VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV MIKROELEKTRONIKY
DEPARTMENT OF MICROELECTRONICS

DETEKCE PŘEKÁŽEK
THE OBSTACLE DETECTION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. Marek Hradiský
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. Pavel Šteffan, Ph.D.
SUPERVISOR

BRNO 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Mikroelektronika**
Ústav mikroelektroniky

Student: Bc. Marek Hradiský ***ID:*** 119439
Ročník: 2 ***Akademický rok:*** 2015/16

NÁZEV TÉMATU:

Detekce překážek

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte možnost snímání a vyhodnocení obrazu získaného z kamery bezdrátově řízeného modelu vozidla. Navrhněte funkční řešení snímání obrazu a vytvořte program, který bude schopný v obrazu detekovat překážku. Obraz zároveň zobrazujte na tabletu nebo mobilním telefonu. Zvažte možnost použití optického snímače pro určení vzdálenosti detekované překážky.

DOPORUČENÁ LITERATURA:

Podle pokynů vedoucího práce

Termín zadání: 8.2.2016 ***Termín odevzdání:*** 26.5.2016

Vedoucí práce: doc. Ing. Pavel Šteffan, Ph.D.
Konzultant diplomové práce:

doc. Ing. Lukáš Fujcik, Ph.D., *předseda oborové rady*

Abstrakt

Diplomová práca sa zaoberá návrhom a popisom realizácie snímania obrazu pomocou minipočítača Raspberry PI a možnosťou detekcie objektu v obraze. V práci je popísaný minipočítač Raspberry PI ako aj snímanie obrazu modulom PiCamera a princíp spracovania obrazu prostredníctvom OpenCV.

Kľúčové slová

Raspberry PI, minipočítač, UDP, TCP, kamera, OpenCV, senzor, PiCamera

Abstract

Diploma thesis deals with design and the processing of video captured by Raspberry PI and the possibility of object detection in video. In this thesis is described Raspberry PI as well as video capturing by PiCamera module. There is also section about video processing by OpenCV.

Keywords

Raspberry PI, minicomputer, UDP,TCP , camera, OpenCV, sensor, PiCamera

HRADISKÝ, M. *Detekce překážek.*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2016. 60 s. Diplomová práce. Vedoucí práce: doc. Ing. Pavel Šteffan, Ph.D.

Prohlášení

Prohlašuji, že svoji diplomovou práci na téma **„Detekce překážek“** jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

podpis autora

Poďakovanie

Touto cestou sa chcem poďakovať vedúcemu diplomovej práce doc. Ing. Pavlovi Šteffanovi, Ph.D. za odbornú pomoc a cenné rady, ktoré mi poskytol pri jej vypracovaní.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Táto diplomová práca sa zaoberá spracovaním obrazu získaného z kamerového modulu pre minipočítač Raspberry Pi model 2. V práci je popísaný minipočítač Raspberry PI, jeho verzie a hardvérové vybavenie. Detailne je opísaná inštalácia operačného systému Raspbian, inštalácia kamerového modulu ako aj modulov OpenCV a PiCamera a získavanie obrazu. Minipočítač je na trhu už cez štyry roky a existuje mnoho aplikácii a rozšírení, ktoré z neho robia výborného pomocníka pre výuku v školách. Práca rozširuje použitie v kategórii spracovávania obrazu, ktoré sa vďaka vydaniu niekoľkých dostupných kamerových modulov stáva dostupnejším. V teórii je stručne popísané a vysvetlené počítačové videnie ako aj základ predspracovania a spracovania obrazu videa, ktoré má viesť ku detekcii objektov v obraze. Rozšírením detekcie obrazu je spolupráca optického senzoru vzdialenosti, ktorý pomáha určovať relevantné objekty ako prekážky. V praktickej realizácii je vytvorený program pre minipočítač Raspberry Pi, ktorý je schopný detekovať prekážku a nasnímaný obraz odoslať prostredníctvom WiFi do tabletu, kde následne zobrazený.

1 Raspberry PI

1.1 Popis

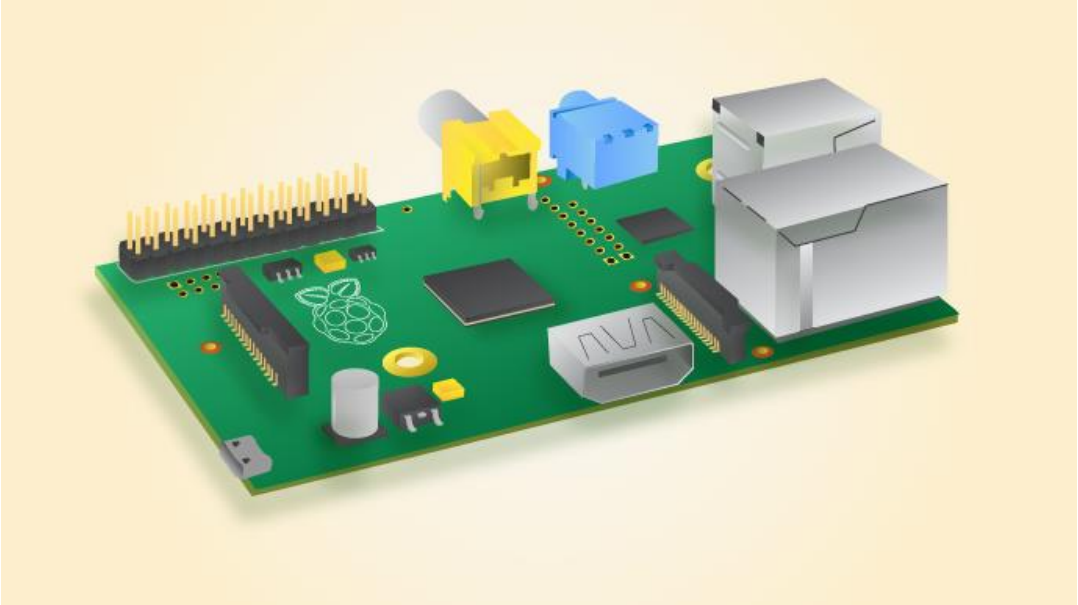
Raspberry PI je minipočítač veľkosti kreditnej karty s ARM procesorom vyvíjaný britskou nadáciou Raspberry Pi Foundation. Primárnym cieľom projektu Raspberry PI bolo vytvorenie prostriedku pre výuku programovania pre deti za rozumnú cenu. Po uvedení si ale Raspberry PI získalo obľubu nielen u detí, ale aj u širšieho spektra programátorov a vývojárov, ktorý uvítali výhodnú cenu minipočítača [1].

Prvým modelom minipočítača Raspberry Pi bol v roku 2012 model B. Stal sa základom, z ktorého vzišlo do dneška ďalších 6 modelov, v rôzných prevedeniach, odlišujúcich sa prevažne hardvérovou výbavou. Výnimku tvoria Raspberry Pi model A+ a model Zero, ktoré sa odlišujú okrem hardvéru aj rozmermi.

1.2 Prvá generácia

Prvá generácia (2012 - 2014) doniesla postupne 4 modely minipočítača. Základom je 700 MHz ARM procesor ARM1176JZF-S (ARM11) ktorý je možné nataktovať na 1 GHz. O grafiku sa stará grafický procesor Broadcom VideoCoreIV schopný prehrávať Full HD video. Periférie je možné pripojiť prostredníctvom USB portov, HDMI, minipočítač tiež obsahuje audio vstup/výstup a výstup kompozitného videa. Pre svoje kompaktné rozmery vývojári namiesto pevného disku zvolili variantu bootovania systému z SD pamäťovej karty. Pre pripojenie karty je na doske implementovaný slot pre SD kartu. Minipočítač je napájaný prostredníctvom mini USB portu. Rozšírením Raspberry Pi oproti klasickým počítačom sú špecializované zbernice pre pripojenie hardwaru a to GPIO, CSI camera interface a DSI display interface. Hlavnými rozdielmi medzi modelmi A a B je veľkosť pamäte RAM, kde modely s označením A majú 256 MB pamäte a modely B 512 MB pamäte. Modely A tiež obsahujú iba jeden USB port pre pripojenie periférií, modely B maju USB porty dva. Model B navyše obsahuje sieťovú kartu s konektorom RJ45.

Roku 2014 boli uvedené modely A+ a B+, ktoré sú prepracovanými modelmi A a B. U A+ sa zmenili rozmery z 85,6x53,98x17mm na 55x65x17mm. U B+ rozmery ostali, ale dostal štyry USB porty. U oboch modelov došlo k zmene u SD karty, kde bola implementovaná úspornejšia micro SD karta, zároveň bol samostatný kompozitný výstup presunutý do A/V konektoru a zvýšil sa počet GPIO portov z 2x13 na 2x20 [1].



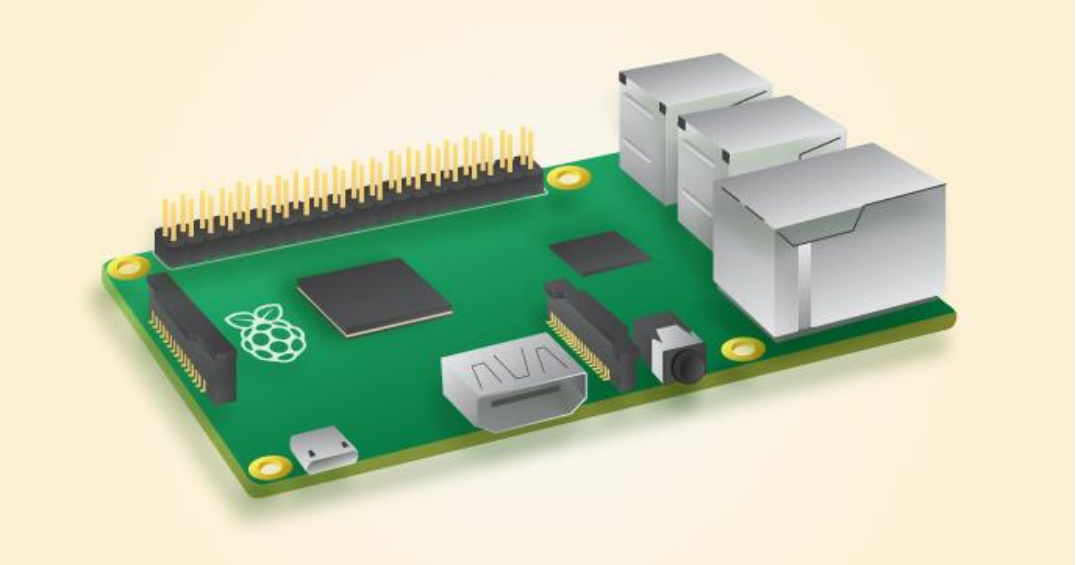
Obr. 1. Raspberry PI model B [1]

1.3 Druhá a tretia generácia

Očakávaná zmena prišla roku 2015 a to uvedením druhej generácie s modelovým označením Raspberri Pi 2 model B. Druhá generácia dostala nový výkonnejší 32b 4-jadrový procesor ARM Cortex-A7 taktovaný na 900 MHz a 1GB pamäte RAM. Vďaka novému procesoru bolo možné rozšíriť aj ponuku operačných systémov ako je Snappy Ubuntu Core a Windows vytvoril s príchodom Windows 10 distribúciu IoT Core pre takzvaný internet vecí.

Koncom roku 2015 bol uvedený minipočitač Raspberry Pi Zero, ktorý vyniká svojimi rozmermi 65 mm x 30 mm x 5 mm. Procesor má rovnaký ako prvá generácia, taktovaný ale na 1 GHz. Pre periférie má k dispozícii mini HDMI, jeden micro USB port a 2x20 neosadených GPIO portov.

Momentálne posledným uvedeným modelom je Raspberri Pi 3 Model B, uvedený v roku 2016. Oproti druhej generácii bol navýšený výkon minipočítača vďaka 4-jadrovému 64b procesoru ARM Cortex-A53 taktovaným na 1,2 GHz. Druhou dôležitou zmenou je integrovanie WiFi 802.11n spolu s Bluetooth 4.1 pri rovnakých rozmeroch ako u predošlých modelov [1].



Obr. 2. Raspberry PI model 2 B [1]

Tab. 1. Modely Raspberry PI 1. generácia

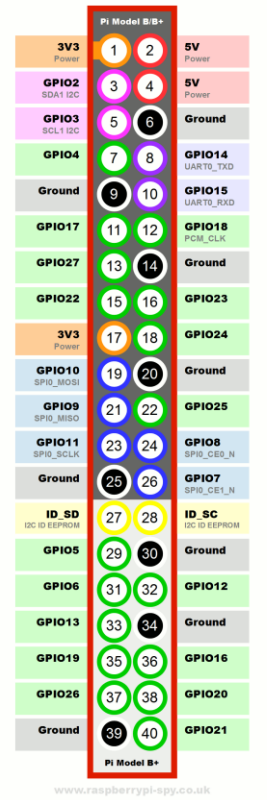
Raspberry Pi	Model A	Model A+	Model B	Model B+
CPU	ARM1176JZF-S	ARM1176JZF-S	ARM1176JZF-S	ARM1176JZF-S
SoC	BCM2835	BCM2835	BCM2835	BCM2835
RAM	256 MB	256 MB	512 MB	512 MB
Pamäťová karta	SD	micro SD	SD	micro SD
USB	1	1	2	4
GPIO	26	40	26	40
Ethernet	nie	nie	áno	áno
HDMI	áno	áno	áno	áno
Rozmery	85x53 mm	65x56 mm	85x53mm	85x53 mm

Tab. 2. Modely Raspberry PI 2. a 3. generácia

Raspberry Pi	2 Model B	Zero	3 Model B
CPU	ARM Cortex-A7	ARM1176JZF-S	ARM Cortex-A53
SoC	BCM2836	BCM2835	BCM2837
RAM	1 GB	1 GB	512 MB
Pamäťova karta	micro SD	micro SD	micro SD
USB	4	1 micro	4
GPIO	40	40	40
Ethernet	áno	áno	áno
HDMI	áno	áno Mini	áno
Rozmery	85x53 mm	65x30 mm	85x53mm

1.2 GPIO

GPIO (general purpose input/optput) je konektor, ktorý obsahuje 2x13 pinov (2x20 pinov od verzie B+), kde je vyvedených 8/17 programovateľných, vstupno-výstupných pinov, komunikačné rozhrania UART, 2x I2C a SPI, ktorých piny sa dajú preprogramovať aj na iné použitie ako vstupno-výstupné. Konektor obsahuje aj dva výstupy s napätím 3,3 a 5V. Všetky piny sú bez ochrany pripojené na procesor, ktorý pracuje s napäťovou úrovňou 3,3 V. To znamená, že všetky aplikácie, zahrnujúce použitie GPIO portu musia byť optimalizované na túto napäťovú úroveň. GPIO porty môžeme využívať priamo z operačného systému minipočítača. Programovať je ich možné pomocou programovacích jazykov ako je C, JAVA, Python, Perl a pod. Pre použitie je potrebné nainštalovať knižnicu bcm2835. Pristupovať k portom bude následne možné pomocou ukazateľov.



Obr. 3. Rozloženie GPIO portu [1]

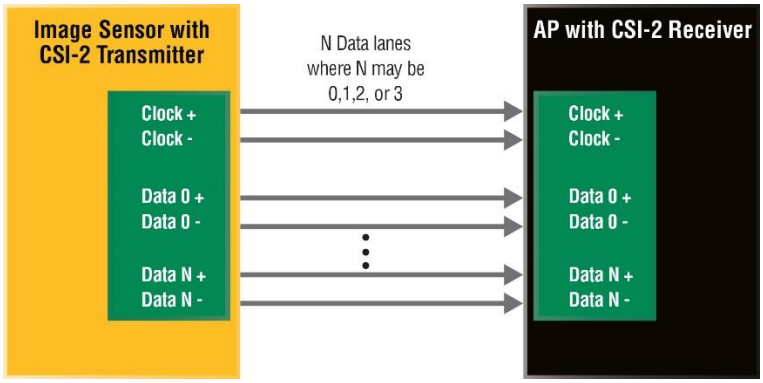
1.3 DSI

DSI display interface je vysokorýchlostný sériový konektor, pomocou ktorého ľahko a rýchlo možme k minipočítaču pripojiť display. DSI je spojený priamo z GPU minipočítača, obchádza tak procesor kvôli čomu dokáže byť rýchlejší ako HDMI. Oficiálny 7“ display má 800 x 480 pixelov a napájaný je z GPIO portu.

1.4 CSI

CSI camera serial interface je vysokorýchlostný sériový port pre pripojenie kamery. Interface podporuje špecifikáciu Mobile industry procesor Interface (MIPI), presnejšie CSI-2. Kamera pripojená cez toto rozhranie získa priame prepojenie s procesorom minipočítača. CSI pracuje podobne ako fyzická vrstva známa ako D-PHY2. Táto fyzická vrstva je známa ako LOW Voltage differential Signalling. Ide o systém nízkonapäťových aplikácii, pracujúcich na potenciálu 1,2 V a prenosom dát rýchlosťou 800 Mb/s v jednom pruhu (lajne). Rýchlosť prenosu môže byť rozdelená do maximálne 4 pruhov s maximálnou teoretickou priepustnosťou 1 Gb/s na jeden Pruh. Raspberry Pi má k dispozícii dva pruhy. CSI potrebuje pre 1Gb pásmo 6 pinov. 2 pre dáta, 2 pre ovládanie a 2 pre hodiny. Teoretická šírka pásma 2Gb zodpovedá približne rozlíšeniu

fotoaparátu 5 Mpx a nahrávaniu videa v rozlíšení HD, t.j. 1920x1080 pixelov pri 30 snímkach za sekundu [2].



Obr. 4. Komunikácia CSI interface [2]

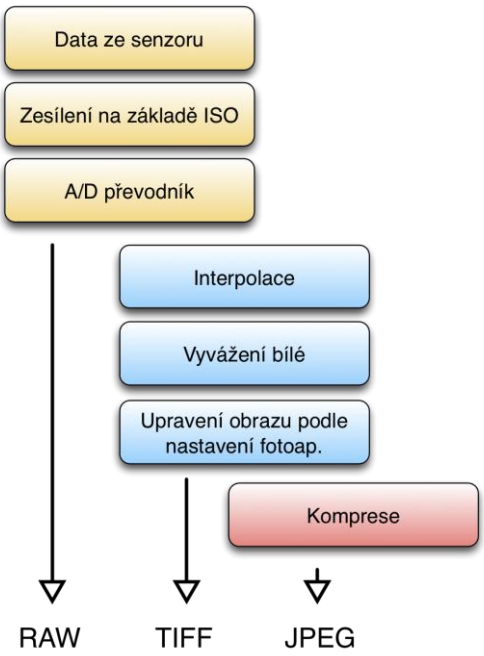
2 Počítačové videnie

2.1 Úvod

Počítačové videnie je odvetvie výpočtovej techniky a vývoja softvéru, zaoberajúce sa vývojom zariadení snímajúcich obraz pod účelom získavať z neho informácie. Vo všeobecnosti ide o rozpoznávanie objektov z videa alebo fotografií. Potenciál počítačového videnia sa využíva v priemysle, medicíne, pri ovládaní procesov, detekcii javov, napomáha pri modelovaní objektov alebo sa využíva pri interakcii užívateľa s počítačom. Počítačové videnie môžeme rozdeliť do niekoľkých krokov:

- Snímanie/digitalizácia obrazu
- Predspracovanie
- Segmentácia obrazu
- Popis objektov a porozumenie obsahu

Snímanie obrazu je prevedenie optickej veličiny na elektrický signál spojitý v čase. Konverzia analógovej informácie do digitálnej podoby. Pre snímanie obrazu je potrebný snímač. V kamerách sa používajú CCD a CMOS snímače. Tie premieňajú fotóny svetla na elektrický signál. Tento signál je potrebné previesť na digitálny pomocou A/D prevodníku – vzorkovanie, kvantovanie. Hodnoty pre jednotlivé pixely sa z A/D prevodníku uložia do matice, ktorá reprezentuje nasnímaný obraz. Tieto surové data sa označujú ako RAW. Je to v podstate formát uloženia zpracovaných dát zo snímača kamery. Enkódery ale umožňujú surové data za nás upravovať a priamo ukládať aj vo formátoch ako je TIFF alebo JPEG [3]. Proces snímania a digitalizácie obrazu je znázornený na obr. 5.



Obr. 5. Snímanie/digitalizácia obrazu [3]

Predspracovanie obrazu má za úlohu zlepšiť kvalitu snímaného obrazu na najnižšej úrovni. Snímanie a digitalizácia vedú k neželaným deformácia a skresleniu ktoré je vhodné potlačiť. Navyše môžeme predspracovaním zvýrazniť parametre alebo črty obrazu, potrebné pre ďalšie spracovanie. Operácie, ktoré sa využívajú v predspracovaní sa väčšinou rozdeľujú do troch základných metód. Sú nimi jasové transformácie, lokálne predspracovanie a geometrické transformácie [3].

Jasové transformácie sú bodové, lokálne a globálne, kde pre bodové sa vypočíta jasová konštanta z jedného pixelu, pre lokálne z hodnôt niekoľkých pixelov a globálne z celého obrazu. Inou možnosťou je vytvorenie obrazu s rovnako rozloženou jasovou konštantou pomocou histogramu.

Lokálne predspracovanie zahrnuje metódy ako vyhladzovanie, ktoré je založené na priemerovaní a účelom je rozostriť hrany alebo potlačiť vyššie frekvencie a detekcia hrán, ktorá využíva gradientné operátory na zvýraznenie hrán, ale aj šumu v obraze.

Geometrické transformácie služia na eliminovanie deformácii, ktoré môžu byť výsledkom snímania. Tie sú napríklad Škálovanie, ktoré slúži na zmenšovanie alebo zväčšovanie obrazu, otáčanie, posunutie alebo skosenie [4].

Segmentácia obrazu má za úlohu rozdeliť obraz na jednotlivé časti (segmenty). Slúži k rozdeleniu obrazu na oblasti so spoločnými vlastnosťami ako je jas a farba. Typickým príkladom segmentácie je identifikácia pozadia a určenie oblasti významného

prvku. Základné metódy segmentácie sú prahovanie, regionálne metódy, detekcia hran, sledovanie hranice, aktívna kontúra a segmentácia rozvodím.

Popis objektov je svojím spôsobom stanovenie parametrov pre snímané objekty, ktoré chceme v obraze zdetekovať. Ako jednoduchý popis môže byť stanovenie rozmerov (plochy), ktorú má objekt v obraze zaberať. Porozumenie obsahu s popisom objektov úzko súvisí. Popis objektu určí, aký má byť objekt veľký, koľko pixelov z obrazu má zaberať, no to je možné len ak objekt bude v predpísanej vzdialenosti od kamery. Ak by sa ale predmet striedavo vzďaľoval a približoval, jeho určenie by bolo podľa samotného popisu objektu nemožné a tak bez hlbšieho porozumenia obsahu, ktoré môže pracovať aj s ďalšími parametrami by počítačové videnie nebolo použiteľné [5].

2.2 Farebné modely

Farebný model je v podstate kombinácia niekoľkých základných farieb a popisuje spôsob ich miešania. Medzi najznámejšie farebné modely patria RGB, CMYK, HSV a YUV.

RGB model je asi najrozšírenejším farebným modelom. Je to z dôvodu, že je používaný v zobrazovacích zariadeniach. Z názvu vyplýva, že ho reprezentujú farby červená, zelená a modrá. Tento farebný model je aditivný. To znamená, že farby sú sčítavané. Každú farbu môže reprezentovať hodnota od 0 do 255. Najnižšia hodnota RGB modelu, teda 0,0,0 je čierna a najvyššia hodnota 255,255,255 reprezentuje bielu [6].

CMYK je farebným modelom známym predovšetkým v tlačiarniach. Farby reprezentujúce tento model sú azúrovná, purpurová, žltá a doplnkovou farbou tohoto modelu je čierna. Je to z dôvodu, že síce zložením farieb CMY vznikne čierná, tá ale nie je úplne kvalitná. Tento farebný model je subtraktívny, čo znamená, že farby sa odčítajú. Farba biela je reprezentovaná hodnotami 0,0,0 a čierna 255,255,255. Modelom CMYK sa nedajú zobraziť všetky farby z modelu RGB [7].

HSV model sa na rozdiel od predošlých modelov nepoužíva na zobrazenie farby ich miešanie, ale farby určuje parametrami tón farby, sytosť farby a jas - množstvo bieleho svetla. Model značne odpovedá ľudskému vnímaniu farieb [8].

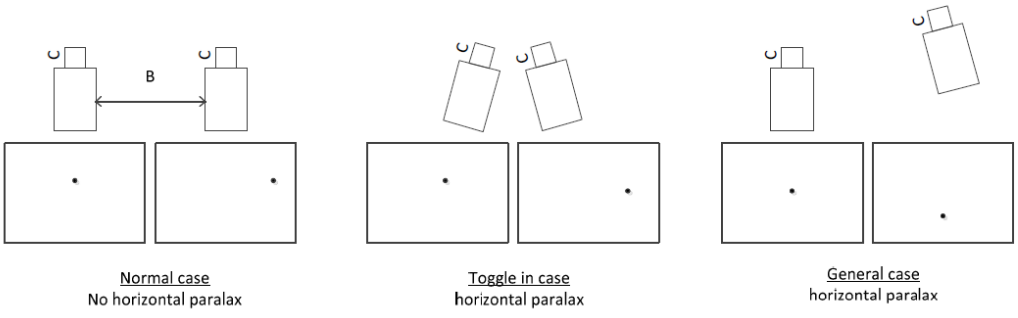
YUV je model, kde Y je jas a U a V sú farebné zložky. Model bol vytvorený pre televýznu techniku, kedy bolo potrebné zariadiť prenos farebného obrazu, kompatibilného s čiernobielym vysielaním [9].

2.3 3D rekonštrukcia

3D je odvetvie, ktoré má vo svete počítačového videnia svoje nezastupiteľné miesto. Ak hovoríme o 3D, hovoríme o svete, ktorý má tretí rozmer a môžeme ho nazvať objem, alebo hĺbka. V tomto svete figuruje všetko okolo nás. V posledných rokoch, s nástupom výkonnejšieho hardvéru a dostupnejších kamier a senzorov je 3D rekonštrukcia stále populárnejšou. Využitie je od medicíny, cez automatizáciu výroby, robotiku, až po natáčanie filmov či vytváranie video hier. Je totižto veľkou výhodou, ak stroj dokáže rozoznávať 3D priestor. Môže tak analyzovať, merať a na základe získaných dat vykonávať svoju úlohu. Z druhej strany umožňuje zobrazenie dát v 3D. 3D obraz je možné získavať pomocou metód aktívnych a pasívnych.

Medzi aktívne metódy patrí 3D skener. Ten umožňuje pomocou laserov snímať trojrozmerný obraz. Lasery pracujú na princípe merania vzdialeností. Týchto meraní je veľké množstvo, až milióny meraní, na základe ktorých je možné následne zrekonštruovať 3D objekt. 3D skener využívá triangluláciu. Laser a kamera, ktorá sníma laserový lúč v definovanej polohe a pomocou trigonometrie sú vypočítavané súradnice objektu. Druhou metódou, s ktorou pracujú laserovné 3D skenery je metóda, založená na počítaní doby letu lúča, kde fázový posun lúča a doba letu a presná poloha kamery slúžia pre výpočet súradníc objektu.

Pasívne metódy na rozdiel od aktívnych vyúživajú iba kamery. Pre 3D rekonštrukciu potrebujú dva snímky s rôznými pozíciami kamier. Kamery môžu byť posunuté v ose x, y ale aj z. Dôležitou podmienkou je mať k dispozícii snímky rovnakej scény. Prvým krokom je vždy nájdenie zhodných bodov v oboch obrazoch. Ďalšími potrebnými údajmi sú ohnisková vzdialenosť kamery, veľkosť pixelov a pozícia zhodného bodu a údaje o vzájomnej polohe kamier vzájomný uhol natočenia a vzdialenosť. Následne je pomocou triangulácie prevedený výpočet priestorovej súradnice [10].



Obr. 6. Možné vzájomné postavenie kamier [13]

2.4 Optimalizácia

Výkon hardvéru nebýva nekonečný, preto samotná optimalizácia je najdôležitejším krokom. Optimalizáciou je v tomto prípade myslené upravenie parametrov obrazu. Veľké rozlíšenie obrázku môže byť neželané. Veľké rozlíšenie znamená viac potrebnej pamäte, ale aj výkonu. Operácia pri rozlíšení 1024 x 768 pixelov, môže trvať 2-krát dlhšie, ako pri rozlíšení 640 x 480 pixelov, pretože máme viac ako dvojnásobný počet bodov. Zároveň zaberá aj viac miesta v pamäti. Záleží však na tom, aké objekty budeme rozpoznávať. Menší objekt vyžaduje jemnejšie rozlíšenie a u veľkého objektu stačí rozlíšenie menšie. S veľkosťou a rozlíšením je spojené, akým spôsobom je uložená informácia o farbe a hĺbke pre jednotlivé pixely. V základe môže byť obrázok čierno-biely, kde je farba uložená v jednom bite, v odtieňoch sivej, kde je farba uložená v 8 bitoch, a farebný, kde je farba uložená v 4 bitoch pri 16-tich farbách, až v 24 bitoch pre režim True Color (16 miliónov farieb). Informácia o farbe môže byť, čo sa veľkosti obrázka týka, zaťažujúca. Dôležitou je v prípade, ak je hľadaný farebný objekt. Ináč, je po zmenšení rozlíšenia na potrebné minimum, odstraňovaná práve informácia o farbe.

2.5 Spracovanie obrazu

Detekcii objektu, teda spracovaniu obrazu, predchádzajú úlohy predspracovania, ktoré majú za úlohu obraz pripraviť. Pomocou segmentačných techník sú oddelené objekty od pozadia. Tým sa odstránia nepotrebné objekty a ostanú len tie, ktoré sa ďalej analyzujú, porovnávajú a upravujú. Ak sa podarí v obraze nájsť očakávaný objekt, dá sa sledovať napríklad jeho pohyb v ďalších snímkoch. Proces je vo všeobecnosti ale omnoho zložitejší. Objekty majú v reálnom živote nepravidelné tvary, hýbajúci sa objekt sa otáča, zväčšuje, zmenšuje a vziať do úvahy všetky aspekty je náročná úloha, ktorá sa vo svete deň čo deň posúva na vyššiu úroveň [5].

2.6 Metódy spracovania obrazu

Odstránenie šumu je proces, prostredníctvom ktorého je možné zredukovať signál šumu zo snímaného obrázku. Šum môže spôsobovať slabé osvetlenie alebo nedokonalosti snímača. Pri spracovaní môže spôsobovať až nepravdivé detekcie objektov. Pre jeho odstránenie existuje niekoľko metód.

Gaussov filter je efektívna metóda pre potlačenie Gausovho šumu. Odstraňuje šum prostredníctvom konvolúcie s maskou. Metóda vedie k rozmazaniu obrázku. Priemerovanie je metóda, kde hodnota pixelu je určená jeho priemerom a jeho najbližších susedov. Mediánový filter vezme hodnotu daného pixelu a jeho okolia a nová hodnota je ich medián. Gaussov filter používa 2D gaussovú funkciu:

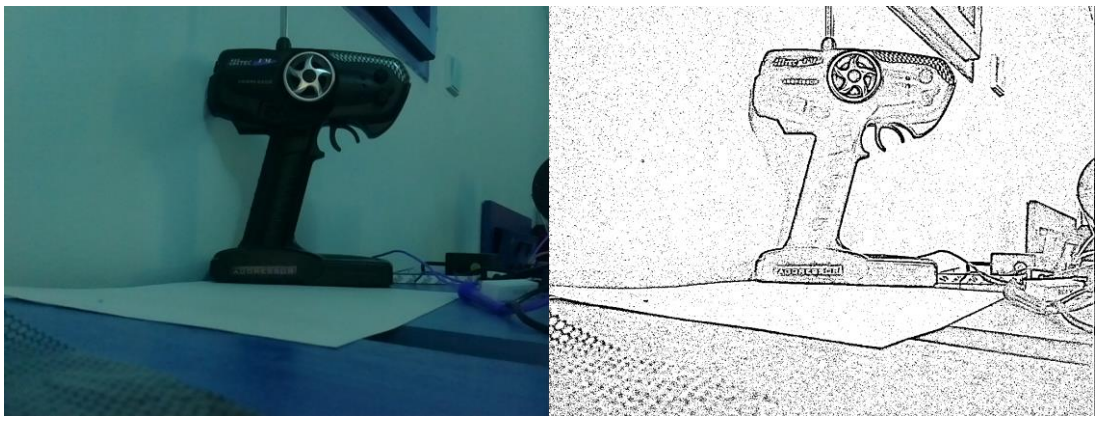
$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{2.1}$$

kde x a y sú súradnice pixelov a σ je smerodajná odchylka [12].

Prahovanie (viz. obr. 7) je najjednoduchšia segmentačná technika, ktorá je výpočtovo nenáročná a rýchla. Technika používa na oddelenie objektu od pozadia jasovú konštantu nazývanú prah. Ako prvé je potrebné priradiť každému pixelu hodnotu, zodpovedajúcu jasu pixelu pomocou histogramu. Následne zvolíme prahovú hodnotu. Všetky pixely, ktoré majú hodnotu vyššiu ako prahovú sú pixely objektu a ostatné pixely sú pozadie. Pre prípad viacerých objektov bude prahových hodnôt viacero. Pre každý objekt oddeľovaný od pozadia. Ak obraz neumožňuje určiť globálny prah je možné použiť adaptívne prahovanie. Prahovanie je definované funkciou:

$$f(c) = \begin{cases} 255 \; ak \; f(x,y) > T \\ 0 \quad ak \; f(x,y) \leq T \end{cases}, \tag{2.2}$$

kde *c* je vstup, *f(c)* je výstup, *T* je prahovacia hodnota, *0* a *255* sú nové hodnoty pre daný bod a *x* a *y* sú súradnice bodu.



Obr. 7. Adaptívne prahovanie

Určovanie hrán v obraze má pri spracovaní tiež svoju úlohu. Všade, kde sa v obraze prudko mení jasová hodnota existuje hrana. Hrana určuje význačné oblasti obrazu. Zvýraznením hrán je možné docieliť pre človeka výraznejší obraz, pretože táto informácia je pre ľudský zrak dôležitejšia ako samotný obsah objektu. V počítačovom videní je možné pomocou určenia hrán vyznačiť hranice objektov a oddeliť od seba jednotlivé oblasti. Týmto spôsobom sa dajú odfiltrovať menej dôležité oblasti a tak výrazne zjednodušiť ďalšie spracovanie [13]. Detektory hrán sú najčastejšie založené na prvej a druhej derivácii. Medzi najstaršie a najjednoduchšie hranové detektory patrí Robertsov hranový detektor. Pracuje s maticou 2x2:

$$h_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \tag{2.3}$$

$$h_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \tag{2.4}$$

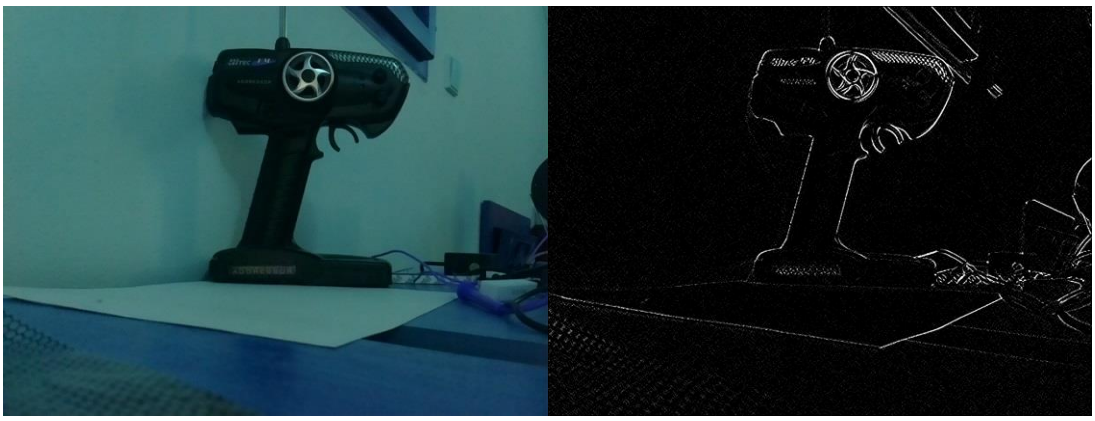
čo vedie často k detekovaní šumu ako hrany, pretože používa deriváciu v jednom bode. Výhodou je ale rýchlosť. Sobelov hranový detektor (viz. obr. 8) na rozdiel od Robertsovho počíta deriváciu z troch bodov:

$$h_1 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, \tag{2.5}$$

$$h_2 = \begin{bmatrix} -2 & 1 & 0 \\ 1 & 0 & 1 \\ -1 & 1 & -1 \end{bmatrix}, \ldots., \tag{2.6}$$

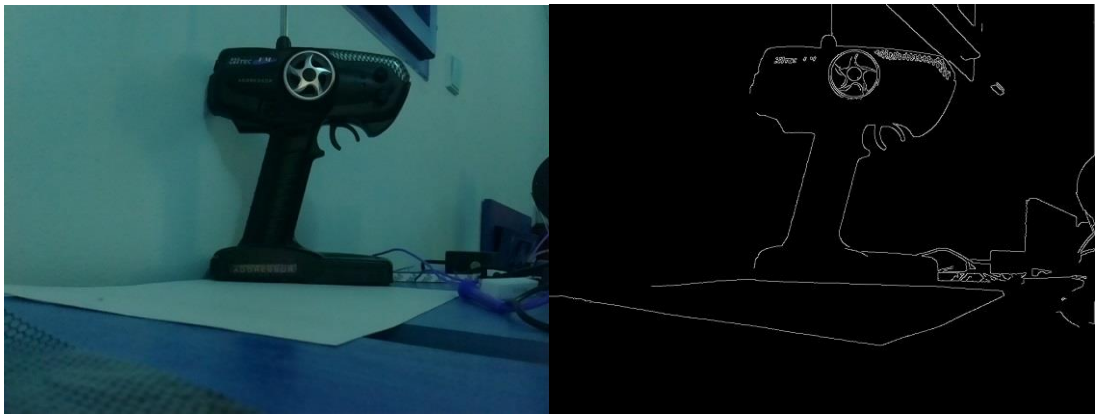
$$h_n = \begin{bmatrix} -1 & 2 & 0 \\ 0 & -1 & 0 \\ 2 & 1 & 0 \end{bmatrix}. \tag{2.7}$$

Týmto vykonáva aj čiastočné vyhladzovanie obrazu a je odolnejší voči šumu.



Obr. 8. Sobelov hranový detektor

Medzi pokročilejšie hranové detektory patrí Cannyho hranový detektor (viz. obr. 9). Na určenie hrán používa pre získanie čo najlepšieho výsledku niekoľko krokov. Prvým krokom eliminuje šum Gaussovým filtrom. Druhým krokom určí hrany napríklad Sobelovým detektorom. Tretím krokom je ztenšenie detekovaných hrán, respektíve vybratie lokálnych maxím gradientov z predchodzieho kroku. Ostanú teda významné hrany. Posledným krokom je prahovanie. Ktoré slúži k eliminácii hrán a tak sú výsledkom dôležité hrany [13].

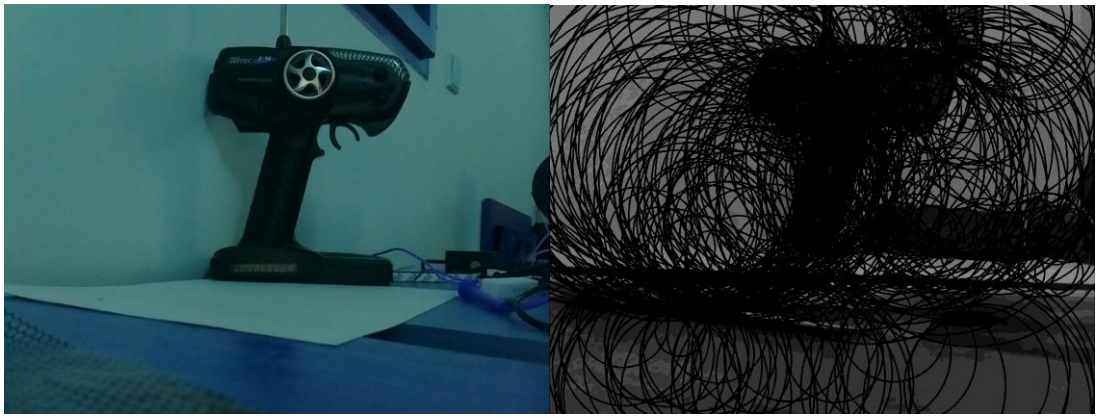


Obr. 9. Cannyho hranový detektor

Kontúry sú zoznamy bodov, ktoré ohraničujú objekt. Kontúry sa môžu vyhľadávať pomocou nachádzania poľa postupnosti bodov, kde jedna postupnosť reprezentuje jednu kontúru. Kontúry sú ale všade, kde nastane určitý zlom vo farbe alebo v jasovej zložke. Na pokrčenom papieri by každý ostrejší ohyb reprezentoval jednu kontúru, ten by bol následne rozdelený na mnoho objektov, preto je vhodné spracovaný obraz rozostriť, aby vznikly vo výsledku plnšie objekty. Inou možnosťou je určovať objem objektov, ktoré kontúry definujú. Ostanú tak len objekty, ktoré reprezentujú väčšiu plochu na obrázku.

Rozdielove snímky je spôsob segmentácie obrazu, kde dochádza k porovnávaniu dvoch po sebe idúcich snímkov. Odčítaním snímkov vznikne výstup, ktorým sú hrany pohybujúceho sa objektu v obraze. Metóda porovnáva jednotlivé pixely snímkov a určujúcim kritériom je zmena jasu.

Houghova transformácia je metóda pre nájdenie objektu v obraze. Je používaná na nájdenie jednoduchých objektov v obraze, pretože potrebujeme poznať popis objektu. Je používaná tam, kde je možné objekt popísať jednoduchými krivkami. Metóda bola pôvodne určená na nájdenie priamky v obraze. Neskôr bola rozšírená na detekciu ľubovoľných pozícii a bola nazvaná Zobecnená Houghova transformácia, ktorá je dnes jedna z najpoužívanejších metód [14].



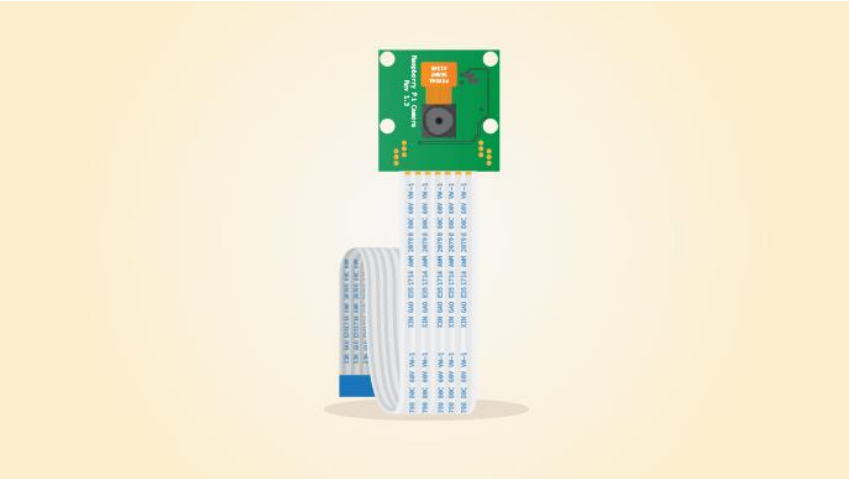
Obr. 10. Houghová kruhová transformácia

3 Pi Camera

3.1 Camera Board

Raspberry Foundation vytvorilo roku 2013 vlastný modul kamery s CSI rozhraním. Modul kamery bol vydaný v dvoch verziách a to Pi Camera Board, ktorá je štandardom a Pi Noir Camera Board, ktorá je bez infračerveného filtra a je možné ju použiť za zhoršených svetelných podmienok či s použitím infračerveného svetla aj v tme. Rozmery kamerových modulov sú 20 x 25 x 9 s váhou len 3 gramy. V kamerách je použitý senzor OmniVision OV 5647 s maximálnym rozlíšením 2592 x 1944 pixelov (5 megapixelov). S týmto maximálnym rozlíšením dokážeme snímkovať do 15 fps. 1080p video (1920 x 1080 pixelov) dokážeme poriadiť s maximálne 30 fps a pri rozlíšení 640 x 480 je to až 90 fps [1].

Roku 2016 bola kamera upgradovaná. Dostala nový senzor od Sony IMX219, ktorý má maximálne rozlíšenie 3280 x 2464 pixelov (8 megapixelov). Rozmery modulu ostali zachované.



Obr. 11. Modul Picamera [1]

3.2 Modul Picamera

Picamera je špeciálne vyvinutý a rýchlo sa rozvíjajúci modul pre Pi Cameru, určený pre programovací jazyk Python 2.7 a vyšší. Modul podlieha licencii BSD [http://opensource.org/licenses/BSD-3-Clause]. Tým, že je modul vyvýjaný priamo pre Pi Cameru zabezpečuje pomerne jednoduché a rýchle nastavenie a používanie kamery. S pomocou modulu Picamera dokážeme snímať video alebo jednotlivé obrázky. Modul umožňuje nastavenie rozlíšenia snímaného obrazu, počtu snímkov za sekundu, ale aj nastavenie výstupného formátu, rýchlosti uzávierky či ukladanie a zobrazovanie nasnímaných obrázkov a videí [15].

3.3 Základna práca s modulom Picamera

Modul umožňuje inicializovať snímanie videa ako aj jednotlivých snímkov, či snímanie videa a popri tom vytvoriť snímok bez toho, aby bolo video akýmkoľvek spôsobom prerušené. Taktiež je možné počas snímania zobrazovať náhľad obrazu na pozadí. Nasnímané video a jednotlivé obrázky môžeme ukladať priamo na disk. Následujúci kód ukáže ako nasnímať 30 sekundové video.

```
with picamera.PiCamera0 as camera:
      Camera.resolution = (1024, 768)
      Frame.rate = (25)
      Camera.start_prewiew = ()
      Camera.start_recording('video.h264')
      Camera.wait_recording(30)
      Camera.stop_recording()
      Camera.stop_preview()
```

Na začiatku kódu je inicializovaná kamera ako „camera“, nastavené rozlíšenie, počet framov, spustí sa náhľad a sníma sa video počas 30 sekúnd, ktoré sa zároveň ukladá na SD kartu.

Pre nasnímanie obrázka použijeme namiesto

```
      Camera.start_recording()
      Camera.capture('obrazok.jpg')
```

čo uloží nasnímaný obrázok na SD kartu. Modul umožňuje do obrazu jednoducho vkladať text v maximálnej dĺžke 255 znakov.

```
      Camera.annotate_text = 'zobraz text'
```

Pre nasnímanie obrázka bez komprimácie, napríklad vo formáte RGB a uloženia do zvolenej premennej

```
      Camera.capture(obraz_rgb, 'rgb')
```

Ďalšie možnosti práce s modulom Picamera nájdeme na [15]

3.4 OpenCV

OpevCV je knižnica, zameriavajúca sa na spracovanie obrazu. Knižnica bola vytvorená firmou Intel roku 1999, je v podstate slobodným softvérom pod licenciou BSD a implementovaná pre systémy Windows, Linux a MAC OS X. Existujú aj jej nadstavby či skôr kópie určené pre rôzne programovacie jazyky, keďže OpenCV je primárne určená pre C/C++. OpenCV poskytuje optimalizované metódy a deklarácie tried pre spracovanie obrazu, nástroje pre užívateľskú interakciu, metódy pre ukladanie

obrazu s podporou rôznych formátov, kvalifikátory pre pripojenie kamery a podobne. Knižnica obsahuje stovky funkcií pre urýchlený vývoj aplikácii.

Knižnica pozostáva zo štyroch základných knižníc. CV, ktorá obsahuje hlavné obrazové funkcie, CXCORE, ktorá obsahuje základné dátové štruktúry, funkcie pre kreslenie a pod, HIGHGUI, ktorá obsahuje funkcie pre zobrazenie okna, načítanie obrazu a pod, a CVAUX, ktorá obsahuje doplnkové funkcie OpenCV.

3.5 Základná práca s OpenCV

Pre načítanie obrázku z adresára a zobrazenie v samostatnom okne použijeme nasledujúce:

```
obrazok = cv2.imread('obrazok.jpg')
cv2.namedWindow('okno s obrazkom',cv2.WINDOW_NORMAL)
cv2.imshow(,okno s obrazkom',obrazok )
```

Kód načita obrázok.jpg do premennej obrazok, cv2.namedWindow vytvorí samostatné okno s názvom „okno s obrazkom“ a cv2.imshow sa postará o zobrazenie obrázku v okne.

Pre nakreslenie vodorovnej čiary do obrázku použijeme nasledujúce:

```
Obrazok = cv2.line(obrazok, (10,10),(100,100),(255,0,0),5)
```

Prvý parameter funkcie je zdroj obrázku. Druhý a tretí parameter sú počiatočná a cieľová súradnica čiary v pixeloch. Štvrtým parametrom je farba čiary a posledný piaty parameter je hrúbka čiary v pixeloch. Podobným spôsobom môžeme nakresliť obdĺžnik či kruh.

```
Obrazok = cv2.rectangle(obrazok,(10,10),(100,100),(255,0,0),5)
Obrazok = cv2.circle(obrazok,(100,100),50,(255,0,0),-1)
```

Do obrázku môžeme pridať aj text a to nasledovne:

```
cv2.putText(obrazok,'nas text',(10,100),
cv2.FONT_HERSHEY_SIMPLEX, 4, (255,0,0),5,cv2.LINE_AA)
```

Pre niektoré funkcie potrebujeme obrázok v odtieňoch sivej. Konverzia je možná dvoma spôsobmi. Buď hneď pri načítaní obrázku, alebo počas práce s obrázkom. Pri načítaní doplníme kód o príznak nasledovne:

```
 obrazok = cv2.imread('obrazok.jpg', 0)
```

Nula znamená, že sa obrázok načita v odtieňoch sivej. Funkcia cvtColor nám umožňuje konverziu nielen do odtieňov sivej, ale aj do iných farebných modelov ako BGR, HSV, YUV:

```
obrazok = cv2.cvtColor(obrazok, cv2.COLOR%RGB2GRAY)
```

Aplikácia filtrov môže byť tiež užitočná. Ich použitie je jednoduché. Rozmazanie obrázka, ktoré je užitočné pred detekciou hrán prevedieme pomocou funkcie blur nasledovne:

```
Blur = cv2.blur(obrazok,(5,5))
```

kde (5,5) nám nastaví veľkosť matice pre priemerovanie. Funkcia vezme priemer všetkých pixelov v matici 5x5 a nastaví novú hodnotu pixelu.

Pre detekciu hrán môžeme použiť funkciu Canny:

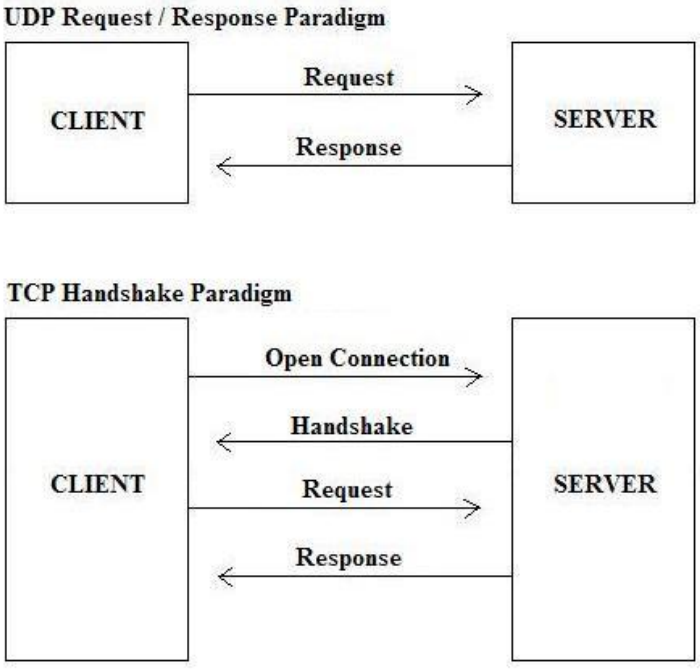
```
Hrany = cv2.Canny(obrazok,100,200)
```

Pre detektor nastavíme prvú a druhú prahovú hodnotu pre riadenie hysterézie a výsledkom budú vykreslené hrany v krivkách.

4 TCP a UDP

Používateľský datagramový protokol UDP je takzvaný nespoľahlivý protokol pre prenos datagramov medzi počítačmi v sieti. UDP je protokolom transportnej vrstvy. Na rozdiel od TCP protokolu nenadväzuje spojenie, z čoho vyplýva, že nie je potvrdené doručenie datagramov, neexistuje kontrolný súčet, dáta nemusia byť doručené v správnom poradí. Preto sa UDP protokol nazýva aj ako nespoľahlivý. Hlavné výhody UDP protokolu sú malá hlavička a malé zaťaženie siete. Nevýhodou je možná strata a duplicita dát a nieje možné zistiť, či dáta skutočne došli. Preto je UDP protokol využívaný pri real-time prenosoch multimediálnych dát, kde sa ich prenáša veľké množstvo a strata niektorého datagramu je vo väčšine nepovšimnutá.

TCP je tak ako aj UDP protokolom transportnej vrstvy. Pri použití TCP protokolu je nutné pre komunikáciu nadviazať spojenie. To znamená, že TCP je spojová služba. Spojenie vždy nadväzuje TCP klient. Ten sa pripojí k TCP serveru. Protokol je označovaný aj ako spoľahlivý. Pri posielaní dat sú zakaždým potvrdené. Ak by data potvrdené neboli, musia sa znovu odoslať. Nestane sa nám teda, že by dáta došli neúplné a poprehadzované. Vždy dôjdu v správnom poradí. To má ale svoje nevýhody. Protokol TCP zaťažuje sieť kvôli väčšom počte informácii v hlavičke, navyše aj kvôli potvrdzovaniu dát. Komunikácia je v oboch smeroch. TCP protokol je vhodné používať na malé objemy dát, napríklad odosielanie príkazov.Vieme zistiť, či príkazy došli a máme istotu, že príkazy dojdu v správnom poradí.



Obr. 12. UDP vs TCP [16]

5 Senzor pre detekciu prekážky

Optické senzory majú vo všeobecnosti veľmi široký záber použitia. Či už sa jedná o meranie vzdialenosti, teploty alebo detekciu prekážky. Sú obľúbené hlavne kvôli ich jednoduchosti, malým rozmerom ako aj hmotnosti. U človeka je viditeľné spektrum elektromagnetického žiarenia, čiže svetla, v rozmedzí od 400 nm do 700 nm. Pre bežne dostupné optické senzory sa využíva infračervené spektrum. Jeho vlnová dĺžka je v rozmedzí 760 nm až 1 mm. Presnejšie je to vlnová dĺžka okolo 940 nm. V niektorých prípadoch, ak je potrebné zamerať objekt sa používa červené svetlo, ktoré má najväčšiu vlnovú dĺžku z viditeľného spektra a to 625 nm až 740 nm. Zaujímavými optickými senzormi sú senzory z rodiny GP2Y0A spoločnosti SHARP. Senzor GP2Y0A20YK0F je na obr. 13.

Obr. 13. Optický senzor GP2Y0A20YK0F [17]

Základné technické špecifikácie:

- Napájacie napätie: 4,5 V – 5,5 V
- Meraná vzdialenosť: 20 cm – 150 cm
- Prevádzková teplota: -10 °C – +60 °C
- Výstupné napätie: 0,5 V – 2,7 V (analógový výstup)

Alternatívou k optickým senzorom sú senzory ultrazvukové. Ultrazvuk je mechanické kmitanie s frekvenciou vyššou ako 20 kHz. Ultrazvukový senzor obsahuje vysielač, ktorý vytvára ultrazvuk s danou frekvenciou. Ultrazvuková vlna sa následne šíri prostredím a ak narazí na prekážku, časť energie vlny prechádza prekážkou, časť energie vlny pohltí prekážka a zvyšok sa od prekážky odrazí. Odrazenú vlnu, nazývanú tiež echo, následne zdeteguje prijímač ultrazvukového senzora. Vzdialenosť prekážky určí vyhodnocovací obvod podľa rozdielu času medzi vyslaním a prijatím ultrazvukovej

vlny. Navyše, ak sa prekážka pohybuje vzhľadom k senzoru, môže byť uplatnený Dopplerov jav. Ten sa prejaví posunom frekvencie odrazenej vlny. Jedným z takýchto senzorov je SRF05 od spoločnosti Devantech. Senzor je na obr. 14. [18]

Obr. 14. Ultrazvukový senzor SRF05 [18]

6 Inštalácia Raspberry PI

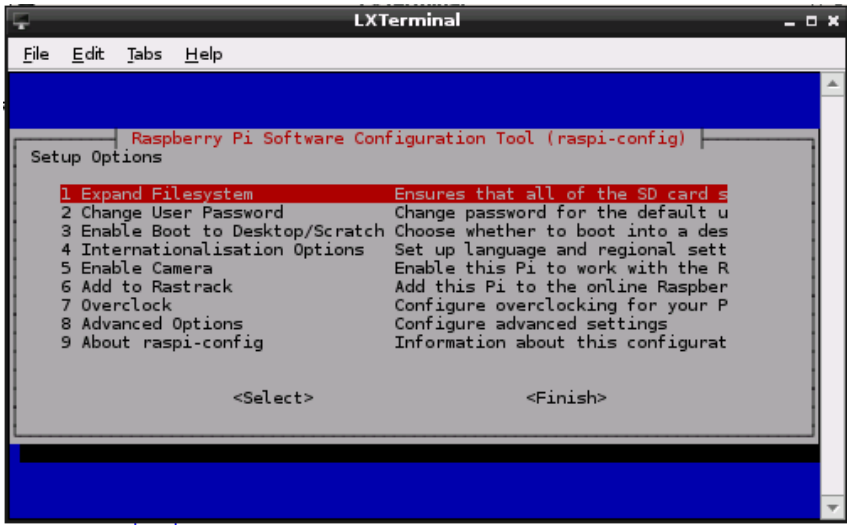
6.1 Operačný systém

Na Raspberry PI je k dispozícii niekoľko distribúcii operačného systému. Najznámejšou optimalizovanou verziou je Raspbian, distribúcia založená na Debiane označovaná tiež Wheezy. Ďalšími sú Pidora, Openelec, Raspbmc a RISC OS, ktoré sú stiahnuteľné zo stránok projektu, a iné neoficiálne verzie ako XBian, Slackware ARM a ďalšie.

Pre zjednodušenie inštalácie OS je zo stránok projektu možné stiahnuť NOOBS (new out of the box software), čo je inštalačný manažér, prostredníctvom ktorého je možné jednoducho zvoliť a nainštalovať jednu z oficiálnych verzií OS.

Inštalácia OS na Raspberry PI je odlišná od toho, na čo sme bežne zvyknutý. V prvom rade je potrebné stiahnuť OS, v našom prípade Raspbian zo stránok *www.raspberrypi.org* vo forme archívu. Stiahnutý archív rozbalíme a obraz disku musíme vložiť na SD (microSD) kartu (minimálne 2GB). Výber SD karty je dôležitým krokom, keďže rýchlosť karty bude určovať aj rýchlosť minipočítača a samotné čo najpohodlnejšie pracovanie s minipočítačom. Odporúčané sú karty Class 10.

Po rozbalení dostaneme obraz OS, ktorý musíme dostať na kartu SD. Vo Windows k tomu poslúži program Win32DiskImager. Po stiahnutí a rozbalení programu ho spustíme, vyberieme obraz OS, SD kartu a zapíšeme obraz na kartu SD. Po nakopírovaní vložíme SD kartu do Raspberry PI. Pripojíme klávesnicu a myš, ideálne cez USB HUB kvôli dostatočnému napájaniu. USB porty Raspberry PI majú totižto obmedzené možnosti napájania. Ďalej pripojíme monitor cez HDMI a nakoniec napájanie, čím automaticky spustíme Raspberry PI. Po nabootovaní nabehne úvodná obrazovka pre základnú konfiguráciu systému. Túto obrazovku je možné spustiť kedykoľvek počas používania Raspberry zadaním príkazu *raspi-config* do terminálu.



Obr. 15. raspi-config

Konfiguračný tool umožňuje následné nastavenia:

1. rozšírenie file systému o nevyužitý priestor na karte SD
2. zmena hesla pre defaultný účet (defaultné meno pi a heslo raspberry)
3. nastaví bootovanie do grafického prostredia
4. zmena miestnych nastavení (Jazyk, časové zóny, klávesnica)
5. povolí kameru pripojenú do CSI portu
6. Pridanie Raspberry PI na mapu http://rastrack.co.uk/
7. umožňuje pretaktovanie procesoru a RAM
8. Všeobecné nastavenia (Hostname, SSH, SPI, I2C, Audio a pod.)
9. Informácie o konfigurátore

Po nastavení príslušných parametrov a zvolení položky Finish sa minipočítač reštartuje a nabootuje už s novými nastaveniami, do grafického prostredia. Hneď po nabootovaní spustíme terminál a príkazmi *sudo apt-get update* skontrolujeme aktualizácie a *sudo apt-get upgrade* aktualizujeme všetky nainštalované balíčky. Tieto dva príkazy budeme využívať po každej inštalácii aplikácie či doplnku. Po zaktualizovaní systému minipočítač reštartujeme príkazom *sudo reboot*.

6.2 Inštalácia kamery

Po nainštalovaní Raspbianu môžeme pristúpiť k inštalácii kamery. Pripojenie kamery je prostredníctvom CSI portu na doske minipočítača. Po pripojení môžeme v *raspi-config* povoliť kameru. Do terminálu zadáme *sudo raspi-config* vyberieme možnosť č. 5, enable camera, zvolíme enable a dáme finish. Po nabootovaní je kamera pripravená na používanie. Jej funkčnosť si overíme pomocou dvoch vstavaných programov vyvinutých priamo pre túto kameru. Sú to raspivid, ktorý slúži pre snímanie videa

a *raspistill,* ktorý slúži na poriadenie fotky. Príkazom *raspivid –f –o video.h264 –t 20000* zosnímame video s defaultnými parametrami o dĺžke 20000 ms, ktoré sa nám zobrazí na celú obrazovku monitora.

Pre možnosti vývoja potrebujeme ku kamere pristupovať. Na to potrebujeme spustiť ovládač, ktorý nám kameru sprístupní ako video zariadenie /dev/video0. Pre tento účel bol vydaný oficiálny ovládač bcm2835-v4l2. Ovládač je v systéme integrovaný, no pred jeho prvým použitím je vhodné spustiť príkazy *sudo apt-get update* a *sudo apt-get upgrade* spolu s aktualizáciou firmvéru na poslednú verziu príkazom *sudo rpi-update*. Ovládač spustíme príkazom *sudo modprobe bcm2835-v4l2*. Teraz máme k dispozícii v /dev/ kameru ako zariadenie video0. Automatickú inicializáciu ovládača pri každom reštarte či spustení systému docielime doplnením súboru /etc/modules o riadok s textom *bcm2835-v4l2*.

6.3 IDE - PYTHON

Pre systém raspbian je k dispozícii hneď niekoľko vývojových prostredí pre rôzne platformy. Aplikácia pre tento projekt bude programovaná v programovacom jazyku Python. Pre tento jazyk sú v operačnom systéme integrované vývojové prostredia Idle a Idle 3. Idle je určené pre Python 2.7 a Idle 3 pre Python 3.

6.4 OpenCV

Pred samotnou inštaláciou OpenCv potrebujeme doinštalovať niekoľko balíčkov, aby sme sa počas programovania nezasekli kvôli chýbajúcim komponentám a prepojeniam. V prvom rade, tak ako pred každou inštaláciou na Raspberry Pi updatujeme už nainštalované balíčky a firmvér minipočítača príkazmi:

```
sudo apt-get update
sudo apt-get upgrade
sudo rpi-update
```

Tým máme pripravený operačný systém na inštaláciu. V prvom kroku nainštalujeme chýbajúce balíčky pre vývojárov príkazom:

```
sudo apt-get install build-essential cmake pkg-config
```

Inštaláciou GTK vývojovej knižnice budeme môcť neskôr používať Grafické uživateľské rozhranie, ktoré umožňuje knižnici highgui zobrazovať výstup na obrazovke:

```
sudo apt-get install libgtk2.0-dev
```

Knižnice pre optimalizáciu operácii pre OpenCv nainštalujeme príkazom:

```
sudo apt-get install libatlas-base-dev gfortran
```

Doinštalujeme knižnicu Numpy potrebnú pre prácu s multi-dimenzionálnymi poliami:

```
sudo apt-get install numpy
```

A môžeme sa vrhnúť na inštaláciu OpenCV. Stiahneme a rozbalíme poslednú verziu OpenCv (1.5.2015):

```
wget -0 opencv-2.4.10.zip
Unzip opencv-2.4.10.zip
Cd opencv-2.4.10
```

Spustíme build:

```
Mkdir build
Cd build
Cmake -D CKAME-BUILD_TYPE=RELEASE -D
CMAKE_INSTALL_PREFIX=/usr/local -D BUILD_NEW_PYTHON_SUPPORT=ON -
D INSTALL_PYTHON_EXAMPLES=ON -D BUILD%EXAMPLES=ON..
```

Skompilujeme:

```
Make
```

Kompilácia môže trvať v závislosti na modele minipočítača 3 – 10 hodín. Nakoniec spustíme finálnu inštaláciu:

```
sudo make install
sudo ldconfig
```

6.5 PiCamera

Modul Picamera je sice vyvíjaný špecáilne pre Raspberry Pi, no nie je priamo implementovaný v operačnom systéme a je ho potrebné doinštalovať.

```
sudo apt-get install python-pip
sudo pip install picamera
```

Pre pohodlnú prácu s modulom Picamera spolu s OpenCV doinštalujeme balíčky

```
sudo apt-get install python-matplotlib
sudo apt-get install python-mpltoolkits.basemap
sudo apt-get install python-numpy
sudo apt-get install python-scipy
sudo apt-get install libjpeg-dev
sudo apt-get install libfreetype6-dev
```

6.6 Inštalácia WiFi modulu edimax

Raspberry Pi 2 bohužiaľ eśte nemá vstavaný WiFi modul tak ako generácia 3. Preto je potrebné konektivitu zaistiť prostredníctvom USB WiFI modulu. Ten je potrebné nastaviť ako router, aby bolo možné sa s minipočítačom bez problémov spojiť [19].

Ako prvý krok musíme nainštalovať aplikáciu hostapt udhcpd príkazom:

```
sudo apt-get install hostapd udhcpd
```

po nainštalovaní je potrebné prepísať súbo v *zložke /etc/udhcpd.conf* na:

```
start 192.168.1.2
end 192.168.1.20
interface wlan0
remainig yes
opt dns 8.8.8.8. 4.2.2.2
opt subnet 255.255.255.0
opt router 192.168.42.1
opt lease 864000
```

Tým si nakonfigurujeme DHCP server. Ďalej musíme zmeniť v súbore */etc/default/udhcpd* z:

```
DHCPD_ENABLED = „no“
```

na

```
#DHCPD_ENABLED – „no“
```

Tým prestavíme defaultné nastavenie DHCP na užívateľské. Ďalším krokom je nastavenie statickej IP adresy:

```
sudo ifconfig wlan0 192.168.1.1
```

Aby sa IP adresa načítala automaticky pri nabootovaní minipočítača, zameníme v súbore */etc/network/interfaces* riadok *„iface wlan0 inet dhcp“* na:

```
iface wlan0 inet static
     address 192.168.1.1
     netmask 255.255.255.0
```

a v súbore na samotnom konci zmeníme:

```
allow-hotplug wlan0
wpa-roam /etc/wpa_suppliant/wpa_suppliant.conf
iface default inet manual
```

na:

```
#allow-hotplug wlan0
#wpa-roam /etc/wpa_suppliant/wpa_suppliant.conf
#iface default inet manual
```

Teraz vytvoríme novú sieť a to tak, ze súboru */etc/hostapd/hostapd.conf* vložíme:

```
interface = wlan0
ssid = moja_sieť
hw_mode = g
channel = 6
auth_algs = 1
wmm_enabled = 0
```

V súbore */etc/default/hostapd* zmeníme:

```
#DAEMON_CONF= „“
```

na:

```
DAEMON_CONF = „/etc/hostapd/hostapd.conf“
```

čím nastavíme spustenie našej wlan.

Teraz nastavíme NAT:

```
sudo sh –c „echo 1 > /proc/sys/net/ipv4/ip_forward“
```

nastavíme automatické nastavenie NAT pri bootovaní v súbore */etc/sysctl.conf* na:

```
net.ipv4.ip_forward = 1
```

a povolíme NAT v kernely sustému:

```
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --state
RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
```

Uložíme aktuálne nastavenia:

```
sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"
```

doplníme do súboru */etc/network/interfaces následujúce*:

```
up iptables-restore < /etc/iptables.ipv4.nat
```

spustíme access point:

```
sudo service hostapd start
sudo service udhcpd start
```

a spustíme našú wlan:

```
sudo update-rc.d hostapd enable
sudo update-rc.d udhcpd enable
```

Teraz po každom reštarte, ak bude pripojený WiFi modul, minipočítač automaticky spustí wlan s názvom „moja_sieť“, na ktorú sa bude dať bez problémov pripojiť. Táto sieť sa navyše automatický premostí s ethernetom, čiže v prípade pripojeného minipočítača k sieti internet bude aj na dotičnom pripojenom zariadení internet.

6.7 Nastavenie UART

V základnom stave Raspberry Pi má nastavený UART pre využívanie vstavanej konzoly, kde môžeme posielať správy, prípadne takzvané logovacie správy, ktoré zjednodušujú prácu s ďalšimi zariadeniami. Ak je ale raspberry nastavené v tomto režime, nedokážeme uart použiť v programe. Nebudeme mať k nemu prístup. Kvôli tomuto je potrebné zmeniť základné nastavenia v systéme. V prvom rade je potrebné zo suboru:

```
/boot/cmdline.txt
```

zmazať:

```
‘console=ttyAMA0,115200’ and ‘kgdboc=ttyAMA0,115200’.
```

výsledny tvar bude:

```
dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p6
rootfstype=ext4 elevator=deadline rootwait
```

a v súbore:

```
/etc/inittab
zakomentovat riadok na konci suboru ‘2:23:respawn:/sbin/getty -L ttyAMA0
115200 vt100’
```

po tomto je potrebné Raspberry reštartovať aby sa nastavenia aplikovali. Existuje ale aj program, respektíve skrypt, ktorý dokáže tieto nastavenia automatiky aplikovať a dokonca ich jednoducho vrátiť naspäť. Program sa volá rpi-serial-console. Nainštalujeme príkazom:

```
sudo wget https://raw.githubusercontent.com/lurch/rpi-serial-
console/master/rpi-serial-console -O /usr/bin/rpi-serial-console && sudo
chmod +x /usr/bin/rpi-serial-console
```

aktuálny stav zistíme príkazom:

```
rpi-serial-console status
```

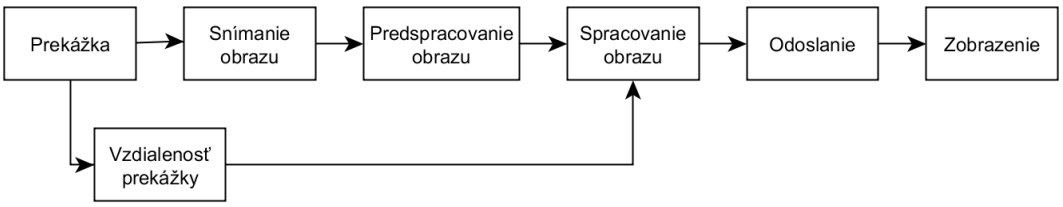
rip konzolu disablujeme príkazom *sudo rpi-serial-console* disable a povolíme *sudo rpi-serial-console enable*. Po každom zmeneí nastavenia je potrebné raspberry reštartovať, aby sa nastavenia aplikovali [20].

7. Navrhnuté riešenie

7.1 Detekcia objektu

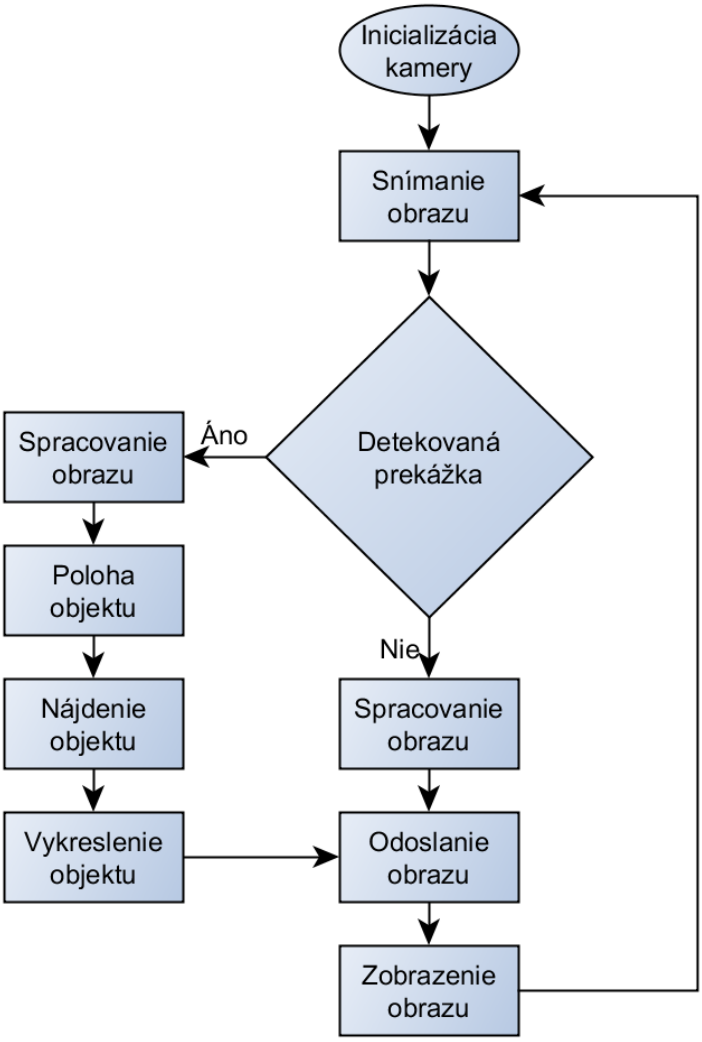
Práca sa zaoberá detekciou prekážky v nasnímanom obraze. To, ako určiť, že je v obraze vôbec nejaká prekážka je obtiažná úloha. Systémov, ako na to, sa dá vymyslieť niekoľko. V podstate sme obmedzený iba samotným hardvérom, ktorý máme k dispozícii. Prekážka môže byť čokoľvek, preto nedokážeme pracovať s rozšírenými osvedčenými metódami. Tie by pracovali, ak by sme to popísali jednoducho, porovnávaním objektov s preddefinovanou databázou objektov, prípadne by aplikácia bežala v režime učenia, kde by na základe preddefinovaných parametrov dokázala určiť objekt, uložiť ho do databáze a týmto spôsobom rozširovala svoje vedomosti o okolí. Toto je zdĺhavý a náročný proces. Jednoduchšou metódou by bolo pracovať s rozdielovými snímkami, kde by sme porovnávali objekty v dvoch, prípadne niekoľkých snímkoch za sebou. Takto dokážeme zistiť objekty ktoré sa pohybujú, výpočtami dokážeme zistiť zväčšovanie a zmenšovanie objektov. Stále ale u neznámych objektov nedokážeme zistiť veľkosť, vzdialenosť. Navyše by bolo toto riešenie vhodné, ak by bola kamera stacionárna. My ale budeme mať kameru na autíčku, ktoré sa bude hýbať a navyše ani objekt nemusí byť stacionárny. K tomuto by bolo vhodné použiť dve kamery. Takto by sme dokázali získať trojrozmerný pohľad na svet. Keďže by boli kamery od seba v definovanej vzdialenosti, porovnávaním snímkov z oboch kamier by bolo možné určiť veľkosť a tak aj vzdialenosť objektov. Minipočítač Raspberry Pi ale umožňuje zapojenie len jednej kamery a táto metóda nieje dostupná. Vhodnou variantou pre túto prácu sa tak stala varianta, kde pre určovanie vzdialenosti objektu, budeme používať optický senzor. Dostaneme tak informáciu o vzdialenosti v rámci možností senzoru.

Snímanie obrazu, detekcia prekážky a vyhodnotenie v obraze má fungovať na základe blokovej schémy na obr. 16.



Obr. 16. Bloková schéma detekcie prekážky

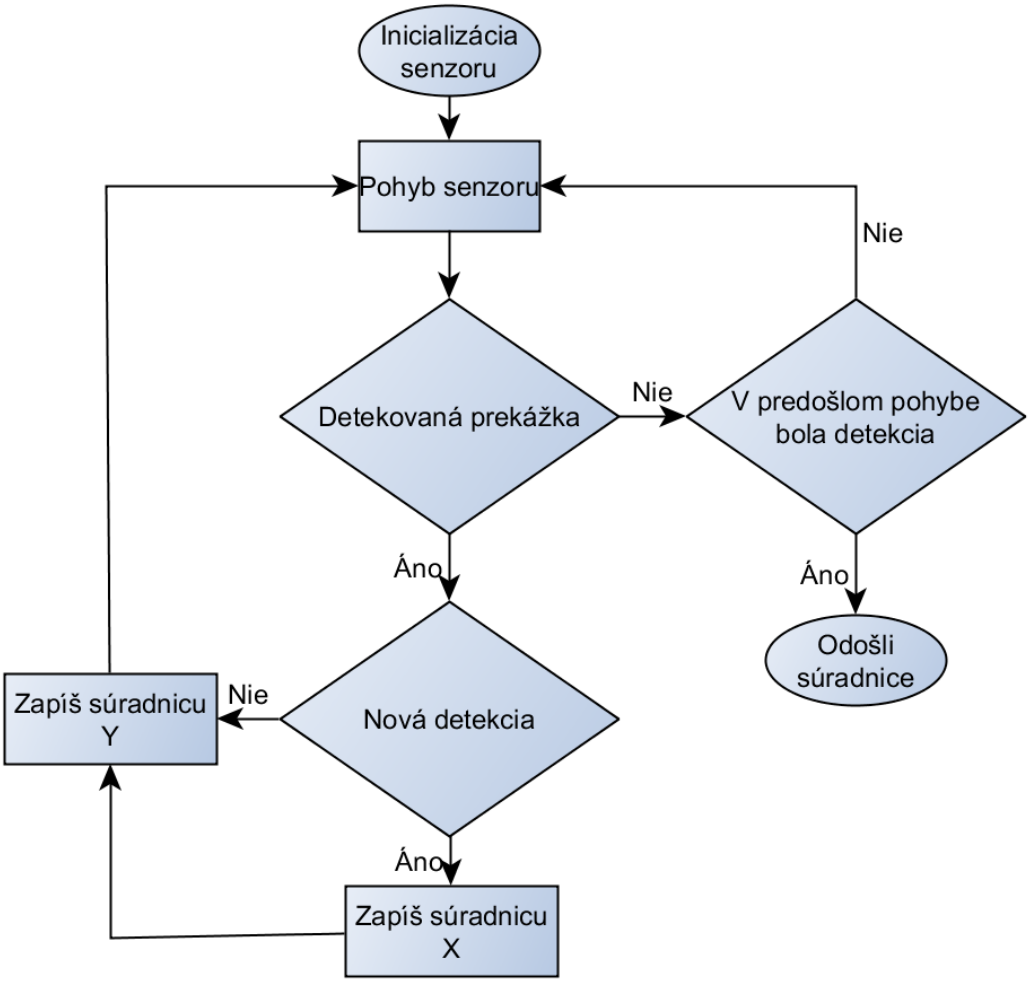
Obraz je snímaný kamerou. Je spracovaný a odoslaný na zobrazenie. V prípade, ak z výstupu z optického senzoru vyčítame, že je nejaký objekt v pozorovacom uhle kamery, táto informácia a informácia o polohe objektu je použitá pri vyhodnocovaní obrazu. Pre presnejší popis celého riešenia je na obr. 17 vývojový diagram.



Obr. 17. Vyvojový diagram detekcie prekážky

Po incializácii kamery nasleduje snímanie obrazu. Pred spracovaním snímaného obrazu zistíme, či optický senzor detekoval objekt. Ak v ceste žiadna prekážka nie je, obraz je upravený a odoslaný na zobrazenie. V prípade, že je v ceste prekážka, obraz sa spracuje a na základe súradníc od optického senzoru nájde objekt v obraze, ktorý vykreslí a odošle na zobrazenie.

Optický senzor má určovať vzdialenosť, ale potrebujeme aj súradnice objektu. Tak vymedzíme priestor, kde má skript v obraze hľadať a eliminovať tak chybné detekcie. Informácie o vzdialenosti a polohe objektu budú predané skriptu, ktorý má za úlohu spracovávať obraz z kamery. Ten podľa súradnic vyberie časť obrazu a tam sa prevedie vyhodnotenie, ktorého úlohou bude vykresliť potencionálnu prekážku. Aby sme ale súradnice vôbec mohli nájsť, musíme zariadiť pohyb optického senzoru v horizontálnom pozorovacom uhle kamery. Princíp fungovania optického senzoru je znázornený vo vývojovom diagrame na obr. 18.



Obr. 18. Vývojový diagram fungovania optického senzoru

8. Praktická realizácia

Pre realizáciu diplomovej práce bol vybraný minipočítač Raspberry Pi 2 Model B. Tento typ bol vybraný vďaka svojmu 4-jadrovému procesoru ako aj 1 GB operačnej pamäti RAM. Snímanie obrazu a jeho následné spracovanie vyžaduje dostatok prostriedkov a prvá generácia minipočítača troška zaostávala. Tretia generácia prišla počas písania práce a po skúsenosti prechodu z generácie 1 na generáciu 2, aj napriek uisteniu autorov minipočítača, že je hardvér kompatibilný, nebol dostatok času experimentovať s najnovšou verziou.

8.1 Porovnanie generácie 1 a 2

Pre porovnanie rýchlosti generácie 1 a generácie 2 bol použitý skript [picamera], prostredníctvom ktorého dokážeme celkom rýchlo a jednoducho demonštrovať rýchlosť snímania minipočítača:

```
Import time
Import picamera

pocet_obrazkov = 10
sirka = 1024
vyska = 768
format_obrazka =' jpeg'
with picamera.PiCamera() as camera:
      camera.resolution = (sirka, vyska)
      camera.framerate = 90
      time.sleep(2)
      start = time.time()
camera.capture_sequence(['image%02d.jpg' % i for i in
range(pocet_obrazkov)], format_obrazka, use_video_port = True)
koniec = time.time()
print('vyfotili sme %d obrazkov s %.2f framov za sekundu' %
(pocet_obrazkov, pocet_obrazkov/(koniec - start)))
```

Skript využíva pre snímanie funkciu *capture_sequence*, ktorá je pre tento účel ideálna. Je to z dôvodu, že enkóder kamery nie je incializovaný pred každým novým snímkom. Týmto spôsobom nie sme obmedzovaný rýchlosťou kamery a rýchlosť snímania obmedzí až minipočítač. Nevýhodou tohoto spôsobu snímania je, že obrázky sú zašumené. To ale v tomto prípade nevadí. V skripte si na začiatku pevne nastavíme počet nasnímaných obrázkov, ktorý nám bude slúžiť pre výpočet, ďalej rozlíšenie a formát, v ktorom budeme snímať. Skript následne zosníma daný počet obrázkov, ktoré uloží na SD kartu a v závere vypočíta, akou rýchlosťou boli snímky poriadené. Výsledky testu sú v tabuľkách 3, 4 a 5.

Tab. 3. Porovnanie generácie 1 a 2 – 10 obrázkov

10 obrázkov	Raspberry Pi 1 model B+		Raspberry Pi 2 model B	
Formát	jpeg	rgb	jpeg	rgb
rozlisenie	1920x1080			
Obrázky za sekundu	12	1	17,5	2,33
rozlisenie	1024x768			
Obrazky za sekundu	27,5	2,98	39.35	6.38
rozlisenie	320x240			
Obrazky za sekundu	45,65	23.32	54.1	37.27

Tab. 4. Porovnanie generácie 1 a 2 – 50 obrázkov

50 obrázkov	Raspberry Pi 1 model B+		Raspberry Pi 2 model B	
Formát	jpeg	rgb	jpeg	rgb
rozlisenie	1920x1080			
Obrázky za sekundu	4,67	0,82	22,21	1,13
rozlisenie	1024x768			
Obrazky za sekundu	24,3	2,07	50.86	6.85
rozlisenie	320x240			
Obrazky za sekundu	76.83	17.85	77.83	57.37

Tab. 5. Porovnanie generácie 1 a 2 – 200 obrázkov

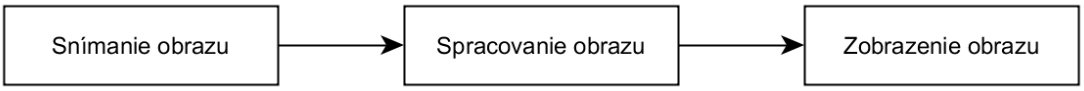
200 obrázkov	Raspberry Pi 1 model B+		Raspberry Pi 2 model B	
Formát	jpeg	rgb	jpeg	rgb
rozlisenie	1920x1080			
Obrázky za sekundu	4,96	0,89	15,22	1,19
rozlisenie	1024x768			
Obrazky za sekundu	15,91	2,06	36,89	3,4
rozlisenie	320x240			
Obrazky za sekundu	63.38	12.74	87.24	57.96

Z tabuliek je vidieť, že v rámci očakávaní je Raspberry Pi 2 model B výkonnejší. Najväčší rozdiel je v snímaní obrázkov vo formáte BGR. JPEG je oproti BGR formátu komprimovaným. To znamená, že výsledná veľkosť obrázka bude menšia a tým musíme zapísať aj menej informácii na SD kartu. Pri rozlíšení 1920 x 1080 pixelov je

veľkosť obrázka snímaného vo formáte BGR 6 MB pričom pri snímaní toho istého obrázku vo formáte JPEG je 640 kB.

8.2 Snímanie obrazu

Snímanie je realizované prostredníctvom kamerovného modulu Pi Noir Camera Board pripojeného k CSI rozhraní minipočítača. Ku kamere pristupujeme prostredníctvom funkcií z modulu Picamera [15]. Tá umožňuje ako snímanie jednotlivých obrázkov, tak aj snímanie videa. Pre účely tejto diplomovej práce bolo zvolené snímanie jednotlivých obrázkov, pretože môžeme k obrázku pristúpiť hneď po nasnímaní a nemusíme ho vyťahovať z prúdu videa. To by bolo nakoniec aj náročnejšie na hardvér.



Obr. 19. Bloková schéma snímania obrazu

Pre čo najrýchlejšie snímanie má modul Picamera k dispozícii dve metódy. Jednou je *capture_sequence*, ktorá je v práci využitá na meranie rýchlosti snímania u generácie 1 a 2. Táto metóda je rýchla, ale kvalita obrázkov je omnoho nižšia. Tam, kde je potrebná rýchlosť ale nie sú vyžadované detaily, je to dobrá voľba. Na zobrazovanie je to už troška horšie. Práca s touto metódou je obtiažnejšia, pretože bola určená predovšetkým na snímanie dopredu daného počtu obrázkov. Aby sme boli schopný kontinuálneho snímania, bolo potrebné použitie metódu troška upraviť:

```
def kamera():
      with picamera.PiCamera() as camera:
            camera.resolution = (640, 480)
            camera.framerate = 30
            camera.start_preview()
            camera.capture_sequence (buffer(), use_video_port =
True)
def buffer():
      obrazok = io.BytesIO()
            buffer_obr = []
      while (True):
            yield obrazok
            buffer_obr.append(obrazok)
            obrazok.seek(0)
```

Takto použitá metóda dokáže zosnímať do 30 fps pri rozlíšení 640 x 480 pixelov.

Druhá metóda je *capture_continuous*. Metóda je vo svojej podstate určená na kontinuálne snímanie. Je ale už v základe pomalšia ako predošlá. Výstup je ale kvalitnejší. Metóda sa používa nasledovne:

```
def kamera():
      camera = PiCamera()
      camera.resolution = (640, 480)
      camera.framerate = 30
      rawCapture = PiRGBArray (camera, size = (640, 480))
      for frame in camera.capture_continouos(rawCapture, 'rgb',
use_video_port = True)
            obrazok = frame.array
```

takto použitá metóda dokáže zosnímať do 20 fps pri rozlíšení 640 x 480 pixelov. Je to dané inicializáciou dekodéru pred každým zosnímaním obrázku. Výsledkom ale je obrázok vo vyššej kvalite ako u predošlej metódy.

Obr. 20. Porovnanie kvality. Vľavo camera_sequence, vpravo camera_continuous

U metódy snímania *capture_sequence* je potrebné obrázky prekonvertovať do formátu pre OpenCV, aby bolo možné ich ďalej spracovávať, pretože snímaný obraz je uložený ako *io.BytesIO*. Obraz tu tak nemá daný formát a preto je potrebné mu ho zadať takto:

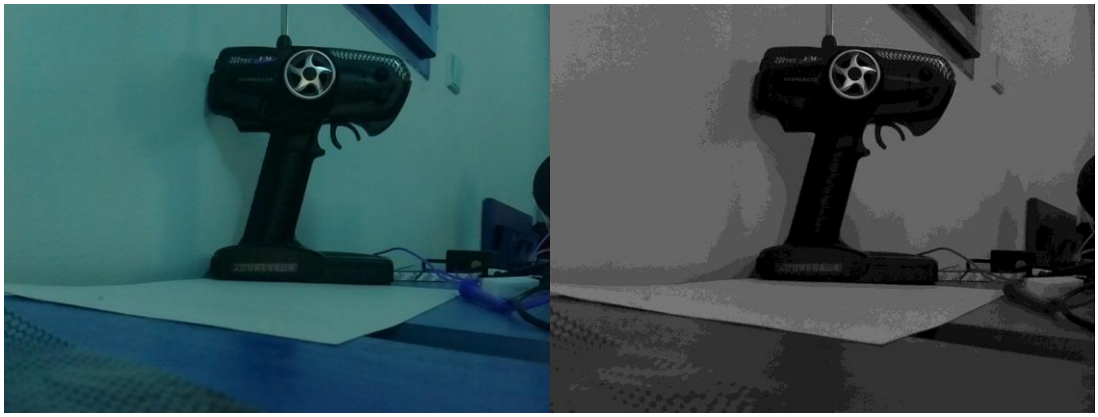
```
def prevod():
      image_prevod = image
      data = np.fromstring(image_prevod.getvalue(), dtype =
np.uint8)
      im_decode = cv2.imdecode(data, 1)
      im_decode = im_decode[:,:,::-1]
      image_prevod = Image.fromarray(im_pom, 'RGB')
```

Metóda *capture_continuous* vracia obraz priamo vo formáte RGB. Preto ho nie je potrebné prevádzať a OpenCV s ním dokáže ďalej pracovať.

Keď dostaneme od senzora správu, že v ceste máme prekážku, určíme súradnice objektu, ktoré budu odoslané cez UART do minipočítača. Vtedy sa aktivuje skript

vyhodnocovania obrazu. Výstup z kamery nieje vhodný pre spracovávanie. Obsahuje veľa informácii ktoré nepotrebujeme. Musíme ho previesť do odtieňov sivej:

```
Gray = cv2.cvtColor (obrázok , cv2.COLOR_RGB2GRAY)
```



Obr. 21. Vľavo originál obrázku, vpravo v otrieňoch sivej

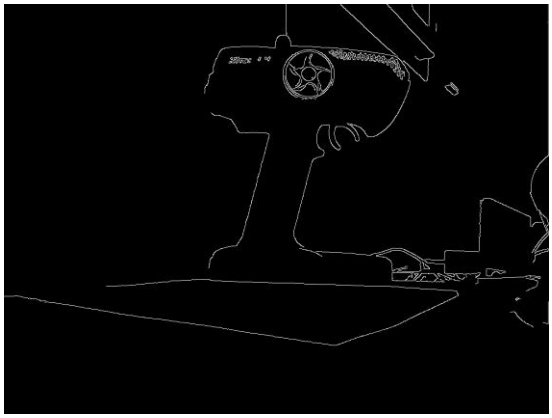
Sivotónový obrázok musíme následne upraviť tak, aby sme predišli falošným alebo čiastočným detekciám. To prevedieme rozmazaním obrázku. Tým eliminujeme všetky hrany na minimum:

```
erosion = cv2.erode(gray, kernel, iterations = 3)
blur = cv2.GaussianBlur (erosion, (5, 5), 0)
```

Obr. 22. Vľavo obrázok po erózii, vpravo rozmazaný obrázok

V takto upravenom obraze môžeme hľadať hrany. Na to použijeme Canny hranový detektor:

```
hrany = cv2.Canny(blur, 5, 255)
```



Obr. 23. Hrany nájdené v obraze pomocou Canny hranového detektoru

Máme vykreslený celý objekt. To nám ale nestačí, musíme nájsť celistvé objekty funkciou na hľadanie kontúr:

```
cnts, hierarchy = cv2.findContours(hrany.copy(), cv2.RETR_LIST,
cv2.CHAIN_APPROX_SIMPLE)
```

Keďže sú hrany detekované v celom objekte, funkcia pre nájdenie kontúr ich nájde v celom obraze. Aby sme našli práve našu prekážku, použijeme dáta z optického senzoru. Tým, že optický senzor kmitá v pozorovacom uhle kamery, dokážeme vyčítať súradnice objektu. Tieto súradnice nám poslúžia na nájdenie kontúry zodpovedajúcej prekážke:

```
for a in cnts:
      contour = cv2.minAreaRect(a)
      box = np.int0(cv2.cv.BoxPoints(contour))
      pts = np.array([box], np.int32)
      pts = pts.reshape((-1, 1, 2))
      x,y,w,h = cv2.boundingRect(a)
      if (x>suradnica_x1 and (x+w)<suradnica_x2):
            cv2.rectangle(image, (x,y), (x+w, y+h), (0,255,0),2)
```

Skript v cykle for prechádza všetky kontúry nájdene v obraze. Z každej kontúry vytiahne súradnice *x, y, w, h*, zodpovedajúcim štvorcu. V závere každého cyklu potom porovná súradnice x-ovej osy so súradnicami, ktoré nám vrátil optický senzor. Všetky relevantné výsledky sú zakreslené v obraze, ktorý je odoslaný do tabletu:

Obr. 24. Nájdené kontúry zodpovedajúce detekcii optického senzoru

8.3 Optický senzor

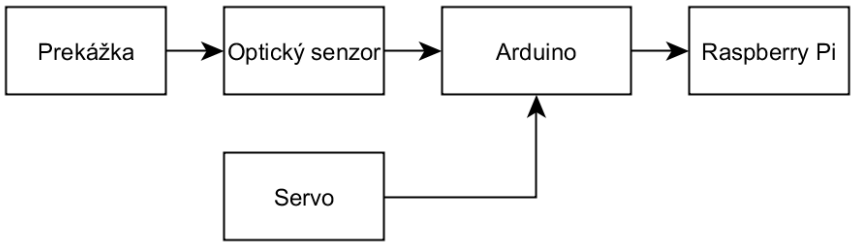
Pre určenie vzdialenosti je použitý senzor GP2Y0A02YK0F. Výstup optického senzoru je analógová hodnota, napätie, v rozmedzí od 0,5 V pre 15 cm do 2,75 V pre 150 cm. Podľa datasheetu máme napätia pre jednotlivé vzdialenosti dané do tabuľky 6. Tieto hodnoty sú uvedené pre ideálne podmienky, kde je ako odrazivá plocha biely papier s reflexiou 90%. Podľa testov senzor dobre reaguje aj na bežné predmety ako čierná látka, lakované drevo, kožené kreslo.

Tab. 6. Vzdialenosť/napätie optického senzora

Napätie [V]	Vzdialenosť [m]
15	2,75
20	2,5
30	2
40	1,5
50	1,25
60	1
100	0,75
150	0,5

Raspberry Pi neobsahuje vstavaný analogový prevodník, jedine ako doplnkový modul. Preto použijeme platformu Arduino, čo je open-source platforma založená na mikrokontroléroch Atmega. Pre programovanie používa grafické prostredie založené na prostredí Wiring [21]. Platforma arduino obsahuje množstvo preddefinovaných modulov pre prácu s AD prevodníkom, UART, SPI, I2C a podobne. Budeme používať dosku Arduno Mega s mikrokontrolérom ATMega 2560, ktorý nám poslúži aj na ovládanie serv. To nie je implementované na minipočítači z dôvodu chýbajúcich hodín.

Raspberry Pi má modul presných hodín k dispozícii znova len ako modul, pripojteľný cez GPIO [22].



Obr. 25. Bloková schéma fungovania optického senzoru

Použitie AD prevodníku vďaka preddefinovaným modulom je jednoduché. Arduino Mega obsahuje šestnásť 10 b AD prevodníkov s označením A0 – A15. Prevodník je stavaný na konverziu napätia medzi 0 – 5 V. Výstupom sú hodnoty v rozmedzí 0-1023. Optický senzor zapojíme na vstup A0. V programe nemusíme AD prevodník nijako inicializovať. Ten je vždy ako analogový vstup, ak neurčíme ináč. Hodnotu napätia zo senzoru vyčítame príkazom:

```
Opticky_senzor = analogRead(0);
```

Vzdialenosť určíme výpočtom:

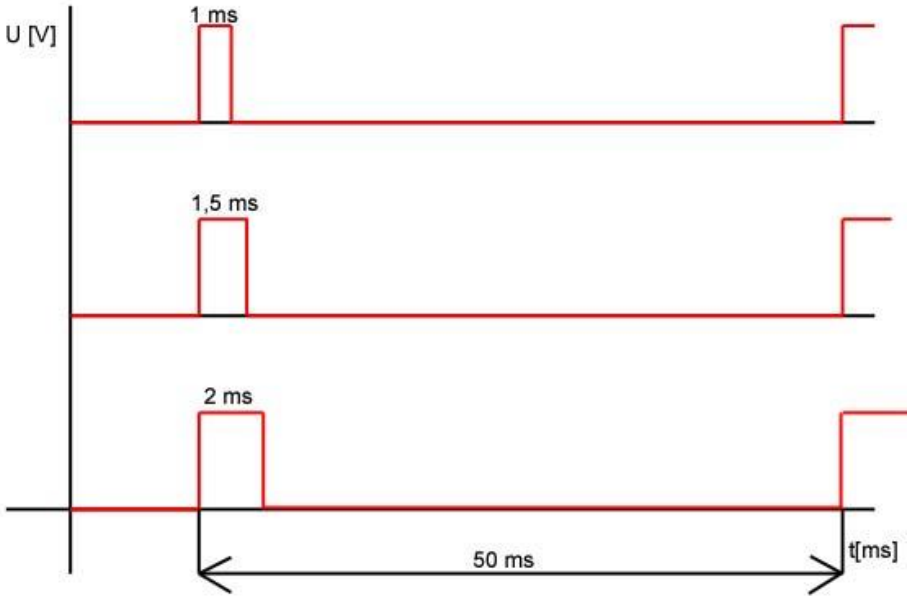
```
71061*pow(opticky_výstup, -1,31)
```

Vzdialenosť takto máme určenú. Teraz potrebujeme určovať súradnice prekážky. Kamera má pozorovací uhol 53,5°. Aby sme pokryli tento uhol, musíme so senzorom v tomto uhle pohybovať. Na to použijeme servo, ktoré budeme ovládať prostredníctvom Arduina. Pre ovládanie serva potrebujeme generovať pulzy o šírke 1 – 2 ms a perióde 20 ms. Arduino má na to vstavaný modul Servo. Týmto modulom dokážeme prostredníctvom Arduino Mega ovládať až 48 serv s využitím všetkých štyroch 16 b časovačov [29]. Nám stačí momentálne jedno, a neskôr ďalšie dva serva. Na to využijeme len jeden časovač. Inicializáciu serva prevedieme následovne:

```
Servo servo_opt_senzor;
Servo_opt_senzor.attach(8);
```

Teraz nám stačí už len nastavovať polohu serva príkazom:

```
Servo_opt_senzor.write(hodnota);
```

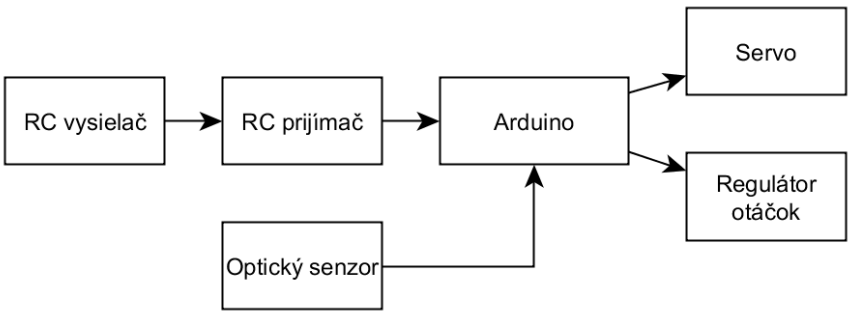


Obr. 26. Signál serva

Modul servo u Arduino dokáže nastaviť buď uhol serva v rozmedzí 0 – 180° alebo šírku pulzu od 1000 do 2000 µs. Pre nás budeme používať prvú variantu. Budeme nastavovať hodnoty od 63° do 117°, čo je 54° a to nám pokryje celý pozorovací uhol kamery. Hodnoty budeme posielať prostredníctvom UART-u do minipočítača.

8.4 Ovládanie

Ovládanie auta prebieha pomocou štandardnej RC vysielačky, aby sme nezaťažovali ovládaním tablet a raspberry. Keďže ovládame auto v reálnom čase, každý výpadok spojenia či zamrznutie programu by mohlo viesť ku kolízii. Pri dostatočnej optimalizácii môže byť ovládanie upravené pre tablet.

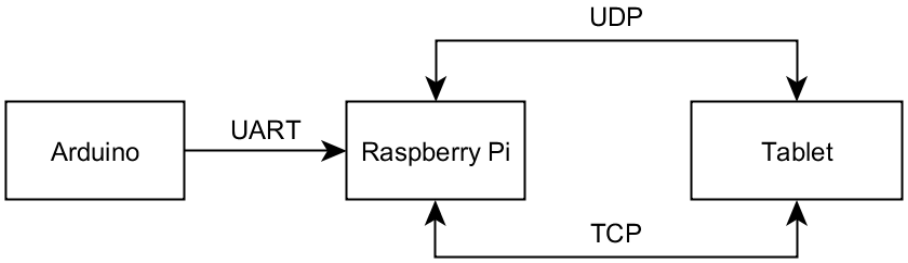


Obr. 27. Bloková schéma ovládania auta

Každopádne ale bude ovládanie auta smerované cez Arduino. V prípade detekcie prekážky a ignorácie od ovládajúceho bude tak možné programovo auto zastaviť. Pre tento účel musí byť signál z prijímača RC súpravy arduinom odchytávaný a preposielaný ďalej servu a regulátoru auta. Odchytávanie je realizované prostredníctvom externých prerušení. Tie majú v štruktúre prerušení najvyššiu prioritu a tak nehrozí, aby sme signál nezachytili. Po navzorkovaní vstupného signálu je hodnota preposlaná do modulu servo.

8.5 Komunikácia

Komunikácia je v práci najdôležitejšou časťou. Potrebujeme mať nadviazanú stabilnú komunikáciu medzi arduinom a Raspberry pi ako aj medzi Raspberry Pi a tabletom. Raspberry Pi 2 ešte nemá integrovaný WiFi modul a tak bolo potrebné ho osadiť USB WiFi modulom Edimax. Ten je nastavený ako WiFi router na ktorý sa tablet pripája. Spojenie je aktívne hneď po zapnutí minipočítača. Od aktualizácie Raspbianu, ktorá implementovala podporu viacerých procesorov pre druhú generáciu nefunguje automatická inicializácia DHCP po nabehnutí OS. Z tohoto dôvodu je potrebné mať na tablete nastavenú IP adresu na pevno.



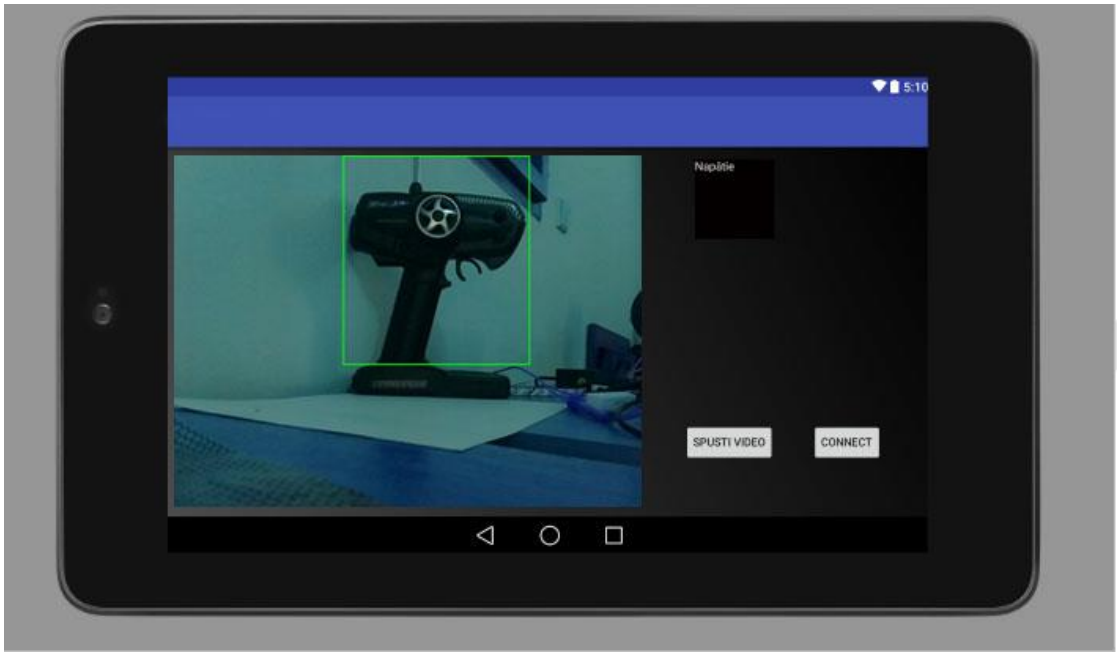
Obr. 28. Bloková schéma komunikácie

Komunikácia medzi minipočítačom a tabletom prebieha prostredníctvom protokolov TCP a UDP. TCP ako takzvaný „spoľahlivý“ protokol slúži na odosielanie príkazov na spustenie komunikácie a videa. Taktiež sú protokolom TCP spätne príjimané dáta o stave bateriek. Protokol UDP je implementovaný z dôvodu odosielania obrázkov.

Na komunikáciu medzi minipočítačom a arduinom slúži protokol UART. Ten slúži na odosielanie hodnôt z optického senzora ako aj napätí z bateriek, ktoré sú preposielané do tabletu.

8.6 Tablet

Tablet so systémom Android slúži ako zobrazovacia jednotka. Aplikácia je vytvorená v Android Studiu, kde je programovacím jazykom Java. Aplikácia obsahuje jednoduché rozhranie, ktoré obsahuje tlačítka na nadviazanie komunikácie, začatie prenosu obrazu, ukončenie prenosu a zobrazovanie videa.



Obr. 29. Aplikácia pre tablet

Záver

Úlohou diplomovej práce bolo naštudovať snímanie a vyhodnotenia obrazu, získaného z kamery u bezdrôtového riadeného modelu vozidla. Mal byť vytvorený program so schopnosťou detekcie prekážky s odosielaním na tablet a zobrazením snímaného videa. Pre účel spracovania práce bol vybraný minipočítač Raspberry Pi model 2 B. V teoretickej časti práce sú popísané staršie, ako aj novší model tohoto minipočítača.

Druhá kapitola sa venuje počítačovému videniu. Sú tam popísané základné princípy počítačového videnia od snímania obrazu, cez digitalizáciu, predspracovanie až po popis objektov. Popísané sú aj metódy spracovania obrazu ako apativne prahovaie, Sobelov hranový detektor a cannyho hranový detektor, doplnené grafickou ukážkou fungovania.

Tretia kapitola sa zaoberá modulom Pi Camera, ktorý je vyvinutý a priamo podporovaný pre minipočítač. Je vysvetlená základná práca s ním ako aj s OpenCV. V štvrtej kapitole je stručne vysvetlený rozdiel medzi TCP a UDP protokolom, a piata popisuje optický senzor vzdialenosti.

Šiesta kapitola sa venuje príprave a inštalácii operačného systému minipočítača a prechádza konfiguráciou systému pre správne fungovanie, ktorá je nevyhnutná pre prácu s ním.

Siedmej kapitola sa venuje popisu navrhnutého riešenia detekcie objektu. Sú tam popísané blokové schémy a vývojové diagramy, podrobne popisujúce fungovanie celého riešenia.

Posledná ôsma kapitola sa venuje samotnej realizácii navrhnutého riešenia. Je tam prakticky overená výkonnosť prvých dvoch generácii minipočítačov. Kapitola pokračuje popisom snímania obrazu a implementáciu optického senzoru, ktorý detekuje vzdialenosť a spolupracuje so skriptom pre hľadanie prekážky v obraze. Záver kapitoly popisuje samotné ovládanie auta spolu s komunikáciou a vyobrazuje ukážku aplikácie pre tablet.

Úlohou práce malo byť zrealizovať funkčný riešenie snímania obrazu, čo sa podarilo. Program je schopný po detekcii prekážky optickým senzorom prekážku nájsť a vykresliť a zosnímaný obraz prostredníctvom WiFi odoslať do tabletu, kde je zobrazovaný. Práca umožňuje ďalšie zdokonaľovanie v oblasti detekcie prekážok rozširovaním senzorov a vývojom dokonalejších riešení v oblasti spracovania obrazu.

Zoznam použitých zdrojov

[1] RASPBERRY PI FOUNDATION. RASPBERRY PI FOUNDATION. [online]. 2016 [cit. 2016-05-23]. Dostupné z: https://www.raspberrypi.org/

[2] Petervis: Raspberry Pi CSI-2 Connector Specifications. J. VIS, Peter. Petervis [online]. 2014 [cit. 2014-12-17]. Dostupné z: http://www.petervis.com/Raspberry_PI/Raspberry_Pi_CSI/Raspberry_Pi_CSI-2_Connector_Specifications.html

[3] UIS MENDELU vývojový tým . UIS MENDELU vývojový tým . [online]. [cit. 2016-05-23]. Dostupné z: https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=9217

[4] DIP – Digital Image Processing. DIP – Digital Image Processing. [online]. 06.2003 [cit. 2016-05-23]. Dostupné z: http://dip.sccg.sk/transf/transf.htm

[5] ŠIKUDOVÁ, E., Z. ČERNEKOVÁ, V. BENEŠOVÁ, Z. HALADOVÁ a J. KUČEROVÁ. Počítačové videnie: detekcia a rozpoznávanie objektov [online]. 1. vyd. Praha: Wikina, 2011, 378 s. [cit. 2014-12-17]. ISBN 978-80-87925-06-5. Dostupné z: http://sccg.sk/~cernekova/Pocitacove_videnie. pdf

[6] ing. Roman Pihan.. fotoroman. FotoRoman. [online]. 2012 [cit. 2016-05-23]. Dostupné z: http://www.fotoroman.cz/techniques3/svetlo05color_model.htm

[7] cs.wikipedia. wikipedia. [online]. [cit. 2016-05-23]. Dostupné z: https://cs.wikipedia.org/wiki/Barevn%C3%BD_model

[8] Holota, Fiřt. Digitalizace a spracování obrazu. home.zcu.cz. [online]. 1.10.2015 [cit. 2016-05-23]. Dostupné z:http://home.zcu.cz/~holota5/publ/DigZprO.pdf

[9] wikipedie. cs.wikipedia.org. [online]. 24.6.2014 [cit. 2016-05-23]. Dostupné z: https://cs.wikipedia.org/wiki/YUV

[10] Ma, Y., Soatto, S., Kosecka, J., Sastry, S.S.. An Invitation to 3-D Vision. Belgian: Belgian Mathematical Society, 2005. ISBN 978-0-387-00893 6

[11] MVDK. http://www.urel.feec.vutbr.cz/. [online]. 3.1.2013 [cit. 2016-05-23]. Dostupné z:http://www.urel.feec.vutbr.cz/web_documents/studium/predmety/MVDK/FRVSg/kompletni.pdf

[12] Odstranění šumu. Wikipedia: the free encyclopedia. [online]. 2001- [cit. 2016-05-23]. Dostupné z:https://cs.wikipedia.org/wiki/Odstran%C4%9Bn%C3%AD_%C5%A1umu

[13] Cannyho hranový detektor. Wikipedia: the free encyclopedia. [online]. 2001- [cit. 2016-05-23]. Dostupné z:http://cs.wikipedia.org/wiki/Cannyho_hranov%C3%BD_detektor

[14] Houghova transformace. Wikipedia: the free encyclopedia. [online]. 2001- [cit. 2016-05-23]. Dostupné z:https://cs.wikipedia.org/wiki/Houghova_transformace

[15] picamera. picamera. [online]. 2016 [cit. 2016-05-23]. Dostupné z: http://picamera.readthedocs.io/en/latest/index.html

[16] National Instruments. National Instruments. [online]. 2016 [cit. 2016-05-23]. Dostupné z: http://www.ni.com/white-paper/6723/en/

[17] GP2Y0A02YK0F. www.sparkfun.com. [online]. 1.12.2016 [cit. 2016-05-23]. Dostupné z:https://www.sparkfun.com/datasheets/Sensors/Infrared/gp2y0a02yk_e.pdf

[18] SRF05 - Ultra-SonicRanger Datasheet.DEVANTECH LTD [online]. 2012 [cit. 2014-12-17]. Dostupné z URL: http://www.robotelectronics.co.uk/htm/ srf05tech.htm

[19] RPI-Wireless-Hotspot. Elinux. [online]. 22.8.2015 [cit. 2016-05-23]. Dostupné z: http://elinux.org/RPI-Wireless-Hotspot

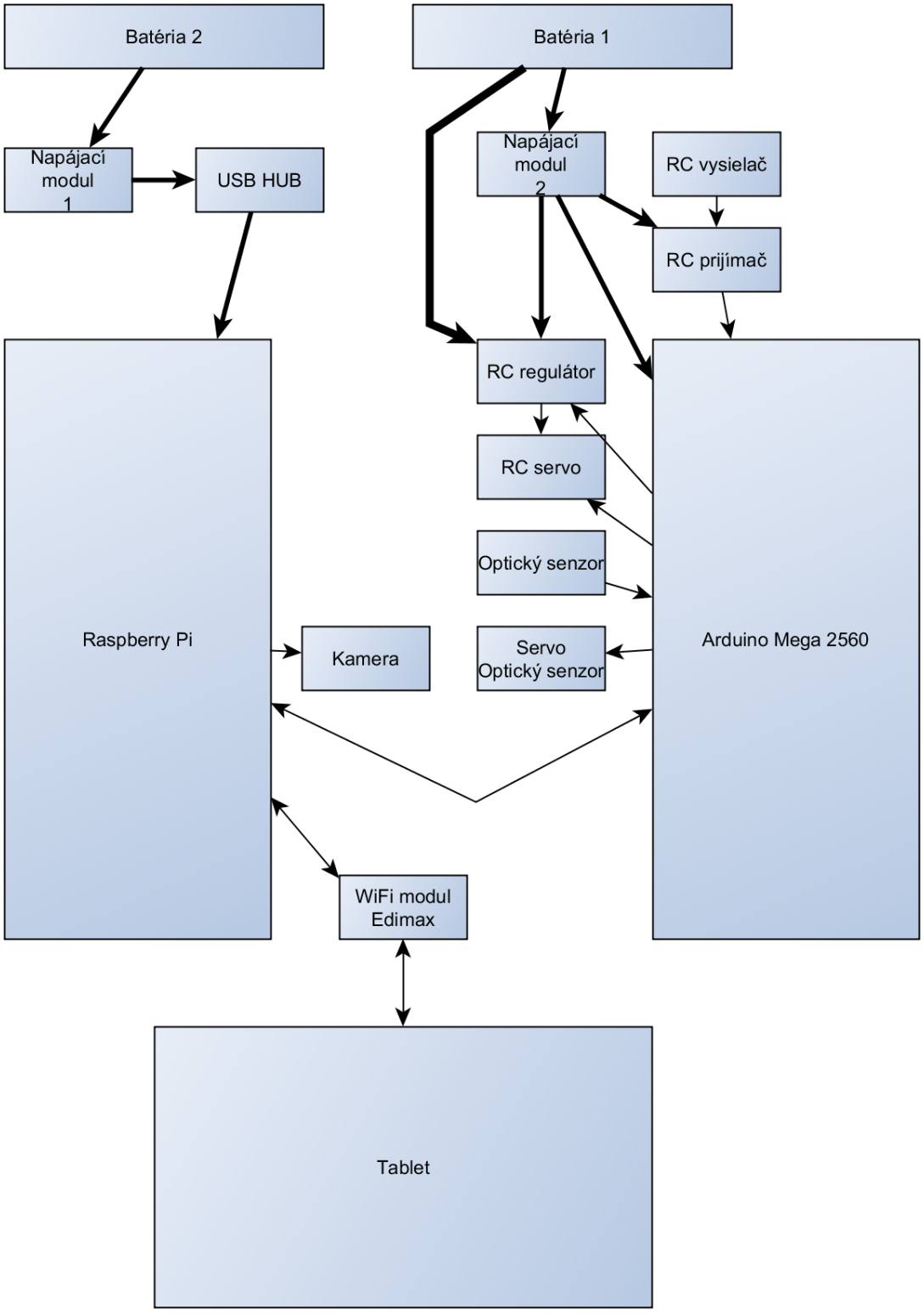
[20] rpi-serial-console. github. [online]. 17.9.2016 [cit. 2016-05-23]. Dostupné z: https://github.com/lurch/rpi-serial-console

[21] Arduino Software (IDE). arduino. [online]. 2016 [cit. 2016-05-23]. Dostupné z: https://www.arduino.cc/en/Guide/Environment

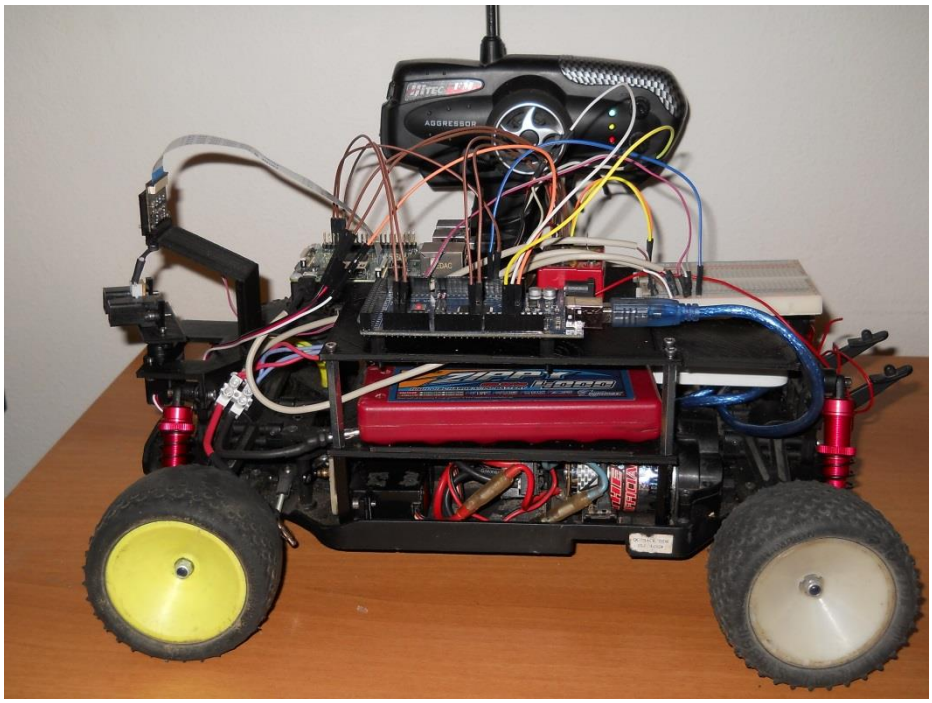
[22] adafruit. adafruit. [online]. [cit. 2016-05-23]. Dostupné z: https://learn.adafruit.com/adding-a-real-time-clock-to-raspberry-pi/

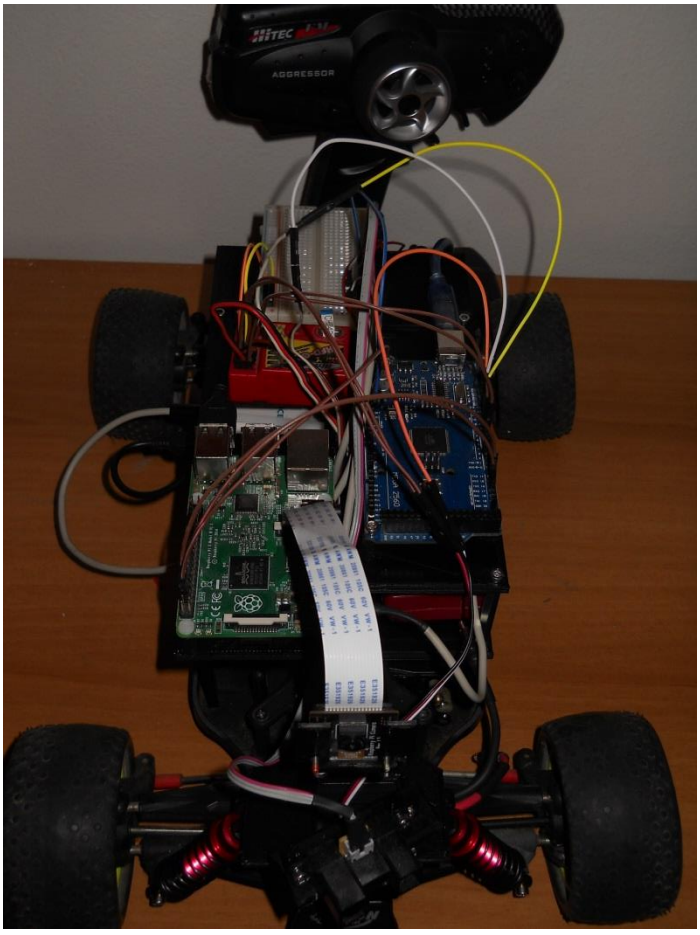
PRÍLOHY

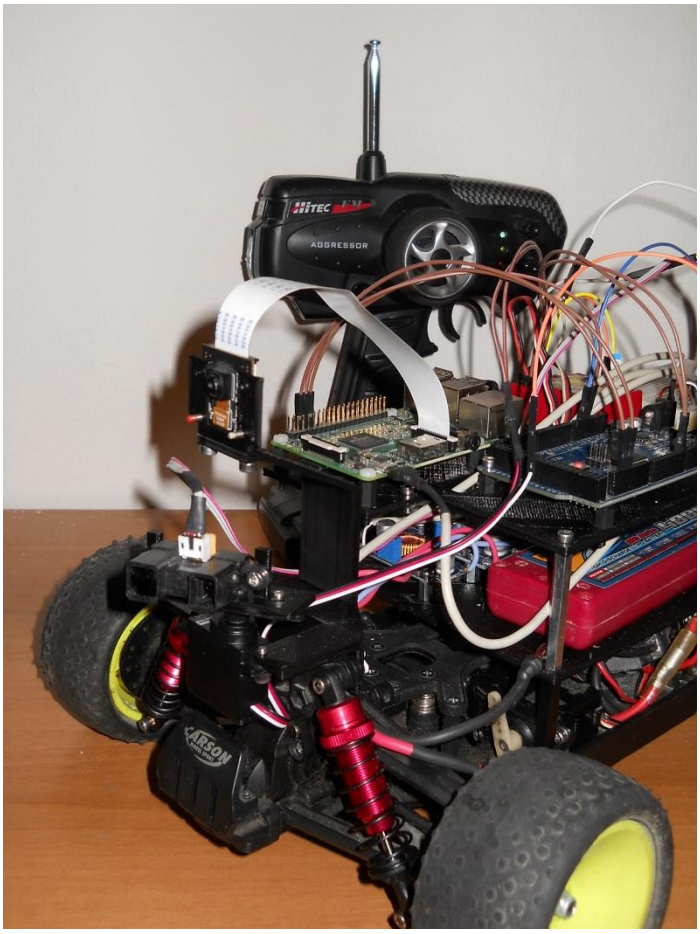
A.1 Bloková schéma



A.2 Praktická realizácia






AGGRESSOR
ARSON