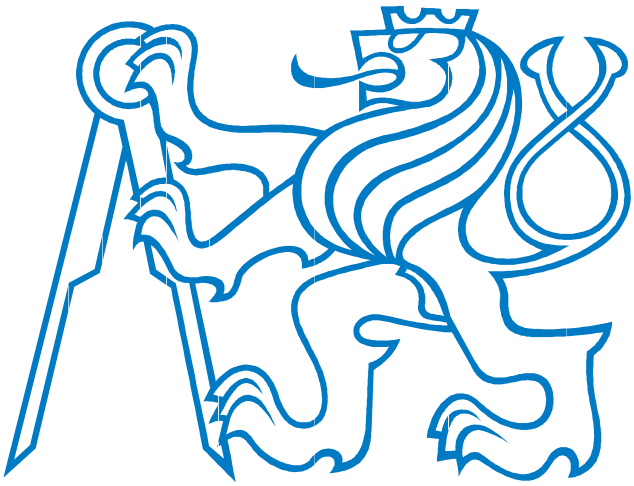
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA ELEKTROTECHNICKÁ
KATEDRA MĚŘENÍ



BAKALÁŘSKÁ PRÁCE

Zpracování obrazu pro sledování optické stopy

Daniel Toms

Vedoucí práce: doc. Ing. Jan Fischer, CSc.

Studijní program: Kybernetika a robotika

Studijní obor: Senzory a přístrojová technika

Praha 2014

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
Katedra měření

Akademický rok **2013-14**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Daniel Toms**

Studijní program: **Kybernetika a robotika**
Obor: **Senzory a přístrojová technika**

Název tématu česky: **Zpracování obrazu pro sledování optické stopy**

Název tématu anglicky: **Image Processing for an Optical Spot Tracking**

Pokyny pro vypracování:

Navrhněte a do modulu STM32F4Discovery implementujte minimalizované metody rychlého zpracování signálu obrazových senzorů pro sledování polohy a velikosti plochy světelných stop tak, aby i při velkém rozlišení snímku bylo možno vystačit pouze s omezenou velikostí vnitřní paměti SRAM v STM32F407. Posuďte možnost využití modulu STM32F429 Disco s rozšířenou pamětí SDRAM a v kladném případě jej také využijte. Prověřte možnost realizace demonstrační aplikace pro zpětnovazební elektromechanickou regulaci polohy světelné stopy tak, aby se nacházela v definovaném, případně markerem označeném místě obrazového pole kamery.

Seznam odborné literatury:

[1] RM0090 reference Manual, STMicroelectronics, 2013; www.st.com
[2] Dokoupil, V.: Bakalářská práce, ČVUT - FEL, 2013
[3] Yiu J.: The definitive Guide to the ARM Cortex- M3. Elsevier 2007, ISBN: 978-0-7506-8534-4

Vedoucí bakalářské práce: doc. Ing. Jan Fischer, CSc.

Datum zadání bakalářské práce: 25. listopadu 2013

Platnost zadání do[1]: 30. ledna 2015

Prof. Ing. Vladimír Haasz, CSc.
vedoucí katedry

L.S.

Prof. Ing. Pavel Ripka, CSc.
děkan

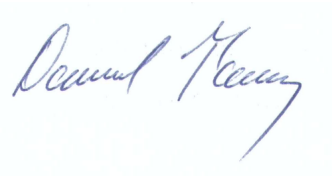
V Praze dne 25. 11. 2013

[1] Platnost zadání je omezena na dobu dvou následujících semestrů.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Tato práce vznikla v laboratoři videometrie katedry měření ČVUT FEL v Praze pod vedením doc. Ing. Jana Fischera, CSc. Navazuje též na výzkum v rámci MSM6840770015 – „Výzkum metod a systémů pro měření fyzikálních veličin a zpracování naměřených dat“ a další práce řečené v laboratoři videometrie, jejichž některé poznatky a výstupy v oblasti aplikace optoelektronických sensorů také využívá.

Datum: 22. 5. 2014 Podpis: 

Poděkování

Děkuji své rodině za duševní a finanční podporu, které se mi v průběhu vytváření této práce i v průběhu celého studia dostalo.

Dále děkuji vedoucímu bakalářské práce doc. Ing. Janu Fischerovi, CSc. za cenné rady, podklady a konzultace, které mi pomohly při vypracování této práce.

Abstrakt (Česky)

Cílem této práce je navrhnout a implementovat algoritmy zpracování obrazu s využitím kitu založeného na mikrořadiči (dále jen MCU) s jádrem ARM Cortex-M4 a monochromatickém WVGA CMOS sensoru Aptina MT9V034. Protože je požadováno zpracování dat v nativním rozlišení sensoru, není možné data vzhledem k omezené velikosti paměti použitého kitu zpracovávat off-line, ale je nutné je zpracovávat průběžně tak, jak je sensor poskytuje. Toto klade nároky na rychlost MCU, ale jak se ukazuje, výkon použitých MCU je dostatečný pro základní algoritmy předzpracování a zpracování obrazu při zachování použitelné snímkové frekvence. V další části je existující hardware a software přizpůsoben pro kit F429 Discovery, mezi jehož hlavní přednosti patří přítomnost 8MB paměti SDRAM a TFT LCD displeje. Nakonec je s pomocí kitu F429 realizována demonstrační aplikace pro sledování laserové stopy s využitím krokového motoru.

Abstract (English)

This thesis focuses on design and development of image processing algorithms using an ARM Cortex–M4-based evaluation kit, together with an Aptina MT9V034 monochrome WVGA CMOS sensor. Since a full-resolution operation is desired, off-line image processing is not possible because of insufficient memory size. It is therefore necessary to process the data on-line, i.e. as soon as they are available. This puts a strain on the used microcontroller (MCU), but as it turns out, the used MCUs are fast enough to allow basic on-line image processing with reasonable framerates. In the next part of this work, parts of existing hardware are ported to accommodate the F429-Discovery kit, which contains 8MBytes of on-board SDRAM memory, and a TFT LCD display. Finally, a demo application for optical tracking is implemented using the F429-Discovery kit, two laser modules, and a stepper motor.

Obsah

Seznam obrázků

Seznam tabulek

1. Úvod

Vzhledem ke zvyšujícím se výkonům mikrořadičů (kdy například v této práci použitý mikroprocesor STM32F407VGT6 běží na frekvenci 168MHz a poskytuje výkon 210DMIPS) je reálné uvažovat o využití těchto mikrokontrolerů pro aplikace zpracování obrazu. Použité mikrokontroléry navíc obsahují rozhraní DCMI – Digital Camera Interface, které je pro tyto účely přímo určeno. Již přítomnost tohoto rozhraní je náznakem, že výrobci těchto MCU s takovým využitím počítají.

Na druhou stranu ovšem podobné mikrokontroléry obvykle obsahují omezené množství uživatelsky dostupné paměti RAM, která nepostačuje k uložení celého obrazu v plném rozlišení. Chceme-li využít sensoru s vyšším rozlišením, musíme buď zredukovat velikost přijímaného obrazu (snížit rozlišení), nebo obrazová data zpracovávat průběžně tak, jak nám je sensor posílá.

Cílem této práce proto bude analyzovat, zda existuje možnost, jak data zpracovávat průběžně, a zda na to mají použité mikrokontroléry dostatek výpočetní síly. Jako první však budou implementovány metody pro zpracování redukovaného obrazu, který lze celý uložit do uživatelské paměti a zpracovávat prakticky neomezeně dlouho. Nebude tedy příliš záležet na rychlosti algoritmu. Úkolem této části práce bude seznámit se s programováním algoritmů pro zpracování obrazu, přičemž výhodou bude neomezeně pomalé krokování programu a obecně snadnější debug.

Hlavní náplní bakalářské práce je ale především analýza druhé možnosti, tj. průběžného zpracování dat v plném rozlišení. Pro realizaci průběžného zpracování obrazu bude využit režim zápisu do paměti, kdy dochází ke střídavému ukládání částí obrazu do dvou paměťových lokací. Důležité bude zpracovat data v jedné lokaci dříve, než se druhá lokace naplní novými daty. To bude klást nároky na rychlost mikrokontroléru a na rychlost a optimalizaci použitého algoritmu.

Tato práce vychází z několika prací vytvořených na katedře měření ČVUT již dříve, jmenovitě práce T. Pavlíčka[2], která se zabývala návrhem algoritmů pro průběžné zpracování obrazu, dále práce A. Michálka[1], která se zabývala implementací těchto algoritmů pro relativně výkonný signálový procesor ADSP-BF533 „Blackfin“, a nakonec práce V. Dokoupila[3], která se zabývala zpracováním obrazu relativně nevýkonným procesorem STM32F4 s jádrem ARM Cortex-M4, avšak pouze v režimu sníženého rozlišení. Tato práce přímo navazuje na práci [3], jejíž hardware zároveň částečně využívá.

Cílem této práce je implementovat a optimalizovat metody strojového vidění tak, aby byly i při použití relativně slabého mikrokontroléru s omezenou pamětí využitelné pro zpracování obrazu ve vysokém rozlišení, a tyto metody využít k vytvoření jednoduché demonstrační aplikace pro sledování optické stopy.

2. Úvod do problematiky zpracování obrazu mikrokontrolérem

Jak již bylo řečeno v úvodu, cílem této práce bude návrh algoritmu pro průběžné zpracování obrazu, snímaného sensorem Aptina MT9V034 a vyhodnocovaného kitem STM32F4-Discovery, který je založen na mikrokontroléru ARM Cortex-M4F, konkrétně STM32F407VGT6 společnosti STMicroelectronics, a který má omezenou kapacitu vnitřní paměti, jež nepostačuje k uložení celého snímku. Vzhledem k tomu, že jsem před započetím této práce neměl žádné předchozí zkušenosti s programováním procesorů architektury ARM, bude pro seznámení s vývojovým kitem vytvořeno několik vzorových programů v jazyce C, které budou demonstrovat funkčnosti vybraných periferií MCU (LED, tlačítka, časovače a další).

Po základním seznámení s možnostmi mikrokontroléru bude implementováno několik základních metod pro zpracování obrazu, kdy především metoda tzv. „labellingu“ slibuje slušné výsledky při rozumné výpočetní náročnosti, a zároveň je vzhledem k rozšířenosti této metody popsáno a vyzkoušeno mnoho metod pro její časovou optimalizaci. Dále bude implementována metoda pro korekci vinětace, což je optická chyba, která se u použitého objektivu výrazně projevuje a ovlivňuje tak schopnosti algoritmů rozeznávat objekty v obraze.

Obraz sejmutý v plném rozlišení senzoru MT9V034, tj. 752x480px, má větší velikost než na kterou dostačuje paměťový prostor na kitu F4-Discovery, což znamená, že při zpracovávání obrazu v plném rozlišení není možné celý obraz uložit do paměti a zpracovávat jej tam. Důsledkem tohoto faktu je nutnost načítat obraz po částech a zpracovávat rovnou. Bude vyzkoušeno využití DMA řadiče v režimu dvojitého bufferingu, kdy jsou data po částech střídavě ukládána do dvou paměťových lokací, ve kterých jsou tato data po omezenou dobu, tj. než jsou přepsána daty novými, dostupná pro zpracování. Tento přístup klade vyšší nároky na efektivitu vybraného algoritmu a na rychlost samotného MCU. Pro snadné zobrazení dat a řízení mikrokontroléru bude rovněž vytvořena PC aplikace pro komunikaci a řízení MCU a navržen jednoduchý systém pro výměnu dat mezi MCU a nadřazeným počítačem.

V další části bude již naprogramovaný a vyzkoušený kód přizpůsoben pro kit STM32F429-Discovery, který byl uveden do prodeje v druhé polovině roku 2013, a který nese mimo jiné mírně výkonnější mikrokontrolér, dotykový LCD displej, ale především 8 megabytů paměti typu SDRAM. Právě tato paměť může být využita pro uložení celého obrazu, se kterým pak může být nakládáno po neomezeně dlouhou dobu. Rozložení vývodů GPIO kitu F429-Discovery bohužel není totožné s kitem F4-Discovery, a tak nebude možné použít již vytvořený hardware a bude třeba sestrojit novou dceřinou desku pro spojení Discovery kitu a CMOS modulu.

V poslední části práce bude ke kitu F429 připojen obvod pro řízení krokového motoru a bude vytvořena jednoduchá demonstrační aplikace využívající naprogramované metody rozpoznávání obrazu ke sledování optické stopy laserem připevněným ke krokovému motoru.

3. Použitý hardware

Jak vyplývá ze zadání práce, řešení práce má být orientováno na využití vývojových kitů založených na procesorech s jádry ARM Cortex-M4, což jsou 32bitové RISC ARM procesory, které se vyznačují slušným výkonem při nízké spotřebě a nízké pořizovací ceně. Pro tuto práci budou využity kity STM32F4-Discovery a STM32F429-Discovery, uvedu proto stručně jejich vlastnosti:

3.1. Kit STM32F4-Discovery

První použitý kit je založen na jednojádrovém 32bitovém procesoru STM32F407VGT6 společnosti STMicroelectronics, s jádrem ARM Cortex-M4F, pamětí Flash o velikosti 1MB a uživatelskou pamětí RAM o velikosti 192kB. Frekvence hodinového signálu procesoru je 168MHz, výkon je 210DMIPS. Procesor obsahuje jednotku FPU (floating point unit) pro práci s čísly s plovoucí desetinnou čárkou.

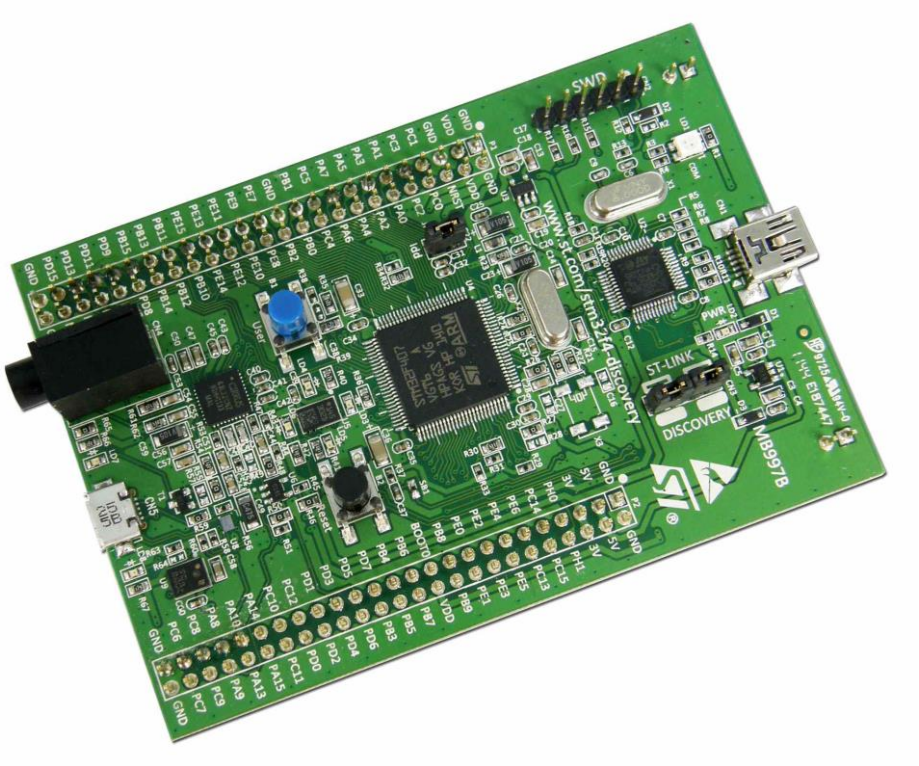
Výběr tohoto kitu byl dán:

- Kompaktností zařízení – v kitu jsou integrovány základní periferie (tlačítka, LED), datový konektor USB, HW debugging (ST-Link/V2 in-circuit debugger)
- Podporou rozhraní DCMI ze strany mikrokontroléru
- Velmi příznivou cenou (330kč, farnell.com) a slušným výkonem (v porovnání s ostatními procesory řad Cortex-M)
- Kvalitní SW podporou (knihovny, ovladače, IDE) pro programování mikrokontroléru

Kit lze napájet z rozhraní USB nebo externě ze zdroje stejnosměrného napětí 5V. Kit samotný pak dokáže napájet připojené periferie na 5V a 3V při maximálním proudovém odběru 100mA.

Kit obsahuje debugger ST-LINK/V2, kterým lze programovat a debugovat program a nese rovněž konektory pro Serial Wire Debug (SWD). Součástí kitu je množství periferií, jmenovitě:

- jedno uživatelské tlačítko, jedno RESET tlačítko, čtyři uživatelské LED diody
- tříosý MEMS akcelerometr LIS302DL
- Audio jack a 24bit audio DAC CS43L22
- MEMS všesměrový digitální mikrofon MP45DT02
- řadič USB OTG s konektorem USB microAB – pouze v režimu Full-speed
- dvě řady headerů s konektory, na které jsou vyvedeny kontakty pouzdra MCU



Obr. 1 - Kit STMF32F4 Discovery

3.2. Kit STM32F429i-Discovery

Druhým použitým kitem je nová verze kitu Discovery, postavená na jednojádrovém 32bitovém procesoru STM32F429ZIT6 ve 144pinovém pouzdru LQFP144, jejíž hlavní přednosti a rozdíly oproti verzi STM32F4 jsou:

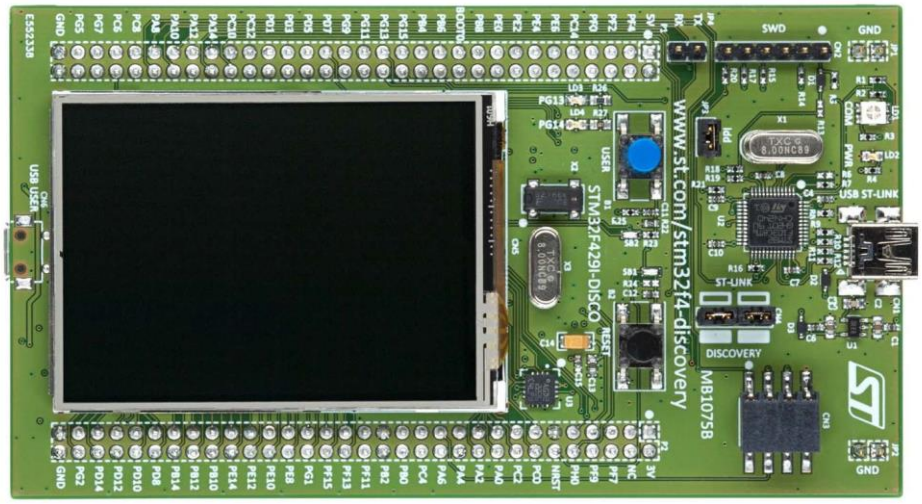
- Vyšší frekvence MCU (180MHz oproti 168MHz u F4 Discovery)
- Větší kapacita vnitřní pamětí
  - Flash: 2MB (F4: 1MB)
  - RAM: 256kB (F4: 192kB)
- Větší pouzdro se 144 piny (F4: 100 pinů)
- 64Mbits (8MBytes) vestavěné paměti typu SDRAM (u F4 zcela chybí)
- 2.4‘ QVGA (320x240px) barevný dotykový TFT LCD displej (u F4 chybí)

Kit dále obsahuje většinu periferií, které jsou přítomny na kitu F4 Discovery (tříosý MEMS akcelerometr, dvě uživatelsky přístupné LED diody, řadič USB OTG, Serial Wire Debug, headery s konektory a další).

Nevýhodou kitu je jiné rozložení GPIO headerů pro připojení dalších periferií, což znamená, že není možné pro spojení kitu a senzorové desky využít propojovací desku z práce [3]. Dalšími nevýhodami kitu jsou jeho téměř dvojnásobná pořizovací cena (580kč, farnell.com) a absence

Audio jacku a audio DAC (což však při předpokládaném využití kitu pro zpracování obrazu nevadí).

V rámci práce bude snaha využít zvýšené rychlosti procesoru a rovněž vestavěné paměti typu SDRAM, čímž odpadne nutnost co nejrychlejšího zpracování obrazu, a rovněž to umožní přenést celý obraz v plném rozlišení do počítače (což u kitu F4 není možné vzhledem k nízké rychlosti rozhraní PC<->Discovery a malé kapacitě vnitřní paměti, která neumožňuje obrázek uložit celý a odeslat jej až po dokončení snímání).

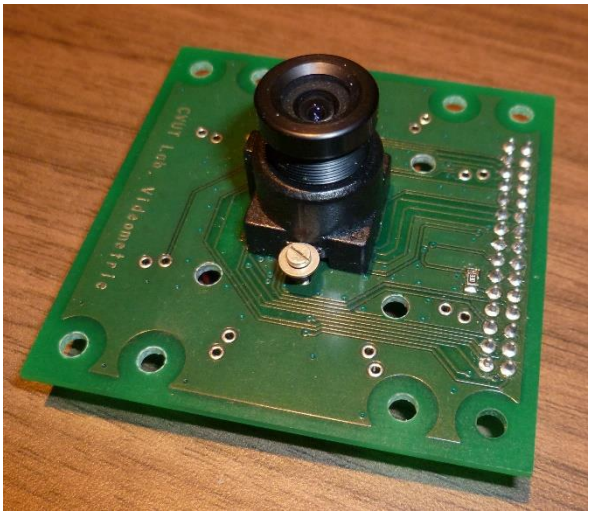


Obr. 2 - Kit STM32F429 Discovery

3.3. Obrazový CMOS sensor Aptina MT9V034

Pro snímání obrazu byl zvolen sensor Aptina MT9V034, což je monochromatický CMOS sensor s rozlišením 752*480 (WVGA) o maximální frekvenci hodinového signálu 27MHz a maximální snímkovací frekvenci 60sn./s. Použitý objektiv má ohniskovou vzdálenost f = 3.6 mm.

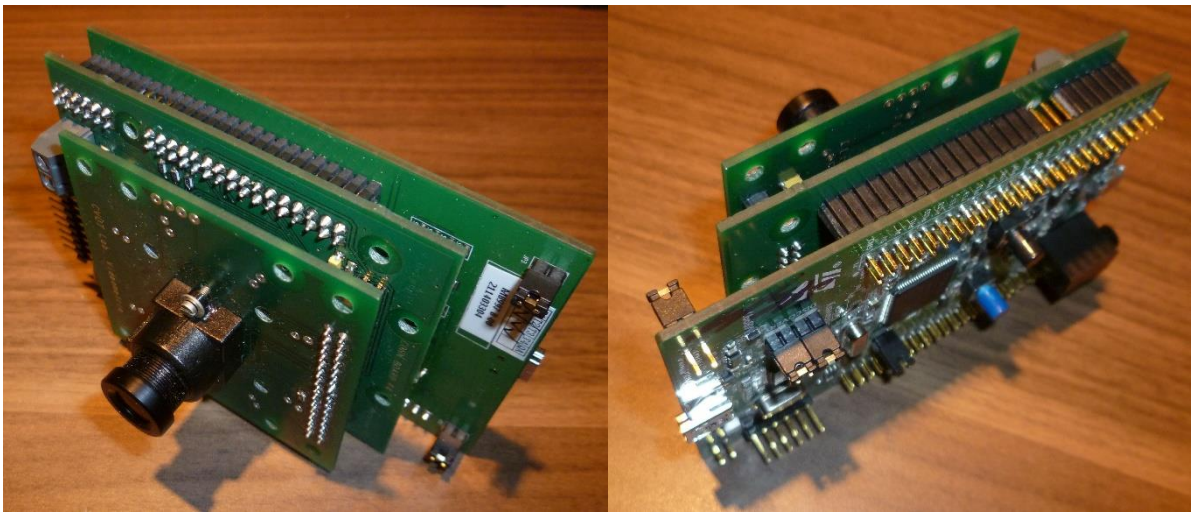
Vzhledem k tomu, že katedra měření má z dřívějšího vývoje dostupný modul včetně PCB využívající právě sensor Aptina MT9V034, byl zvolen právě tento sensor. Byla zvolena monochromatická verze sensoru, neboť předpokládaná aplikace tohoto sensoru v algoritmech průběžného zpracování obrazu by utrpěla zvýšenou výpočetní náročností práce s barevnými daty a zároveň by taková volba nenesla žádný přínos.



Obr. 3 - Obrazový sensor Aptina s nasazeným objektivem

Přenos obrazu ze sensoru probíhá pomocí rozhraní DCMI, k řízení a nastavení sensoru se využívá sběrnice I^2C. DCMI je synchronní paralelní rozhraní přítomné v procesoru STM32F4, určené pro práci s digitálními senzory, které podporuje DMA, a které je schopno číst data ze sensoru rychlostí až 54MB/s. DCMI podporuje až 14bitový paralelní datový tok, tzv. CROP feature (zmenšení obrazu oříznutím), volbu mezi jednorázovým a kontinuálním snímáním, a další.

Vedle samotného kitu Discovery a modulu s CMOS sensorem je k dispozici ještě propojovací modul, který zajišťuje spojení těchto dvou komponent, a obsahuje rovněž dostatečně proudově dimenzovaný napěťový regulátor 5V -> 3.3V, potřebný pro napájení CMOS sensoru. Tento propojovací modul byl vytvořen v rámci bakalářské práce V. Dokoupila z roku 2013 [3] a poskytuje kompaktní propojení celého modulu.



Obr. 4 – Kompletní modul

Tabulka popisující propojení sensoru CMOS a kitu Discovery je uvedena níže:

Signál modulu CMOS sensoru	Pin Discovery kitu
D0	-
D1	-
D2	PC6
D3	PC7
D4	PC8
D5	PC9
D6	PE4
D7	PB6
D8	PE5
D9	PE6
PIXCLK	PA6
SYSCLK	PB0
HSYNC	PA4
VSYNC	PB7
STANDBY	PA3
RESET	PA2
OE (Output Enable)	PC2
EXPOSURE	PA1
I2C_SDA	PB9
I2C_SCLK	PB8
LEDOUT	PC1

Tab. 1 - Propojení sensoru CMOS a kitu F4-Discovery

- Signály SCLK a SDA náleží k rozhraní I^2C (na straně mikrokontroléru konkrétně I^2C2)
- Signály D0-D9, PIXCLK, HSYNC, VSYNC náleží k rozhraní DCMI
- Zbývající signály slouží k řízení CMOS sensoru a jeho napájení

Protože je využíváno 8 datových signálů DCMI, nejsou dolní dva signály D0 a D1 zapojeny a data prochází pouze přes D2-D9. Propojovací deska dále podporuje připojení LED displeje na dalších pinech, tato funkcionalita ale nebude využita.

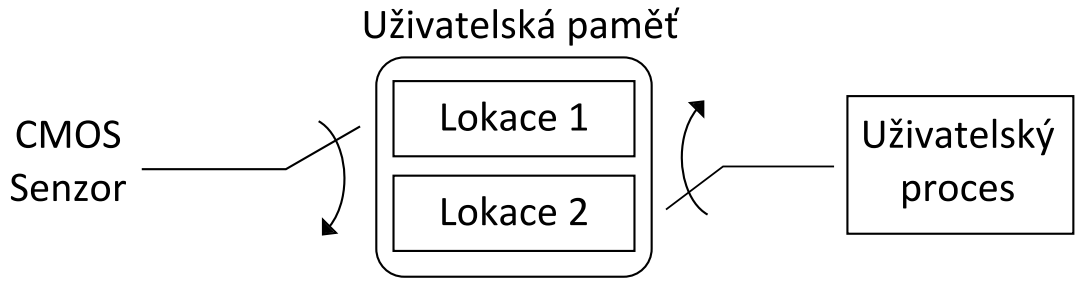
4. Snímání obrazu kitem s omezenou pamětí

Hlavním problémem, kterému čelí tato práce, je fakt, že použité mikrokontroléry mají velmi omezenou kapacitu vnitřní paměti RAM, která nedostačuje pro uložení celého snímku. Jedním z možných řešení je omezit rozlišení vstupního snímku tak, aby se tento snímek vešel do paměti mikrokontroléru. Nevýhodou tohoto přístupu je, že se ztrácí obrazová informace a drobnější objekty nemusí být v obrazu vůbec detekovány. Druhou variantou je zpracovávat data již

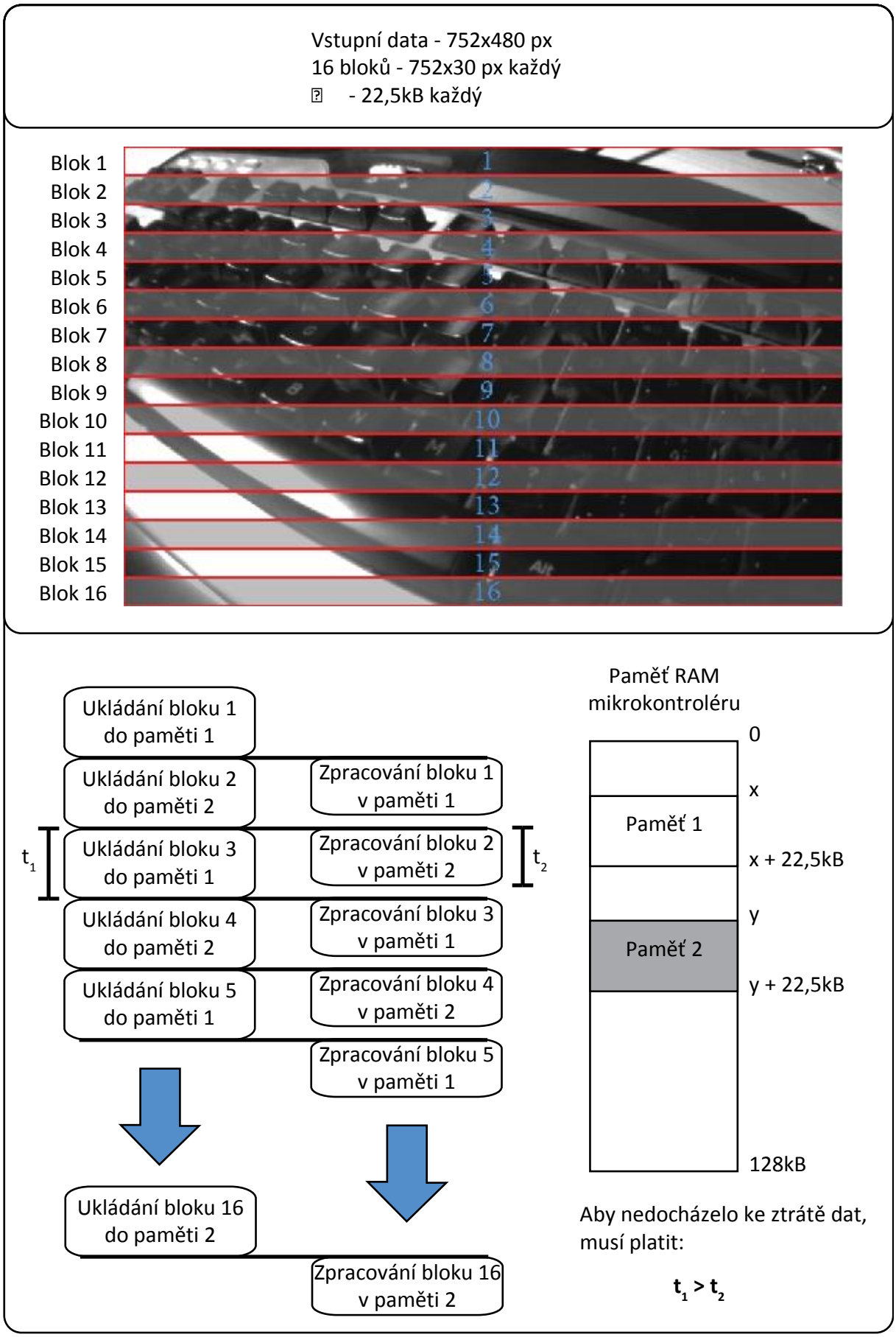
v průběhu snímání a ihned po zpracování je zahazovat. Tím se zamezí ztrátě obrazových informací, ke které by došlo v důsledku snížení rozlišení. Nevýhodou ale je, že data jsou k dispozici jen po omezenou dobu a pak jsou nenávratně ztracena. Samotný algoritmus zpracování obrazu tedy musí být rychlejší než samotné snímání, což v případě použití relativně pomalého mikrokontroléru dává prostor pouze pro relativně nenáročné a dobře optimalizované algoritmy.

Nejjednodušší možností je zpracovávat obraz pixel po pixelu tak, jak je posílán ze sensoru. V případě použití této nejzákladnější možnosti bude mít použitý algoritmus přesně daný a neměnný čas na zpracování jednoho pixelu, což není úplně ideální řešení. Druhá varianta je načítat obraz po větších částech, tedy například vždy po několika řádcích, což použitému algoritmu umožňuje efektivněji pracovat s dostupným časem. Doba, po kterou v této práci použité algoritmy zpracovávají „prázdný“ pixel, tj. pixel, ve kterém není nalezen žádný objekt, je totiž zpravidla kratší, než doba zpracování „objektového“ pixelu, u kterého je většinou nutné nějak naložit se získanou informací. Načítání po větších částech tak mikrokontroléru umožní vytvořit si na „prázdných“ pixelech časovou rezervu, kterou pak může využít pro zpracování „objektových“ pixelů. To samozřejmě funguje, pouze pokud je poměr „prázdných“ a „objektových“ pixelů dostatečně malý. Metoda se tak jeví ideální například pro detekci laserových stop, které jsou snadno detekovatelné, neboť jejich jas ve většině případů výrazně převyšuje jas okolí, ale zároveň jsou plošně velmi malé, což napomáhá rychlosti algoritmu.

Jednou z možností, jak obraz načítat po částech, je střídavý zápis do dvou paměťových bloků umístěných v paměti RAM procesoru tak, že zatímco budou data zapisována do prvního bloku, může algoritmus zpracovávat data v bloku druhém. Tyto dva bloky by měly ideálně být co největší (ale tak, aby se vešly do paměti mikrokontroléru), abychom použitému algoritmu poskytli co největší časovou volnost. K tomuto účelu lze s výhodou využít DMA řadič, který je dnes přítomný na prakticky všech mikroprocesorech s výjimkou těch nejzákladnějších a těch, které jsou určeny pro velmi specifické aplikace.



Obr. 5 - Ilustrace funkce dvojitého bufferingu řadiče DMA



Obr. 6 - Ilustrace načítání obrazu po částech

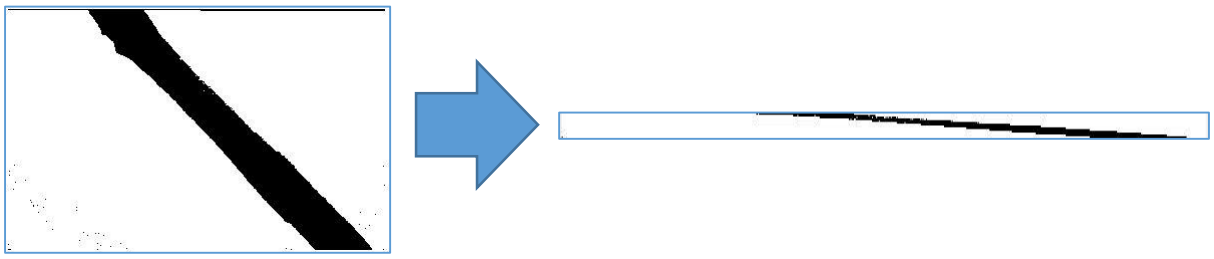
4.1. Režim dvojitého bufferingu řadiče DMA

DMA řadič je periferie, která slouží k přesunu dat z jedné lokace do druhé, tedy například z datového výstupu CMOS sensoru do paměti RAM, a to zcela bez asistence mikrokontroléru, což uživateli umožňuje využít výkon mikrokontroléru ke zpracování obrazu a neplýtvat výkonem na řízení toku dat. Pro načítání obrazu po větších částech lze využít například právě režim dvojitého bufferingu řadiče DMA. Principem dvojitého bufferingu je střídavé zapisování do dvou paměťových lokací, přičemž DMA řadič není nutné přeprogramovávat při každém střídání cílové paměťové lokace. Celá tato činnost probíhá bez asistence mikrokontroléru.

Černobílý obraz v plném rozlišení 752x480 px má velikost přibližně 361kB (752x480x8 bitů), a nevejde se tedy do paměti RAM mikrokontroléru (která má velikost 192kB, z čehož 64kB je navíc typu Core-Coupled Memory, která se pro DMA nedá použít). S využitím double bufferingu je možné obraz načítat „po částech“ (například po částech o velikosti 752x30px, každá o velikosti zhruba 22,5kB) střídavě do dvou paměťových lokací (celkem je tedy potřeba 45kB paměťového prostoru).

Je ovšem nutné data v jedné paměťové lokaci zpracovat dříve, než DMA dokončí zápis do druhé paměťové lokace a začne zapisovat opět do první lokace. Zde je tedy kritická především rychlost mikrokontroléru, společně s rychlostí použitého algoritmu. Výhodou je možnost zpomalit hodinový signál sensoru ze základní rychlosti 27MHz až na pouhé 3MHz, a to bez viditelné změny v kvalitě snímaného obrazu. Také lze zvýšit hodnoty horizontal/vertical blankingu (zatmívacích intervalů) senzoru, čímž se snímání dále zpomalí. Takové úpravy mají pochopitelně nepříznivý vliv na počet snímků za vteřinu.

Ověření správné funkčnosti double buffer DMA jsem provedl snímáním předem známého objektu (diagonální čáry přes celý obraz), kdy jsem z každé oblasti o velikosti 752*30px zkopíroval jeden řádek o velikosti 752px. Výsledný obrázek je tedy na výšku 30x menší, ale graficky znázorňuje správnou funkčnost DMA – pokud by DMA bylo špatně nastaveno, nebo pokud by algoritmus nestíhal zpracovávat data dodávaná sensorem, došlo by k deformaci výsledného obrazu, tedy například k promíchání jednotlivých řádků obrazu. Výsledným obrázkem by pak nebyla souvislá čára jako na obr. 7. K otestování jsem stvořil jednoduchou aplikaci schopnou výsledný obraz, zaslaný do počítače přes Virtual COM Port, zobrazit pro vizuální kontrolu uživatelem.



Obr. 7 - Ilustrace funkčnosti DMA v režimu double buffer na vzorových datech

5. Metody zpracování obrazu pro detekci objektů v zorném poli

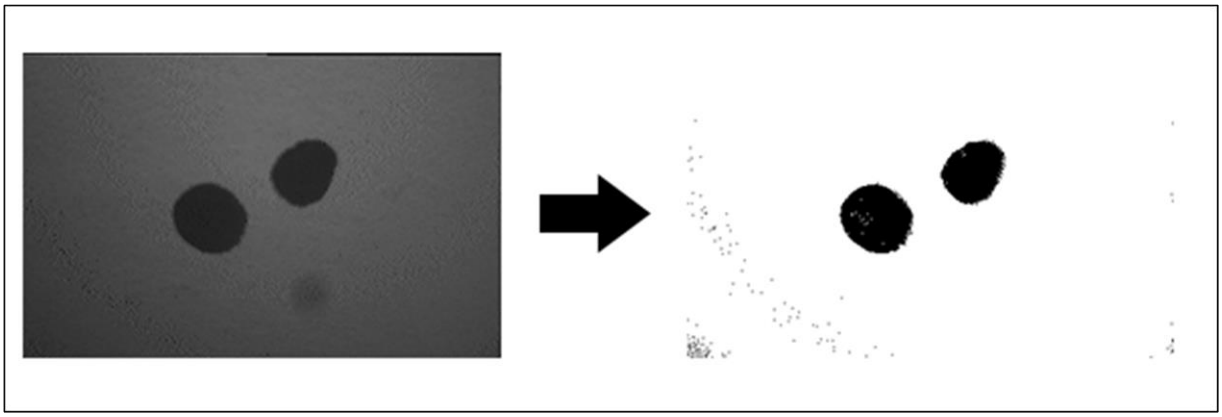
Jedním z cílů této práce je detekce světelné stopy v obraze sejmutém připojenou kamerou. Aby bylo vůbec možné získat z obrazu užitečnou informaci (například polohu objektů), musí obraz projít zpracováním pomocí algoritmů strojového vidění. V této kapitole bude proto rozebráno a implementováno několik algoritmů obecně sloužících k detekci objektů v obrazu.

Jako první krok je nutné odlišit všechny objekty od pozadí, a rovněž vzájemné odlišit jednotlivé objekty, je-li jich více (obecně je počítáno s tím, že tomu tak je). Pro potřeby průmyslového využití lze předpokládat, že snímaná scéna je předem definovaným způsobem nasvětlena, a objekty v obrazu se od pozadí liší svou hodnotou jasu (jasovou funkcí).

Pro odlišení objektů od pozadí lze využít metody prahování. Prahování je jedna z nejzákladnějších metod zpracování obrazu, která nám vstupní obraz ve stupních šedi (grayscale) převede na obraz binární (binary), tedy obraz složený pouze z jedniček a nul (bílá/černá, pozadí/objekt). Obraz je zpracován sekvenčně řádek po řádku, pixel po pixelu, a pro každý pixel se pouze vyhodnotí následující funkce a výsledek se zapíše do právě vyhodnocovaného pixelu.

$$h(x,y) = \begin{cases} 1 \; pro \; f(x,y) > p \\ 0 \; pro \; f(x,y) < p \end{cases}$$

V tomto případě p je prahová hodnota jasu a $f(x,y)$ je jasová funkce pixelu.



Obr. 8 - Aplikace prahování

Výhodou prahování je jeho nízká náročnost na výpočetní výkon procesoru a již z principu vyplývající vhodnost pro řádkové zpracování obrazu. Nevýhodou je nutnost určení správně prahovací hodnoty, která se mění s úrovní osvětlení scény, což obecně představuje složitý problém, a rovněž citlivost na šum v obraze a rovnoměrnost osvícení scény. Pro potřeby této práce je ovšem předpokládáno (jak již bylo řečeno), že je snímaná scéna odpovídajícím

a neměnným způsobem nasvícena, a tak není problém nastavit hodnotu prahování při prvním použití ručně.

5.1. Výpočet plochy objektu

Jednou ze základních informací, které lze z obrazu získat, je plocha detekovaných objektů. Tu lze vypočítat buď v pixelech, nebo i v jiných jednotkách (například cm^2), je-li znám převodní vztah. Velikost objektu v pixelech se spočítá jednoduše jako

$$S = \sum_{j=0}^{y} \sum_{i=0}^{x} f(i,j),$$

kde x, y jsou rozměry obrazu a $f(i,j)$ je jasová funkce, která je v tomto případě binární. Obrazová funkce má hodnotu 1, náleží-li pixel k objektu, nebo 0, náleží-li k pozadí. Pro výpočet v cm^2 je třeba výsledek vynásobit konstantou K[cm/px].

5.2. Určení polohy těžiště objektu

Poloha těžiště objektu se počítá s využitím modifikovaného fyzikálního vzorce

$$\vec{R} = \frac{1}{M} \int \vec{r}\, dm,$$

kde $\vec{R}$ je polohový vektor těžiště, M je celková hmotnost tělesa a $\vec{r}$ je obecný polohový vektor. Integrace probíhá podle hmotnostního elementu dm.

V tomto případě se jedná o diskrétní problém, integrál je nahrazen sumou (v dvourozměrném prostoru dvojící sum pro x a y), a rovněž hmotnostní element dm je nahrazen jasovou funkcí $f(i,j)$, která obecně není binární, ovšem v případě oprahovaného obrazu binární je. Výsledkem jsou vzorce

$$T_i = \frac{\sum_{j=0}^{y} \sum_{i=0}^{x} i \cdot f(i,j)}{\sum_{j=0}^{y} \sum_{i=0}^{x} f(i,j)} \qquad T_j = \frac{\sum_{j=0}^{y} \sum_{i=0}^{x} j \cdot f(i,j)}{\sum_{j=0}^{y} \sum_{i=0}^{x} f(i,j)}$$

kde T_i přestavuje souřadnici x těžiště, a T_j představuje souřadnici y těžiště, i a j jsou souřadnice právě zpracovávaného pixelu a $f(i,j)$ je již zmíněná jasová funkce.

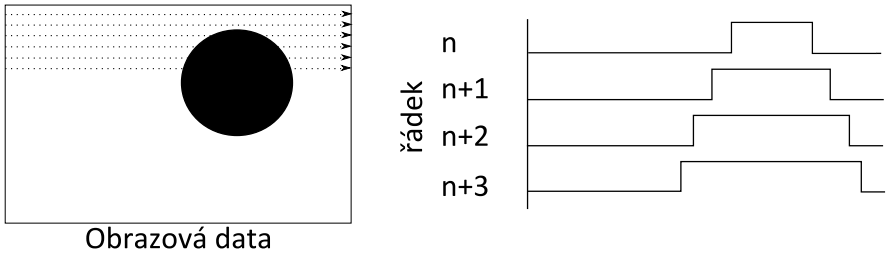
Porovnáním souřadnic těžišť jednotlivých objektu ve dvou po sobě jdoucích snímcích lze získat informaci o pohybu objektů v zorném poli kamery. Základní metody rozpoznání obrazu jako takové neumožňují v případě detekce více objektů bezpečně určit, který objekt se pohybuje kterým směrem, a tak by v případě potřeby sledovat pohyb více objektů bylo nutné využít další algoritmy, například pro odhad pohybu (motion estimation). Pak by bylo možno odhadovat, kde zhruba který objekt po změně jeho pozice hledat, a tento odhad pak propojit se skutečně

zaznamenanými daty. Ani poté však není výsledek zaručen, například v případě, že se objekty v zorném poli kamery prolnou.

5.3. Metoda překrývajících se pruhů

Metoda s pracovním názvem „Metoda překrývajících se pruhů“ je vhodná pro řádkové zpracování a spočívá ve sledování, zda se pixely objektu na právě zkoumaném řádku alespoň částečně překrývají s pixely téhož objektu na řádku minulém. Ve chvíli, kdy je vyhodnocován jeden řádek, pak pixely náležící k nějakému objektu představují na tomto řádku pruh, který má svou počáteční a koncovou souřadnici. Souřadnice tohoto pruhu se pak porovnají se souřadnicemi pruhů detekovaných na předchozím řádku a dojde-li k alespoň částečnému překrytí souřadnic, je jasné, že tyto pixely patří ke stejnému objektu. Z tohoto principu vyplývá i pracovní název metody.

Je zřejmé, že je tato metoda použitelná pouze pro objekty jednoduchých tvarů. Přesněji, na jednom řádku smí mít jeden objekt jen dvě hrany – jednu náběžnou a jednu sestupnou. Nelze tedy detekovat objekty tvaru U, X a podobné (viz obr. 9). Takové objekty jsou sice detekovány, ale jsou nesprávně interpretovány. Metoda je vysoce citlivá na špatně sejmuté obrazy a vyžaduje kvalitní vstupní data. Zásadní, a zároveň pravděpodobně jedinou, výhodou této metody je její výpočetní a prostorová nenáročnost, v paměti je totiž nutné mít uložený pouze jeden řádek obrazu a prostor pro data obsahující informace o detekovaných objektech. Při implementaci této metody jsem vycházel z poznatků v práci [2].



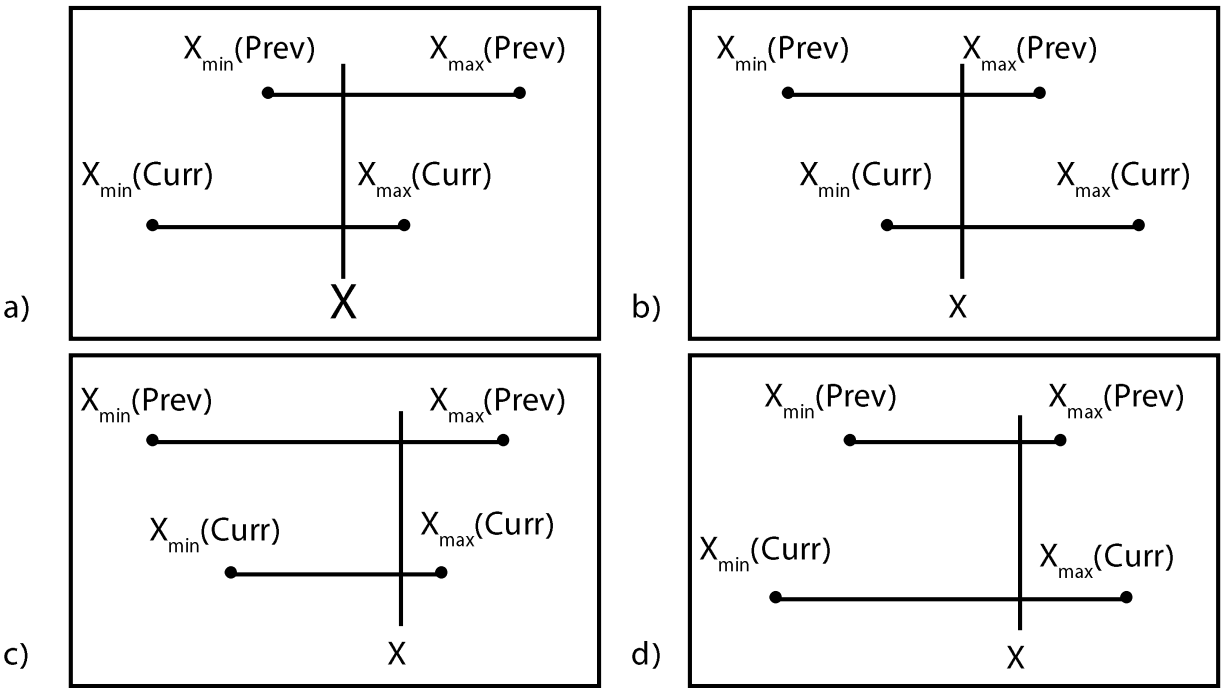
Obr. 9 - Ilustrace k metodě překrývajících se pruhů

Algoritmus metody překrývajících se pruhů se dá popsat následovně (ilustrace je na obrázku 8):

- Pro každý řádek do pracovní paměti uložit minimální a maximální hodnotu souřadnice X objektu (pruhu). Poznamenat informaci o obsahu a poloze těžiště.
- Tyto hodnoty porovnat se souřadnicemi již nalezených, ale neukončených objektů (tzn., že spodní hrana objektu ještě neprošla zpracováním).
    - Není-li nalezena shoda mezi hodnotami v pracovní paměti a právě nalezenými hodnotami X_{min} a X_{max}, jedná se o nový objekt, jehož hodnoty X_{min} a X_{max} je třeba uložit do pracovní paměti.
    - Je-li zaznamenána shoda, uložit data do paměti k danému objektu.

- Na konci řádku označit jako ukončené ty objekty v pracovní paměti, ke kterým na současném řádku již nepřibyly další pixely (tj. spodní hrana objektu již prošla zpracováním).

Při vyhodnocování překrývání pruhů mohou nastat následující situace:



Obr. 10 - Ilustrace možných situací při vyhodnocování překrývání pruhů

Pruhy se tedy překrývají, splňují-li například tyto podmínky:

a) $\{X_{min}(Prev) \leq X_{max}(Curr)\} \wedge \{X_{min}(Prev) \geq X_{min}(Curr)\}$,
b) nebo $\{X_{max}(Prev) \leq X_{max}(Curr)\} \wedge \{X_{max}(Prev) \geq X_{min}(Curr)\}$,
c) nebo $\{X_{min}(Prev) \leq X_{min}(Curr)\} \wedge \{X_{max}(Prev) \geq X_{max}(Curr)\}$,
d) nebo $\{X_{min}(Prev) \geq X_{min}(Curr)\} \wedge \{X_{max}(Prev) \leq X_{max}(Curr)\}$.

Problém lze dále zjednodušit pouze na dva případy, neboť například případy C a A lze popsat jednou podmínkou, a případy D a B také jednou podmínkou (rovněž je možné spojit případy A a D, a případy B a C).

Výsledné podmínky tedy vypadají následovně:

a) $\{X_{max}(Curr) \leq X_{max}(Prev)\} \wedge \{X_{min}(Prev) \leq X_{max}(Curr)\}$,
b) $\{X_{max}(Prev) \leq X_{max}(Curr)\} \wedge \{X_{min}(Curr) \leq X_{max}(Prev)\}$.

Vzhledem k tomu, že je snaha získat veškerá užitečná data již při prvním průchodu obrazem, bude vypočten rovnou i obsah a těžiště detekovaného pruhu.

Obsah pruhu lze „násilně“ vypočítat součtem všech pixelů náležících k detekovanému pruhu, nebo chytřeji (a výpočetně efektivněji) jako

$$S(obj) = X_{max}(Curr) - (X_{min}(Curr) - 1),$$

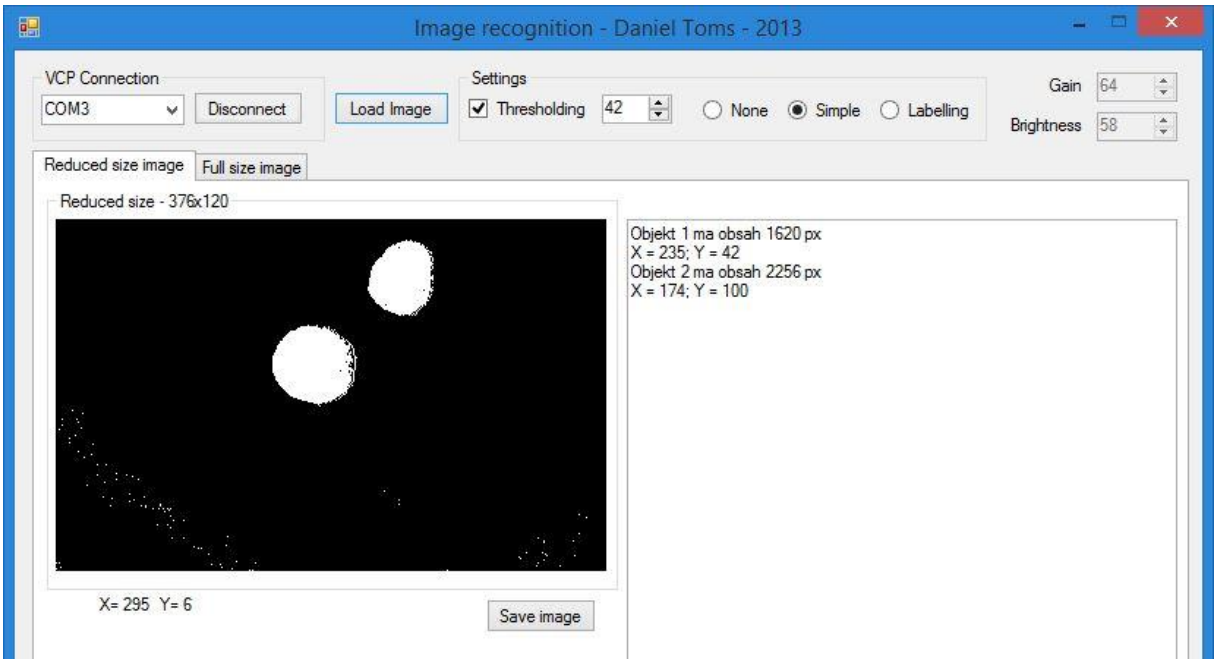
čili jako rozdíl souřadnic maxima a minima zmenšený o 1.

K výpočtu souřadnic těžiště lze využít vzorců z kap. 5.2. Suma v čitateli bude načítána v průběhu zpracování a suma ve jmenovateli odpovídá při jednotkové váze všech pixelů celkovému obsahu objektu. Sumu v čitateli lze opět získat „násilně“ součtem přes všechny pixely na daném řádku, nicméně i zde je možné výpočet urychlit – použitím aritmetické posloupnosti. Sumy v čitateli souřadnic x, y pak lze získat jako

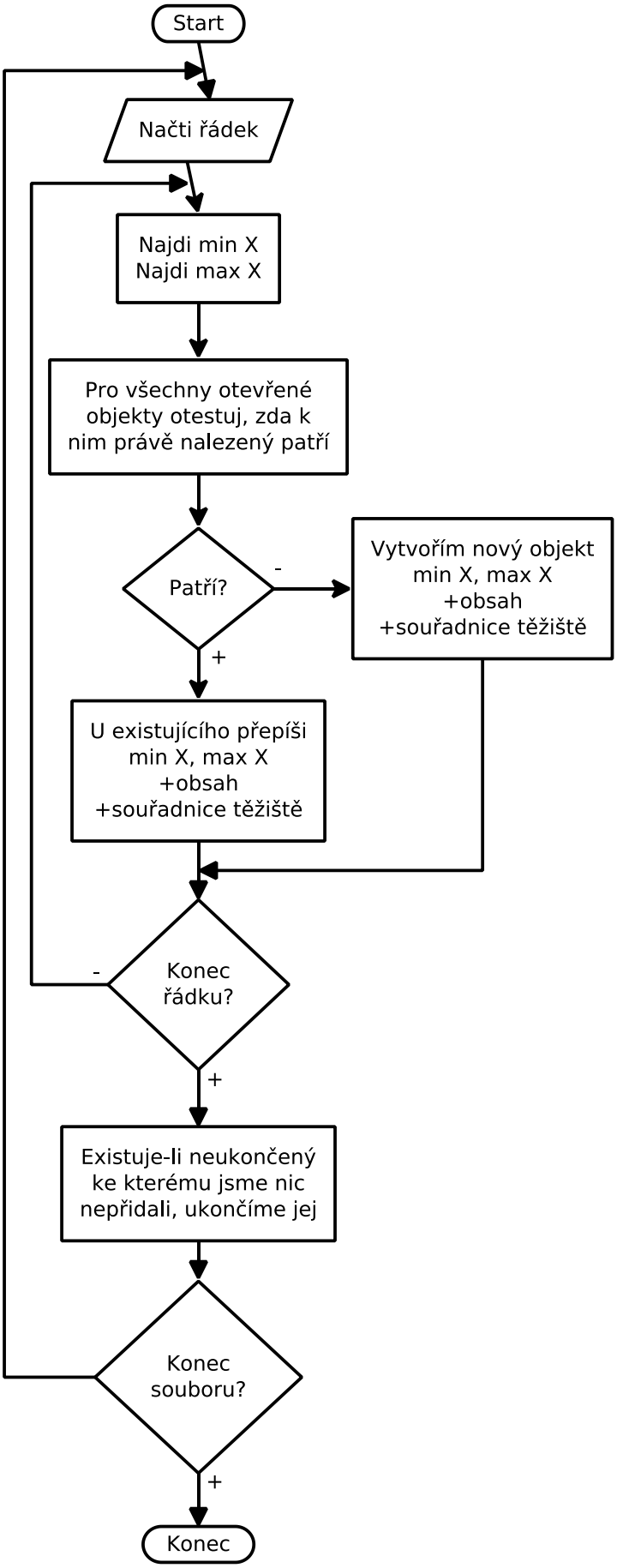
$$S_x = (X_{max} - X_{min} + 1) \cdot \left(\frac{X_{min} + X_{max}}{2}\right)$$

$$S_y = (X_{max} - X_{min} + 1) \cdot y$$

Na konci zpracování pak stačí tyto hodnoty pouze vydělit celkovým obsahem objektu a získat tak hledané souřadnice středu-těžiště objektu.



Obr. 11 - Demonstrace funkčnosti metody překrývajících se pruhů při detekci objektů



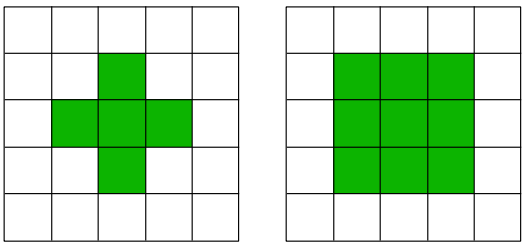
Obr. 12 - Vývojový diagram metody překrývajících se pruhů

5.4. Metoda Labellingu (Connected Component Labelling)

Metoda „connected-component labelling“ představuje další metodu pro detekci objektů v obrazu, která je však, na rozdíl od metody „překrývajících se pruhů“, použitelná pro všechny objekty bez omezení jejich tvaru. Nevýhodou této metody je její vyšší náročnost na výpočetní výkon (náročnost je navíc závislá na vstupních datech), a dále i náročnost na paměťový prostor. Při implementaci této metody jsem vycházel z poznatků v pracích [1] a [2].

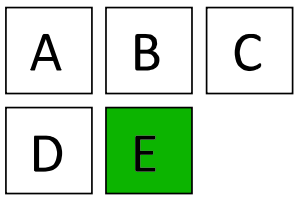
Metoda labellingu představuje implementaci teorie grafů, kdy pixely příslušící objektu představují uzly grafu, sousedství jednotlivých pixelů představují hrany grafu, a jednotlivé objekty v obraze představují komponenty souvislosti. Výsledkem použití metody labellingu je detekce a označení jednotlivých komponent souvislosti grafu (jednotlivých objektů v obrazu). V této implementaci je výstupem obraz, kde pixely s nulovou hodnotou přísluší k pozadí obrazu, a pixely s nenulovou hodnotou představují objekty. Jednotlivé objekty jsou pak odlišeny právě svou hodnotou.

Základní forma algoritmu vyžaduje data z předchozího a současného řádku a počítá se dvěma průchody obrazem. Při prvním průchodu obrazem je zkoumán pixel po pixelu celý obraz, a každému pixelu, který nenáleží k pozadí, je přiřazena značka. Pro každý pixel se při přiřazování značky zohledňuje tzv. „8-connected neighborhood“, tedy pixely, které spolu sousedí hranami nebo okraji (viz následující obrázek).



Obr. 13 - 4connected(vlevo) a 8connected(vpravo) neighborhood

Při dopředném řádkovém zpracování se zohledňuje předchozí a současný řádek až do právě zpracovávaného pixelu, tedy dohromady 4 sousední pixely, viz obr. 14.

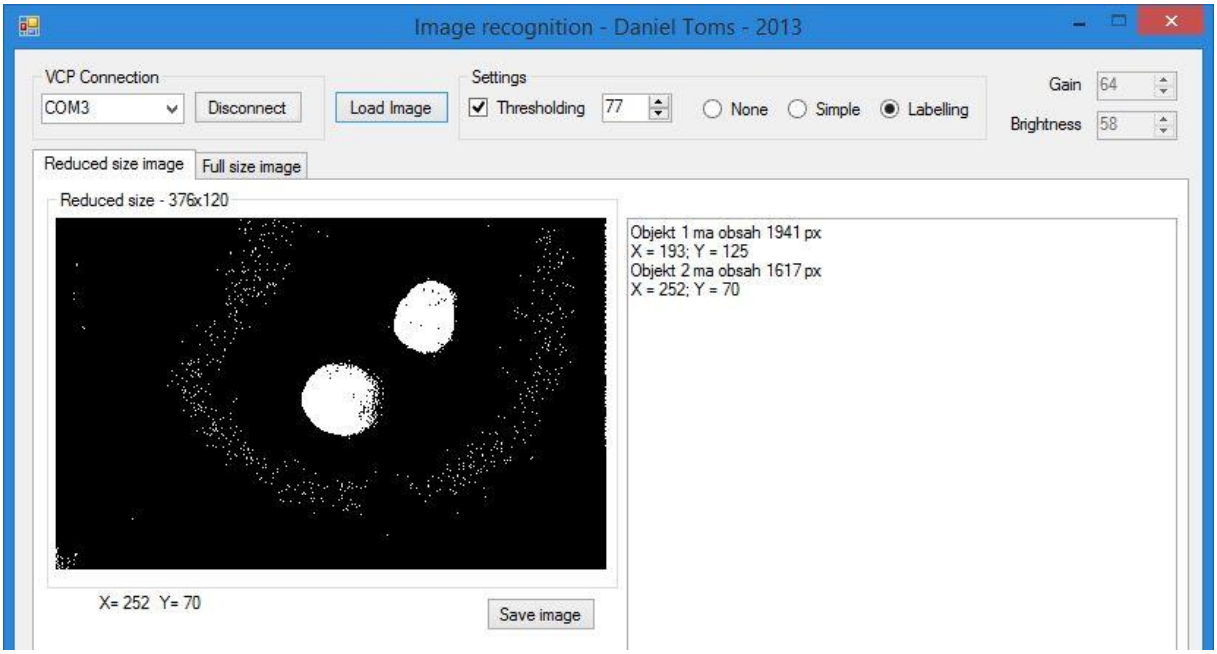


Obr. 14 - Pixely zohledňované při řádkovém zpracování

Přiřazování značky pak probíhá následujícím způsobem:

- Patří-li všechny pixely A-D k pozadí, dostává pixel E novou značku
- Pokud některý z pixelů A-D již značku má, její hodnota se zkopíruje i na pixel E
- Dojde-li ke spojení dvou dříve nesouvislých oblastí (například u objektu tvaru U, kde pixely A a C přísluší k různým oblastem), je třeba toto zaznamenat do jakési tabulky ekvivalentních značek, aby existovala informace, které oblasti k sobě patří. Z důvodu zjednodušení se přiřazuje vždy nejnižší ze sousedních značek.

Po dokončení prvního průchodu obrazem je tabulka ekvivalentních značek zjednodušena tak, aby všem značkám náležícím k jednomu objektu odpovídala pouze jedna značka, a při druhém průchodu obrazem jsou pak pixely označeny dle této zjednodušené tabulky. Tím je získán obraz, ve kterém jsou rozlišené a označené jednotlivé objekty. Ze získaných dat lze ještě odfiltrovat příliš malé objekty (šum – viz demonstrační obrázek).



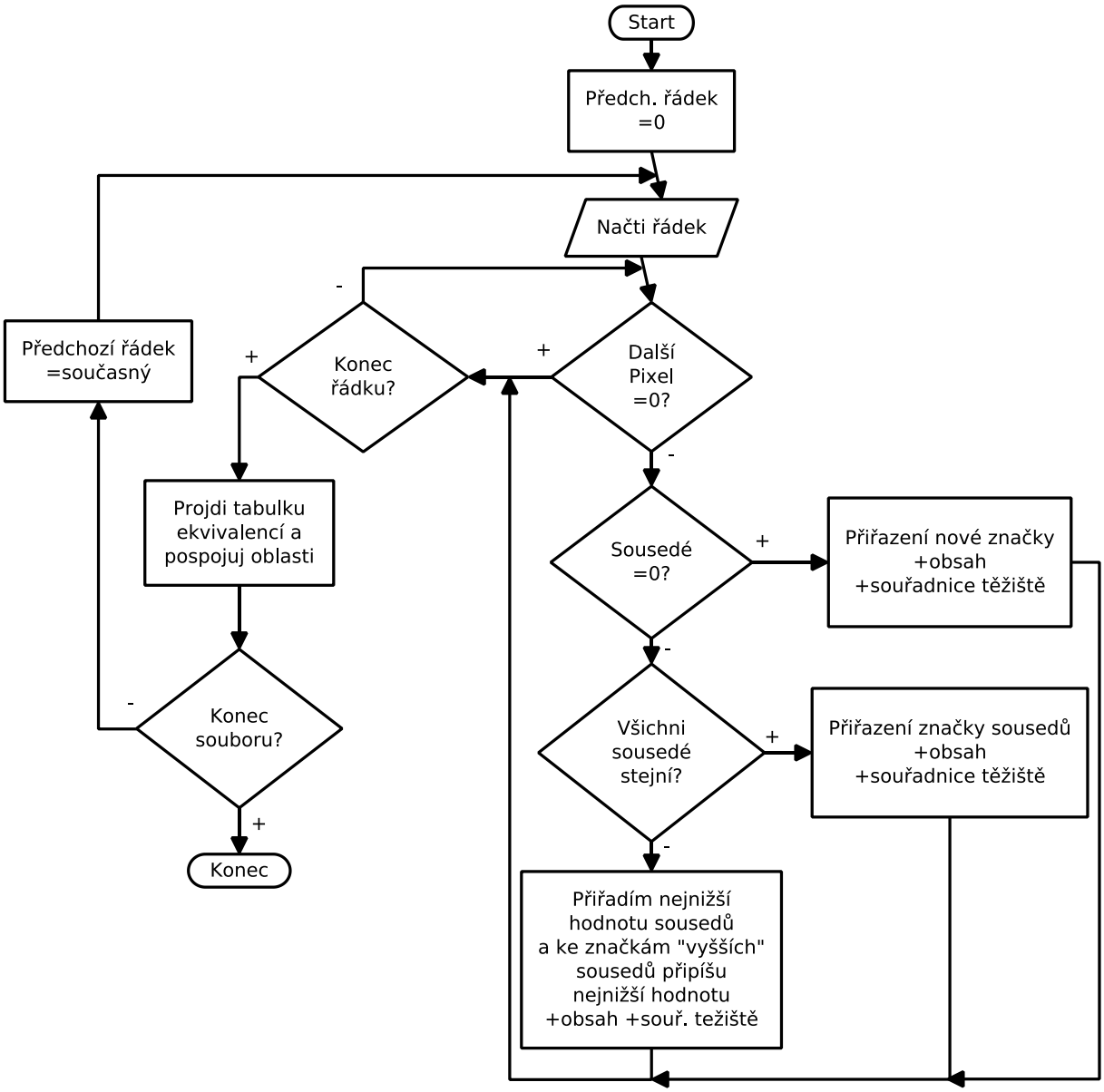
Obr. 15 - Demonstrace funkčnosti metody labellingu při detekci objektů

5.4.1. Úprava pro průběžné zpracování

Vzhledem k potřebě projít celý obraz dvakrát není tato metoda v základní formě použitelná pro průběžné zpracování. Metoda ale již při prvním průběhu poskytuje potřebnou informaci o jednotlivých regionech obrazu, a tabulku vztahu mezi nimi (tabulku ekvivalentních značek). Není tedy problém zpracovat informace o jednotlivých regionech a po zpracování celého obrazu pouze tyto informace spojit dle tabulky ekvivalentních značek. Vzhledem k tomu, že po získání informací o obsazích detekovaných objektů již zdrojový obrázek není k ničemu potřeba, je zbytečné jej znovu procházet a označovat každý pixel příslušnou značkou, čímž nutnost procházet obraz podruhé odpadá úplně.

Mezi určované parametry objektů opět patří především obsah a souřadnice těžiště objektu. Vychází se opět ze stejných vzorců, které byly použity pro předchozí algoritmus, protože jde však o obecnější algoritmus, je použit výpočetně náročnější způsob. Ve chvíli, kdy je určena příslušnost pixelu k některému z objektů, dojde k inkrementaci v paměti uložené hodnoty obsahu objektu, a rovněž se zvýší hodnota čitatele pro výpočet souřadnic těžiště o aktuální souřadnice pixelu.

Spojení obsahů podle tabulky ekvivalentních značek je provedeno prostým součtem, spojení čitatelů pro výpočet souřadnic těžišť rovněž tak. Po zpracování celého obrazu pak jsou čitatelé vzorců pro výpočet souřadnic těžišť vyděleny obsahy jednotlivých objektů. Vývojový diagram této metody následuje.



Obr. 16 - Vývojový diagram metody Labellingu

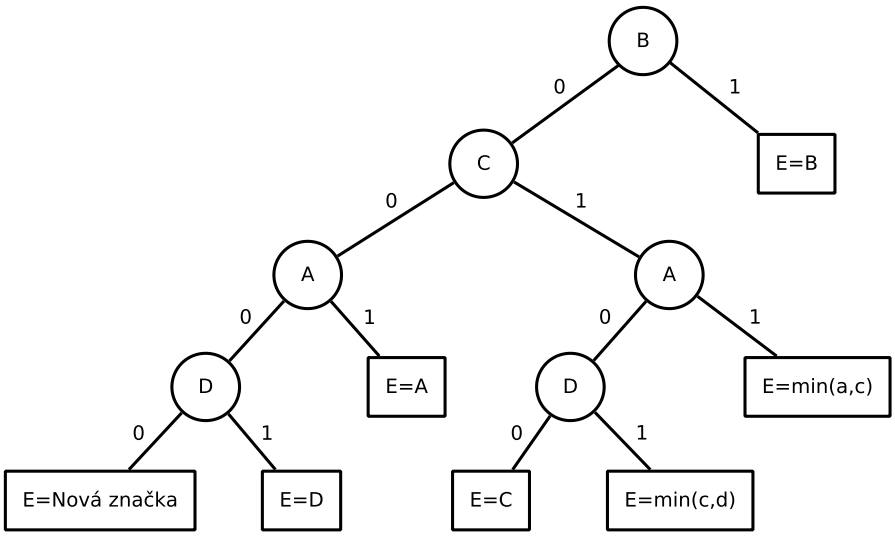
5.4.2. Optimalizace rychlosti

Naivní implementace algoritmu labellingu je výpočetně relativně neefektivní, proto byly implementovány některé metody zrychlující běh algoritmu. První optimalizační krok je využití rozhodovacího stromu pro navštěvování okolních pixelů (v 8-connected neighborhood), druhý krok je využití Union-Find algoritmu při práci s tabulkou ekvivalentních značek. Oba tyto algoritmy jsou do detailu popsány v práci [1], kde byly rovněž implementovány pro signálový procesor ADSP-BF533 ‚Blackfin'.

5.4.2.1. Rozhodovací strom

Optimalizace s využitím rozhodovacího stromu se snaží zredukovat počet navštívených pixelů sousedících s právě zkoumaným pixelem na minimum. Dle [7] použití rozhodovacího stromu zrychlí algoritmus až na 12/7 rychlosti naivní implementace.

Z obr. 18 je zřejmé, že všechny pixely v této skenovací masce jsou sousedy pixelu B. Pokud tedy pixel B náleží k objektu (B=1), není třeba zkoumat ostatní pixely vůbec. Patří-li pixel B k pozadí, je vyvolán rozhodovací strom:



Obr. 17 - Rozhodovací strom pro labelling

Patří-li pixel B k pozadí (B=0), pak zkoumáme pixely A a C – patří-li oba k nějakému objektu, je pixelu E přiřazena menší ze značek reprezentujících objekty, ke kterým náleží pixely A a C. Pokud patří k objektu pouze pixel A, pak zkoumaný pixel dostává jeho značku rovnou. Patří-li k objektu pouze pixel C, pak je v závislosti na pixelu D přiřazena zkoumanému pixelu buď menší ze značek reprezentujících objekty, ke kterým náleží pixely C a D, nebo je přiřazeno rovnou E=C. Pokud ze všech sousedů patří k objektu pouze pixel D, pak je přiřazeno E=D, a konečně pokud žádný pixel nenáleží k žádnému objektu, je pixelu E přiřazena značka nová.

5.4.2.2. *Union-Find algoritmus*

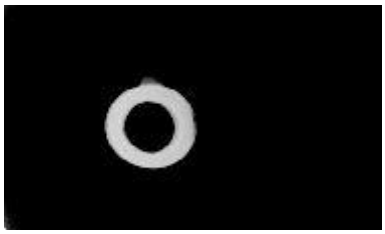
Způsob, jakým algoritmus labellingu spojuje dvě oblasti v tabulce ekvivalencí, lze rovněž podrobit optimalizaci. Naivní způsob znamená opakovaně procházet celou tabulku ekvivalentních značek, a upravovat značky tak, aby $P[X] = P[P[X]]$, kde P je tabulka ekvivalencí. Jakmile dojde k jedné změně, je prohledávání přerušeno a tabulka je procházena opět od začátku. Celý algoritmus končí až potom, co mezi dvěma průchody tabulkou nedojde ke změně značek.

Algoritmus Union-Find představuje efektivní metodu pro sledování vztahů mezi jednotlivými labely. Union-find ukládá labely, které náleží ke stejnému objektu, do disjunktních množin (tj. tak, že každý label je právě v jedné množině) ve formě stromů, kde labely jsou uzly stromů, a hrany mezi uzly značí, že značky náleží stejnému objektu. Kořen stromu pak slouží jako reprezentant objektu. Místo spojových seznamů realizovaných pointery je využito jednorozměrného pole, jehož index představuje značku, a hodnota představuje ukazatel na nadřazený label. Funkce Union(uzel1,uzel2) pak spojí dva stromy, funkce Find(uzel) vyhledá kořen stromu, do kterého uzel patří. Z názvu těchto funkcí pak plyne název algoritmu. Kořeny stromů představují výsledné značky objektů. Při přiřazení nové značky je založen nový strom, a tato značka se stává kořenem tohoto stromu.

Je-li dále funkce Union(uzel1,uzel2) implementována tak, že vždy za kořen nového (spojeného) stromu zvolí kořen s nižším číslem labelu, pak ve výsledném stromu má vždy rodič nižší číslo než potomek. Toho je pak využito v poslední funkci, flatten(). Tato funkce upraví značky podobně jako naivní implementace tak, aby $P[X] = P[P[X]]$, na rozdíl od naivní implementace se ale nyní pole P prochází jenom jednou, což je důsledkem toho, že každý rodič má nižší číslo než jeho potomek. Práce [7] dokazuje, že použití Union-Find algoritmů zajišťuje, že výsledná náročnost úpravy tabulky ekvivalencí je nezávislá na složitosti této tabulky.

5.4.2.3. *Výsledná rychlost algoritmu labellingu*

Stručné shrnutí výsledků optimalizace při detekci objektu zabírajícího zhruba 5% zorného pole (viz obr. 18) kamery je v tab. 2. Přesné hodnoty byly získány softwarovým měřením pomocí časovačů. Výsledná maximální rychlost snímání a vyhodnocování algoritmem labellingu je cca 11,1 sn./s.



Obr. 18 - Vzorový objekt pro testování rychlosti algoritmů

ALGORITMUS	DOBA ZPRACOVÁNÍ CELÉHO OBRAZU	ZRYCHLENÍ PROTI NAIVNÍ IMPLEMENTACI
Naivní	6,005,418 cyklů MCU	0
S rozhodovacím stromem a union-find	5,114,130 cyklů MCU	14,8%

Tab. 2 - Shrnutí výsledků optimalizace labellingu

Vzhledem k tomu, že použité optimalizační metody ovlivňují pouze dobu zpracovávání „objektových" pixelů, tj. pixelů, které v obrazu představují objekt a nikoliv pozadí, je jejich přínos pro zpracování obrazu s nízkým celkovým množství „objektových" pixelů celkově menší. Úzkým hrdlem celého programu je ale ten nejpomalejší sektor (viz obr. 6), neboli program je tak rychlý, jak je rychlá jeho nejpomalejší část. V případě zpracování obr. 18 tvoří rozdíl mezi dobou zpracování nejpomalejšího sektoru před a po optimalizaci 30,9%, což již představuje výrazné zrychlení.

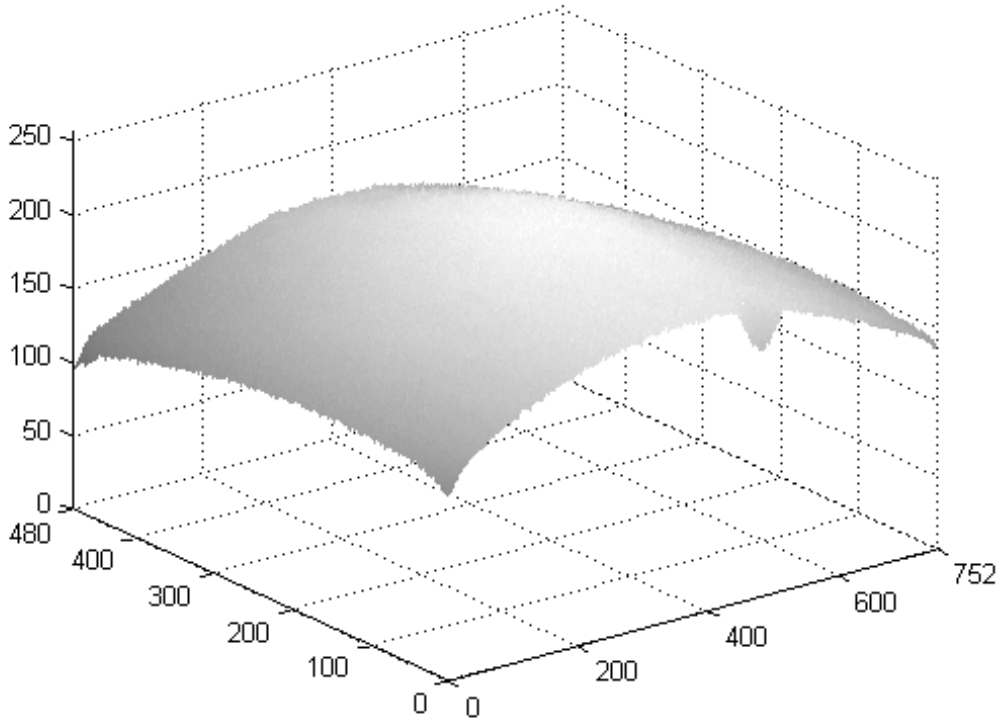
5.5. Korekce vinětace

Snímání obrazu je obecně ovlivněno různými vlastnostmi a vadami použitých objektivů a jejich čoček. Mezi tyto vady patří například sférická aberace, která je způsobená tím, že se paprsky na okrajích čočky lámou více než u středu čočky a nejsou tedy zaostřeny na sensor, ale mírně před něj, nebo chromatická vada, která vyplývá z faktu, že indexy lomu jednotlivých barev ve spektru se liší.

Nejpalčivější problém však u použitého objektivu představuje vinětace. Vinětace je vada snímací sestavy, která se projevuje sníženým jasem na okrajích snímaného obrazu. Vznik vinětace je způsobem tím, že při použití běžného objektivu dopadají paprsky kolmo pouze na střed sensoru, a úhel dopadu se zvětšuje směrem k okrajům obrazu, což znamená, že světelný paprsek dopadá na větší plochu a intenzita světelného záření na jednotku plochy je tak nižší. To pak má za následek tmavnutí okrajů obrazu. Tento efekt dále prohlubuje samotný digitální CMOS sensor, jehož světelná citlivost se liší v závislosti na úhlu dopadajícího světla. Fyzické rozměry sensoru jsou 4.51x2.88 mm.

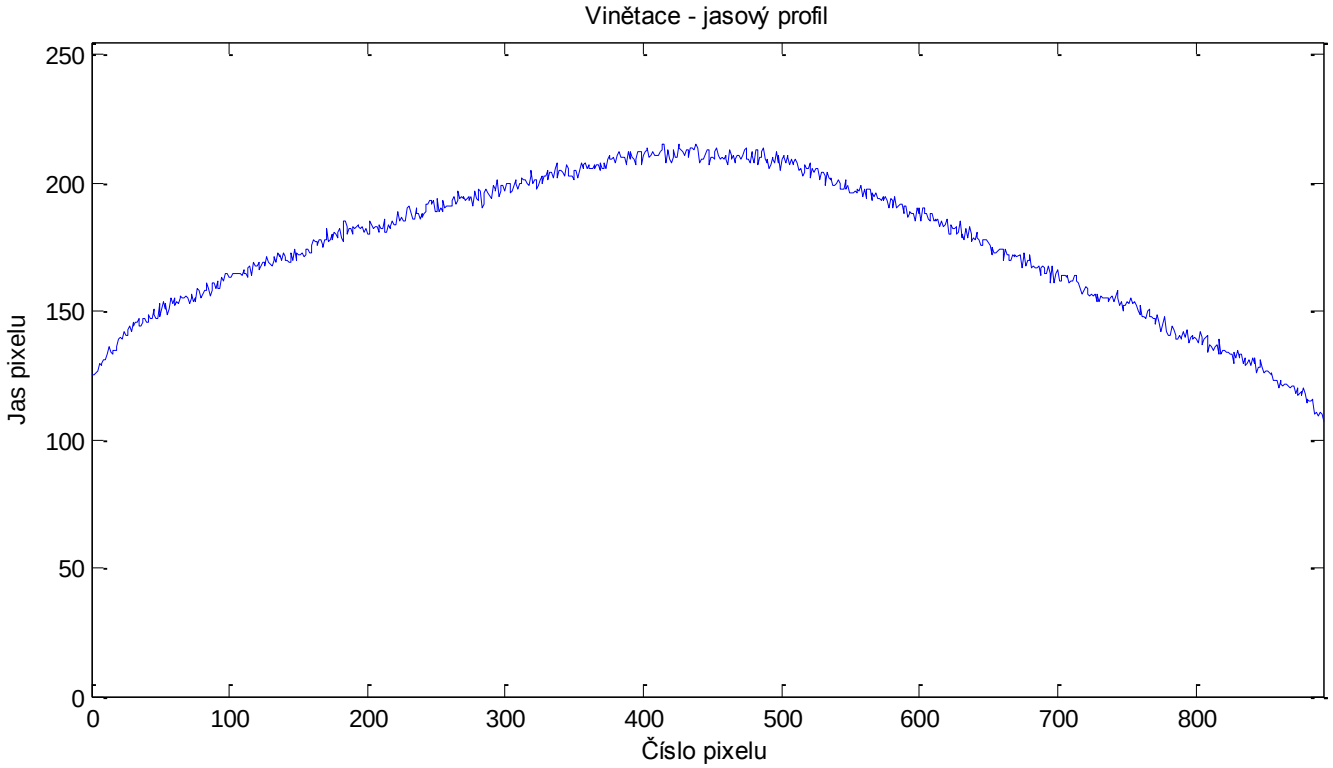
Obr. 19 - Efekt vinětace na obrázku sejmutém sensorem MT9V034 (zmenšeno)

Použitý objektiv výrazně trpí efektem vinětace, a ačkoli tento efekt nepředstavuje pro lidského pozorovatele větší problém, schopnosti algoritmů strojového vidění zpravidla touto optickou deformací výrazně utrpí. Bude-li tedy k detekci objektů v obrazu využito prostého prahování, je výhodné (za určitých podmínek, tj. například při detekci málo kontrastních objektů, dokonce nezbytné) korigovat tento efekt. Na obr. 19 je graficky znázorněn pokles jasu v rozích obrazu pomocí funkce Surface plot (surf) programu MATLAB. Jako podklad byl použit snímek z obr. 18, který byl pořízen přímo použitým CMOS sensorem.



Obr. 20 - Znázornění efektu vinětace na snímku homogenního pozadí

Obr. 20 znázorňuje jasový profil úhlopříčného řezu snímkem, ze kterého je opět zřejmý pokles jasu v rozích obrazu. Pro zobrazení posloužila funkce improfile programu MATLAB, a jako podklad byl opět použit snímek z obr. 18.



Obr. 21 - Úhlopříčný řez snímkem ovlivněným efektem vinětace

Jedna z nejpřímočařejších variant korekce vinětace je tzv. „Flat-field“ korekce. Pro určení vlivu vinětace na snímaný obraz je sejmut kalibrační snímek homogenní a rovnoměrně nasvícené scény (např. odrazné desky). Expozici snímače je vhodné nastavit tak, aby sejmutý snímek dosahoval v průměru zhruba 30-50% saturační úrovně. Tím se značně omezí vliv případné nelinearity sensoru, ale především se tím zajistí, že nebude sejmut saturovaný snímek (tj. snímek, ve kterém jeden nebo více obrazových bodů dosáhlo saturační úrovně). Takový snímek by pro flat-field korekci nebyl použitelný.

Rozdíly v jasu jednotlivých pixelů takto sejmutého snímku jsou způsobeny vinětací. Průměrný jas celého snímku je pak vydělen hodnotami jednotlivých pixelů, čímž se každému pixelu

- s nižším než průměrným jasem přiřadí hodnota mírně vyšší než 1,
- každému pixelu s vyšším jasem hodnota mírně nižší než 1.

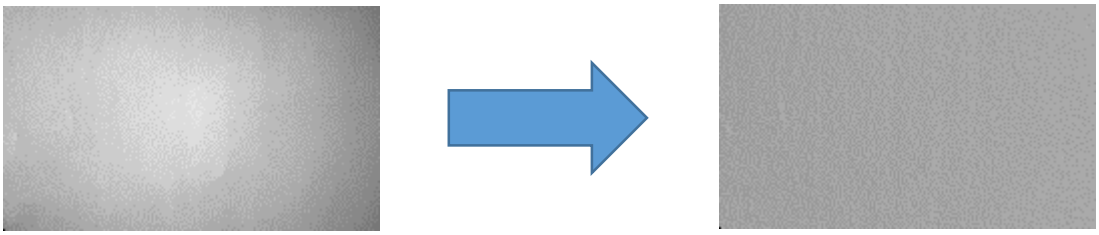
Vzhledem k větší paměťové náročnosti čísel s plovoucí desetinnou čárkou (4 byty na pixel, v porovnání s prostými hodnotami jasu jednotlivých pixelu, které vyžadují 1 byte na pixel) bylo

zvoleno oddělené uložení korekčního snímku a hodnoty průměrného jasu všech pixelů tohoto snímku. Výpočet korekčního koeficientu se tedy vykonává až během samotné korekce.

$$P(x,y) = \frac{R(x,y) * m}{F(x,y)},$$

kde $P(x,y)$ je jas pixelů výsledného obrazu, $R(x,y)$ je jas pixelů vstupního obrazu, m je průměrný jas korekčního snímku, a $F(x,y)$ je jas pixelů korekčního snímku.

V zájmu šetření paměťovým prostorem je možné ukládat pouze zmenšený korekční snímek, ve kterém jednotlivé pixely budou představovat průměr z více sousedních pixelů. V této implementaci je originální obraz o rozlišení 752x480px zredukován na obraz o velikosti 188x120px, kde každý pixel představuje průměrnou hodnotu koeficientu ze 16ti pixelů původního obrazu (4x4px). Taková úprava značně sníží paměťovou náročnost korekce, a její vliv na účinnost korekce je zanedbatelný.



Obr. 22a - Snímek před korekcí

Obr. 21b - Snímek po korekci

Flat-field korekce je velmi intuitivní, jednoduchá pro implementaci, a v případě, že je možné sejmout kvalitní korekční snímek, také poskytuje skvělé výsledky. Nevýhodou Flat-field korekce je právě nutnost pořízení korekčního snímku, který vyžaduje rovnoměrně nasvícenou scénu, a rovněž nutnost pořídit nový korekční snímek vždy, když dojde k pohybu či jen přeostření kamery.

5.5.1. Ukládání korekčních koeficientů do Flash

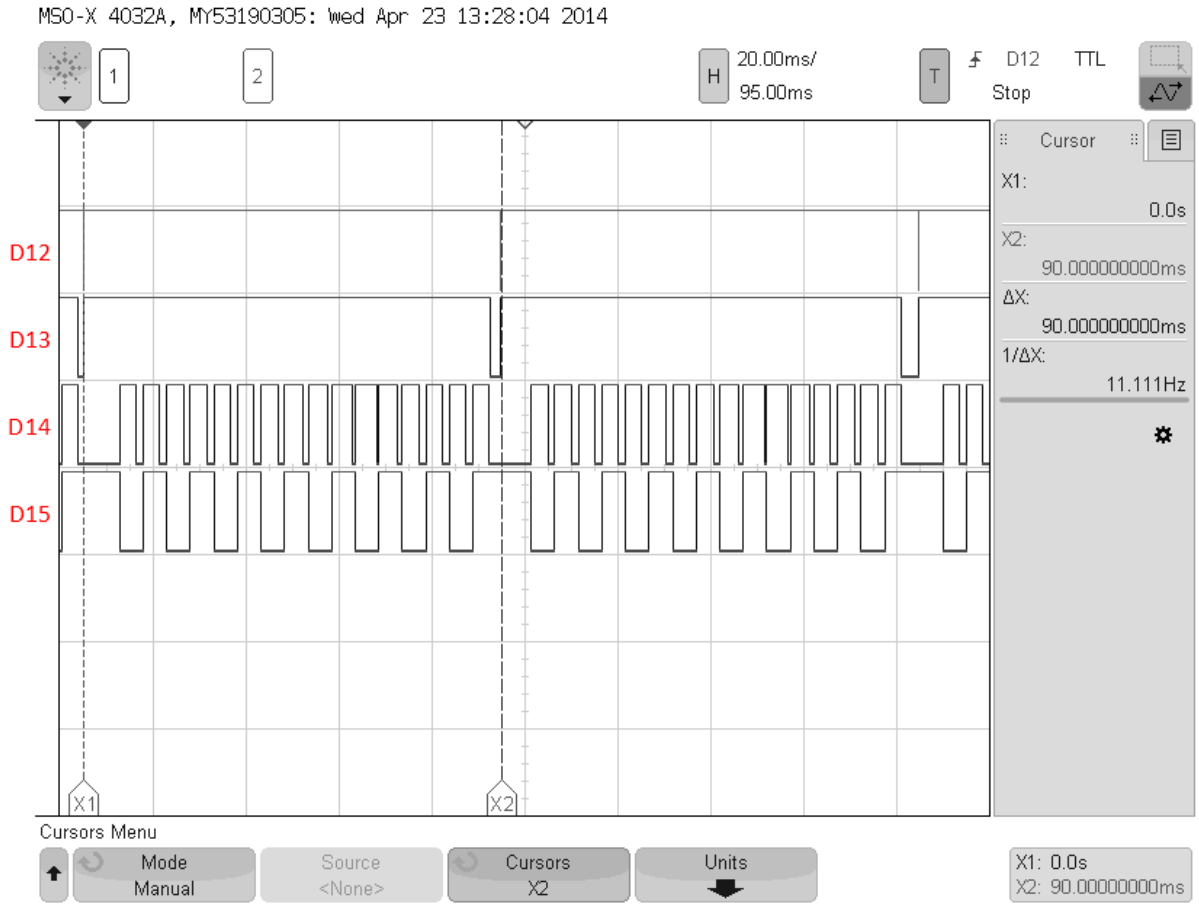
Pro dlouhodobé uložení korekčního snímku (a rovněž pro uložení případných dalších dat) lze využít nevolatilní paměť typu Flash, kterou disponují oba kity (F4: 1MB; F429: 2MB), a která je sdílená s prostorem pro programový kód. Tato paměť je rozdělena na sektory o velikostech 16-128kB (což je důležité vzhledem k tomu, že paměť Flash lze mazat pouze po celých sektorech). Je tedy třeba si dát pozor, byla data zapisována do volných sektorů.

5.6. Kontrola funkčnosti programu osciloskopem

V průběhu programování vyvstala potřeba hardwarově testovat funkčnost programu, neboť v případě průběžného zpracování není možné program krokovat v in-circuit debuggeru, a zobrazení osciloskopem je tak jedinou možností, jak získat alespoň rámcový přehled o tom, zda program funguje tak, jak má. K tomuto účelu lze využít volných GPIO pinů

mikrokontroléru, jejichž úroveň lze programově měnit například vždy na začátku a konci vykonáváni algoritmu. Tím lze vyhodnotit reálnou rychlost programu, implementovaných algoritmů i přínos optimalizačních metod.

Vizualizace na osciloskopu poskytuje názornější náhled na rychlost algoritmu, což je zřejmé z obr. 21. Vizualizace zobrazuje průběh dvou programových smyček, tj. sejmutí, vyhodnocení dvou po sobě jdoucích snímků, včetně následného odeslání části dat do počítače.



Obr. 23 - Vizualizace na osciloskopu

Signál D12 představuje činnost celé programové smyčky, tj. snímání obrazu, zpracování obrazu a odesílání dat do počítače. Signál D13 představuje aktivitu CMOS sensoru, tj. expozici a následné odesílání dat přes DCMI do paměti mikrokontroléru. Signál D15 ukazuje, do které lokace právě DMA zapisuje obrazová data. Vysoká úroveň signálu D15 značí zápis do první paměťové lokace, nízká úroveň značí zápis do druhé paměťové lokace (viz obr. 6). Na průběhu tohoto signálu lze vidět princip střídavého zápisu do dvou paměťových lokací v praxi. Signál D14 znázorňuje dobu vyhodnocování algoritmu. Pokud je tento signál na vysoké úrovni, probíhá zpracování obrazových dat, pokud je tento signál na nízké úrovni, je procesor v nečinnosti a čeká na nová data. Ze signálu D14 je rovněž patrná závislost rychlosti algoritmu na komplexitě zpracovávaných dat (neboli jak často se algoritmus musí nořit hlouběji do

rozhodovacího stromu, zbytek operací běží v lineárním čase). Ze snímku je přibližně vidět, že v první polovině obrazu nebyl detekován žádný objekt, v druhé polovině ano.

5.7. Automatické nastavení rychlosti snímání

Jelikož je žádoucí, aby výsledná aplikace běžela maximální možnou rychlostí, bylo implementováno softwarové řízení frekvence hodinového signálu CMOS sensoru. Vzhledem k tomu, že rychlost běhu programu je závislá na náročnosti vstupních dat, tedy na tom, kolik pixelů je nad/pod komparační úrovní, je v programu implementováno softwarové měření rychlosti s využitím časovačů. Měřenými hodnotami jsou doba snímání a doba vyhodnocování dat algoritmem zpracování obrazu. Rozdíl těchto hodnot pak představuje časovou rezervu. Podle velikosti rezervy je následně regulována rychlost snímání.

Hodinový signál pro CMOS sensor je generován jedním z časovačů mikrokontroléru, a jeho frekvence je dána jako

$$f = \frac{84 \cdot 10^6}{2 * D}\ [Hz],$$

kde f je frekvence a D je konstanta nastavení tohoto čítače. Měněním této konstanty program mění frekvenci senzoru. Základní frekvence je nastavena na situaci, kdy všechny detekované objekty dohromady zabírají maximálně zhruba 10% obrazu.

Zjistí-li program, že je časová rezerva mezi jednotlivými zpracovávanými buffery dostatečná, zvyšuje automaticky rychlost snímání (změnou frekvence hodin CMOS sensoru). Naopak ve chvíli, kdy při snímání náročnějšího obrázku program zjistí, že zpracování některého z DMA bufferů (viz obr. 6) trvá déle než snímání dalšího (což znamená, že ještě nezpracovaná data jsou již přepisována daty novými), je následně zpracování programově přerušeno a zbytek snímku je zahozen. Následně je snížena rychlost snímání a je provedeno nové snímání.

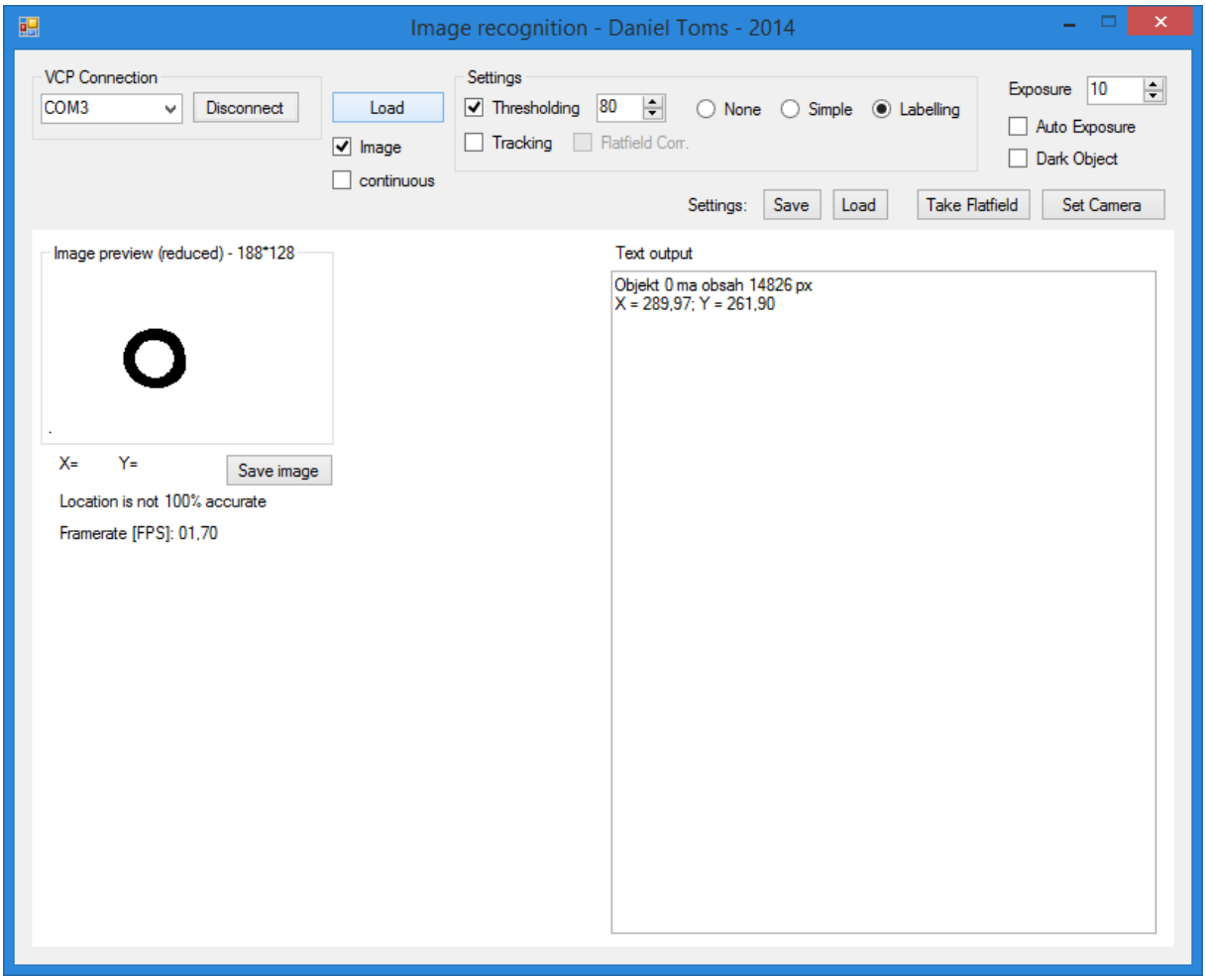
Při měnění frekvence hodin CMOS sensoru je třeba si dát pozor na to, že s měnící se frekvencí sensoru se mění i doba expozice, která je u sensoru MT9V034 vyjádřena počtem řádků, resp. dobou jejich čtení. Tato hodnota je uložena v registrech sensoru a je možné ji programově měnit. Rychlost čtení řádků je pak přímo závislá na frekvenci hodinového signálu sensoru. Snížení frekvence hodinového signálu pak znamená delší dobu snímání, a tedy delší reálnou dobu expozice. Tuto změnu je nutné kompenzovat, aby se změnou rychlosti senzoru neměnila snímaná data. Závislost doby expozice na frekvenci hodinového signálu sensoru je lineární, a změna expozice se vypočte jako

$$E = \frac{R_n}{R_{old}} * E_{old},$$

kde E je nová doba expozice, R_n je nová rychlost, R_{old} je stará rychlost, a E_{old} je původní expozice.

6. PC aplikace pro komunikaci a ovládání kitu

Pro komunikaci s kitem, tj. k ovládání a příjmu (a intepretaci) dat z kitu jsem vytvořil grafickou aplikaci v jazyce C#. Náhled aplikace následuje.



Obr. 24 - Náhled PC aplikace

Navržená aplikace podporuje komunikaci pomocí rozhraní COM, v tomto případě jde o spojení přes sběrnici USB v režimu Virtual COM Port. Při startu aplikace dojde k detekci dostupných COM portů, z nichž si poté uživatel může vybrat pomocí rolovacího menu v levé horní části obrazovky. Po stisknutí tlačítka „connect“ dojde k pokusu o spojení se zařízením a odeslání nastavovacího paketu, který obsahuje informace k nastavení připojeného kitu.

Volbami v rámečku „settings“ lze ovlivnit chování kitu, lze nastavit prahování (thresholding) a prahovací hodnotu, a rovněž nastavit konkrétní metodu zpracování obrazu či zapnutí korekce vinětace. Není-li aktivní volba prahování, program nepovolí nastavení metod zpracování obrazu (odpovídající check-boxy nejsou viditelné). V pravém horním rohu aplikace lze nastavit dobu expozice nebo zapnout automatiku.

Tlačítko „Load Image“ pošle kitu příkaz k jednorázovému sejmutí obrázku (snapshot) a jeho odeslání do PC aplikace. Obrázek se poté vykreslí do obrazového rámečku pod tlačítkem. Při najetí kurzorem na vykreslený obrázek lze odečíst souřadnice umístění kurzoru v obrázku. Tlačítko „save image“ pak již jen uloží právě vykreslený obrázek do zvoleného souboru ve formátu .BMP (rastrový obraz s rozlišením 8 bitů na pixel). Tlačítka „Save“ a „Load“ slouží k uložení, resp. načtení, korekčních koeficientů pro korekci vinětace z nevolatilní paměti Flash.

Tlačítko „Dark Object“ určuje, zda je snímaný objekt světlejší nebo tmavší než pozadí snímané scény. Pole vpravo slouží pro datový výstup kitu pro jiná, než obrazová data (v naší aplikaci především informace o těžišti a obsahu detekovaných objektů). Pokud budou z kitu přijata data v předdefinovaném formátu (viz dále), pak dojde k interpretaci přijatých dat a jejich formátovaný výstup, v opačném případě jde o neformátovaný výstup.

6.1. Formát datové komunikace s vývojovým kitem

6.1.1. Komunikace PC -> Discovery

Komunikaci ve směru PC->Discovery obstarává několik jednobytových příkazů a dva vícebytové pakety - paket pro nastavení módu snímání (mode), a paket pro nastavení parametrů snímání a zpracování obrazu (control).

Příkazy:

- „s“ – spuštění snímání
- „t“ – zastavení snímání
- „f“ – sejmutí korekčního obrazu pro korekci vinětace
- „u“ – uložení nastavení do FLASH
- „n“ – nahrání nastavení z FLASH
- „L“ – přenos obrazu v plném rozlišení (pouze F429-Discovery)
- „G“ – spuštění regulace polohy optické stopy (pouze F429-Discovery)

Pakety:

- MODE – velikost 2B

BYTE	VÝZNAM	PŘÍPUSTNÉ HODNOTY
0	Identifikátor paketu	Vždy “m” (0x6D)
1	Kontinuální/snapshot režim	0 – snapshot režim 1 – kontinuální režim

Tab. 3 - Byty paketu MODE

- Paket CONTROL – velikost 9B

BYTE	VÝZNAM	PŘÍPUSTNÉ HODNOTY
0	Identifikátor paketu	Vždy “r” (0x72)
1	Zapnutí/vypnutí Prahování	0 - vypnuto 1-255 – hodnota prahu
2	Výběr použitého algoritmu	0 - žádný 1 - jednoduchý 2 - labelling
3+4	Doba expozice	0-65535
5	Zapnutí/vypnutí sledování polohy	0 - vypnuto 1 - zapnuto
6	Zapnutí/vypnutí flat-field korekce	0 - vypnuto 1 - zapnuto
7	Zapnutí/vypnutí přenosu obrazu do PC	0 - vypnuto 1 - zapnuto
8	Objekt je jasnější/tmavší než pozadí	0 – objekt je jasnější 1 – objekt je tmavší

Tab. 4 - Byty paketu CONTROL

Na straně mikrokontroléru pak dochází k dekódování těchto paketů a aplikaci příslušných nastavení.

6.1.2. Komunikace Discovery->PC

Komunikaci ve směru Discovery->PC obstarávají buď pakety obsahující obrazová data ze sensoru (k jejich rozpoznání slouží jejich fixní délka – obrazová data mají stále stejnou velikost), nebo speciální pakety obsahující data o detekovaných objektech a o jejich posunu v zorném poli kamery. Všechna ostatní data jsou pouze vypsána na textový výstup programu bez další interpretace.

Paket DATA – velikost 17B

BYTE	VÝZNAM	PŘÍPUSTNÉ HODNOTY
0	Identifikátor paketu	Vždy “o” (0x6F)
1-4	Obsah objektu	$0 – (2^{32}-1)$
5-8	Souřadnice X těžiště objektu	$0 – (2^{32}-1)$
9-12	Souřadnice Y těžiště objektu	$0 – (2^{32}-1)$

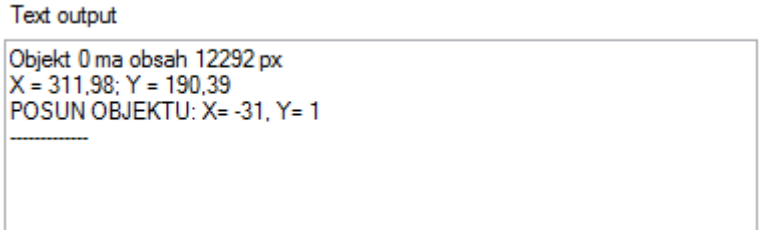
Tab. 5 - Byty paketu DATA

Paket SHIFT – velikost 9B

BYTE	VÝZNAM	PŘÍPUSTNÉ HODNOTY
0	Identifikátor paketu	Vždy “s” (0x73)
1-4	Posun v ose X	0 – (2^{32}-1)
5-8	Posun v ose Y	0 – (2^{32}-1)

Tab. 6 - Byty paketu SHIFT

Každý paket obsahuje informace o jednom objektu, maximální počet objektů je závislý na nastavení algoritmu v MCU (ve finální verzi je to 256 objektů). PC aplikace tato data interpretuje a vypisuje v uživatelsky srozumitelné formě.



Obr. 25 - Ukázka dat přijatých PC aplikací

7. Přizpůsobení projektu pro kit F429-Discovery

V druhé polovině roku 2013 byl společností STMicroelectronics představen kit STM32F429-Discovery, jehož hlavními zlepšeními jsou mírně vyšší frekvence procesoru, přítomnost vestavěného LCD displeje s rozlišením 240x320px, a především vestavěné paměti SDRAM o velikosti 8MBytes (více o kitu v kap. 3.2). Bylo rozhodnuto pokusit se nový kit využít pro zpracování obrazu, primárně z důvodu přítomnosti větší paměti, a případně i LCD displeje. Přítomnost velké paměti znamená, že obraz lze snímat maximální rychlostí CMOS sensoru a není nutné čekat na průběžné zpracování dat mikrokontrolérem. Zároveň jsou však tato data, na rozdíl od průběžného zpracování s využitím dvojitého bufferingu, po dokončení snímání dostupná po neomezeně dlouhou dobu, což umožňuje pro jejich zpracování využít i složitějších algoritmů.

V další části této práce byla nejprve vytvořena propojovací deska pro kit F429-Discovery a CMOS modul (viz kap. 3.3). Účelem této desky je především poskytnout spojení datových a napájecích pinů kitu a CMOS modulu. Vzhledem k tomu, že rozložení GPIO headerů na kitech F4 a F429 se liší, nebylo možné využít propojovací desku vytvořenou pro kit F4-Discovery (viz práce [3]), ale bylo nutné vytvořit desku novou. Jako základ nové desky byl použit univerzální vrtaný plošný spoj, na který byly připájeny dvě dvouřadé dutinkové lišty pro nasunutí kitu a jedna dvouřadá kolíková lišta pro připojení CMOS modulu. Odpovídající

signály pak byly spojeny stejným způsobem, jako u propojovací desky ke kitu F4-Discovery. Modul dále obsahuje dostatečně proudově dimenzovaný napěťový regulátor 3,3V (typ LF33CV v pouzdře TO220, s maximálním výstupním proudem 500mA při 3,3V) a dále konektor pro externí napájení a desetipinový dvouřadý konektor pro připojení desky s řídicím obvodem pro krokový motor. Krokový motor včetně řídicího obvodu bude popsán v následujících kapitolách.

Po vytvoření propojovací desky byl kit oživen. Vzhledem k chování GPIO pinů PA1 a PA2, na které bylo připojeno ovládání CMOS sensoru, a které na vysoké logické úrovni poskytovaly napětí pouze 600mV namísto standardních 3V, byly signály CMOS_EXPOSURE a CMOS_RESET přemapovány na piny PB1 a PB2. Dále byl přemapován pin DCMI_D5 z PB6 na PD3, protože pin PB6 je využíván řadičem FMC (Flexible Memory Controller) pro ovládání SDRAM paměti, což by zamezilo ukládání obrazu ze sensoru přímo do SDRAM. Tabulka popisující propojení sensoru CMOS a kitu Discovery je k vidění na následující stránce. Pro úplnost jsou v tabulce uvedeny i signály zajišťující řízení krokového motoru, který bude využit v dalších kapitolách.

5V	5V
PF4	PF5
PF2	PF3
PF0	PF1
PC14	PC15
PE6	PC13
PE4	PE5
PE2	PE3
PE0	PE1
PB8	PB9
BOOT0	Vdd
PB6	PB7
PB4	PB5
PG15	PB3
PG13	PG14
PG11	PG12
PG9	PG10
PD7	PD6
PD5	PD4
PD3	PD2
PD1	PD0
PC12	PC11
PC10	PA15
PA14	PA13
PA12	PA11
PA10	PA9
PA8	PC9
PC8	PC7
PC6	PG8
PG7	PG6
PG5	PG4
GND	GND

3V	3V
PF6	N/C
PF8	PF7
PF10	PF9
PH1	PH0
GND	NRST
PC1	PC0
PC3	PC2
	PA0
PA3	
PA5	PA4
PA7	PA6
PC5	PC4
PB1	PB0
GND	PB2
PF12	PF11
PF14	PF13
PG0	PF15
PE7	PG1
PE9	PE8
PE11	PE10
PE13	PE12
PE15	PE14
PB11	PB10
PB13	PB12
PB15	PB14
PD9	PD8
PD11	PD10
PD13	PD12
PD15	PD14
PG3	PG2
GND	GND

Obr. 26 - Kit F429 Discovery, GPIO headery, pohled shora

Signál CMOS sensoru	Pin Discovery kitu
D0	PC6
D1	PC7
D2	PC8
D3	PC9
D4	PE4
D5	PD3
D6	PE5
D7	PE6
D8	-
D9	-
PIXCLK	PA6
SYSCLK	PB0
HSYNC	PA4
VSYNC	PB7
STANDBY	PA3
RESET	PB2
OE (Output Enable)	PC2
EXPOSURE	PB1
I2C_SDA	PB9
I2C_SCLK	PB8
LEDOUT	PC1
SPI_SCK	PC10
SPI_MOSI	PC12
HCT595 Latch	PG3

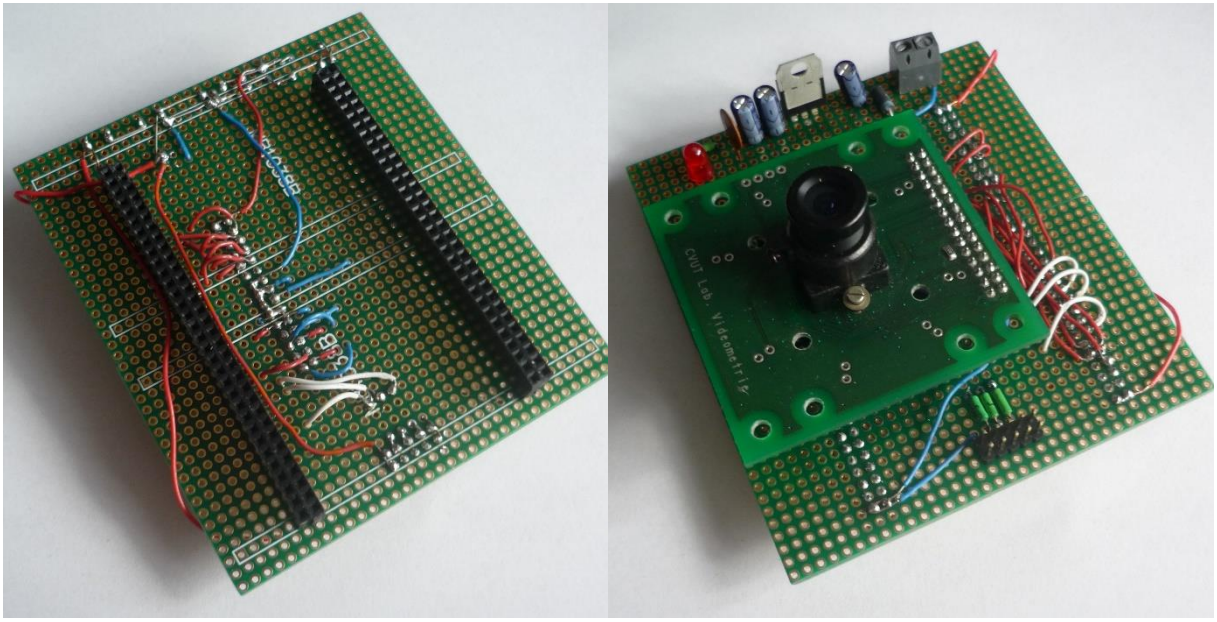
Tab. 7 - Propojení sensoru CMOS a kitu F429-Discovery

- Signály SCLK a SDA náleží k rozhraní I²C (na straně mikrokontroléru konkrétně I²C2)
- Signály D0-D9, PIXCLK, HSYNC, VSYNC náleží k rozhraní DCMI
- Signály PA10, PC10, PC12 slouží k řízení krokového motoru
- Zbývající signály slouží k řízení CMOS sensoru a jeho napájení
- Tmavě vyznačené signály slouží ke komunikaci s SDRAM

Pro připojení LCD displeje je třeba vzít na vědomí, že piny displeje částečně kolidují s piny DCMI rozhraní, proto není možné zároveň snímat obraz a zobrazovat informace na vestavěném LCD displeji. Piny ostatních periferií, včetně SDRAM, naštěstí vzájemně nekolidují. LCD je hardwarově připojeno v režimu RGB565. Seznam pinů je uveden v tab. 8, kolize s rozhraním DCMI jsou vyznačeny hvězdičkou.

SIGNÁL LCD	PIN DISCOVERY	SIGNÁL LCD	PIN DISCOVERY
LCD_TFT R3	PB0*	LCD_TFT G7	PD3*
LCD_TFT R4	PA11	LCD_TFT B3	PG11
LCD_TFT R5	PA12	LCD_TFT B4	PG12
LCD_TFT R6	PB1*	LCD_TFT B5	PA3*
LCD_TFT R7	PG6	LCD_TFT B6	PB8*
LCD_TFT G2	PA6*	LCD_TFT B7	PB9*
LCD_TFT G3	PG10	LCD_TFT HSYNC	PC6*
LCD_TFT G4	PB10	LCD_TFT CLK	PG7
LCD_TFT G5	PB11	LCD_TFT VSYNC	PA4*
LCD_TFT G6	PC7*	LCD_TFT DE	PF10

Tab. 8 - Připojení LCD displeje ke kitu F429 Discovery

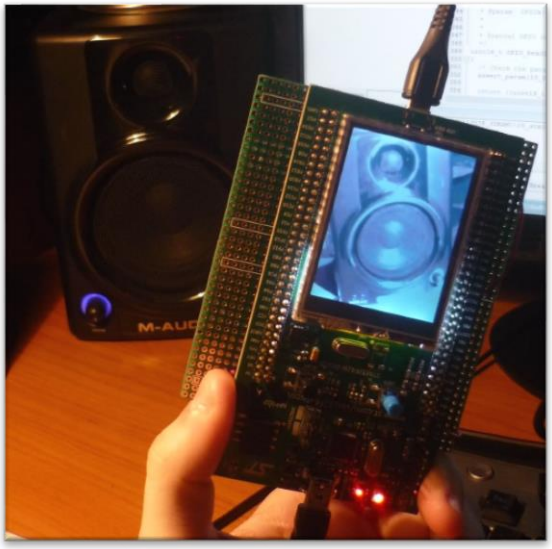


Obr. 27 - Propojovací deska kitu F429-Discovery

Schéma desky je uvedeno v příloze [2].

Po oživení kitu byla naprogramována aplikace pro vyzkoušení potřebných periferií, včetně těch, které jsou nové pro kit F429 (především DCMI, LCD a SDRAM). Bylo vyzkoušeno, že kolize pinů DCMI a LCD lze částečně vyřešit přenastavováním kolidujících pinů, tj. deaktivací rozhraní DCMI a aktivací LCD řadiče pro dobu zobrazení, a deaktivací LCD řadiče a opětovnou aktivací rozhraní DCMI pro dobu snímání. Po přenastavení pinů z režimu LCD zpět na režim DCMI nedokáže bohužel LCD displej udržet poslední informaci, a tak není možné zobrazovat obraz na LCD a současně snímat nový snímek CMOS sensorem.

Pro demonstraci kooperace těchto periferií byla vytvořena aplikace FOTOAPARÁT, která po stisknutí uživatelského tlačítka na kitu sejme kamerou obrázek a až do dalšího stisknutí tlačítka jej zobrazí na LCD displeji. Pro zjednodušení programu byla využita CROP funkce sensoru, která obrázek sama ořízne na velikost 240x320px, aby jej bylo možné na LCD displeji zobrazit v měřítku 1:1.



Obr. 28 - Demonstrace aplikace FOTOAPARÁT

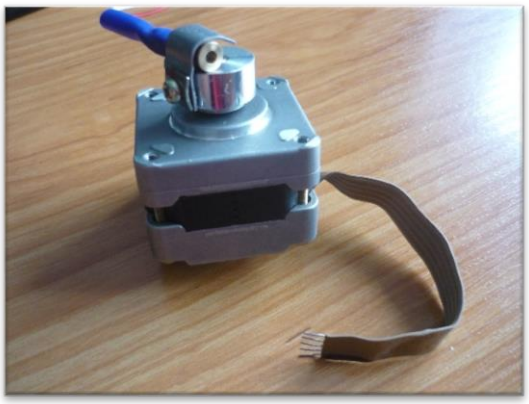
V další části této práce tak byl software, původně naprogramovaný pro kit F4-Discovery, modifikován a oživen i na kitu F429-Discovery. Modifikace softwaru spočívala především v odstranění kolizí pinů (viz předchozí strana), v inicializaci FMC řadiče a paměti SDRAM, a rekonfiguraci DMA řadiče pro zápis do SDRAM.

Vzhledem k nízké rychlosti rozhraní PC<->Discovery a omezené vnitřní paměti RAM MCU nebylo v případě použití kitu F4-Discovery možné odesílat do počítače obraz v plném rozlišení (dostupná paměť RAM po odečtení paměti typu CCM (Core-Coupled Memory), kterou nelze použít pro DMA, je 128kBytes na kitu F4, resp. 192kBytes na kitu F429, zatímco velikost celého obrazu je přibližně 351kBytes). Kit STM32F429 dává možnost uložit celý obraz, případně dokonce přes 20 snímků najednou, do vestavěné paměti SDRAM, a odeslání do PC provést

libovolně malou rychlostí až po dokončení snímání. Pro zobrazení snímku v plném rozlišení musela být rovněž mírně upravena PC aplikace.

8. Elektromechanická regulace směru laserového paprsku

Jak vyplývá ze zadání, součástí této práce je jednoduchý regulační systém využívající elektromechanického prvku pro regulování směru laserového paprsku vytvářejícího optickou stopu v zorném poli kamery. Tento paprsek bude směrován tak, aby kopíroval stopu druhého laserového paprsku, který je ovládán uživatelem. Pro polohování laserového modulu byl zvolen unipolární krokový motor.



Obr. 29 - Unipolární krokový motor s připevněným laserovým modulem

Princip řízení krokového motoru je jednoduchý. Pohyb rotoru je vyvolán aktivací cívek statoru v určitém pořadí (viz obr. 30), což je realizováno přiváděním napětí na jednotlivé cívky statoru. Kolem těchto cívek se pod napětím indukuje magnetické pole, které přitahuje zuby rotoru. Tím dochází k otáčení rotoru. Z právě uvedeného principu vyplývá fakt, že je prakticky nutné využít pro řízení krokového motoru alespoň základní logický obvod, neboť není možné motor řídit pouhými změnami napájecího napětí. Tato vlastnost krokového motoru sice znamená, že není možné roztočit motor pouhým připojením ke zdroji napětí, ale ve chvíli, kdy je vytvořený a zprovozněný řídicí obvod, však lze začít využívat podstatné výhody, které princip řízení krokového motoru přináší.

Jelikož má krokový motor konstrukčně daný počet kroků na jednu otáčku, má řídicí jednotka vždy přesnou informaci o tom, v jaké poloze se krokový motor nachází. Z toho vyplývá další podstatná výhoda krokového motoru, a tou je fakt, že motor lze řídit pouze přímovazební regulací. To ovšem platí, pouze pokud není překročena hodnota mezního zatížení či rychlosti (nedojde k „utržení“ motoru). V ten moment se totiž ztratí informace o tom, v jaké poloze se krokový motor právě nachází. Mezi další výhody pak patří nízká cena, vysoká spolehlivost, prakticky bezúdržbový provoz, a vysoký točivý moment v nízkých otáčkách. Nevýhodami jsou

nízký točivý moment ve vysokých rychlostech, nespojitý pohyb v nízkých rychlostech (rotor se pohybuje v krocích), a trvalý odběr proudu nezávislý na otáčkách motoru.

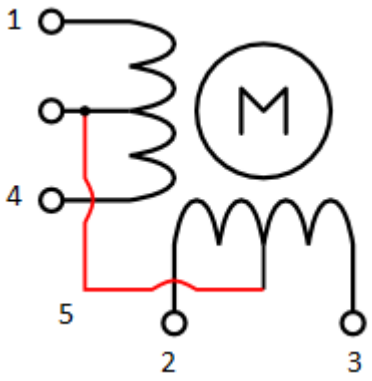
Ke krokovému motoru je připevněn miniaturní červený laserový modul 3VDC/20mA s vyzářeným výkonem zhruba 1mW, kterým bude sledována stopa druhého laseru, ovládaného uživatelem.

8.1. Řízení krokového motoru

Unipolární krokový motor je nejjednodušší variantou krokového motoru, a princip jeho funkce je velmi jednoduchý. Stator obsahuje cívky, ve kterých se pod proudem generuje magnetické pole. Rotor pak obsahuje prstenec permanentních magnetů, které tvoří „zuby" rotoru. Tyto jsou přitahovány magnetickým polem indukovaným na cívkách statoru. Postupným střídáním cívek statoru pak dochází k otáčení rotoru. Rotor mnou použitého motoru má 50 zubů, pro posun o jeden zub je potřeba vykonat čtyři kroky. Pro plnou rotaci je třeba udělat 200 celých kroků, takže jeden krok otočí rotorem o

$$\varphi = \frac{360}{200} = 1.8°.$$

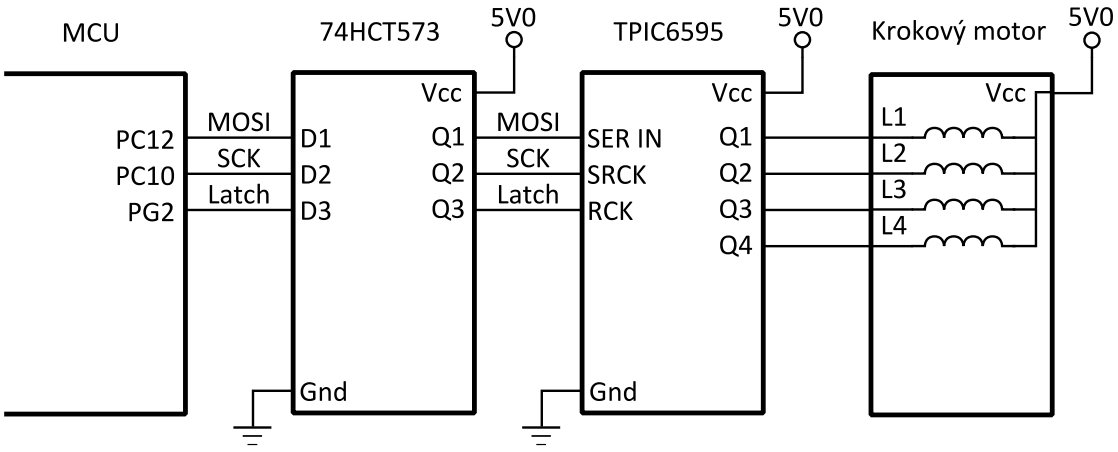
Využít lze rovněž alternativních variant řízení (půlkroky, mikrokroky), které zvyšují úhlové rozlišení, ale snižují maximální točivý moment motoru.



Obr. 30 - Schéma unipolárního krokového motoru
Zdroj: http://www.esuli.it/2011/10/11/converting-an-unipolar-stepper-to-bipolar

Pro řízení krokového motoru je vedle samotného mikrokontroléru využit ještě obvod TI TPIC6595, což je 8bitový posuvný registr se sériovým vstupem a paralelním výstupem, na jehož výstup je připojen záchytný registr tvořený klopnými obvody typu D s navázanými spínacími transistory NMOS (low side switch). Tento obvod se vyznačuje napájenými výstupy, což umožňuje tyto výstupy využít pro buzení součástek s relativně vysokým proudovým odběrem (jako je například použitý krokový motor). Vstup tohoto obvodu bohužel není kompatibilní s TTL úrovněmi ($V_{iH} = 0.85V_{CC}; V_{iL} = 0.15V_{CC}$), a tak je nutné mezi MCU a TPIC6595 zařadit

ještě převodník logických úrovní. K tomu poslouží obvod TI SN74AHCT573, což je osmibitový paralelní záchytný registr tvořený klopnými obvody typu D, jehož vstup je kompatibilní s TTL úrovněmi, a na jehož výstupu jsou úrovně $V_{oH} = V_{CC}; V_{oL} = 0$.



Obr. 31 - Zjednodušené schéma řídicího obvodu pro krokový motor

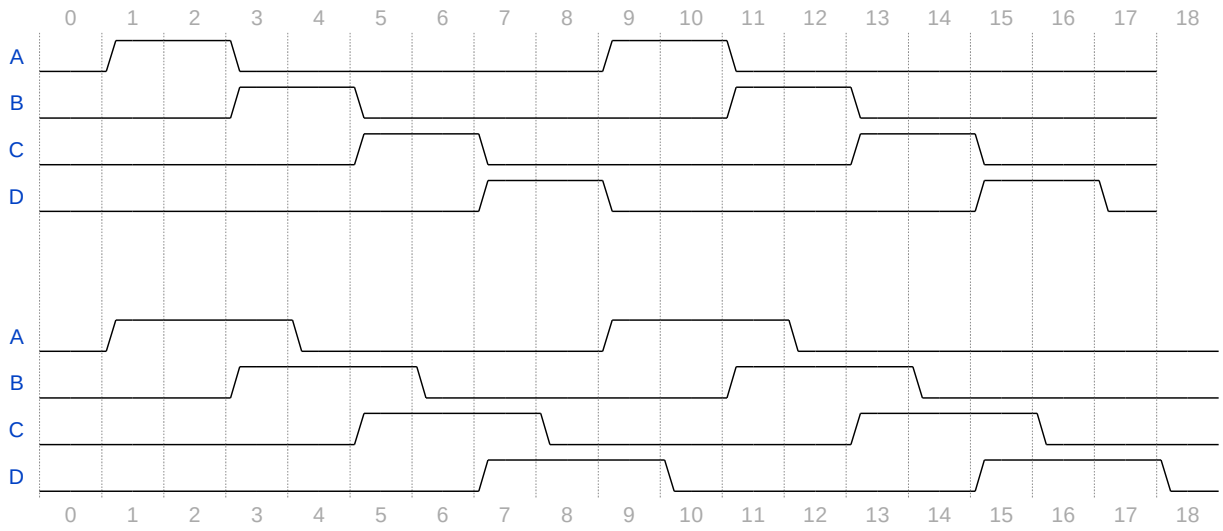
K samotnému řízení motoru se používají řídící byty (přesněji pouze půlslabiky, 1 půlslabika = 4 bity), pomocí kterých určujeme, kterými cívkami krokového motoru protéká proud, a které tedy budou v důsledku působení indukovaného magnetického pole přitahovat „zuby“ rotoru. Řídicí půlslabiky lze buď uložit do pole v paměti RAM a ve správném pořadí je číst, nebo stačí využít binárního posuvu, což je zřejmé z tabulky na následující straně.

Krok:	1.	2.	3.	4.	5.	6.	7.	8.
A	1	0	0	1	1	0	0	1
B	1	1	0	0	1	1	0	0
C	0	1	1	0	0	1	1	0
D	0	0	1	1	0	0	1	1

Tab. 9 - Řízení krokového motoru po celých krocích

Krok:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.
A	1	1	0	0	0	0	0	1	1	1	0	0	0	0	0	1
B	0	1	1	1	0	0	0	0	0	1	1	1	0	0	0	0
C	0	0	0	1	1	1	0	0	0	0	0	1	1	1	0	0
D	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1

Tab. 10 - Řízení krokového motoru po půlkrocích



Obr. 32 - Řídicí signály krokového motoru, nahoře plný krok, dole půlkrok

K odesílání řídicích dat je využito rozhraní SPI (Serial Peripheral Interface), čímž je ušetřen procesorový výkon pro další aplikace. Výstup MOSI (Master Output, Slave Input) periferního rozhraní je připojen přes převodník logických úrovní na vstup Serial In obvodu TPIC6595, výstup SCK (Serial Clock) je připojen na vstup SRCLK (Shift Register Clock) TPIC6595, a výstup mikrokontroléru PG3 je v režimu „Output“ připojen na vstup RCLK (Register Clock) TPIC6595, který na vysoké úrovni kopíruje data z posuvného registru na výstupní záchytný registr, a který je zároveň jako jediný softwarově ovládán. K výstupnímu registru obvodu TPIC6595 je pak připojen samotný krokový motor. K jednotlivým vinutím krokového motoru jsou ještě antiparalelně připojeny obyčejné diody, jejichž úkolem je odvést zpět do zdroje napěťové špičky, které vznikají naindukováním na vinutích elektromotoru. Schéma řídicího obvodu je uvedeno v příloze 2.

8.2. Použití laserového modulu

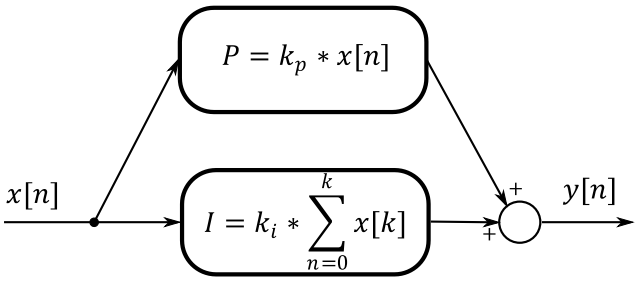
Ke krokovému motoru je připojen laserový modul 3VDC/20mA s vyzářeným výkonem zhruba 1mW. Laserová stopa je kamerou velmi dobře snímatelná, neboť má výrazně jinou hodnotu jasu než okolní prostředí, a její vyhodnocení algoritmem labellingu je díky její malé ploše velmi rychlé. Problém při sledování jedné laserové stopy druhou stopou spočívá v detekci dvou objektů v obraze a nemožností jednoduše určit která stopa je sledovaná, a která sledující. Možné řešení je periodicky vypínat sledující laser, což prodlužuje dobu vyhodnocení minimálně na dvojnásobek, neboť je nutné pro každý krok sejmout dva snímky. Jednodušší variantou je vertikálně oddělit obě stopy tak, aby z pohledu algoritmu nemohlo být pochyb o tom, která stopa je která.

Jsou-li obě laserově stopy vertikálně odděleny (z pohledu snímače), pak vzhledem k použitému řádkovému zpracování obrazu bude vždy zřejmé, která stopa je řízená, a která stopa je řídicí. Pro regulaci vzájemné polohy v jednom rozměru (vzhledem k použití jednoho krokového

motoru) stačí najít rozdíl v horizontálních souřadnicích detekovaných objektů, a převést tuto informaci na počet kroků, které musí krokový motor vykonat.

8.2.1. PI(PS) regulátor pro řízení polohy laserové stopy

Pro řízení polohy laserové stopy byl zvolen PI regulátor, respektive jeho diskrétní, proporcionálně-sumační (PS) varianta. Oproti použití čistě proporcionálního regulátoru PS regulátor eliminuje trvalou regulační odchylku a zlepšuje stabilitu regulátoru. Regulátor PS vznikne paralelním spojením proporcionálního a sumačního regulátoru. Základní schéma PS regulátoru je uvedeno níže, přičemž $x[n]$ regulační odchylka (vzdálenost bodů), a $y[n]$ je žádaný počet kroků.



Obr. 33 - Základní schéma PS regulátoru

Pro realizaci regulátoru jsou v programu implementovány vedle funkce turnMotor(steps), která otáčí motorem o daný počet kroků, dále funkce findDifference(), která spočte vzdálenost bodů, a funkce adjust(), která provede samotnou regulaci a volá funkci turnMotor. Funkce adjust() vypadá následovně:

```
signed int rozdil, numOfSteps;
findDifference();
if (difference > n && difference < -n) integratorSum += difference;
else numOfSteps =     (signed int)(regP * (float)difference)
                    + (signed int)(regI * integratorSum);

turnMotor(numOfSteps);
```

Protože krokový motor má konečné rozlišení, je před integrátorem implementováno malé pásmo necitlivosti, které by se přibližně mělo rovnat rozlišení krokového motoru), a které má za úkol zamezit integrování kolem stabilní polohy.

8.3. Rychlostní přizpůsobení aplikace

Aby regulace polohy probíhala co nejrychleji, byl program upraven tak, aby docházelo k co nejmenšímu plýtvání časem. S využitím paměti SDRAM, která eliminuje nutnost zpracovávat data průběžně, lze sensoru nastavit maximální frekvenci hodinového signálu 21MHz. Samotné zpracování obrazu je provedeno až po sejmutí celého snímku. Výkon tohoto algoritmu je zhruba 8,5sn./s. K řešení tohoto problému lze však také s výhodou využít průběžného

zpracování i přesto, že je nyní k dispozici dostatek paměti pro uložení celého snímku. Tento přístup zrychlí vykonávání programu zhruba na 12 sn./s., což je mírně vyšší frekvence než s použitím kitu F4-Discovery. Tento nárůst lze připsat mírně vyšší frekvenci MCU na kitu F429-Discovery (viz kap. 3.2).

Jak bylo vysvětleno v kapitole 4, algoritmus Connected-component labellingu neběží v konstantním čase, ale je závislý na náročnosti vstupních dat. Proto musí při průběžném zpracovávání existovat časová rezerva mezi dobou snímání a dobou zpracování. Jak je vidět v kapitole 5.6, průměrná rezerva představuje zhruba čtvrtinu celkového času potřebného pro sejmutí jednoho bloku. Proto je snímání celého snímku při průběžném zpracování o necelou čtvrtinu delší, než by mohlo být v ideálním případě. Odstraněním těchto mezer, společně v kombinaci s vysokou frekvencí hodin sensoru při snímání do SDRAM, se proto mírně sníží výhoda průběžného zpracování vůči off-line zpracování.

Úzkým hrdlem celé demonstrační aplikace je však použitý krokový motor. I v případě, že bude docházet jen k drobným změnám polohy, není použitý krokový motor schopen provést více než přibližně 6,5 regulačních zásahů za sekundu. Proto v tomto případě příliš nezáleží na metodě, která je pro řešení této úlohy zvolena.

9. Závěr

V rámci bakalářské práce jsem úspěšně implementoval minimalizované metody zpracování obrazu s využitím procesorů ARM Cortex-M4F na kitech STM32F4-Discovery a STM32F429-Discovery. V rámci řešení jsem přizpůsobil již existující hardware pro využití s kitem F429-Discovery s rozšířenou pamětí typu SDRAM. Nakonec jsem vytvořil demonstrační aplikaci pro zpětnovazební řízení polohy světelné stopy v závislosti na poloze druhé stopy, ovládané uživatelem. K tomuto účelu jsem využil unipolárního krokového motoru a dvou laserových modulů.

Na začátku této práce jsem se seznámil s procesorem ARM Cortex-M4F (STM32F407VGT6 a STM32F429ZIT6), který je součástí kitů STM32F4-Discovery a STM32F429-Discovery společnosti STMicroelectronics. Začal jsem úpravou vzorového programu pro předmět 38NVS – Návrhy vestavných systémů, psaného v assembleru a pokračoval zprovozněním a ovládáním základních periferií těchto kitů. Po tomto základním seznámení s možnostmi mikrokontrolérů jsem implementoval dvě metody pro zpracování obrazu, metodu s pracovním názvem „překrývající se pruhy“, a obecnější a náročnější metodu Connected-component Labellingu. Tuto metodu jsem navíc optimalizoval použitím rozhodovacích stromů a algoritmu Union-Find pro zajištění maximální rychlosti.

Tyto metody zpracování obrazu jsem nejdříve aplikoval na obraz ve sníženém rozlišení pro ověření jejich funkčnosti. Poté jsem analyzoval možnosti průběžného zpracování obrazu a ověřil využitelnost DMA řadiče pro režim střídavého ukládání snímaných dat do dvou paměťových lokací v paměti RAM mikrokontroléru. Zmíněné metody zpracování obrazu jsem upravil pro průběžné zpracování, a úspěšně je implementoval pro kit F4-Discovery při zachování rozumné snímkové frekvence přibližně 11sn./s. Dále byla implementována metoda pro korekci vinětace, kterou systém, vzhledem k použitému objektivu, výrazně trpí. Použití korekce vinětace je však výpočetně náročné a snižuje snímkovou frekvenci přibližně na 6,5sn./s. Aby byla za každých podmínek zachována co nejvyšší rychlost snímání, implementoval jsem softwarové měření rychlosti, na jehož základě program neustále vyhodnocuje, zda existuje prostor pro zrychlení, nebo naopak zda se stíhá obraz při dané rychlosti snímání zpracovávat. V závislosti na velikosti časové rezervy při zpracování pak program sám upravuje frekvenci hodinového signálu CMOS sensoru.

V další části této práce jsem navrhnul a realizoval prototypovou desku pro připojení obrazového sensoru ke kitu F429-Discovery, neboť nebylo z důvodu odlišného rozložení konektorů možné využít tu, která byla již dříve vytvořena pro kit F4-Discovery. Kit F429-Discovery obsahuje 8MB vestavěné paměti SDRAM, kterou lze využít pro uložení celého obrazu, čímž odpadá nutnost průběžného zpracování. Po propojení kitu a CMOS modulu jsem naprogramoval demo aplikaci

„fotoaparát“, která sejme obraz z CMOS sensoru a zobrazí jej na vestavěném TFT LCD displeji kitu, a to i přesto, že připojení kamery a LCD displeje kolidují, což jsem vyřešil softwarově. Následně jsem pak metody zpracování obrazu z kitu F4-Discovery přizpůsobil pro využití kitu F429-Discovery a jeho vestavěné SDRAM paměti.

V poslední části práce jsem vytvořil demonstrační aplikaci využívající CMOS sensor, krokový motor a dva laserové moduly, z nichž jeden je uchycen na krokový motor. Principem této aplikace je sledování jedné laserové stopy, ovládané uživatelem, druhou laserovou stopou, ovládanou krokovým motorem. Ke sledování laserových stop a vyhodnocování jejich poloh jsem využil vytvořených algoritmů zpracování obrazu. Pro řízení krokového motoru jsem vytvořil obvod sestávající z obvodu 74HCT573, fungujícího jako převodník logických úrovní, a z obvodu TPIC6595, což je osmibitový posuvný registr se záchytným výstupním paralelním registrem. Pro regulaci polohy řízeného laseru, připevněného na krokovém motoru, jsem využil PI regulátoru, resp. jeho diskrétní varianty. Výsledný regulátor sleduje a vyhodnocuje vzájemnou polohu obou laserů s frekvencí přibližně 6,5sn./s.

Pro snadné zobrazení veškerých dat a pro řízení mikrokontroléru jsem vytvořil PC aplikaci v jazyce C#, která s mikrokontrolérem komunikuje po virtuálním sériovém portu přes sběrnici USB.

10. Literatura

[1] MICHÁLEK, Adam. Počítání a třídění objektů metodou průběžného zpracování obrazu. Praha, 2011. Bakalářská práce. ČVUT FEL.

[2] PAVLÍČEK, Tomáš. Metody průběžného zpracování obrazu a jejich implementace do signálového procesoru Blackfin ADSP-BF532. Praha, 2008. Diplomová práce. ČVUT FEL.

[3] DOKOUPIL, Vojtěch. *Zpracování obrazu vestavěným mikrořadičem*. Praha, 2013. Bakalářská práce. ČVUT FEL.

[4] STM32F4 Reference Manual. In: *STM32F4 Reference Manual* [online]. 2013 [cit. 2014-01-06]. Dostupné z: http://www.st.com/web/en/resource/technical/document/reference_manual/DM00031020.pdf

[5] STM32F4 Discovery Kit User Manual. In: *STM32F4 Discovery Kit User Manual* [online]. 2013 [cit. 2014-01-06]. Dostupné z: http://www.st.com/st-web-ui/static/active/en/resource/technical/document/user_manual/DM00039084.pdf

[6] ARM Cortex-M4 Technical Reference Manual. In: *ARM Cortex-M4 Technical Reference Manual* [online]. 2013 [cit. 2014-01-06]. Dostupné z: http://infocenter.arm.com/help/topic/com.arm.doc.ddi0439d/DDI0439D_cortex_m4_processor_r0p1_trm.pdf

[7] WU, Kesheng, Ekow OTOO a Kenji SUZUKI. Optimizing two-pass connected-component labeling algorithms. *Pattern Analysis and Applications*. 2009, vol. 12, issue 2, s. 117-135. DOI: 10.1007/s10044-008-0109-y. Dostupné z: http://link.springer.com/10.1007/s10044-008-0109-y

[8] YIU, Joseph. *Definitive guide to the ARM Cortex-M3*. Boston: Newnes, c2007, xix, 359 p. ISBN 978-075-0685-344.

[9] Aptina MT9V034 Data Sheet. In: Aptina MT9V034 Data Sheet [online]. 2008 [cit. 2014-05-22]. Dostupné z: http://www.aptina.com/assets/downloadDocument.do?id=406

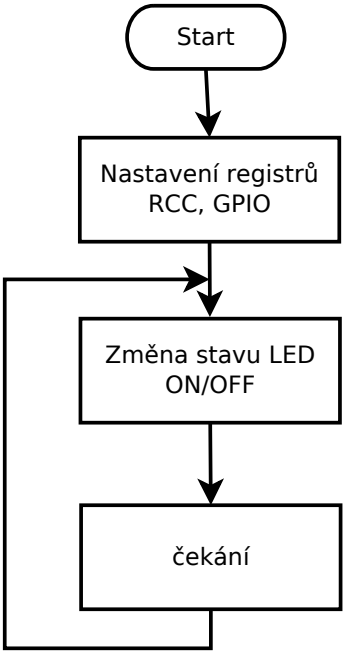
Příloha 1 – Programy sloužící k seznámení se s kity Discovery

Vzhledem k tomu, že jsem před započetím práce na samotné bakalářské práci neměl žádné předchozí zkušenosti s programováním mikroprocesorů architektury ARM, vytvořil jsem několik jednoduchých programů pro seznámení se s principy programování pro ARM a rovněž pro ověření správně funkčnosti kitu. Tyto příklady zde uvádím pouze z důvodu, že se mohou hodit těm z mých případných následovníků, kteří budou, stejně jako já, začínat s nulovou znalostí programování ARM mikroprocesorů.

Vzorové programy

Blikání LED v Assembleru

Pro seznámení se s programováním procesoru STM32F4 na nejnižší úrovni, tj. v assembleru jsem modifikoval program LED_BLIK vytvořený p. Michalem Tomášem pro potřeby výuky v předmětu 38NVS – Návrhy vestavných systému. V tomto předmětu se pracuje s vývojovými kity STM32VL Discovery, které jsou osazeny procesory ARM Cortex-M3.



Obr. 34 - Vývojový diagram programu „LED v Assembleru“

Původní program byl vytvořen pro demonstraci základního oživení kitu STM32VL Discovery – nastavení hodin periferií, nastavení parametrů vstupně-výstupních portů, a samotné užitečné aplikace, v tomto případě blikání uživatelských LED diod.

Po resetu procesoru jsou základní registry, například pro nastavení frekvence jádra kontroléru, automaticky nastaveny na správné hodnoty, což nám ulehčí práci. Chceme-li využívat LED diody, je ovšem potřeba povolit hodiny pro periferie a správně nastavit registry periferií (GPIO).

Naše LED dioda je na pinu P15 GPIO brány D (PD15)[5], je tedy nutné povolit hodiny pro bránu GPIOD, což provedeme RCC AHB1 peripheral clock enable registrem (RCC_AHB1ENR). Prozkoumáním Referenčního manuálu[4] zjistíme, že je potřeba nastavit bit 4 na hodnotu 1(high). Následně je potřeba nastavit registry GPIOD_MODER, GPIOD_OTYPER a GPIOD_OSPEEDR, tak abychom získali parametry „výstup, 50MHz, Push-Pull, No-Pull“.

6.3.12 RCC AHB1 peripheral clock enable register (RCC_AHB1ENR)

Address offset: 0x30

Reset value: 0x0010 0000

Access: no wait state, word, half-word and byte access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reser-ved	OTGHS ULPIEN	OTGHS EN	ETHMA CPTPE N	ETHMA CRXEN	ETHMA CTXEN	ETHMA CEN	Reserved		DMA2EN	DMA1EN	CCMDATA RAMEN	Res.	BKPSR AMEN	Reserved	
	rw	rw	rw	rw	rw	rw			rw	rw			rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CRCEN	Reserved			GPIOIE N	GPIOH EN	GPIOGE N	GPIOFE N	GPIOEEN	GPIOD EN	GPIOC EN	GPIOB EN	GPIOA EN
			rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

Obr. 35 - Ukázka z referenčního manuálu

Nastavení registru provedeme v assembleru následujícím způsobem[6]:

```
LDR   R0, =RCC_AHB1ENR ;Kopie adresy RCC_AHB1ENR do registru R0
LDR   R1, [R0]         ;Nacteni stavu registru RCC_AHB1ENR do R1
LDR   R2, =0x08        ;Konstanta pro zapnutí hodin pro bránu C
ORR   R1, R1, R2       ;Maskování
STR   R1, [R0]         ;Uložení nové hodnoty
```

RCC_AHB1ENR je pouze definované makro s adresou registru, která je 0x40020C00 (zjistíme z Referenčního manuálu pod klíčovým slovem register boundary addresses) s offsetem 0x30, tedy výsledná adresa je 0x40020C30.

Podobným způsobem nastavíme zbývající registry (nyní se již se nemusíme zabývat maskováním):

```
MOV   R2, #0x40000000
LDR   R0, =GPIOD_MODER
STR   R2, [R0]
LDR   R2, =0x8000
LDR   R0, =GPIOD_OTYPER
STR   R2, [R0]
MOV   R2, #0x80000000
LDR   R2, =GPIOD_OSPEEDR
STR   R2, [R0]
```

Principy nastavování ostatních periferií se od tohoto příkladu příliš neliší. Nyní je již potřeba jen kód, který bude blikat LED diodou zapisováním hodnot na Output Data Register (GPIOD_ODR). Toto celé můžeme uzavřít do nekonečné smyšky užitím návěští (např. SMYCKA)

```
SMYCKA
MOV   R1, #0      ; Vložení hodnoty 0 do R1
STR   R1, [R2]    ; Zápis hodnoty v R1 na adresu v R2,
                  ; tj. nulování všech bitu.
                  ; na bráně D (LED na PD15 nesviti)
MOV   R0,#100000 ; Vložení do RO hodnoty prodlevy,
                  ; tj. např. 100000 dekadicky
                  ; RO je v tomto případě jako vstupní parametr
                  ; podprogramu DELAY
BL    DELAY       ; Volání podprogramu DELAY s uložením návratové
                  ; adresy do LR
MOV   R1, #0x8000; Vložení hodnoty 0x00008000 do R1, konstanta pro
                  ; bit 15
STR   R1, [R2]    ; Zápis hodnoty v R1 na adresu v R2, tj. nastaveni
                  ; bitu 15
                  ; na bráně D (LED na PD15 sviti), ostatní bity
                  ; jsou nulovány
BL    DELAY       ; volání podprogramu pro zpoždění
B     SMYCKA      ; skok na navěští LOOP, tj. nekonečné opakování
                  ; smyčky (LED bliká)
```

Podprogram DELAY vypadá následovně:

```
DELAY             ; Navesti zacatku podprogramu
PUSH  {LR}        ; Ulozeni hodnoty navratove adresy - LR do zasobniku
                                                      ;
WAIT
SUBS  R0, R0, #1 ; Odecteni 1 od R0,
                  ; tj. R0 = R0 - 1 a nastaveni priznakoveho registru
BNE   WAIT        ; Skok na navesti pri nenulovosti

POP   {LR}        ; Navrat z podprogramu, obnoveni LR ze zasobniku
BX    LR          ; Navrat do hlavního programu
```

Využití LED, tlačítka v režimu externího přerušení a časovače v režimu PWM, jazyk C

Pro seznámení s programováním procesoru STM32F4 Discovery s využitím jazyka C v prostředí Keil µVision 4 jsem zhotovil program využívající uživatelské LED diody, tlačítka v režimu externího přerušení a časovač v režimu PWM pro řízení jasu LED diody.

Program vyčkává, dokud nepřijde externí interrupt od tlačítka připojeného na portu PA0, po přijetí interruptu začne blikat diodou LD4 na portu PD12. Jas diody je řízen časovačem Timer4 v režimu PWM. Výsledkem je hezký „fading“ efekt blikání.

Nejdřív je třeba povolit hodinový signál pro potřebné periferie - LED diody, tlačítko a časovač. Konzultací s referenčním manuálem[4] zjistíme, že je potřeba nastavit příslušné bity v registrech RCC_AHB1ENR (Pro LED jde o bit GPIOD, pro tlačítko GPIOA), RCC_APB1ENR (Timer – bit TIM4). Využijeme-li knihovny STM32F4xx_StdPeriph, jde o jednoduchou operaci.

```
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE); //LED
RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM4, ENABLE);  //TIMER
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE); //tlacitko
```

Dále je potřeba nastavit parametry jednotlivých periferií – LED, tlačítka, timeru, externího interruptu a NVIC (Nested Vectored Interrupt controlleru). Knihovna STM32F4xx_StdPeriph pro tyto účely využívá pro zpřehlednění a zjednodušení kódů tzv. inicializačních struktur, do kterých zapíšeme jednotlivé požadované parametry (definované množstvím direktiv DEFINE a struktur), a pak provedeme inicializaci. Příklad inicializace LED ve standardním režimu následuje.

```
GPIO_InitTypeDef  GPIO_InitStructure;                  //struktura
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_13;             //vyběr pinu
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;          //výstupní pin
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;     //rychlost pinu
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;         //typ Push-Pull
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP ;          //režim Pull Up
GPIO_Init(GPIOD, &GPIO_InitStructure);                 //inicializace
```

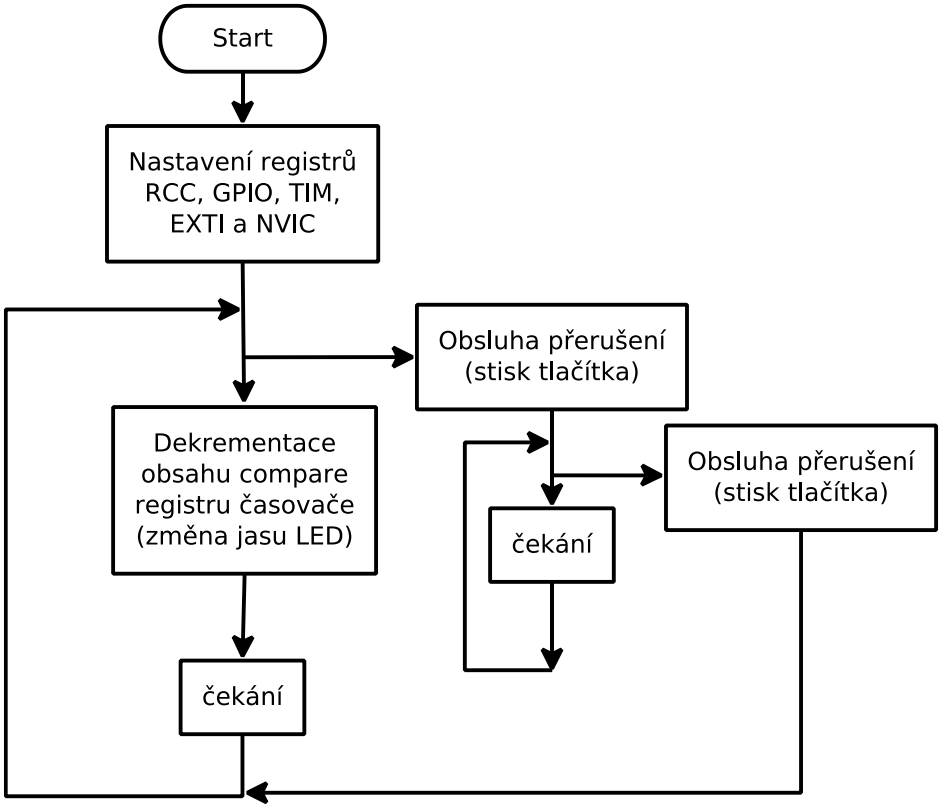
Funkce GPIO_Init se pak již sama postará o ověření a zápis hodnot do registrů GPIOD_MODER, GPIOD_OSPEEDR, GPIOD_OTYPER a GPIOD_PUPDR. Podobným způsobem lze nastavit časovač (TIM), vnější přerušení (EXTI), a Nested Vectored Interrupt Controller (NVIC), který má na starosti mimo jiné též řízení priorit přerušení.

Obsluhu externího přerušení má na starost funkce void EXTI0_IRQHandler(void) v souboru stm32f4xx_it.c, která pouze přepíná příznak programBezi, podle kterého se řídí hlavní funkce programu.

```
void EXTI0_IRQHandler (void) {
      if(EXTI_GetITStatus(EXTI_Line0) != RESET) {
            switch (programBezi){
                  case 0:    programBezi = 1;
                             break;
                  case 1:    programBezi = 0;
                             break;
                  }
            //vymazani priznaku preruseni
            EXTI_ClearITPendingBit(EXTI_Line0);
            };
}
```

Hlavní funkce programu pak provádí dekrementaci hodnoty output compare registru, což má za následek kratší PWM pulsy a snižování jasu LED diody.

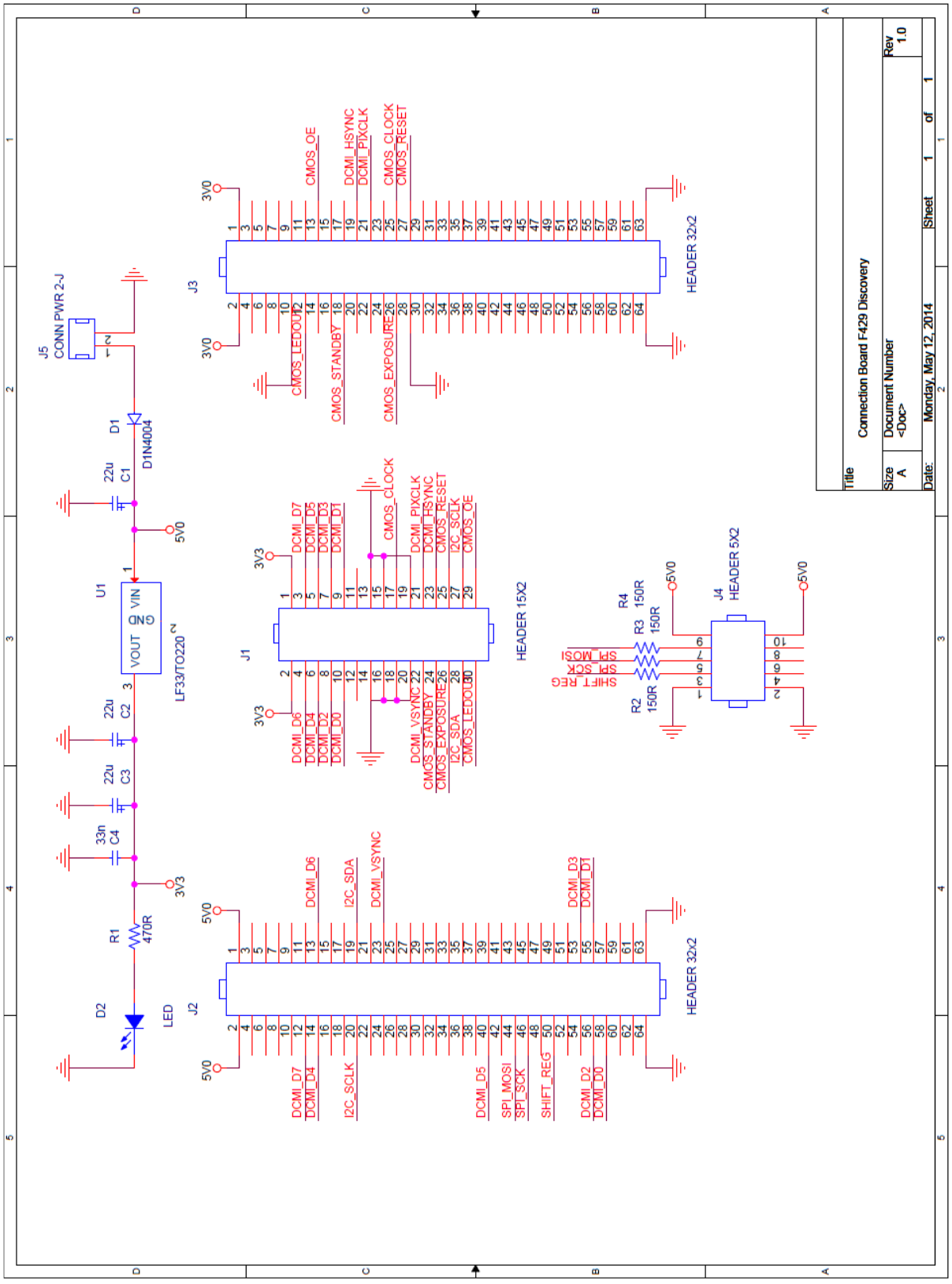
```
while(1){
      if (programBezi == 0) continue;       //dokud programBezi==0
                                            //program stoji
      intensita++;                          //inkrementuj OC registr
      TIM4->CCR1 = 666 - (intensita % 666); //zapis od OC registru
      for(i=0;i<60000;i++);                 //cekani
}
```



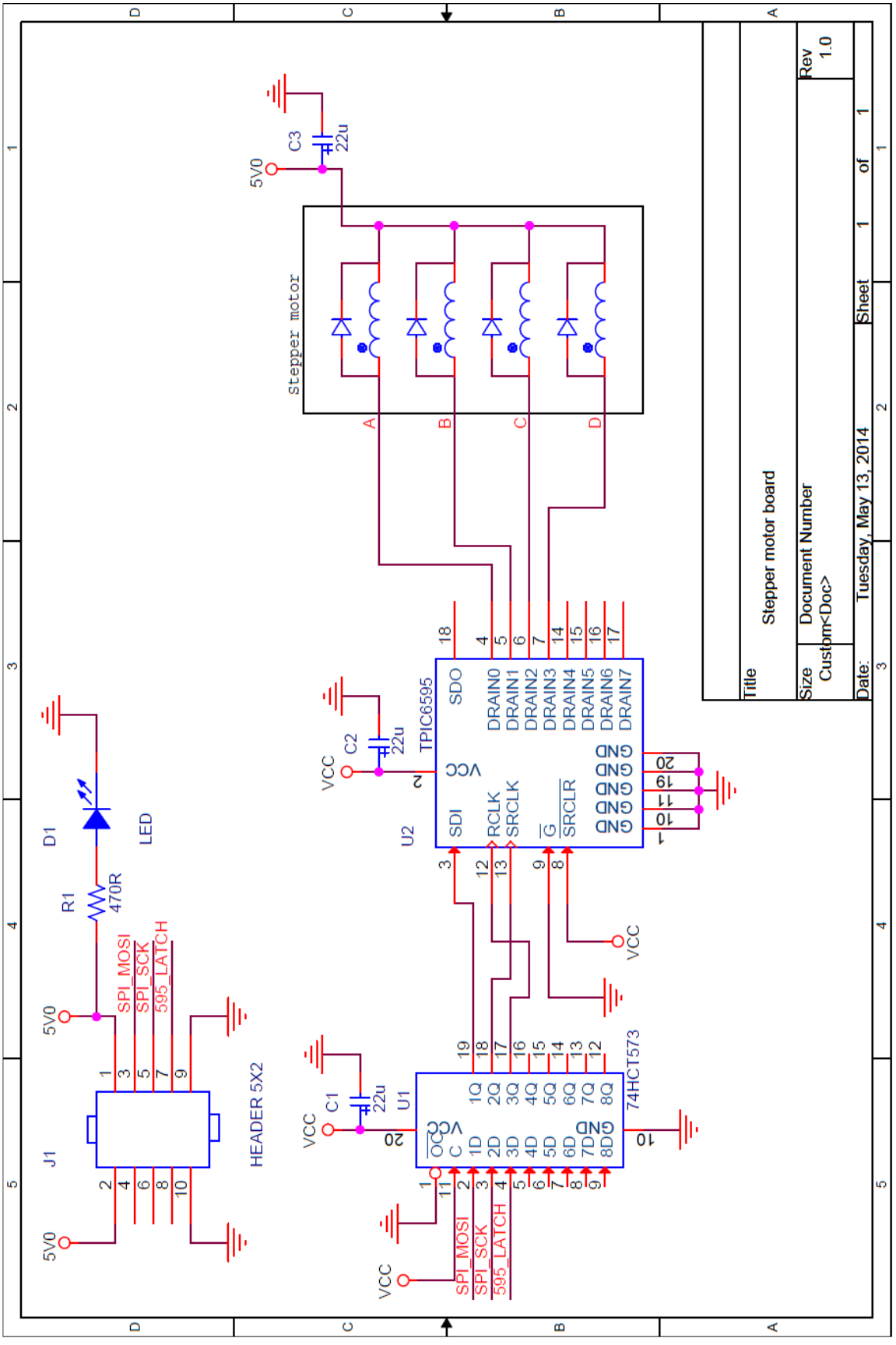
Obr. 36 - Vývojový diagram demonstračního programu v C

Příloha 2 – obvodová schémata

Propojovací deska CMOS sensoru a kitu F429 Discovery



Obvod pro řízení krokového motoru



Příloha 3 – Obsah přiloženého CD

- Bakalářská práce ve formátu PDF
- Aplikace a programy
    - Programy pro MCU
        - LED v Assembly
        - LED, external interrupt a PWM v jazyce C
        - DMA Test
        - STM32F4 Průběžné zpracování
        - STM32F429 Fotoaparát
        - STM32F429 Průběžné zpracování s regulací polohy
        - STM32F429 Offline zpracování s regulací polohy
    - PC Aplikace
        - DMA Test
        - F4 image Processing
        - F429 Image Processing
- Zdroje BP
- Další materiály