže uživatel stiskne klávesu 'q', program se ukončí uvolněním prostředků (uzavřením přístupu k USB kameře) a zavřením všech otevřených okenních grafik pomocí funkcí cap.release() a cv2.destroyAllWindows().

Funkce přerušení programu pomocí klávesy 'q' je implementována čistě z testovacích důvodů a může být ve finální verzi kódu vynechána. Tato aplikace zobrazuje snímky spolu se vzdáleností pomocí „vyskakovacího“ okna, které se objeví na ploše po spuštění kódu. Nejedná se o nejlepší možné řešení, avšak pro svou jednoduchost se dá považovat za dostačující.

5.2 Doplnění stávajícího kódu o zobrazení na webové stránce

Pro vyšší uživatelskou přívětivost a snadnější používání bylo rozhodnuto implementovat aplikaci do webového rozhraní. Tato změna umožňuje uživatelům přistupovat k funkcím aplikace prostřednictvím webového prohlížeče, což zvyšuje flexibilitu a dostupnost aplikace.

Webové rozhraní je vytvořeno pomocí frameworku Flask, jenž je snadno použitelný a efektivní pro vytváření webových aplikací. Díky Flasku je možné definovat různé routy zpracovávající uživatelské požadavky a poskytující vhodné odpovědi. Z tohoto důvodu byl kód číslo 3 doplněn o následující řádky kódu.

Kód 4: Doplněné funkce ke kódu 3 pro zobrazení výstupu ve webovém rozhraní

```
from flask import Flask, render_template, Response
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/video_feed')
def video_feed():
    return Response(gen_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')
def gen_frames():
    camera = cv2.VideoCapture(0)
    while True:
        success, frame = camera.read()
        if not success:
            break
        else:

            distance = measureDistance()
            distance_text = f"Vzdalenost: {distance:.2f} cm"


            if distance > 100:
                color = (0, 255, 0)
            elif distance > 50:
                color = (0, 165, 255)
            else:
                color = (0, 0, 255)


            cv2.putText(frame, distance_text, (10, 50), cv2.FONT_HERSHEY_SIMPLEX, 1.5,
color, 2)

            ret, buffer = cv2.imencode('.jpg', frame)
            frame = buffer.tobytes()
            yield (b'--frame\r\n'
                   b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
if __name__ == "__main__":
    app.run(host='0.0.0.0', port=5000, debug=True)

```

Kód provádí několik úkolů pomocí frameworku Flask pro tvorbu webových aplikací v jazyce Python. Nejprve jsou importovány potřebné moduly z Flasku, jako jsou Flask, render_template a Response.

Dekorátor @app.route('/') určuje, že funkce index() bude obsluhovat URL adresu '/', což bude hlavní stránka aplikace. Funkce index() renderuje HTML šablonu 'index.html' pomocí funkce render_template() a vrátí ji jako odpověď na požadavek klienta.

Dekorátor @app.route('/video_feed') určuje, že funkce video_feed() bude obsluhovat URL adresu '/video_feed', která poskytuje streamovaný video obsah. Funkce vytváří odpověď typu Response s obsahem streamovaného videa generovaného funkcí gen_frames().

Funkce gen_frames() má za úkol generovat jednotlivé snímky videa, jež budou následně odeslány klientovi jako odpověď. Měření vzdálenosti i zachycení videa je stále stejné, pouze s tím rozdílem, že text s vzdáleností je poté vykreslen na aktuální snímek pomocí metody cv2.putText(). Snímek je nakonec konvertován do formátu JPEG pomocí metody cv2.imencode(), aby bylo možné ho odeslat jako bytový řetězec. Nakonec je vytvořen odpovídající rámec obsahující snímek v bytovém formátu s definovaným formátem hranice. Rámec je vrácen pomocí příkazu yield, což zajišťuje streamování jednotlivých snímků jako odpověď na žádost klienta.

Funkce app.run() spouští webový server pomocí frameworku Flask. Parametr host='0.0.0.0' určuje, že server bude naslouchat na všech dostupných síťových rozhraních, čímž bude umožněn přístup k serveru z jakéhokoli zařízení v síti. Parametr port=5000 určuje číslo portu, na němž bude server naslouchat. Pokud není specifikováno jinak, použije se výchozí hodnota 5000. Volba debug=True zapne režim ladění, což zajistí výpis chyb a další ladící informace přímo do konzole. Tento režim je užitečný při vývoji aplikace, ale měl by být vypnutý pro produkční nasazení z důvodu bezpečnosti a výkonu.

5.3 Vytvoření webové stránky

K tomu, aby byl záznam pořízený z kamery správně zobrazen na webovém rozhraní, je třeba ještě udělat webovou stránku. Ve složce s aktualizovaným kódem je vytvořena složka „templates“ a v ní je následně vytvořen soubor s názvem index.html. V tomto souboru je následně napsán a laděn kód webové stránky.

Kód 5: Webová stránka pro zobrazení výstupu z kamery

```
1.   <!DOCTYPE html>
 2. <html>
 3. <head>
 4.     <link rel="shortcut icon" href="{{ url_for('static', filename='favicon.ico') }}">
 5.     <meta name="viewport" content="width=device-width, initial-scale=1">
 6.     <style>
 7.         .navbar {
 8.             overflow: hidden;
 9.             position: fixed;
10.             bottom: 0;
11.             width: 100%;
12.             background-color: black;
13.             opacity:0.6;
14.         }
15.
16.         .navbar a {
17.             float: left;
18.             display: block;
19.             color: #f2f2f2;
20.             text-align: center;
21.             padding: 14px 16px;
22.             text-decoration: none;
23.             font-size: 17px;
24.         }
25.
26.         .navbar a.active {
27.             background-color: #4CAF50;
28.             color: white;
29.         }
30.
31.         .main {
32.             padding: 16px;
33.             margin-bottom: 30px;
34.         }
35.
36.         .camera-bg {
37.             position: fixed;
38.             top: 0;
39.             left: 0;
40.             min-width: 100%;
41.             min-height: 100%;
42.             height: 100%;
43.             background-position: center;
44.             background-repeat: no-repeat;
45.             background-size: cover;
46.         }
47.
48.         body {
49.             margin: 0;
50.             padding: 0;
51.             width: 100vw;
52.             height: 100vh;
53.             overflow: hidden;
54.             background-color: black;
55.         }
56.     </style>
57. </head>
58. <title>Parking</title>
59. <meta name="viewport" content="width=device-width, initial-scale=1">
60. <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">
61. <body>
62. <div class="main" id="newpost">
63.     <img  class="camera-bg" style="width: 100%; height:80%; background-attachment: fixed;"
id="bg" class="center" src="{{ url_for('video_feed') }}">
64. </div>
65. </body> </html>
```

V záhlaví dokumentu HTML, zahrnujícím značky <!DOCTYPE html> a <html>, je inicializována struktura webového dokumentu. V úvodní sekci <head> jsou umístěny klíčové informace a definice stylů, které formují vzhled a chování prezentované stránky.

Následuje oblast <style>, kde jsou detailně specifikovány stylistické vlastnosti, jako je formátování navigačního pruhu (.navbar), hlavního obsahu stránky (.main) a pozadí dokumentu (.camera-bg). Tento segment slouží k definici estetických aspektů, včetně barev, umístění prvků a velikosti textu.

V sekci <title> a <meta> v rámci <head> jsou uvedena metadata stránky včetně názvu a konfigurace zobrazení pro různá zařízení, což zahrnuje i obsah mobilních zařízení (viewport).

Následuje úsek těla dokumentu, který započíná ukončením záhlaví (</head>). Zde je prostor pro prezentaci obsahu, jakým je například hlavní sekce stránky. Prvotním prvkem v těle je hlavní blok , obklopující obsah. V jeho nitru se nachází obrázek <img> zprostředkovávající video stream z funkce video_feed.

V závěrečné části dokumentu (</body> a </html>) jsou ukončeny veškeré strukturální prvky HTML dokumentu, což uzavírá jeho strukturu a obsah.

ZÁVĚR

Tato bakalářská práce se zabývá výběrem vhodných komponent pro konstrukci parkovací kamery navržené pro použití zemědělskou technikou. Po pečlivém průzkumu trhu a analýze požadavků jsme byly identifikovány klíčové komponenty pro tento systém. Mikropočítač Raspberry Pi 4B s dostatečným výkonem pro zpracování obrazu a měření vzdálenosti v reálném čase byl základním stavebním kamenem. K tomu byla přidána kvalitní kamera SriHome SH001 a ultrazvukový senzor HC-SR05, které společně umožňují spolehlivou detekci překážek a bezpečné parkování zemědělské techniky.

Analýza dostupných programů pro zpracování obrazu poskytla hlubší pochopení fungování celého systému. Postupné programování a testování jednotlivých částí kódu bylo klíčovým krokem při vývoji, umožnilo lépší porozumění dané problematice a získání komplexního přehledu o realizaci daného úkolu.

Vývoj tohoto systému byl postaven na pevných základech systematického přístupu a iterativního vylepšování jednotlivých částí. Tento přístup umožnil dosáhnout optimálního výsledku v podobě spolehlivého a efektivního zařízení pro parkování zemědělské techniky.

SEZANAM POUŽITÉ LITERATURY

[1] *Dějiny počítačů*, 2024. Online. In: Wikipedia: the free encyclopedia. San Francisco (CA): Wikimedia Foundation. Dostupné z: https://cs.wikipedia.org/wiki/Dějiny_počítačů. [cit. 2024-04-08].

[2] *Mikropočítače*, 2018. Online. Mikropočítače. Dostupné z: https://physics.mff.cuni.cz/kfpp/skripta/elektronika/kap9/mikropo.html. [cit. 2024-04-08].

[3] *Architektura počítače*, 2014. Online. KUTÝ, Michael. Michaelkuty.github.io. Dostupné z: https://michaelkuty.github.io/ssz-ai-hk-3/tech/2.html. [cit. 2024-04-08].

[4] *Difference between Von Neumann and Harvard Architecture*, 2021. Online. Geeksforgeeks. Dostupné z: https://www.geeksforgeeks.org/difference-between-von-neumann-and-harvard-architecture/. [cit. 2024-04-08].

[5] *MIKROPROCESOR*, Online. Navio. Dostupné z: https://www.navio.cz/slovnik-pojmu/mikroprocesor/. [cit. 2024-04-08].

[6] OLIVKA, Petr, 2020. *Paměti počítačů*. Online, Studijní materiál pro předmět Architektury počítačů a paralelních systémů. Ostrava: katedra informatiky FEI VŠB-TU Ostrava. Dostupné z: https://poli.cs.vsb.cz/edu/apps/down/pameti.pdf. [cit. 2024-04-08].

[7] *Objektiv*, 2019. Online. Megapixel. Dostupné z: https://www.megapixel.cz/objektiv. [cit. 2024-04-08].

[8] *CCD vs. CMOS - srovnání senzorů*. Online. W-technika. Dostupné z: https://www.w-technika.cz/ccd-vs-cmos-srovnani-senzoru/. [cit. 2024-04-08].

[9] *Mnoho předností, málo omezení: princip ultrazvukové technologie*, ©2024. Online. Ifm. Dostupné z: https://www.ifm.com/cz/cs/shared/technologies/ultrazvukove-senzory/mnoho-prednosti-malo-omezeni-princip-ultrazvukove-technologie. [cit. 2024-04-08].

[10] ŠÍRKOVÁ, Marie, 2024. *Raspberry Pi – co to je a jaký vybrat?* Online. Eticky.cz. Dostupné z: https://www.eticky.cz/raspberry-pi-co-to-je-a-jaky-vybrat/. [cit. 2024-04-08].

[11] ŠEBEK, Cyril, © 2024. *Lekce 1 - Úvod do Raspberry Pi Zdroj*. Online. Itnetwork. Dostupné z: https://www.itnetwork.cz/hardware-pc/raspberry-pi/uvod-do-raspberry-pi. [cit. 2024-04-08].

[12] ARMENTA, Antonio, 2022. *Introduction to Arduino: History, Hardware, and Software*. Online. Control.com. Dostupné z: https://control.com/technical-articles/introduction-to-arduino-history-hardware-and-software/. [cit. 2024-04-08].

[13] HORÁK, Jiří, 2023. *PM DPZ 6 – Filtrace obrazu*. Online. Idoc. Dostupné z: https://homel.vsb.cz/~hor10/Vyuka/PMDPZ/PMDPZ6.pdf. [cit. 2024-04-08].

[14] HLADÍK, Jiří, 2015. *Průběžné zpracování obrazu v DSP pro sledování objektů*. Online, Diplomová práce. Praha: České vysoké učení technické, Fakulta elektrotechnická, Katedra měření. Dostupné z: https://dspace.cvut.cz/bitstream/handle/10467/61449/F3-DP-2015-Hladik-Jiri-Prubezne%20zpracovani%20obrazu%20v%20DSP%20pro%20sledovani%20objektu.pdf?isAllowed=y&sequence=1. [cit. 2024-04-08].

[15] *Measure Distance using Raspberry PI with Ultrasonic Sensor* [@Collvy], 2022. Online. 2022. Dostupné z: YouTube, https://www.youtube.com/watch?v=nAopJsSkOEQ&t=153s. [cit. 2024-04-08].

[16] *How To Make A Remote Viewable Camera With Raspberry Pi (Beginner Project)* [@Make:], 2020. Online. 2020. Dostupné z: YouTube, https://www.youtube.com/watch?v=zfBHD4v8hD0. [cit. 2024-04-08].

[17] *Working with Raspberry Pi and Ultrasonic Sensor*, 2022. Online. Dostupné z: https://www.collvy.com/blog/measuring-distance-using-raspberry-pi-and-ultrasonic-sensor. [cit. 2024-04-08].

[18] *About*, © 2024. Online. OpenCV. Dostupné z: https://opencv.org/about/. [cit. 2024-04-08].

[19] *Getting Started with Videos*, 2024. Online. Dostupné z: https://docs.opencv.org/3.4/dd/d43/tutorial_py_video_display.html. [cit. 2024-04-08].

[20] FROZE, Roger. *Augmented reality for beginners : principles and practices for augmented reality and virtual computers*. CreateSpace Independent Publishing Platform, 2016. ISBN 9781539919377.

[21] SCHMALSTIEG, Dieter a Tobias HÖLLERER. *Augmented reality : principles and practice*. Boston: Addison-Wesley, 2016. ISBN 9780321883575 0-321-88357-8.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ALU	Aritmeticko-logická jednotka
HDMI	High-Definition Multimedia Interface
USB	Universal Serial Bus
DPS	Digitální signálový procesor
CPU	Centrální procesorová jednotka
LED	Light Emitting Diode
SD	Secure Digital
ARM	Advanced RISC Machine
MCU	Microcontroller Unit
VDC	Volts Direct Current
OS	Operační systém
IDE	Integrated Development Environment

SEZNAM OBRÁZKŮ

SEZNAM TABULEK