TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

MECHATRONICKÝ SYSTÉM PRO TŘÍDĚNÍ LEGO DÍLKŮ

Semestrální projekt

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 1802T007 - Informační technologie

Autor práce: **Petr Kaspar**
Vedoucí práce: doc. Ing. Josef Chaloupka, Ph.D.

Liberec 2016

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, zejména § 60 – školní dílo.

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

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

Bakalářskou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum:

Podpis:

Abstrakt

Semestrální projekt se zabývá obecnou problematikou zpracování a rozpoznávání obrazu. Dále se práce v teoretické části věnuje seznámení s robotickou stavebnicí LEGO MINDSTORMS NXT a jejím hardwarovým specifikacím.

V praktické části práce je popsáno konstrukční řešení mechatronického systému, který má za úkol třídit LEGO dílky podle určitých kritérií. Ten je rozdělen na několik prvků, u kterých je samostatně nastíněn princip jejich činnosti. K tomuto systému je vytvořen ovládací program, u kterého je popsán postup vedoucí k nalezení objektu v obraze.

Klíčová slova

LEGO, MINDSTORMS NXT, zpracování obrazu, robot, programování

Abstract

Semester project deals with the general progress of image processing. The thesis deals with the theoretical part of the introduction to robotic LEGO MINDSTORMS NXT and its hardware specifications.

The practical part describes the design solutions of mechatronic system, which is responsible for sorting LEGO pieces according to specific criteria. It is divided into several elements, for which is described principles of their activities. For this systém is created a control program for which is described the procedure leading to the location of the object in the image.

Key words

LEGO, MINDSTORMS NXT, image processing, robot, programming

Obsah

Seznam obrázků

Úvod

V první části celé práce bylo úkolem seznámit se s jednotlivými kroky pro zpracování obrazu. Tyto kroky se mohou měnit v závislosti na zadání konkrétní aplikace. Do první části patří i seznámení se se stavebnicí MINDSTORMS NXT, kterou uvedla na trh společnost LEGO. Tato stavebnice umožňuje do klasických umělohmotných kostek zabudovat funkční prvky, jako jsou různé senzory a motory, které společně s mikropočítačem mohou vytvořit nejrůznější roboty, schopné pohybové nebo manipulační činnosti. V práci jsou představeny její hardwarové specifikace a možnosti programování NXT kostky.

Byl navržen a sestrojen mechatronický systém, především ze stavebnice LEGO. Pouze některé části musely být nahrazeny z důvodu nedostatku některých dílků. Byla použita také kamera určená pro snímání jednotlivých dílků.

Vstupem do celého systému je nějaké nesetříděné množství kostiček LEGO. Cílem třídícího systému, který je tvořen nejrůznějšími dopravníky, zvedáky a koly, je oddělit od sebe jednotlivé dílky tak, aby se pod kamerou ocitl vždy pouze jeden. Pouze tak se může provést úspěšné rozpoznání a vyhodnocení. V práci je stručně popsána činnost jednotlivých prvků systému.

Programovatelné bloky NXT jsou připojeny k PC pomocí Bluetooth a USB. V počítači běží ovládací program, který má na starosti vyhodnocování obrazového přenosu z kamery a na tom závislé řízení všech použitých servomotorů.

1 Zpracování digitálního obrazu

Obrazem se rozumí jakákoli reprodukce reálného světa, který je trojrozměrný, do nějaké vizuální scény, která je v případě této práce dvourozměrná. Obraz zachycený na sítnici lidského oka či na TV kameru je tedy vícerozměrný signál. Může být modelován matematicky pomocí spojité skalární funkce několika proměnných. Pro uložení a následné zpracovávání obrazu v počítači nebo v jiném systému je nezbytné převést jej do digitální podoby. V této formě je obrazová funkce $f(x, y)$ reprezentována maticí, kde prvky matice jsou obrazové elementy (pixely). Argumenty funkce $f(x, y)$ jsou souřadnice ve dvou rozměrech. Jedná-li se o šedotónový obraz, pak každý prvek na těchto souřadnicích nese informaci o jasové hodnotě prvku. [1][3] V případě barevného modelu RGB je každý prvek funkce $f(x, y)$ vektor, nesoucí informaci o třech jasových složkách barevných pásem.

Průběh zpracování obrazu lze rozdělit do několika základních kroků, které se můžou měnit v závislosti na zadání konkrétní aplikace:

- Snímání a digitalizace
- Předzpracování
- Segmentace
- Rozpoznávání
- Klasifikace

1.1 Snímání a digitalizace

Důležitým krokem pro úspěšné zpracování obrazu je nasnímání sledovaného objektu. K získání kvalitního obrazu scény je zapotřebí mít zkoumaný objekt vhodně osvětlený. Vzhledem k tomu, že se mechatronický systém mohl přemísťovat z místa na místo, nemohly být zaručeny stejné světelné podmínky. Rušivým vlivem může být také denní světlo, nebo jiné umělé osvětlení na pracovišti. Pro získání stále stejného jasového obrazu je třeba tyto světelné šumy eliminovat. To lze vyřešit například technickým odstíněním, nebo volbou osvětlovače, který vydává více světla, než jsou nežádoucí zdroje. Při návrhu optické soustavy je volba osvětlovače závislá na interakci zkoumaného předmětu se světlem a na úloze, která se s pořízeným obrazem má vykonávat. Zadní osvětlovač se používá k zobrazení obrysu objektu tím, že vytváří kontrast mezi pozadím a objektem. Vzhledem k tomu, že kromě tvaru bylo nutné zkoumat i barvu objektu,

nebylo by použití zadního osvětlovače příliš vhodné. Také by řešení s osvětlovači bylo konstrukčně náročnější a vyžadovalo by další funkční prvky.

V případě řešení semestrálního projektu bylo přistoupeno k softwarovému řešení. Uživatel si v první fázi nastaví například správnou mez pro prahování, nebo velikost objektu, spadající do naší oblasti zájmů, tady LEGO kostky.

Čidla pro vstup obrazové funkce jsou většinou zdrojem spojitého signálu. Abychom obrazovou funkci mohli zpracovat v počítači, musíme ji digitalizovat. Digitalizace spočívá ve vzorkování obrazu v matici $M \times N$ bodů a v kvantování spojité jasové úrovně každého vzorku do K intervalů. Díky kvantování nabývá jasová funkce v digitalizovaných obrazech celočíselných hodnot. Čím jemnější je vzorkování (čím větší M, N) a kvantování, tím lépe je aproximován původní spojitý obrazový signál. [1]

Shannonova věta, kterou se vzorkování řídí, nám říká, že nejmenší detail v digitálním obraze musí být minimálně dvojnásobkem vzorkovací frekvence. Frekvencí rozumíme rychlost střídání intenzit barev. Volba vhodného rozlišení obrazu je důležitá, protože při nízkém rozlišení se bude ztrácet informace o detailech, a naopak při velkém rozlišení bude stoupat výpočetní náročnost při dalším zpracování obrazu. Jedná-li se o zpracovávání obrazu z kamery v reálném čase, rychlost zpracování je důležitější než detailní obraz. Rozlišení obrazu z kamery bylo tedy zvoleno nízké také pro to, že postačilo získávat informaci jen o barvě a hrubých obrysech. [3]

1.2 Předzpracování obrazu

Po úspěšném získání obrazu a jeho digitalizaci máme k dispozici digitální obraz pozorované scény. Obraz může být zkreslen díky způsobu snímání, nebo nevhodných podmínkách při průběhu snímání. Pokud je znám charakter zkreslení, je možné tuto chybu opravit pomocí korekcí, které jsou jednou z metod, které usnadňují další analýzu obsahu obrazu, identifikaci objektů nebo jen zvýrazňují důležité rysy obrazu pro snazší pozorování člověkem. [3]

rozdělení metod předzpracování obrazu:

- jasové transformace
- geometrické transformace
- filtrace a ošetření

Z praktického hlediska je transformace jasové stupnice důležitá zejména pro úpravy obrazu, které zajišťují pozorovateli snazší interpretaci vizualizovaného obrazu.

Příkladem může být snaha o zvýšení kontrastu původně nekontrastního rentgenového obrazu. Pro automatickou analýzu, kde obrazová data člověk neinterpretuje, nemají transformace jasové stupnice žádaný význam. [1]

Geometrické transformace vypočtou na základě souřadnic bodů ve vstupním obraze souřadnice bodů ve výstupním obraze. Geometrické transformace jsou obvyklé v počítačové grafice, která často pracuje s objekty ve vektorovém tvaru. V digitálním zpracování obrazu tedy dovolují odstranit geometrické zkreslení vzniklé při pořízení obrazu (např. korekce geometrických vad objektivu kamery). [1]

Filtrace nebo také vyhlazení obrazu slouží k zvýraznění nebo naopak potlačení určité informace která je v obraze obsažena. Obraz se může zpracovat několika metodami. Může se z něj odstranit šum, který daný obraz zkresluje, může se vyhladit, aby se zvýraznili plochy, může se pracovat s kontrastem nebo se mohou v obraze hledat hrany, linie nebo tvary. [4]

1.3 Segmentace

Jedním z nejdůležitějších kroků zpracování obrazu je segmentace obrazu. Jedná se o analýzu obrazu vedoucí k nalezení objektů v obraze. Za objekty se zde považují části obrazu, které jsou bodem zájmu v dalším průběhu zpracování. Cílem segmentace je tedy rozdělení obrazu do částí odpovídající předmětům či oblastem reálného světa. Výsledkem segmentace by měl být soubor oblastí, které odpovídají objektům ve vstupním obraze. Jedná se pak o tzv. kompletní segmentaci. Pokud ale oblasti neodpovídají přesně objektům, tak se tato segmentace nazývá částečná. Kompletní segmentace obecně využívá vyšší úrovně zpracování, která je založena na znalostech řešeného problému. Částečná segmentace je založena na principu homogenity obrazových vlastností (např. jas, barva) uvnitř segmentu. [3]

Pro segmentaci obrazu existuje celá řada segmentačních algoritmů, nejpoužívanějšími jsou metody založené na prahování. Příkladem je prahování podle úrovni šedé, které patří k nejjednodušším a také nejstarším segmentačním metodám. Prahování je funkce, která provádí transformaci vstupního obrazu na výstupní binární obraz, přičemž každý obrazový bod ze vstupu se porovnává s definovaným prahem. [2]

Pro segmentaci obrazu lze využívat dále metody založené na detekci významných hran v obraze. Lokální hrany jsou detekovány pomocí hranových detektorů na základě

rozdílu hodnot okolních pixelů. Hranový detektor je algoritmus, který vyhledává množinu hran (bodů, pixelů) v obraze.

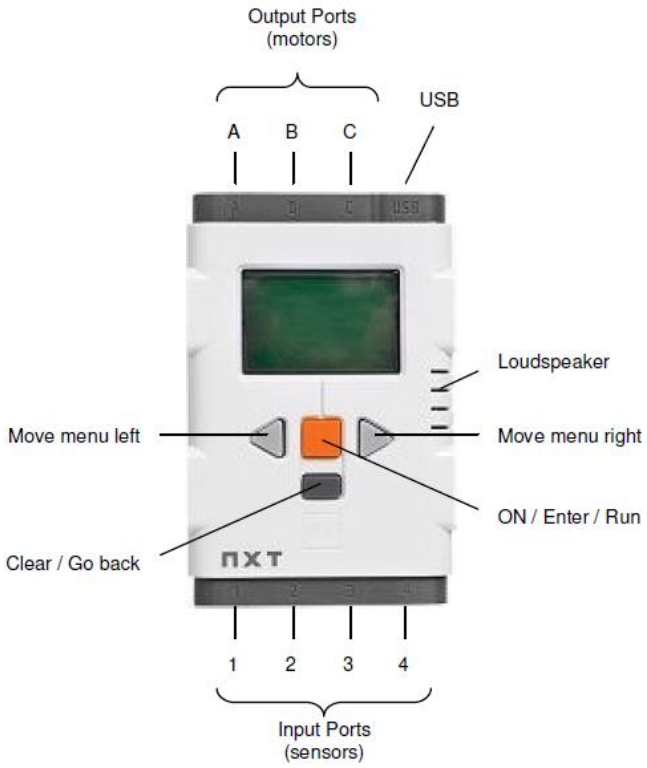
Dalšími metodami jsou metody založené na hledání regionů v obraze. Pokud lze identifikovat hrany, měly by teoreticky ohraničovat regiony. Kontury regionů však mohou být porušené, nemusí ohraničovat celý region. Není také zaručeno, že hranice regionů nalezené metodou detekce hran budou stejné, jako ty nalezené metodou hledání regionu. Při segmentaci obrazu je to důležitou etapou identifikace oblasti. Jednou z metod je využití řetězových kódů pro popis hranice objektu. [2]

2 Stavebnice LEGO MINDSTORMS

Stavebnice LEGO MINDSTORMS NXT je vyráběná společností LEGO a lze pomocí ní sestrojit robota, doplněným vlastním programem. Pomocí této stavebnice si mohou nejen děti osvojit základy robotiky. Používají ji školy i univerzity jako doplněk při výuce. Dává také možnost použití řady programovacích jazyků. LEGO vydalo stavebnici v několika verzích a všechny dílky jsou spolu kompatibilní, čímž dává spoustu možností konstrukčních řešen. V projektu byly použity soupravy LEGO MINDSTORMS NXT, vydaná v roce 2006 a nesoucí produktové označení 9797.

2.1 Hardwarové vybavení robota

Jedna souprava obsahuje řadu stavebních prvků, hlavně pak ale moduly, které umožňují vykonávat pohyby a získávat informace z okolí. Patří sem tři interaktivní servomotory, dva dotykové sensory, ultrazvukový senzor, optický senzor a zvukový senzor. Další snímače kompatibilní se systémy LEGO MINDSTORMS vyrábí společnost HiTechnic a patří mezi ně například barevný snímač, gyroskop, akcelerometr a kompas. Tyto motory a snímače by však samy o sobě nefungovaly. Nejdůležitějším prvkem je NXT inteligentní kostka.



Obrázek 2.1: NXT kostka s porty a tlačítky – převzato z [6]

2.1.1 NXT brick

NXT brick (kostka) je řídicí jednotkou celé stavebnice. Pohání jí 32bitový procesor Atmel ARM a umožňuje řídit a využívat ostatní moduly. Díky paměti typu RAM se mohou do NXT kostky ukládat soubory, které zůstanou uloženy i po odpojení od zdroje napětí. Těmito soubory se myslí zvukové nahrávky, jednoduché obrázky, ale hlavně programy, ovládající celého robota. K dispozici jsou čtyři vstupy a tři výstupy pro snímače a motory (Obrázek 2.1). Snadné ovládání kostky umožňují čtyři tlačítka a grafický LCD display. Připojení NXT kostky k počítači lze realizovat buď pomocí USB kabelu, nebo je možné použít bezdrátové rozhraní Bluetooth. K napájení jednotky slouží Li-Ion baterie, ale v případě potřeby lze použít i tužkové baterie (6x 1,5V AA). Pro podrobnější přehled technických parametrů NXT kostky je zde tabulka. [5]

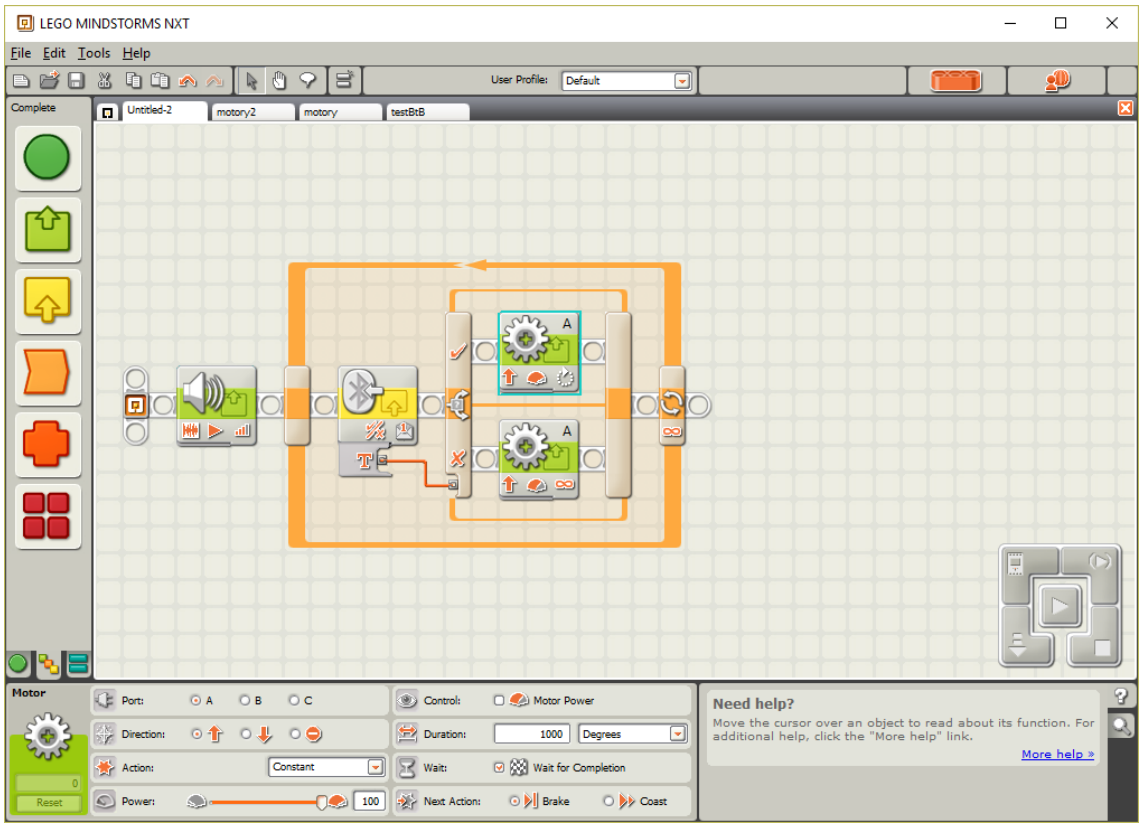
Tabulka 2.1: Technické parametry řídící jednotky NXT - převzato z [5]

Parametr	**Specifikace**
Procesor	**Atmel ARM, AT91SAM7S256** • 48 MHz • 32 bit • 256 KB FLASH • 64 KB RAM
Co-procesor	**Atmel AVR, ATmega48** • 8 MHz • 8 bit • 4 KB FLASH • 512B RAM
Připojení k PC	**USB 2.0** • 12 Mbit/s **Bluetooth, CSR BlueCore 4 v 2.0 + EDR System** • podpora Serial Port Profile (SPP) • interní 47 KB RAM • externí 8 Mb FLASH • 26 MHz • 460,8 Kbit/s

Vstupy / výstupy	**4 vstupní porty (1, 2, 3, 4)** • 6ti žilové rozhraní, konektor RJ12 • podpora digitálního i analogového rozhraní • 1 vysokorychlostní port IEC 61158 Typ 4/EN 50170 **3 výstupní porty (A, B, C)** • 6ti žilové rozhraní, konektor RJ12 • podpora vstupu pro enkodéry
Display	**Grafický LCD** • rozlišení 100 x 64, černobílý • viditelná oblast 26 x 40,6 mm
Reproduktor	**Výstupní kanál s 8bit rozlišením** • podporované vzorkování 2 – 16 KHz
Ovládání	4 tlačítka
Napájení	Li-Ion baterie nebo 6x 1,5 V AA tužkové články

2.2 Možnosti programování NXT

K práci s robotem je zapotřebí mimo jiné základy programování. Nejjednodušeji však lze přimět robota k nějaké řízené činnosti vytvořením jednoduchého programu v LEGO MINDSTORMS NXT softwaru (Obrázek 2.2). Zde se pracuje v grafickém jazyce NXT-G, který spočívá ve skládání připravených bloků do schématu, které odpovídají požadovaným funkcím. Tento dodaný software je také užitečný pro případný update NXT firmwaru.



Obrázek 2.2: Ukázka prostředí NXT-G vytvořené společností National Instruments

Pro práci s NXT lze využít i další programovací prostředí, například RoboLab, LabVIEW, Microsoft Robotics Studio, MATLAB a Simulink. Pro komplexnější programy založené také z části na interakci s uživatelem, bude lépe využít rozšířenější textové jazyky, jako třeba Java, C, C++, C# a Python.

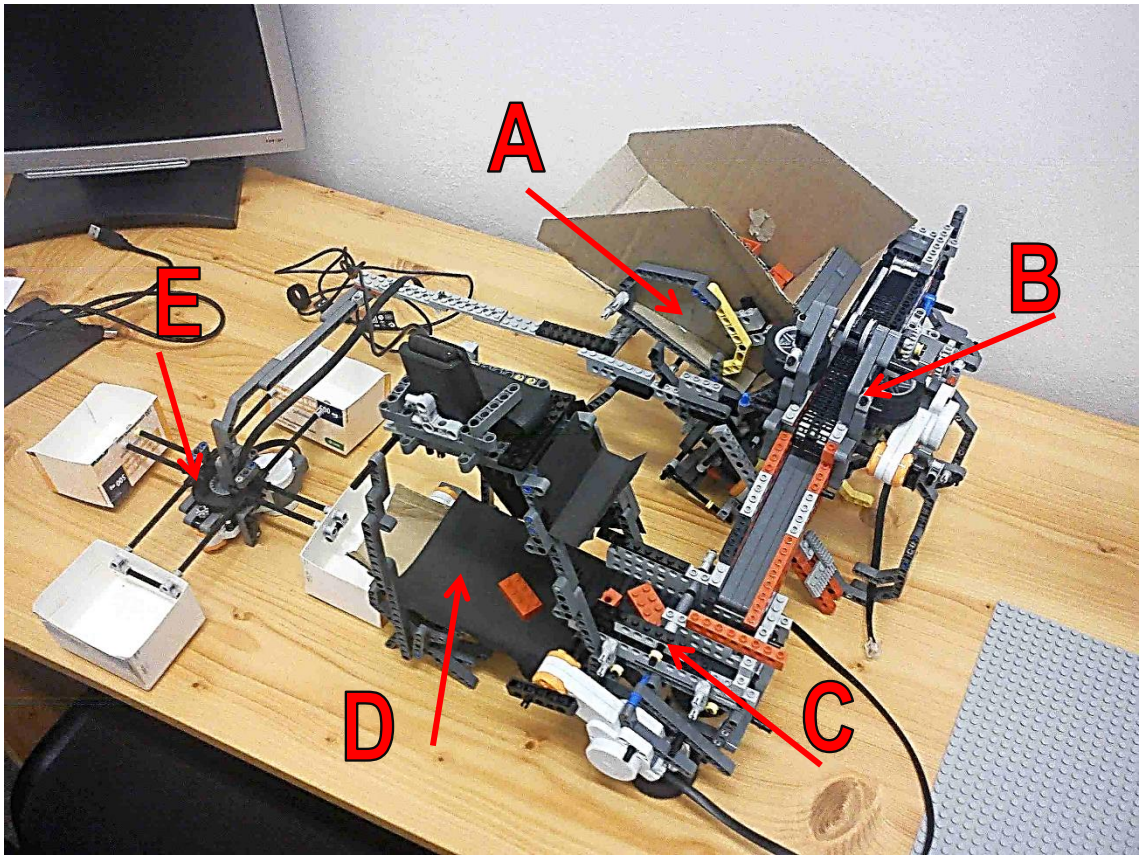
Jak je vidět v tabulce nahoře, NXT kostka se může s PC spojit přes Bluetooth a USB. Pomocí Bluetooth mohou komunikovat i kostky mezi sebou. Tato technologie umožňuje také komunikaci pře mobilní telefon. Pokud je řídicí jednotka NXT s počítačem spojena, jsou k dispozici dva způsoby řízení. Prvním je tak zvané On-Board řízení, kdy je program po napsání uložen do paměti NXT. Zde se daná sekvence příkazů ovládající běh robota může kdykoliv spustit. Druhým způsob řízení je posílání přímých příkazů do řídící jednotky. V případě tohoto projektu je využita tato druhá možnost, kdy se za pomocí knihoven v jazyce C# zapínají a vypínají motory v celém třídícím systému.

Programovací jazyk C# byl vyvinutý firmou Microsoft, která jej založila na jazycích C++ a Java a má velmi podobnou syntaxi. C# lze využít kromě tvorby formulářových aplikací ve Windows také k vytváření databázových programů, webových aplikací a softwaru pro mobilní zařízení. [7]

3 Konstrukční řešení mechanického systému

Zkráceně lze říci, že účel celého mechatronického systému je vzít z nějakého nesrovnaného množství LEGO kostek jednotlivé dílky, jednotlivě je vyhodnocovat a uložit do správné přihrádky.

V projektu byly využity dvě NXT řídící jednotky, z nichž jedna byla připojená přes USB a druhá přes Bluetooth. Dále bylo použito šest servomotorů, zajišťující pohyb všech funkčních prvků. Celá třídící jednotka se dá rozdělit do pěti dílčích částí (Obrázek 3.1). Toto rozdělení umožňuje snadnější manipulaci v případě přenášení. Je to také výhoda při hledání vhodného prostoru pro uskladnění. Celková konstrukce by měla být navržena takovým způsobem, aby byla zajištěna stabilita. Při stavbě však bylo k dispozici jen omezené množství stavebních prvků. Je tedy vyloučen přenos všech částí dohromady a musí se přistoupit k částečnému rozebrání.



Obrázek 3.1: Konstrukce mechanického systému

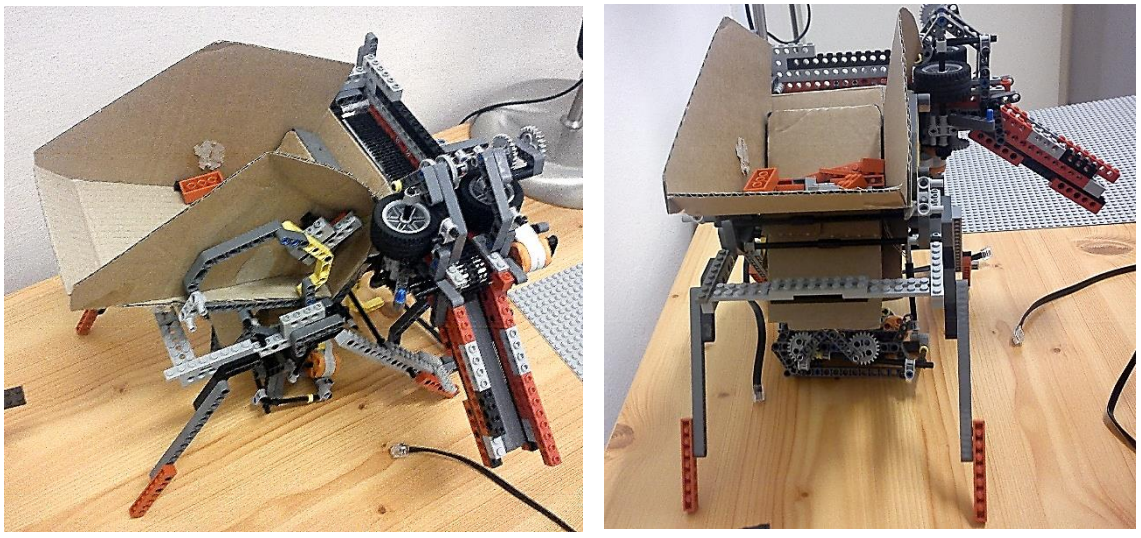
- **A. zásobník se zdviží**
- **B. dopravní pás s koly**
- **C. druhý dopravník**
- **D. sklápěcí mechanizmus s kamerou**
- **E. koncové úložné oddíly**

3.1 Obecný popis chodu systému

Vstupem do třídičky je nějaké nesrovnané množství dílků různé barvy, tvaru a velikosti. Tyto kostky jsou na začátku nasypány do zásobníku (Obrázek 3.1A). Odtud jsou pomocí zdviže vyzvednuty na první pás, odkud jsou přepraveny na pás druhý. Při tomto přesunu jsou podniknuty kroky pro to, aby na konci druhého pásu již jeli jednotlivé kostky lega jednotlivě za sebou. Pokud z konce druhého dopravníků spadne kostka pod kameru, všechny dosavadní pásy se zastaví. Kamera je součástí konstrukce D. Po vyhodnocení dílku počítačem se pootočí motorem z části E tak, aby pod sklápěcí konstrukcí byla správná krabička.

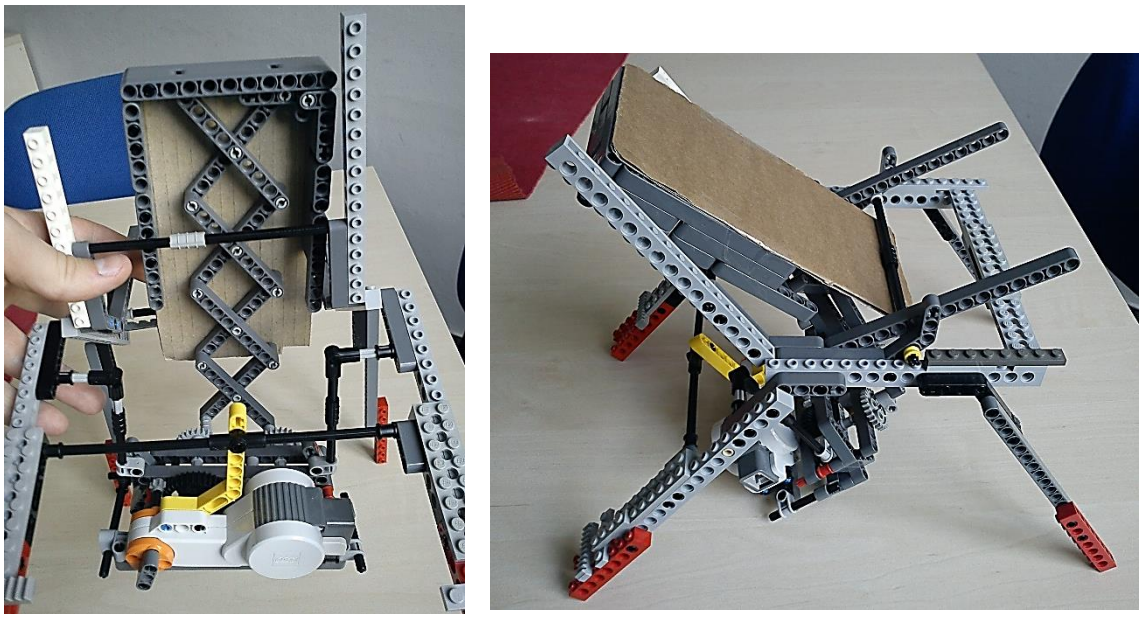
3.2 Zásobník se zdviží

Zásobník se zdviží je počáteční místo celé soustavy. Z důvodu nedostatku dílků lega byl použit karton pro vytvoření „korby“. Přesun lega z této hromádky na první dopravník zajišťuje výtah. Ten je poháněn jedním motorem, který se točí neustále jedním směrem. Je však jedno jakým. Nosná plocha zdviže je ve své nejnižší poloze lehce pod úrovní dna korby. V tom okamžiku se díky naklonění přesune část hromádky kostek na výtah a ten je přepraví nahoru. Ve svém nejvyšším bodě je opět lehce nad okrajem přední stěny korby. Díky dostatečnému sklonu nabrané kostičky přepadnou na dopravní pás. Tyto dvě části jsou vidět na obrázku (Obrázek 3.2).



Obrázek 3.2: Konstrukce prvků A a B

Na obrázku (Obrázek 3.3) je vidět prvek A z celé soustavy ve fázi, kdy ještě není zakryt náhradními kartonovými stěnami. Také ukazuje, jak vypadá zespod řešení výtahu s motorem.



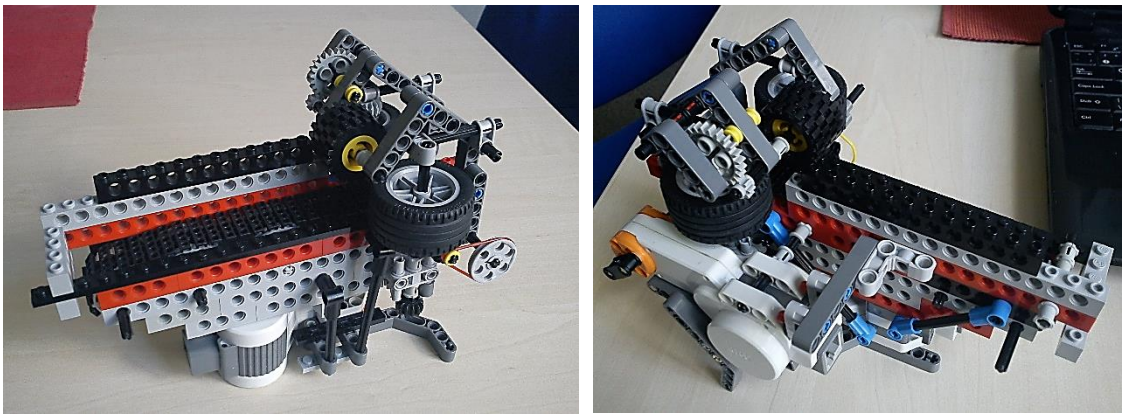
Obrázek 3.3: Řešení zdviže a zásobník bez kartonových stěn

3.3 Dopravní pás s koly

Na plastový pás, který je také součástí stavebnice, dopadají kostky lega z výtahu a pomalu se posouvají k dopravní rampě. Ta se nijak nepohybuje a kostky po ní sklouznou na prvek C.

Kostky vycházející z tohoto pásu na rampu jsou již částečně seřazeny jednotlivě za sebou. Trojice kol, které se otáčí v protisměru vzhledem k pásu, má za úkol vrácení nevhodně poskládaných dílků zpátky do výchozího boxu. Velikost otvoru mezi koly a pásem odpovídá jedné lego kostce velikosti 2×y. Pokud by měly systémem procházet pouze tyto dílky, mohla by za tímto pásem být rovnou umístěná kamera a detekovat. Kostky by totiž nemohly ležet ani na sobě, ani vedle sebe. Různá velikost vstupních dílků však nevhodné poskládání nevylučuje a pod kameru by se mohly dostat dva dílky najednou.

Z důvodu různé rychlosti a směru pohybu mají kola a pás samostatný servomotor. Jsou zde tedy použity dva, jak je částečně i vidět na obrázku (Obrázek 3.4).

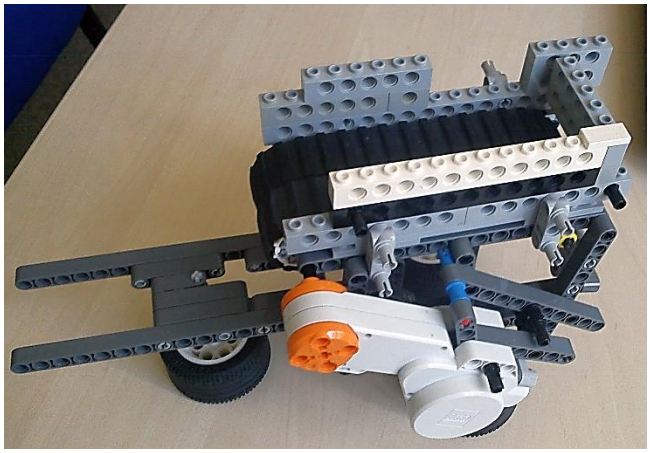


Obrázek 3.4: Dopravní pás s koly

3.4 Druhý dopravník

Tentokrát se jedná o dvojici gumových pásů. Byly také součástí balení stavebnice a patrně měli primárně sloužit pro pásové robotické vozítko. Dopravníky jsou poháněny opět jedním servomotorem (Obrázek 3.5).

Na začátek pásu dopadají dílky z rampy, která je připevněná k předchozímu prvku. Po průchodu horní přepážkou, která je nad pásem a opět zamezuje průchod několika kostek na sobě, dopadají jednotlivé dílky na sklápěcí mechanismus D. Dopravník se pohybuje dostatečně pomalu a ovládací program má tak mnoho času na detekci kostky a zastavení systému.



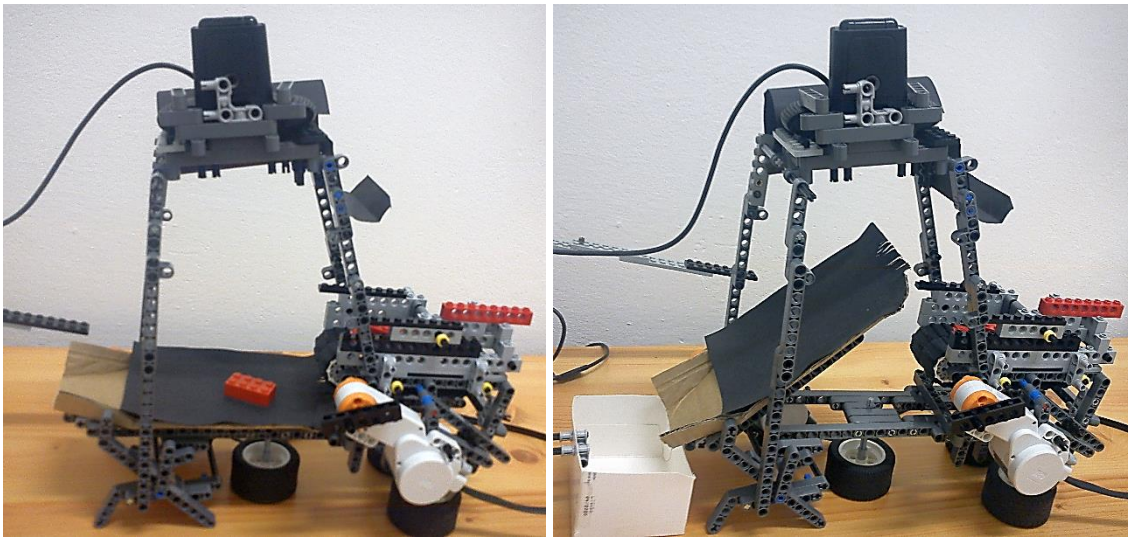
Obrázek 3.5: Dopravníkový pás číslo 2

3.5 Sklápěcí mechanizmus s kamerou

V tomto prvku je kromě stavebnice lega, která vytváří konstrukci, použita také kamera a černá papírová podložka. Webová kamera je značky Logitech s označením HD PRO WEBCAM C92. Umožňuje pořizovat záznam ve vysokém rozlišení Full HD 1080p a je připojena k počítači rozhraním USB.

Pod kameru, která je umístěna na vršku konstrukce tohoto prvku, přijíždějí už samostatné dílky lega. To je zajištěno díky včasnému zastavení všech motorů, které doposud měli na starosti pohyb pásů, koleček a zdviže. Toto zastavení zaručuje správný chod celé třídičky. Příkaz k zastavení motorů vyšle řídící program ve chvíli, kdy detekuje objekt na černé ploše snímané kamerou. Onu černou plochu představuje obyčejný černý papír, který je přilepený na plochou sklápěčku vyrobenou z lega.

Příchozí obraz z kamery, na kterém se provádí detekce objektů, musí přicházet v reálném čase. Pokud by nastavené rozlišení bylo příliš vysoké, docházelo by ke zpoždění a na plochu pod kamerou by ve skutečnosti mohlo dopadnout z druhého pásu více kostek.



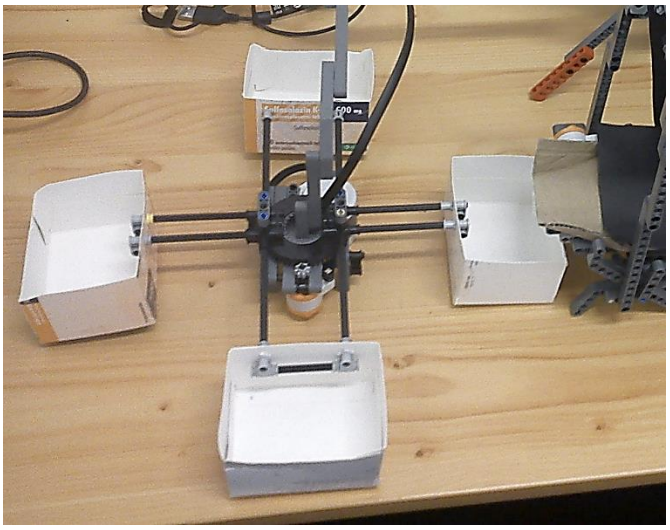
Obrázek 3.6: Konstrukce sklápěčky v dolní a horní poloze

Na obrázcích nahoře jsou vidět sklápěcí prvky společně s předešlým dopravním pásem. V horní části se nachází kamera, která snímá černou podložku. V prvním případě se na sklápěčce nachází červená kostka lega a v druhém je v poloze pro vyklopení do krabičky. Vyklopení se provede v okamžiku, kdy se po dokončení správné detekce dílku vykoná posun příslušného boxu pod sklápěčku.

3.6 Koncové úložné boxy

Jeden motor rotuje se čtyřmi rameny, na jejichž konci jsou umístěny krabičky. Při nadbytečnosti stavebnicových kostek by nebyl problém boxy vyrobit taktéž z lega. Jak je vidět na obrázku (Obrázek 3.7), náhradní papírové plní účel stejně dobře.

Ze zbylých dílků stavebnice bylo vytvořeno rameno, které tento konečný prvek udržuje ve správné pozici.



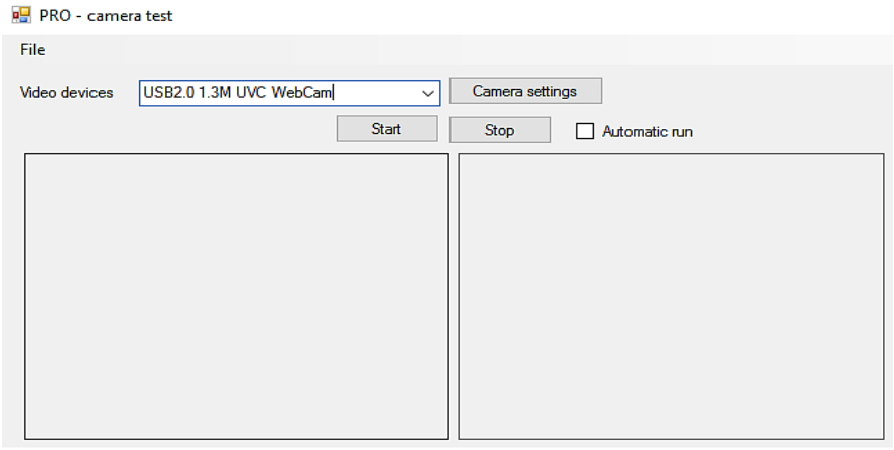
Obrázek 3.7: Koncové úložné boxy

4 Tvorba ovládacího programu

Obě použité NXT kostky byly propojeny s PC, kde běžel ovládací program zajišťující zpracování obrazových dat a posílání příkazů jednotlivým motorům. Tento program byl napsán v programovacím jazyce C# za použití vývojového prostředí Microsoft Visual Studio Community 2015.

4.1 Postup získávání obrazu

Zdrojový kód celého programu je značně rozsáhlí. V této kapitole je ukázáno několik nejzajímavějších částí kódu a metod, které jsou postupně aplikovaných na vstupní obraz. Cílem uvedeného postupu je získání obrazu separovaného dílku stavebnice lega, jenž se dostal pod kamerou průchodem skrz mechatronický systém. Na obrázku (Obrázek 4.1) je část GUI, ve kterém jsou vidět dvě ohraničené oblasti vytvořené pomocí *AForge.Controls*. V první se zobrazuje čistý obraz z kamery. Ve druhém jsou na něj aplikovány postupy vedoucí k nalezení a rozpoznání LEGO dílku. Je tedy možné v reálném čase pozorovat, jak se obraz bude chovat v případě změny proměnných.



Obrázek 4.1: Část GUI

4.1.1 Odstranění pozadí

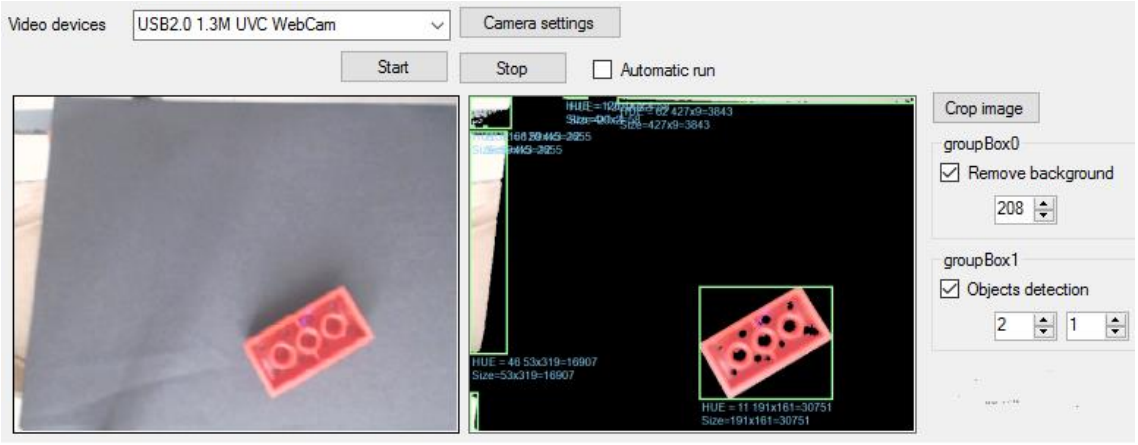
Metoda *removeBackground* odstraňuje z obrazu tmavě šedou podložku, kterou tvoří černý papír. Pozadí mění na čistě černé a v obraze zůstane pouze zkoumaný barevný dílek lega. Proměnnou *thresholdBG* má uživatel možnost měnit v závislosti na světelných podmínkách pracoviště. Řešením je následující kus kódu využívající třídu *ColorFiltering* z knihovny *AForge.Imaging.dll*. Pro každou barevnou složku v pixelu lze určit rozsah a rozhodnout, zda se v obraze ponechá, nebo bude nahrazena černou barvou.

```
private static Bitmap removeBackground(Bitmap image) {
// odstraneni tmavého pozadi
ColorFiltering colorFilter = new ColorFiltering();
colorFilter.Red = new IntRange(0, thresholdBG);
colorFilter.Green = new IntRange(0, thresholdBG);
colorFilter.Blue = new IntRange(0, thresholdBG);
colorFilter.FillOutsideRange = false;

Bitmap processedImage = colorFilter.Apply(image);
return processedImage;
}
```

4.1.2 Hledání osamocených objektů

Po odstranění nedokonalého pozadí a nahrazením čistě černým by teoreticky v obraze mohl zůstat jen hledaný objekt zájmu. Ve skutečnosti se v obraze často vyskytuje například plocha stolu, na kterém celá konstrukce stojí (Obrázek 4.2). Může to být způsobeno nevhodným umístěním kamery při sestavování.



Obrázek 4.2: Výsledek metody *BlobDetection*

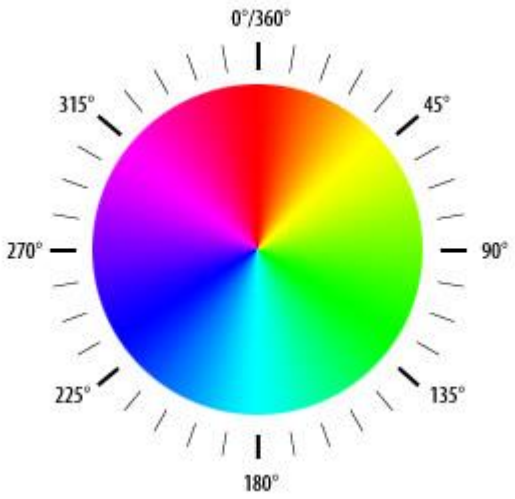
Pro metodu *BlobDetection* je vstupem bitmapa, které je zároveň výstupem z předchozí popsané funkce. Třída *BlobDetection,* která je opět z *AForge.Imaging.dll,* dokáže extrahovat samostatné objekty, jejichž obrazové body mají vyšší hodnotu, než je hodnota černého pozadí.

V programu se pro přehlednost vytváří okolo každého nalezeného objektu zelený rámeček, u kterého je popisek s naměřeným průměrem barevného tónu (hue) a rozměry. Tyto rozměry jsou uvedeny pouze jako šířka a výška daného obdélníku, ohraničujícího objekt. Ovládacím prvkem v GUI lze určit, aby se nezobrazovaly všechny nalezené objekty, ale pouze objekty určité velikosti, nebo jen ten největší. Právě největší nalezený objekt v obraze je obvykle hledaná kostička.

4.1.3 Zjišťování barvy

V této fázi zpracování je k dispozici pouze oraz hledaného objektu na černém pozadí. Lze ho tak lépe podrobit dalším procesům, jako je hodnocení rozměrů. Pro měření by bylo postačující mít pouze binární obraz objektu. Černé pozadí již je a zbytek pixelů by se snadno mohl při procházení všech obrazových bodů nastavit na bílou. Součástí zadání bylo ale i zjišťování barvy což by z binárního obrazu nešlo.

V barevném modelu HSL (Hue, Saturation, Lightness) barevný tón představuje úhlová hodnota. Výsledkem metody *getAVG_Hue,* do které vstupuje bitmapa, je celé číslo v rozsahu 0 až 359.



Obrázek 4.3: Barevné tóny rozděleny do kruhu na 360° - převzato z [8]

V ovládacím programu se nezjišťoval barevný tón pro hodnoty příliš světlé a také naopak velmi tmavé barvy. Je více způsobů jak provést transformaci do šedotónové stupnice. Výsledná hodnota bude vždy záviset na všech třech barevných složkách.

V proměnné *grayValue* je uložena vypočítaná hodnota 0 až 255, reprezentující stupeň šedi. Následuje podmínka, která určuje, pro které pixely se má spočítat barevný tón. Ten vrací metoda *getHueFromRGB.*

```
private int getAVG_Hue(Bitmap bitmap) {
…
      for (int x = 0; x < width; x++) {
            for (int y = 0; y < height; y++) {
                  myColor = bitmap.GetPixel(x, y);

            grayValueD = 0.2989 * (double)myColor.R + 0.5870 *
(double)myColor.G + 0.1140 * (double)myColor.B;

            grayValue = (int)Math.Round(grayValueD);

            if (grayValue > 10 && grayValue < 245)
```

```
                    hueValues[i] = getHueFromRGB(
(int)myColor.R, (int)myColor.G, (int)myColor.B);
                }
                i++;
            }
        }

        int sum = 0, n = 1, avg = 0;
        foreach (int item in hueValues) {
            if (item != 0) {
                sum = sum + item;
                ++;
            }
        }
        if (n != 1) n--;
        avg = sum / n;
        return avg;
}
```

Postup pro získání barevného tónu by se dal pro zjednodušení rozdělit do tří kroků:

1. nejdříve se hodnoty RGB převedou ro rozsahu 0 až 1
2. nalezne se z těchto tří složek maximální a minimální hodnota
3. v závislosti na tom, pro kterou barvu z RGB vyšla maximální hodnotase vypočítá hodnota odstínu z jednoho ze tří různých vzorců

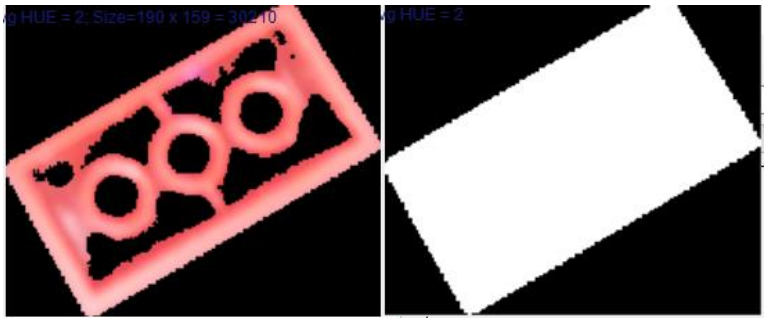
Aplikace uvedeného postupu je možné vidět v kódu níže. Ze získaného čísla udávající průměrné *hue*, lze snadno určit, o kterou barvu kostičky se jedná. Podle toho se pak vyklopí do příslušné krabičky. Vyjde-li například proměnná *avg* = 5, bude jasné, že se jedná o červený dílek (Obrázek 4.3). Stavebnice LEGO obsahuje také černé, šedé a bílé dílky. Ty budou tímto postupem nerozeznatelné. Cílové boxy jsou čtyři. Je tak možné ukládat dílky podle čtyř kritérií. Rozeznatelné barvy jsou červená, modrá, žlutá a zelená.

```
private int getHueFromRGB(int r, int g, int b) {
      float r2=r / 255.0f, g2 = g / 255.0f, b2 =b/ 255.0f;
      int max = Math.Max(r, Math.Max( g, b)), min = Math.Min(r,
Math.Min(g, b));
      float max2 = Math.Max(r2, Math.Max( g2, b2)), min2 =
Math.Min(r2, Math.Min(g2, b2));
      int hue;
      if (max == min) {
            hue = 0;
      } else {
            if (max == r) {
                  hue = (int)(60*((g2 - b2) / (max2 - min2)  ) +
(g < b ? 6.0 : 0.0));//+(g < b ? 6 : 0)
            } else if (max==g) {
                  hue = (int)(60 * (((b2 - r2) / (max2 - min2)) +
2.0));
            } else {
```

```
                hue = (int)(60 * (((r2 - g2) / (max2 - min2)) +
4.0));
            }
        }
        return hue;
}
```

4.1.4 Detekce velikosti

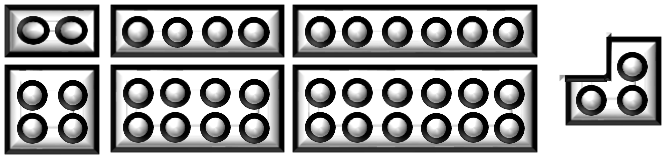
Pro korektní spočítání všech pixelů tvořící objekt bylo nutné provést vyplnění děr, které vznikly při odstraňování pozadí pomocí prahování (Obrázek 4.4). Tyto díry by se daly definovat jako plochy v objektu, které mají černou barvu a jejich obvod tvoří nečerné obrazové body. Zajišťuje to funkce *FillHoles()* z *AForge.Imaging.dll*. Při té příležitosti se provedl i převod na binární obraz.



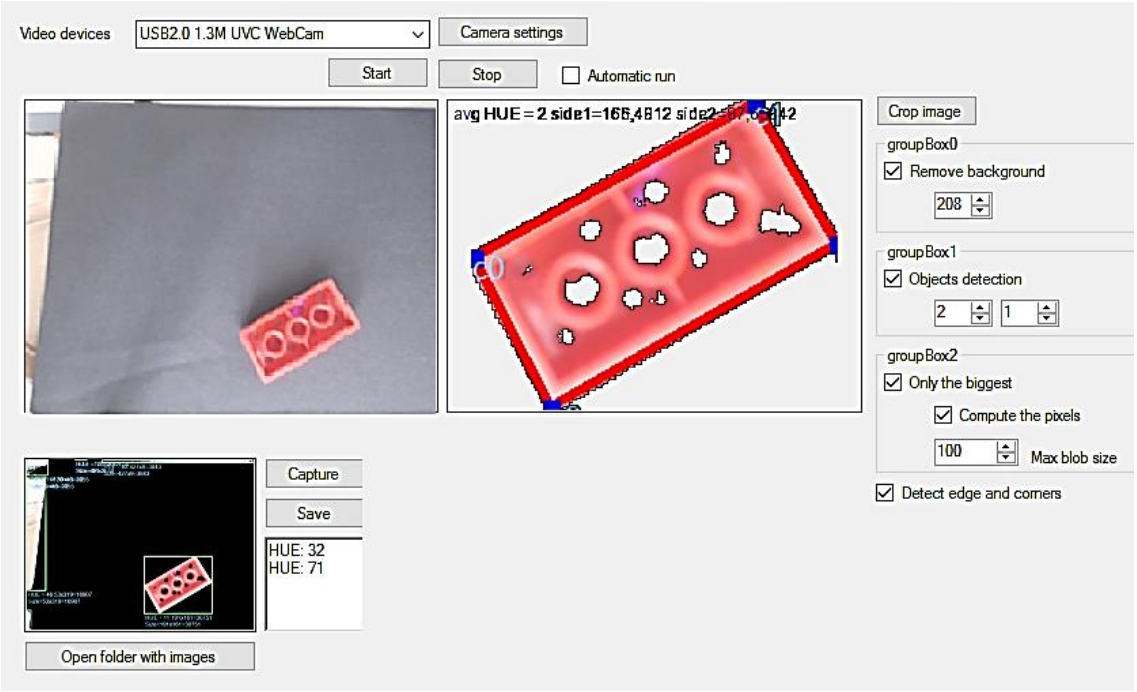
Obrázek 4.4: Vytvořený binární obraz

Nyní z hotového obrazu, který je tvořen pouze dvěma barvami, lze spočítat všechny obrazové body tvořící dílek lega, jak je vidět v kódu níže. Tato metoda by mohla selhávat pouze u dílků velikosti 1×4 a 2×2, které by měli mít stejný obsah.

```
private int ComputePixels(Bitmap bitmap) {
      var filterCountPix = new FiltersSequence(
Grayscale.CommonAlgorithms.BT709, new Threshold(2), new
FillHoles());
      Bitmap bitmapCountPix = filterCountPix.Apply(bitmap);
      int nPixels = 0;
      for (int xx = 0; xx < bitmapCountPix.Width; xx++) {
            for (int yy = 0; yy < bitmapCountPix.Height; yy++) {
                  if (!bitmapCountPix.GetPixel(xx,
yy).ToArgb().Equals(Color.Black.ToArgb())) nPixels++;
            }
      }
      return nPixels;
}
```



Obrázek 4.6: Různé tvary LEGO dílků průchozí systémem



Obrázek 4.5: Kompletní GUI ovládacího programu

Závěr

Vytvořená práce pojednává o základních principech počítačového zpracování obrazu a možnostmi aplikace některých metod na konkrétní úlohy. Po získání představy o možnostech stavebnice LEGO MINDSTORMS a práci s inteligentní kostkou NXT, byl navržen vlastní třídící systém. Jednotlivé části tohoto systému jsou popsány ve třetí kapitole.

Problémy v konstrukci se z počátku objevovaly velmi nečekaně a nepravidelně, kdy různá velikost vstupních dílků zapříčiňovala zaseknutí některých pohyblivých částí. Průběžně byly ale podnikány kroky k zamezování těchto problémů. Vlivem stárnoucích baterií v NXT kostce se od určité chvíle motory nemusí vůbec pohybovat, nebo může být jejich rotace nepravidelná. Aby se motory chovaly korektně i ve chvíli slabší baterie, musí se v programu přenastavit proměnná, udávají sílu jejich rotace.

Projekt umožnil zajímavou kombinaci programování a stavbu mechatronického systému. V budoucnu by mohl být rozšířen o některé další postupy v oblasti zpracování obrazu.

Literatura

[1] HLAVÁČ, Václav a Miloš SEDLÁČEK. *Zpracování signálů a obrazů*. 2. přeprac. vyd. Praha: ČVUT, 2007, 255 s. ISBN 978-80-01-03110-0.

[2] VLACH, Jaroslav. *Metody zpracování obrazu pro časově náročné úlohy*. Liberec, 2012. Dostupné z: http://www.mujweb.cz/jvlach/Disertace_Vlach_2012.pdf. Disertační práce. Technická univerzita v Liberci. Vedoucí práce doc. Ing. Milan Kolář, CSc.

[3] FIŘT, Jaroslaf a Radek HOLOTA. Digitalizace a zpracování obrazu. In: *Digitální mikroskopie a analýza obrazu v metalografii: sborník z 1. mezinárodní konference, Plzeň, 25.9.2002*. Plzeň: Západočeská univerzita, 2002. ISBN 80-7082-917-6.

[4] HÁJOVSKÝ, Radovan, Radka PUSTKOVÁ a František KUTÁLEK. *Zpracování obrazu v měřící a řídící technice*. Vyd. 1. Ostrava: Vysoká škola báňská - Technická univerzita Ostrava, 2012. 1 DVD-ROM. ISBN 978-80-248-2596-0. Diplomová práce. Mendelova univerzita v Brně Provozně ekonomická fakulta. Vedoucí práce prof. RNDr. Ing. Jiří Šťastný, CSc.

[5] FOJTÍK, David, Jaromír ZAVADIL a Petr PODEŠVA. *Návody ke stavebnici LEGO MINDSTORMS pro týmová cvičení v předmětu výpočetní technika*. Ostrava, 2010.

[6] RINDERKNECHT, Mike. Tutorial for Programming the LEGO® MINDSTORMS™ NXT. University of Zurich, Department of Informatics, Artificial Intelligence Laboratory.

[7] C Sharp. In: Wikipedia [online]. Wikimedia Foundation, 2001- [cit. 2016-08-10]. Dostupné z: https://cs.wikipedia.org/wiki/C_Sharp

[8] Light and Color. Lode Vandevenne [online]. 2004 [cit. 2016-08-14]. Dostupné z: http://lodev.org/cgtutor/color.html