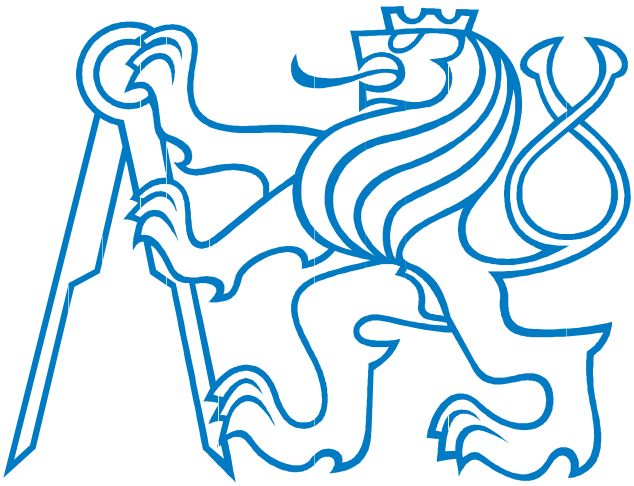
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA ELEKTROTECHNICKÁ
KATEDRA MĚŘENÍ



DIPLOMOVÁ PRÁCE

Detekce optických značek pro řízení polohy

Bc. Daniel Toms

Vedoucí práce: doc. Ing. Jan Fischer, CSc.
Studijní program: Kybernetika a robotika
Studijní obor: Senzory a přístrojová technika

Praha 2016

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
Katedra měření

Akademický rok **2015-16**

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student:	**Bc. Daniel Toms**
Studijní program:	**Kybernetika a robotika**
Obor:	**Senzory a přístrojová technika**
Název tématu česky:	**Detekce optických značek pro řízení polohy**
Název tématu anglicky:	**Detection of Optical Markers for Position Control**

Pokyny pro vypracování:

Navrhněte metody pro polohování objektu s využitím zpracování obrazu a detekce optických značek. Orientujte se na mikrořadiče řady STM32F4xx, případně STM32F7xx a obrazové senzory CMOS. Navrhněte formu optických značek a prověřte jejich vhodnost experimentálně. Realizujte systém, který bude mechanicky nastavovat polohu na základě detekce takových značek. Vytvořte potřebné programy pro mikrořadič i pro nadřazené PC.

Seznam odborné literatury:

[1] Yiu, J.: Definitive Guide to ARMR CortexR-M3 and CortexR-M4 Processors
[2] STMicroelectronics: RM0090 Reference manual
[3] STMicroelectronics: DM00071990, STM32F429 Data

Vedoucí diplomové práce:	doc. Ing. Jan Fischer, CSc.
Datum zadání diplomové práce:	14. prosince 2015
Platnost zadání do[1]:	30. září 2017

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta elektrotechnická

Doc. Ing. Jan Holub, Ph.D.
vedoucí katedry

Prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 14. 12. 2015

[1] Platnost zadání je omezena na dobu tří 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: 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 diplomové práce doc. Ing. Janu Fischerovi, CSc. za cenné rady, podklady a konzultace, které mi pomohly při vypracování této práce.

Abstrakt (Česky)

Cílem této práce byl návrh metod pro detekci a přesné polohování objektu s využitím kamerového systému a polohovacího systému. Součástí práce je návrh a realizace tohoto polohovacího systému. Objekt, který je doplněný optickou značkou a je uchycený k polohovacímu systému, je snímán kamerovým systémem a jeho pozice je regulována na základě vyhodnocení snímku tímto systémem pořízeného. Vedle samotné polohovací mechaniky je součástí celého systému rovněž mikrokontrolér s jádrem ARM Cortex-M4F a monochromatický WVGA CMOS sensor Aptina MT9V034. Byly navrženy dva druhy optických značek, které se dají systémem snímat a vyhodnocovat – jednoduchá značka ve formě jednoho bodu a pak také soustava značek tvořících jednorozměrné měřítko, ve kterém je zakódována aktuální poloha podobně jako na klasickém pravítku, pouze bez použití čísel. Zpracování dat probíhá v nativním rozlišení sensoru. Důraz je kladen na rychlost zpracování tak, aby výsledný systém řízený obrazovou informací byl dostatečně rychlý pro praktické použití. Přesto však byly základní algoritmy zpracování obrazu doplněny o další pre-/post-processingové algoritmy, které zvýšily spolehlivost a přesnost řídicího systému.

Abstract (English)

The main focus of this thesis was to propose and create methods for the detection and precise regulation of an object using an optical camera system and a mechanical movement system. The mechanical movement system itself also had to be developed during the course of this work. The object, which has an optical marker attached to it and is connected with the movement system, is captured by the camera system and its position is regulated based on the data gathered from this captured image. Aside from the movement itself, the system also consists of an ARM Cortex–M4 based evaluation kit and an Aptina MT9V034 monochrome WVGA CMOS sensor. Two types of detectable optical markers have been developed – a simple one-point marker and also a more complex system of multiple markers that has its position encoded into it, much like a regular ruler but without the numbers. The CMOS sensor operates at full resolution. The speed of the application is the main concern so that the resulting system provides reasonable framerates for real-world operation. The used image processing algorithms have, however, been enhanced by some more pre-/post-processing algorithms in order to improve the reliability and accuracy of the entire system.

Obsah

Seznam obrázků

Seznam tabulek

1. Úvod

Tato práce se bude zabývat návrhem metod pro detekci a přesné polohování objektu s využitím kamerového systému a polohovacího systému. Součástí práce je také návrh a realizace tohoto polohovacího systému. Tento polohovací systém se pak bude pohybovat na základě dat získaných snímáním a vyhodnocením detekovaných objektů, které budou mít formu optických značek a budou uchyceny k polohovacímu systému. K tomu bude sloužit sestava CMOS sensoru a mikrokontroléru s jádrem ARM Cortex-M.

Optické značky by měly být svázány s polohou polohovacího systému tak, aby změnou pozice polohovacího systému došlo zároveň ke změně polohy optických značek v zorném poli snímacího systému. Výsledný produkt by měl mít podobu demonstrační aplikace, dostatečně veliké pro praktické použití například v rámci výuky v laboratoři videometrie, ale zároveň dostatečně malé pro občasné přenášení na různé promo akce pořádané katedrou či fakultou.

Smyslem snahy provádět regulaci polohy posuvného systému na základě obrazových dat je otestování možnosti použití takového řešení v případech, kdy není možné regulovat polohu na základě dat z jiného snímače (například inkrementálního snímače). Další výhodou zamýšleného řešení je možnost rozšířit celé řešení na dvouosou 2-D regulaci.

V rámci této diplomové práce by mělo být navrženo více variant regulace polohy na základě snímání různých druhů značek. Vedle snímání jednoduché značky, kdy je snaha posouváním pojezdu posunout značku na předem danou polohu v zorném poli kamery, by tak měla být navržena a vyzkoušena i pokročilejší regulace s využitím měřítka, ve kterém je nějakým způsobem zakódována i informace o aktuální poloze v rámci měřítka (podobně jako u čísel na klasickém centimetrovém pravítku). Zřejmou výhodou pokročilejší varianty regulace je, že rozsah regulace může být podstatně větší, než je zorné pole kamery (tedy že stačí snímat pouze malou část uvažovaného měřítka).

Uživatel by měl být schopen měnit parametry regulace, ať již prostřednictvím ovládací aplikace pro počítač, nebo pomocí vestavěného LCD displeje na použitém hardwaru.

K regulaci polohy je třeba implementovat snímání obrazu optickým senzorem a detekci optických značek v sejmutém obrazu. K tomu bude použit již hotový hardware, založený na monochromatickém WVGA senzoru a mikrokontroléru s jádrem ARM Cortex-M4/M7. Krátce před započetím této práce byly uvedeny do prodeje nové a výkonnější mikrokontroléry s jádrem ARM Cortex-M7, proto bude snaha přizpůsobit navrhovanou aplikaci i pro použití s nimi. To však bude znamenat i nový vývoj některých hardwarových komponent.

Výkony těchto mikrokontrolérů by byly ještě před několika lety něčím nevídaným, a tak je dnes reálně možné používat takový hardware i k výpočetně náročným úlohám snímání a zpracování obrazu. Součástí těchto mikrokontrolérů jsou navíc také dedikovaná rozhraní, která jsou přímo určena ke zpracování obrazu z kamer (STM32 DCMI – Digital Camera Interface), což dále podporuje myšlenku využití pro zpracování obrazu. S výkony použitých mikrokontrolérů Cortex-M4F (180MHz, 225DMIPs) a Cortex-M7 (216MHz, 462DMIPs) se tak dá předpokládat

možnost poměrně důkladného zpracování obrazu o rozlišeních VGA až XGA (v závislosti na náročnosti použitých algoritmů) a to při zachování použitelné snímkové frekvence (cca 10Hz+). Výkonnějším mikročipům (například ARM Cortex-A či signálové procesory) samozřejmě tyto mikrokontroléry svým výkonem konkurovat nemohou, proto je stále nutná i analýza náročnosti navrhovaných algoritmů a pochopitelně i dobrá optimalizace kódu.

Smyslem snahy umístit náročné algoritmy image processingu přímo do embedded zařízení (mikrokontroléru se snímačem) je vyloučení výkonného stolního počítače z celého procesu. To má mnohé výhody: není nutné mít velký a drahý stolní počítač a výsledné zařízení je tedy velmi skladné a energeticky i cenově nenáročné, také není nutné řešit výměnu velkého množství dat mezi tímto počítačem a mikrokontrolérem. Ve srovnání kontrolérů Cortex-M s jinými mikrokontroléry (Cortex-A, signálové procesory) pak mezi výhody patří především nižší pořizovací cena a nižší spotřeba energie.

Základem algoritmů detekce optických značek budou metody zpracování obrazu popsané v [1], které budou dále optimalizovány a doplněny dalšími algoritmy pro zpracování a předzpracování obrazu tak, aby byl získán co nejkvalitnější (a především nejrobustnější) výstup. Zároveň ale budou metody zpracování obrazu navrhované v této práci optimalizovány pro co nejmenší časovou náročnost. Příliš nízká snímkovací frekvence by negativně ovlivňovala reakční schopnost celého snímacího a regulačního systému.

Stručně vzato je cílem této práce tedy implementovat a optimalizovat metody strojového vidění pro detekci optických značek (jednotlivé značky či složitější soustavy značek). Na základě dat ze senzoru a jejich zpracování navrženými algoritmy pak bude řízen polohovací systém. Cílovou polohu polohovacího systému určí uživatel pomocí uživatelského rozhraní. Výsledná aplikace musí být dostatečně rychlá na to, aby byla i při použití relativně slabého mikrokontroléru prakticky využitelná pro snímání ve vysokém rozlišení s dostatečnou snímkovou frekvencí.

2. Problematika embedded zpracování obrazu

V této kapitole bude provedena stručná rekapitulace rozborů a výsledků dosažených v [1]. Tam byla analyzována možnost návrhu rychlých algoritmů zpracování obrazu, které by byly dostatečně rychlé pro použití s mikrokontroléry ARM Cortex-M a jejich omezeními (malá paměť RAM, relativně nízký výkon ve srovnání například s procesory Cortex-A nebo signálovými procesory). Byly použity některé obecně známé algoritmy (jako například connected-component labelling pro detekci kontrastních objektů v obraze), které byly analyzovány a optimalizovány pro co nejvyšší rychlost zpracování (především eliminací slepých větví algoritmu). Tato algoritmy byly vyzkoušeny na kitu STM32F4-Discovery a v závislosti na rozsahu preprocessingu (korekce vinětace, Gaussovo rozostření) bylo docíleno snímkové frekvence přibližně 6-11sn./s.

Závěrem tedy bylo, že tyto mikrokontroléry jsou s určitými omezeními použitelné pro aplikace základního zpracování obrazu. Bezpochyby největším omezením výsledné aplikace byl nedostatek paměti RAM, do které nebylo možné uložit celý snímek (pokud nebyl v opravdu nízkém rozlišení), a tak bylo nutné obraz zpracovat průběžně, a tedy minimálně takovou rychlostí, jakou nám jej poskytoval kamerový modul.

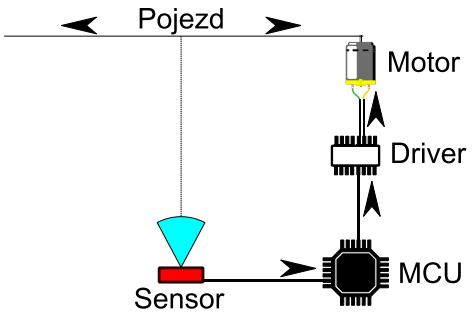
V této práci bude využit pokročilejší hardware, disponující externí pamětí typu SDRAM. Využití externí paměti umožní věnovat snímkům dostatek procesorového času a tedy výpočetního výkonu. Tím bude eliminováno omezení popsané v předchozím odstavci - CMOS senzory použité v této práci mají obecně nějakou minimální frekvenci, se kterou poskytují data. Chceme-li je tedy zpracovávat mikrokontrolérem, který nemá dostatek paměti ani na uložení jednoho celého snímku, je nutné tato data zpracovávat alespoň stejnou rychlostí, jakou jsou nám poskytovány. Toto omezení tedy s využitím externí paměti padá. V zájmu zachování použitelnosti celé aplikace ale bude snaha udržet celý systém co nejrychlejší.

Díky použití lepšího hardware bude možné navrhovanou aplikaci rozšířit o další algoritmy zpracování obrazu – toto je tedy jedna z oblastí, které se tato práce bude věnovat. Výsledná aplikace by tak měla být robustnější - méně náročná na kvalitu snímané scény i na intenzitu a homogenitu osvětlení.

Jak již bylo řečeno, dnešní mikrokontroléry disponují nevídanou výpočetní silou v poměru k ceně a také spotřebě. Algoritmy image processingu nicméně zpravidla spoléhají na hrubou výpočetní sílu, a tak je jejich použití i tak komplikované a velmi závislé na dobré optimalizaci. Podobně jako v [1] bude obraz snímán monochromatickým sensorem MT9V034 s rozlišením 752x480px, který představuje rozumný kompromis mezi kvalitou obrazu a velikostí poskytovaných dat. Více o použitém hardwaru v kapitole 4.

3. Problematika regulace polohy

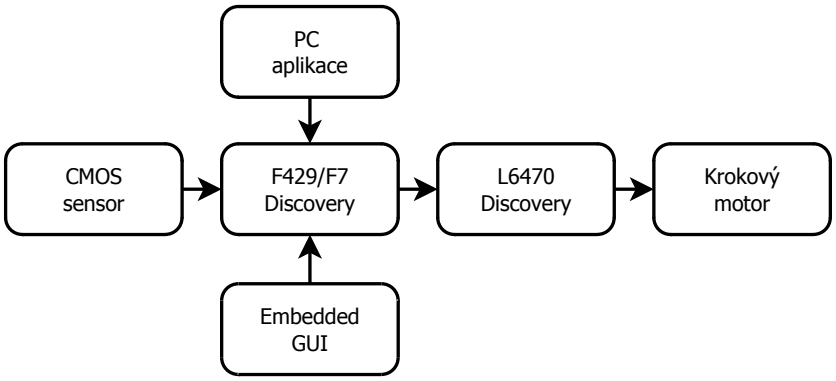
Základním bodem této diplomové práce je využití dat, získaných algoritmy zpracování obrazu, pro regulaci polohovacího systému (pojezdu) a demonstrace jejich funkčnosti ve formě funkčního dema. K tomu lze zvolit libovolný pojezd poháněný elektrickým motorem. Možným kandidátem na základ celého dema může být například pojezd ze staré tiskárny nebo plotteru, jehož součástí zpravidla bývá krokový motor a také koncový spínač, který by bylo možné využít ke kalibraci koncové polohy.



Obr. 1 - Ideové schéma celého systému

Řízení krokového motoru lze realizovat jak přímým ovládáním mikrokontrolérem (generováním řídicích sekvencí, popsáno v kap. 12.2), tak využitím dedikovaného řídicího obvodu, kterému by stačilo předávat příkazy typu „posun o x kroků“ nebo „posun směrem x rychlostí y“. V době řešení této diplomové práce byl v laboratoři videometrie k dispozici specializovaný kit pro řízení krokových motorů EVAL6470H-DISC R1 s inteligentním obvodem ST L6470H a řídicím mikrokontrolérem STM32F105 (ARM Cortex-M3). Jde o kit vše-v-jednom, který je možné řídit buď z nadřazeného počítače pomocí dodávané aplikace, nebo je možné osazený mikrokontrolér STM32F1 naprogramovat vlastním firmwarem a vytvořit si vlastní ovládací PC software, nebo využít dostupného rozhraní UART a přijímat příkazy skrze něj. Tato deska také kromě daného obvodu obsahuje i oddělené napájení pro vinutí řízeného motoru a možnost omezení proudu (řízeno obvodem L6470 a nastavováno trimrem na desce).

V případě, že se uživatel rozhodne naprogramovat si vlastní firmware, je ke kitu EVAL6470H dostupná knihovna „dSPIN“, která zajišťuje konfiguraci a ovládání čipu L6470. Pro použití této knihovny je pouze potřeba nakonfigurovat několik parametrů (ve formě #define direktiv) jako je například maximální rychlost použitého motoru (kroků/s), maximální zrychlení, střída napájecího napětí pro režim v pohybu a pro režim „hold“ (tedy v případě, že motor stojí na místě) a dalších. Pak je již možné používat předpřipravených funkcí k ovládání krokového motoru. Bude-li tak úkol otáčení motorem delegován na tuto desku, uleví se tím hlavnímu mikrokontroléru na kitu F429-Discovery.

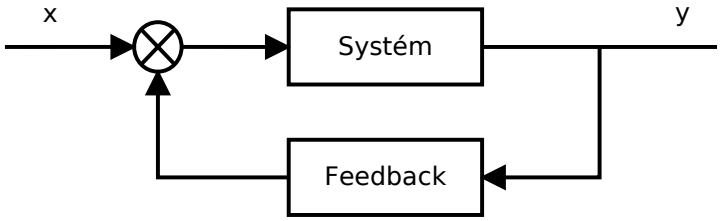


Obr. 2 - Vztahy jednotlivých funkčních celků navrhované aplikace

Pro zobecnění řešení regulace polohy pomocí stejnosměrných motorů budou implementovány dva režimy regulace – jednak klasický režim krokového motoru, kdy jsou dávány příkazy typu „posun o x kroků“, tak i režim stejnosměrného motoru „posun směrem x rychlostí y“, tzn. bez možnosti využít základní výhody krokových motorů – znalost jejich aktuální polohy v kterémkoliv okamžiku. Funkčnost tohoto režimu je hlavní výhodou celé aplikace – u krokových motorů je jejich poloha vždy známá, u stejnosměrných motorů nikoliv a tak je notné polohu zjistit jinak – například použití inkrementálního snímače, nebo práce obrazové informace.

Řízení klasického krokového motoru je přímovazební (zjednodušeně tedy bez kontinuálního snímání aktuálního stavu systému a úpravy řízení na základě této informace), protože díky přesně definovaným krokům vždy známe aktuální stav systému (pokud jej na začátku zjistíme). K počáteční kalibraci polohy krokového motoru lze využít koncový spínač, který je součástí zvoleného pojezdu. Jde o jednoduchý mechanický nebo optický spínač, který lze připojit k řídicímu obvodu a detekovat jeho sepnutí – pojezd tedy tak dlouho posouváme směrem ke koncovému spínači, dokud na něj nenarazíme. Tato poloha je označena jako „0“ a od tohoto okamžiku je stav posuvu vždy znám. Pokud navíc známe (například zjistíme měřením) počet kroků na otáčku/počet kroků na centimetr posuvu, pak vždy víme, kolik kroků musí krokový motor udělat, aby posuv nastavil na požadovanou pozici.

Druhé zvolené řešení předpokládá zavedení zpětnovazebního řízení – jediná informace, kterou zpravidla máme o stejnosměrném motoru je směr jeho otáčení a rychlost otáčení (přesněji velikost budícího napětí). Jako regulátor bude použit PSD (proporcionálně-sumačně-diferenční) regulátor neboli diskrétní varianta PID regulátoru. Regulační odchylkou regulátoru je rozdíl cílové a detekované pozice snímaného objektu.



Obr. 3 - Zpětná vazba

4. Použitý hardware

Jak vyplývá ze zadání práce, řešení práce má být orientováno na využití vývojových kitů založených na procesorech s jádry ARM Cortex-M4 a M7, neboli 32bitových RISC ARM procesorů, vyznačujících se mimo jiné slušným výkonem při nízké spotřebě a nízkou pořizovací cenou. Pro tuto práci bude využit kit STM32F429-Discovery a nově také F7-Discovery, uvedu proto stručně jejich vlastnosti:

4.1. Kit STM32F429i-Discovery

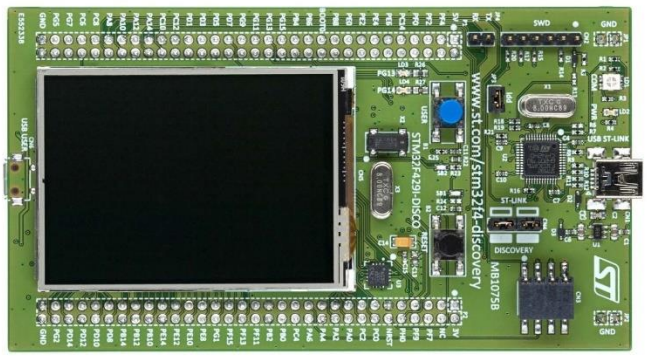
Prvním použitým kitem je kit STM32F429-Discovery, jehož základem je jednojádrový 32bitový RISC procesor STM32F429ZIT6 ve 144pinovém pouzdru LQFP144. Hlavními vlastnostmi tohoto kitu jsou:

- Mikrokontrolér s jádrem Cortex-M4 o frekvenci 180MHz (225DMIPS@180MHz)
  - Single-precision FPU jednotka
  - DCMI (Digital Camera Interface)
- Paměti: Flash: 2MB, RAM: 256kB
- Vestavěná paměť typu SDRAM o velikosti 64Mbits (8MBytes)
- Barevný dotykový TFT LCD displej o velikosti 2.4‘ QVGA (320x240px)

Kit dále obsahuje: tříosý MEMS akcelerometr, dvě uživatelsky přístupné LED diody, řadič USB OTG, Serial Wire Debug, headery s konektory a další. Ke kitu existuje kvalitní SW podpora (knihovny, ovladače, vývojové prostředí).

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

V rámci práce bude snaha využít zvýšené rychlosti procesoru (oproti kitu F4-Discovery použitém v [1]) 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-Discovery není možné vzhledem k nízké rychlosti rozhraní PC<->Discovery, absenci externí paměti RAM a zároveň malé kapacitě interní paměti RAM (192kB), která neumožňuje obrázek uložit celý a odeslat jej až po dokončení snímání).



Obr. 4 - Kit STM32F429-Discovery

4.2. Kit STM32F7-Discovery

Druhým použitým kitem je kit STM32F7-Discovery, jehož základem je nový mikrokontrolér STM32F746NG, který je postaven na základě zbrusu nového jádra ARM Cortex-M7, mezi jehož hlavní přednosti v porovnání s jádrem Cortex-M4 jsou:

- Vyšší maximální frekvence (216MHz) a výkon (462DMIPS@216MHz)
  - Téměř dvojnásobný výkon (ale i spotřeba) při stejné frekvenci
- Double-precision Floating-point aritmetika (výpočty s plovoucí desetinnou čárkou)
- Dvojnásobná šířka datových a instrukčních sběrnic (64bit)

STM32F7-Discovery pak kromě tohoto mikrokontroléru obsahuje následující komponenty:

- USB OTG (včetně fyzické vrstvy (PHY) pro High Speed variantu - 480Mbit/s)
- 10/100Mbps Ethernet včetně PHY
- Konektor pro MicroSD karty
- Audio DAC, stereo MEMS mikrofony
- Paměť SDRAM o velikosti 64Mbit (pozn.: skutečná velikost je 128Mbit, zmenšení je dáno 16bit adresováním)
- Nevolatilní paměť typu NOR Flash o velikosti 128Mbit
- Barevný TFT LCD displej s kapacitní dotykovou vrstvou, s rozlišením 480x272 a úhlopříčkou 4,3“
- plochý konektor typu FFC pro kamerový modul (využívající signálů DCMI)

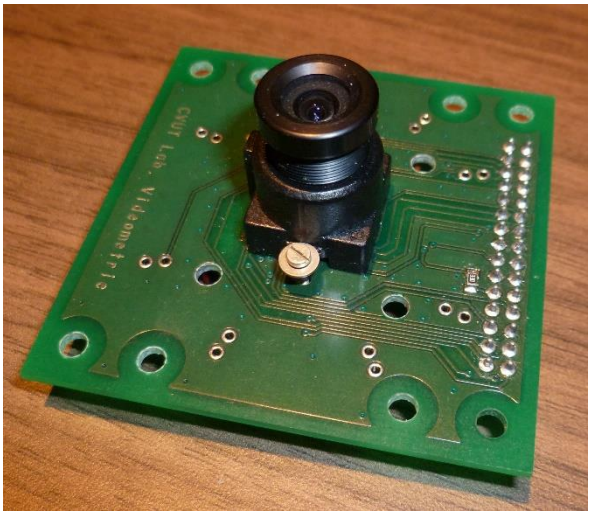
Výraznou komplikací ve využití kitu F7-Discovery pro naše účely je vyvedení kamerového interface na FFC ploché konektory (30pin, rozteč 0.5mm), jejichž dostupnost je v České republice špatná, navíc není ani možné prototypovat s využitím klasických 100mil konektorů a je nutné rovnou vytvořit plošný spoj s odpovídajícím konektorem.

Obr. 5 - Kit STM32F7-Discovery

4.3. 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. Sensor je vybaven uzávěrkou typu global shutter. Použité objektivy mají ohniskové vzdálenosti f = 3.6 mm, f = 8mm a nakonec f = 16mm. Další parametry objektivů jsou neznámé.

Vzhledem k tomu, že katedra měření má z dřívějšího vývoje dostupný modul včetně PCB využívající právě sensor Aptina MT9V034 (případně také jeho barevný ekvivalent MT9M001), byl zvolen právě tento sensor. Byla zvolena monochromatická verze sensoru, neboť předpokládaná aplikace sensoru v algoritmech zpracování obrazu by utrpěla zvýšenou výpočetní náročností práce s barevnými daty a přínos takového řešení by byl minimální.

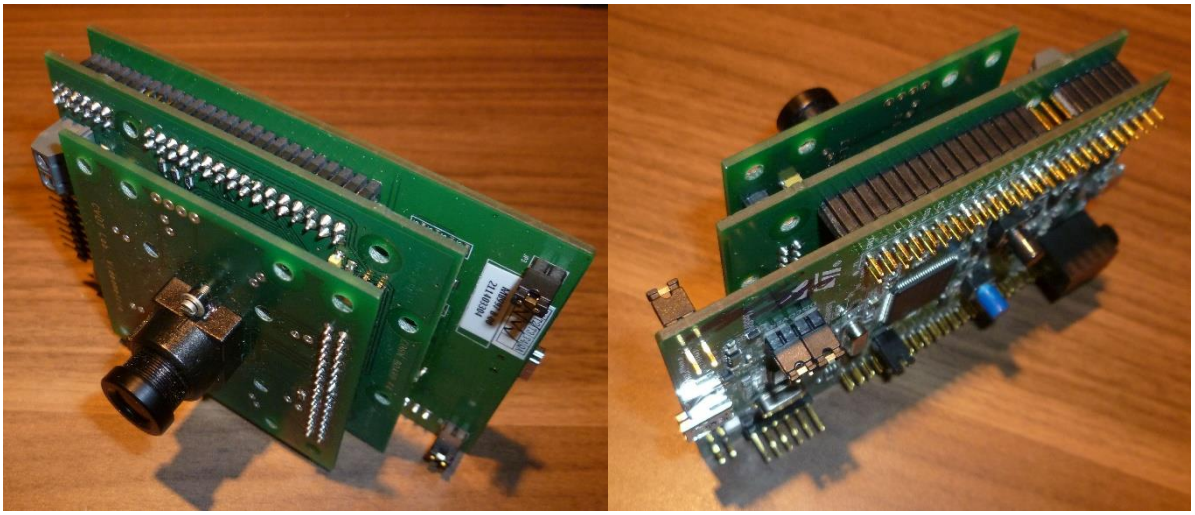


Obr. 6 - Obrazový sensor Aptina s nasazeným objektivem

K senzoru je z dřívějšího vývoje na katedře měření dostupná knihovna v jazyce C se základním nastavením mikrokontroléru a CMOS senzoru. Komunikace se sensorem probíhá přes sběrnici I^2C v případě řízení sensoru, a přes sběrnici DCMI v případě čtení obrazových dat ze sensoru.

DCMI je synchronní paralelní rozhraní schopné číst data ze sensoru rychlostí až 54MB/s, které podporuje DMA. 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 (který může špičkově odebírat až 100mA, v barevné variantě až 250mA). Prototyp tohoto modulu byl vytvořen v rámci [1] a následně finalizován do podoby kompaktního modulu v diplomové práci [5] pana Dokoupila z roku 2015, a poskytuje kompaktní propojení celého modulu.



Obr. 7 - Kompletní modul

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

SIGNÁL MODULU CMOS SENSORU	PIN DISCOVERY KITU
D0	PC6
D1	PC7
D2	PC8
D3	PC9
D4	PE4
D5	PD3
D6	PE5
D7	PE6
D8 (pouze 10-bit režim)	PC10
D9 (pouze 10-bit režim)	PC12
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

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

- Signály SCLK a SDA náleží k rozhraní I²C (na straně mikrokontroleru konkrétně I²C2)
- 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 řídicí elektroniky pro krokový motor či LED displeje, tato funkcionalita ale nebude využita. Deska podporuje i 10bitový DCMI přenos, pro jeho použití je však třeba provést několik hardwarových změn (pájení propojek – více v [5]).

4.4. ST EVAL6470H-DISC R1

Jak bylo naznačeno v kapitole 2, pro řízení krokového motoru je využíván kit ST EVAL6470H-DISC R1. Tento kit je dalším z řady ST-Discovery kitů, tentokrát ale postaven především kolem obvodu L6470, což je plně integrovaný „inteligentní obvod" pro řízení dvoufázových bipolárních krokových motorů. Obvod L6470 se automaticky stará o generování řídicích sekvencí pro bipolární krokové motory včetně otáčení směru toku proudu jednotlivými vinutími (součástí obvodu je duální DMOS můstek). Mezi vlastnosti tohoto obvodu patří:

- Trvalý proud 3A (špičkový 7A) pro každou fázi
- Pracovní napětí 8-45V (oddělené od napájení řídicího kontroléru)
- Current sensing obvody pro snímání odběru proudu a ochranu proti nadproudu (overcurrent)
- Detekci zastavení (utržení)
- Detekci koncového snimače (s konektorem pro připojení jednoduchého spínače, například tlačítkového)
- 5Mbit/s SPI interface
- Uživatelsky definované pohybové vzorce: akcelerace, decelerace, cílová rychlost či poloha

Pro řízení tohoto obvodu je na kitu EVAL6470H-EVAL osazen ještě mikrokontrolér STM32F105RB, do kterého je možné nahrát vlastní firmware pro ovládání obvodu L6470H. V rámci podpory od výrobce je k dispozici knihovna psaná v jazyce C, obsahující funkce pro nastavení a ovládání obvodu L6470 (inicializace, kalibrace, posun o n kroků, posun rychlostí n kroků za minutu a další). Pro programování je dostupný JTAG konektor plné velikosti (20 pinů). Na rozdíl od mnoha jiných Discovery kitů od ST není však na kitu přítomné flash/Debug rozhraní ST-Link, je tedy třeba mít vlastní programátor.

U většiny ostatních kitů Discovery/Nucleo s ST-Linkem je však možné jejich vestavěný ST-Link programátor využít i pro programování a debug jiných procesorů, podporují-li tyto programování přes SWD (Serial Wire Debug) – stačí pouze propojit několik pinů (SWDIO, SWCLK, GND, volitelně V_{cc} a RESET). Tyto signály jsou podmnožinou signálů JTAGu, a jsou tedy přítomny i na 20pin JTAG konektoru na kitu EVAL6470H-DISC R1.

Ke kitu je dále dostupný počítačový software SPINFamily, kterým lze vývojový kit po připojení přes USB ovládat – ne ale v případě, že se uživatel rozhodne si napsat firmware vlastní, v tom případě je v embedded kódu pouze několik krátkých ukázek bez komunikace s okolím (UART, Virtual COM). V případě použití UART pro řízení je tak potřeba si napsat vlastní komunikační protokol.

Obr. 8 - Kit EVAL6470H-DISC R1

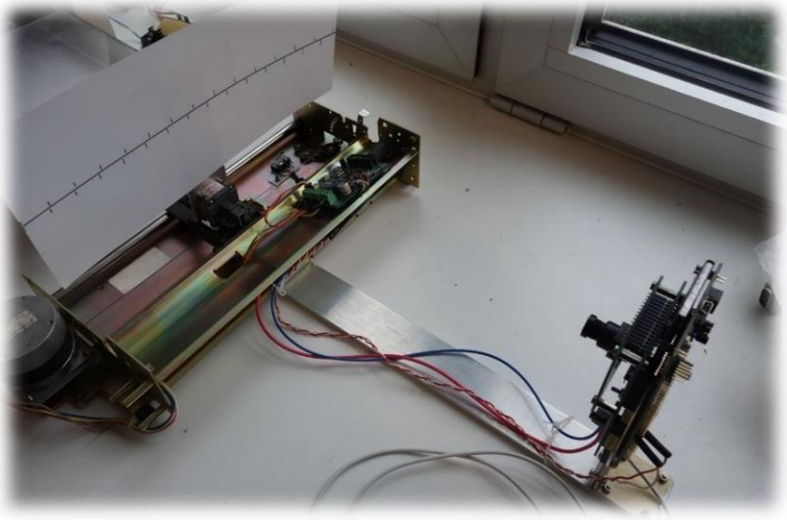
5. Sestavení hardwaru a jeho omezení

5.1. Sestavení hardwaru

Základ snímacího systému je tvořen kamerovým modulem, propojovací deskou a kitem F429-Discovery (jak bylo uvedeno v 4.3). Dále je využit regulovatelný prvek – pojízdný mechanismus z jehličkové tiskárny, který je osazen bipolárním krokovým motorem a jezdcem připevněným k lankovému mechanismu. Součástí pojezdu je také mechanický koncový spínač (ten lze využít bez dalších komplikací díky připravenému konektoru na kitu EVAL6470H).

Pojezd s krokovým motorem byl očištěn od všech komponent bezprostředně nesouvisejících s předpokládaným využitím – především od kompletního mechanismu posuvu tiskového papíru včetně několika válců a tiskové hlavy. Na místo tiskové hlavy byl umístěn sloupek k uchycení snímaných předmětů. Do čela pojezdu byla umístěna konstrukce držící kit F429-Discovery i s kamerovým modulem a kit EVAL6470H. Výsledná soustava není příliš mobilní, je ale jednoduše spustitelná – stačí připojit dva napájecí adaptéry – pro kity F429-Discovery a EVAL6470H (5V) a pro driver krokových motorů (8-45V). Konektory jsou umístěny na okraji pojezdu a označeny.

Při napájení obou kitů je maximální odebíraný proud přibližně 400mA při 5V (tedy dostupný i z běžného počítačového portu standardu USB 2.0, tedy 500mA), v případě využití kitu F7-Discovery je ale potřeba zvolit externí adaptér s minimálním proudem 1A při 5V. K napájení krokového motoru je pro optimální výsledky vhodné využít zdroj schopný dodat alespoň 12V/1A. V případě využití PC aplikace je nutné připojit ještě USB konektor (připojen do portu „USB USER“ na Discovery kitu) – ten má ale nulový proudový odběr.



Obr. 9 - Fotografie výsledné sestavy

5.2. Omezení zapojení

V rámci tohoto projektu se využívá mnoho periferií procesoru a navíc i mnoho dalších externích komponent: DCMI (Digital Camera Interface), SDRAM, LTDC (LCD Controller), SPI (pro dotykový panel), UART (pro propojení s deskou EVAL6470H). Proto dochází nutně k překrývání funkcí některých vývodů (pinů) na mikrokontroléru, které je třeba vyřešit, aby celá aplikace mohla fungovat. V tabulce na následující straně jsou uvedeny využívané piny a jejich kolize (více v [1]):

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

SIGNÁL LCD	PIN DISCOVERY
LCD_TFT R3	**PB0***
LCD_TFT R4	PA11
LCD_TFT R5	PA12
LCD_TFT R6	**PB1***
LCD_TFT R7	PG6
LCD_TFT G2	**PA6***
LCD_TFT G3	PG10
LCD_TFT G4	PB10
LCD_TFT G5	PB11
LCD_TFT G6	**PC7***
LCD_TFT G7	**PD3***
LCD_TFT B3	PG11
LCD_TFT B4	PG12
LCD_TFT B5	**PA3***
LCD_TFT B6	**PB8***
LCD_TFT B7	**PB9***
LCD_TFT HSYNC	**PC6***
LCD_TFT CLK	PG7
LCD_TFT VSYNC	**PA4***
LCD_TFT DE	PF10
LCD_NCS	**PC2***

Tab. 2 - Připojení periferií ke kitu F429-Discovery

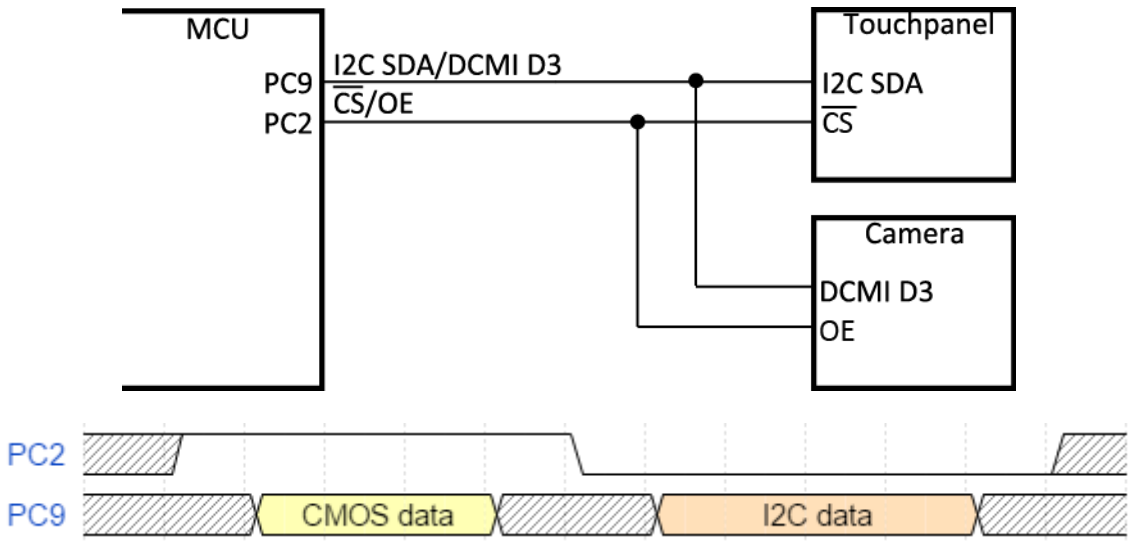
Nově je navíc využívána i dotyková vrstva LCD displeje připojená skrze komunikační rozhraní I^2C následujícími piny:

- SCL -PA8 (bez kolizí)
- SDA - ***PC9***

Tím vznikla další kolize daná sdílením pinu PC9 mikrokontroléru mezi sběrnicí I^2C dotykového panelu a datovým vodičem D3 kamerového rozhraní DCMI. CMOS modul tedy není možné využívat v době, kdy je využíván touchscreen (vzhledem ke kolizí s LCD signály ale stejně nemůže být CMOS sensor využíván ani kdykoliv je aktivní LCD displej) - proto jsou v době snímání obrazu displej i touchscreen vypínány a po dokončení snímání opět znova inicializovány (v případě, že je požadavek na výstup na LCD displej).

Aby bylo možné deaktivovat výstupní buffery CMOS sensoru, je třeba vzít jiný signál - PC2 (CMOS Output Enable) – a nastavit jej na nízkou úroveň. Tím se vstupy a výstupy CMOS sensoru přepnou do režimu vysoké impedance a neovlivňují tak ostatní komunikaci po sdílených pinech. V opačném případě by CMOS sensor svými daty (D3) znemožňoval I^2C komunikaci po PC9 přesto, že jeho výstup vlastně není vůbec žádán a využíván.

Tento druhý signál (CMOS Output Enable na pinu PC2) je ale navíc sdílen se signálem LCD_NCS (Not Chip Select LCD řadiče ILI9341). Driver LCD řadiče tento signál nastavuje na nízkou úroveň jen při zápisu nebo čtení z registrů LCD řadiče, po zbytek času je na vysoké úrovni, čímž je zároveň nastaven Output Enable CMOS sensoru, čímž je znemožněna komunikace dotykové vrstvy. Situaci lépe ilustruje obr. 10:



Obr. 10 - Ilustrace kolize zapojení touchscreenu a senzoru

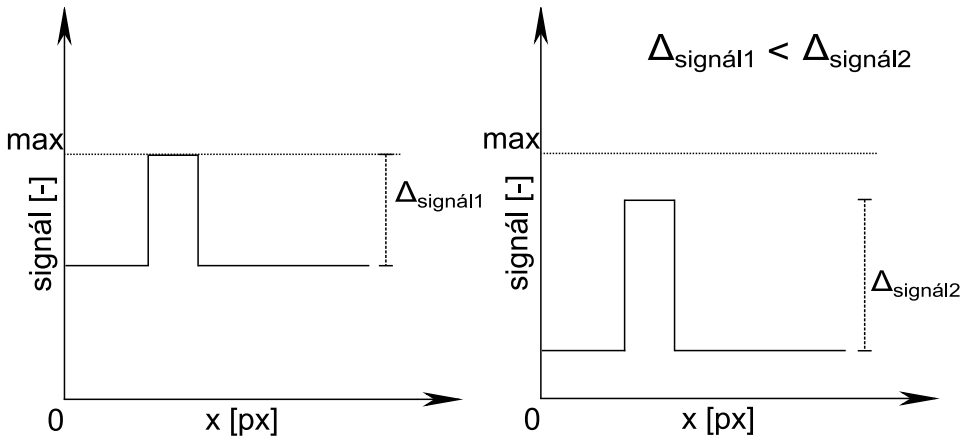
LCD řadič ILI9341 naštěstí připouští trvale nastavenou nízkou úroveň (tedy logickou 1) signálu Chip Select bez omezení funkčnosti – tento fakt společně s tím, že CMOS Output Enable a LCD Chip Select mají vzájemně inverzní logiku, lze snadno využít k řešení této situace. Úpravou driveru LCD řadiče je zcela zamezeno nastavování PC2 na vysokou úroveň a CMOS sensoru je tak zamezeno v ovlivňování signálů na připojeních datových linkách.

Poslední přidanou funkcí je UART (UART5), využívající jedny z posledních volných pinů - PC12 a PD2. Tento je využíván pro komunikaci kitu F429-Discovery a EVAL6470H.

6. Robustní nastavení doby expozice

Většina používaných CMOS senzorů (byť ne úplně všechny) zpravidla disponuje funkcí automatického nastavení doby expozice, které za běžných podmínek odvádějí poměrně dobrou práci. Nicméně i přesto, že předpokládaná oblast využití této aplikace je v laboratoři za dobrých světelných podmínek, jedním z cílů této aplikace bylo vytvořit a odladit robustní model řízení snímání (zisk, doba expozice, …). V případě prezentování na akcích pořádaných školou/katedrou totiž nemusí být zajištěny dostatečně podmínky pro automatiku. Její chování se navíc nedá příliš ovlivnit.

Specifické parametry scény při aplikacích image processingu (zpravidla tmavá či naopak světlá scéna s několika malými kontrastními body) by tedy mohly benefitovat z vlastního regulátoru, který se bude umět lépe přizpůsobit těmto specifickým parametrům – na obr. 11 je jednoduché znázornění řezu obrázkem, který obsahuje jeden kontrastní bod. Jak je vidět, automatická regulace, která nastaví průměrný jas scény zhruba na polovinu rozsahu může být méně vhodná než manuální regulace, která nastaví průměrný jas nižší, čímž zvýší kontrastnost snímaného objektu:



Obr. 11 - Vliv doby expozice na kontrastnost snímaného objektu

Vzhledem k tomu, že většinu výkonu mikrokontroléru spotřebují algoritmy zpracování obrazu, neprobíhá regulace expozice průběžně, ale jen „na vyžádání“ uživatelem, tedy např. stisknutím tlačítka (nebo periodickým voláním). V případě použití PC aplikace je pak možné nastavit cílový průměrný jas obrazu tak, aby vyhovoval zamýšlenému použití.

6.1. Jednoduchý výpočet optimální doby expozice z celého obrazu

Nejjednodušším způsobem výpočtu optimální doby expozice je využití celého obrazu, kde všechny body mají stejnou váhu. Výhoda je jediná – výpočetní nenáročnost. Nevýhodou je, že

je špatně použitelná pro snímání objektů na vysoce kontrastním pozadí (tvář proti slunci, žárovka proti tmavému pokoji).

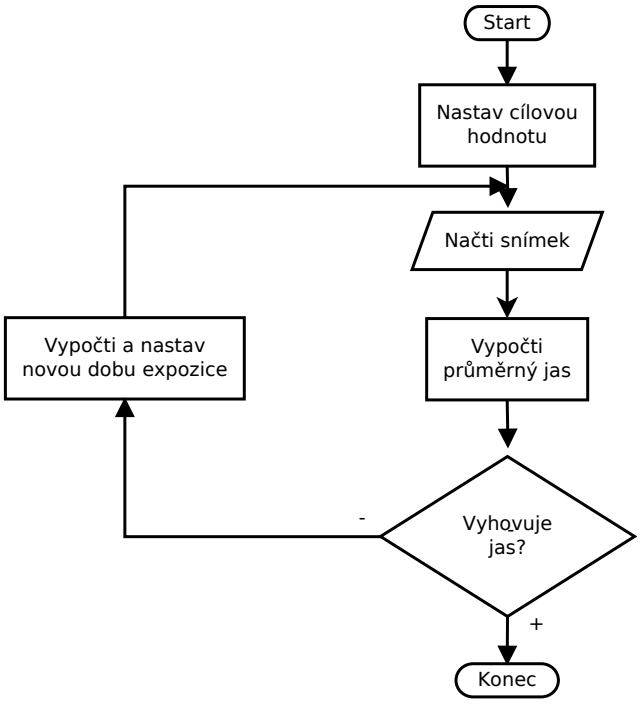
Využije-li se faktu, že průměrný jas (aritmetický průměr) celého snímku je přímo úměrný (přibližně) době expozice, pak je nejjednodušší možností, jak docílit žádané doby expozice, následující algoritmus:

1) Sejmutí obrazu se známou hodnotou doby expozice (EV_0)
2) Výpočet hodnoty průměrného jasu celého snímku (B_0)
3) Vypočítat optimální dobu expozice:

$$EV_{optimal} = EV_0 \frac{B_{optimal}}{B_0}$$

kde B_{opt} může být například polovina rozsahu (128), nebo libovolné jiné číslo podle požadavků na danou aplikaci. Tato hodnota se zpravidla volí empiricky.

4) Nastavení nové doby expozice
5) Sejmutí nového obrazu a porovnání B_0 a $B_{optimal}$ (s určitou tolerancí)
6) V případě nedostačujícího výsledku návrat k bodu 1

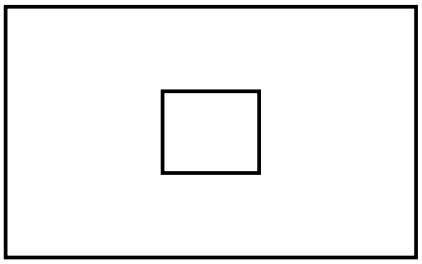


Obr. 12 - Nastavení doby expozice

6.2. Bodové měření, segmentové měření, váhované měření

Lepší variantou výpočtu optimální doby expozice je bodové či segmentové měření. Nejjednodušší variantou bodového měření je měření pouze malého výřezu uprostřed zorného pole kamery (nebo kdekoliv jinde – v tom případě je třeba zadat kde, například manuálním

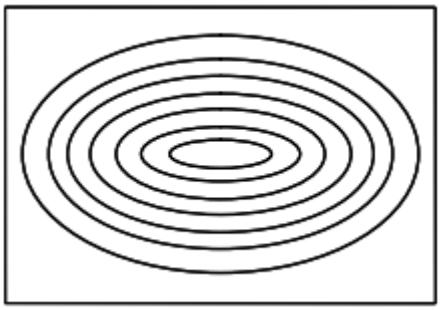
označením). Snímaný objekt musí být ve vybraném výseku zorného pole. Oblastí může být samozřejmě zvoleno i více. Algoritmicky se tato varianta od předchozí neliší, pouze se počítá s jasem omezeného segmentu.



Obr. 13 - Bodové/Segmentové měření

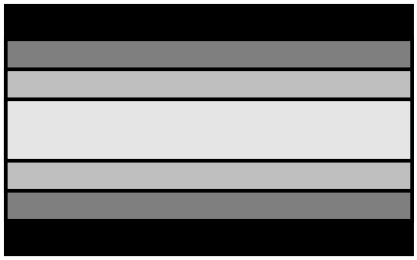
Segmentové měření je velmi podobné bodovému měření, vyhodnocovaná oblast je ale o něco větší (pokud jsme v předchozím případě vyhodnocovali pouze oblast odpovídající cca 1-3% z celého obrazu, nyní je to přibližně 10-15%). Takové zpracování trvá sice o něco déle, není však tak náchylné na lokální extrémy a zpravidla poskytuje lepší výsledky.

Poslední implementovanou variantou základních algoritmů nastavení optimální doby expozice je váhované měření – Zde je vyhodnocován celý obraz (nebo jeho většina), čím dále jsou ale body od středu obrazu, tím má jejich hodnota menší váhu. Jde o nejběžnější variantu automatického výpočtu doby expozice, pokud není použita nějaká složitější metoda (například s využitím histogramu či peak analýzy).



Obr. 14 - Váhované měření

Toto měření bylo dále upraveno pro potřeby snímání pravítka nebo jiného 1-D měřítka – kdy nejvyšší váha je přiřazena pixelům na prostředním řádku a váha ostatních řádků se snižuje s jejich vzdáleností od prostředního řádku.



Obr. 15 - Váhované měření pro snímání pravítka

7. Použité algoritmy zpracování obrazu

Algoritmy zpracování obrazu, použité v této práci, vycházejí z algoritmů použitých v mé vlastní [1] (u těchto provedu pouze stručnou rekapitulaci) a jsou doplněny několika dodatečnými preprocessingovými filtry, volně inspirovanými pracemi [2, 3].

Vzhledem k přítomnosti onboard SDRAM paměti o velikosti 8MB nebylo nutné používat algoritmy průběžného zpracování, a tak byly tyto algoritmy upraveny pro offline zpracování obrazu, uloženého v SDRAM paměti. To mi rovněž umožnilo využít i další algoritmy, které se při průběžném zpracování provádějí obtížně nebo nemohou být aplikovány vůbec. Sejmutý obraz má velikost přibližně 361kB (752*480*8), do paměti RAM o velikosti 256kB by se tedy nevešel. Další (drobnou) výhodou kitu F429-Discovery oproti kitu F4-Discovery je o několik procent vyšší frekvence (180MHz oproti 168MHz) a tím i větší výpočetní výkon (225DMIPS oproti 210DMIPS).

7.1. Filtry – Gaussův filtr (rozostření)

Gaussův filtr je jeden ze základních prvků pro image processing. Algoritmus spočívá v aplikaci dané konvoluční masky na každý pixel zpracovávaného obrazu, čímž dojde k rozostření obrazu – výsledkem čehož je mimo jiné i potlačení nevýznamných hran.

V rámci PC aplikace lze nadefinovat a použít libovolnou konvoluční masku, výchozí konvoluční maska je aplikací váženého průměrování – zvýraznění středu:

$$\frac{1}{16}\begin{pmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{pmatrix}$$

Výsledkem takové operace je rozostřený obrázek, který sice ztratil část detailu, na druhou stranu ale ztratil také nevýrazné hrany a šumy v obraze. To zvyšuje spolehlivost algoritmů hledání hran či prahování. Násobení každého pixelu maticí 3x3 je ale výpočetně poměrně náročná operace, její efekt za to ale zpravidla stojí.

Výsledek operace rozostření hran pomocí Gaussova filtru je vidět na obr. 16:

Obr. 16 - Použití gaussova filtru

7.2. Filtry – operátor Prewittové/Sobelův operátor

Operátor Prewittové, stejně jako Sobelův operátor, slouží k hledání hran v obrazu. Máme-li obraz ve stupních šedi a vizualizujeme si ho jako výškovou mapu, pak operátor Prewittové vyhledává směry největšího gradientu – nejstrmějšího stoupání. V případě zpracování obrazu tak jde o hledání největších rozdílů v jasech sousedících pixelů.

Ve spojitém světě se výsledku dosahuje použitím gradientu – v našem diskrétním světě obrazů, který má jen 256 úrovní šedi, je použita konvoluce s konkrétními maskami, kterými je počítána aproximace gradientů v různých směrech – především horizontálně (1. maska) a vertikálně (2. maska), kde A je zpracovávaný obraz a $*$ představuje operátor konvoluce. Stejným způsobem je možné vypočítat i spočítat gradient v úhlopříčných směrech (3. maska + její zrcadlový ekvivalent), případně dalších směrech (směry detekce jsou ale vždy kvantovány, s maskou 3x3 není například možné detekovat gradient ve směru jiném než jsou tyto čtyři).

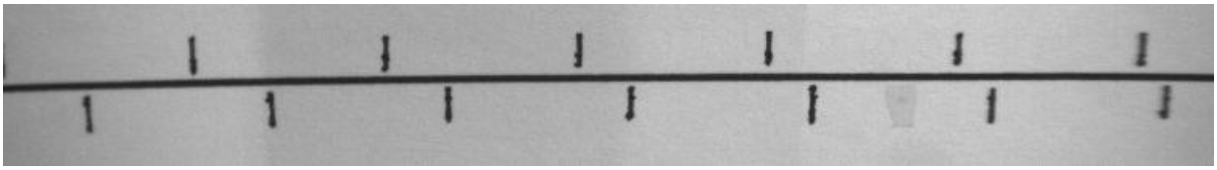
$$G_x = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} * A, G_y = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} * A, G_u = \begin{pmatrix} 0 & -1 & -1 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{pmatrix} * A$$

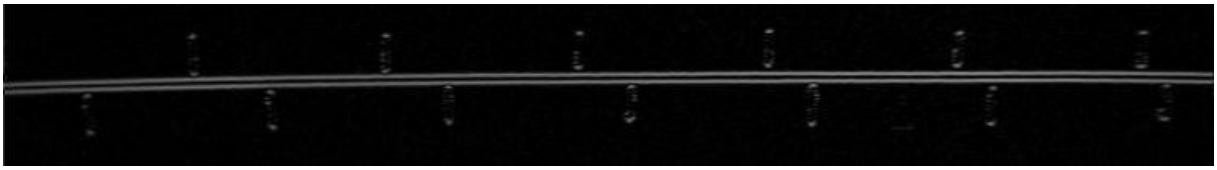
Výsledkem celé operace pro každý pixel je gradient jasu v daném bodě a směru. Čím větší je tato hodnota, tím je hrana strmější, tam, kde žádná hrana není, konvoluce s těmito maskami vrací nulu. Celková velikost gradientu při použití vertikální i horizontální detekce v daném bodě je pak dána jako

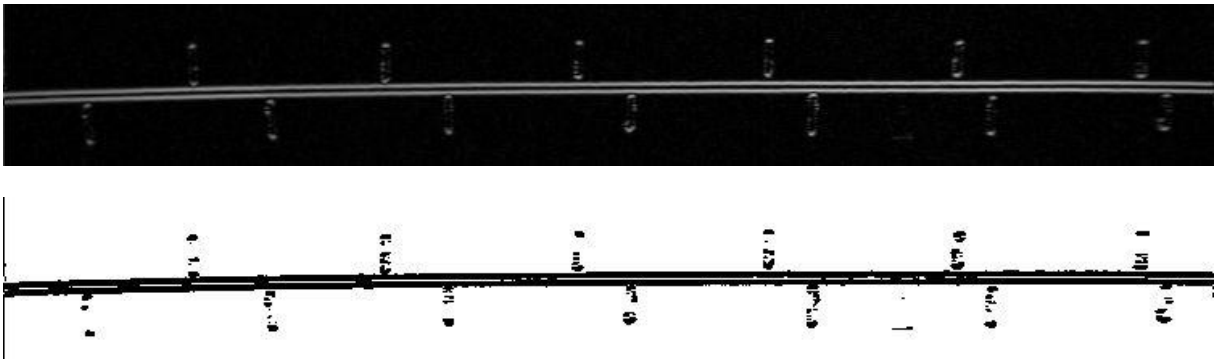
$$G = \sqrt{G_x^2 + G_y^2}.$$

Tento vzorec ale není nutno použít, používá-li se operátor pouze pro detekci gradientů v jednom směru.

Tato operace se jeví ideální například pro detekci pravítka, čehož tato práce využívá. Výstup tohoto algoritmu je navíc vylepšen použitím Gaussova filtru, který potlačuje nevýznamné hrany a šumy v obraze.







Obr. 17 - Hledání hran – originál (1), Prewitt(2), Sobel(3), Sobel+prahování(4)

Vedle operátoru Prewittové byl rovněž implementován i Sobelův (Sobel-Feldmanův) operátor, který funguje na identickém principu, pouze s jinými konvolučními maskami. Obraz zpracovaný Sobelovým operátorem má více zvýrazněné hrany, což je vidět na obr. 17 a což je při použití následujícího algoritmu (prahování) užitečné – zvýší se odstup jasu předmětu od jasu pozadí, což dává větší volnost v nastavování prahovací úrovně.

$$G_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} * A, G_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} * A$$

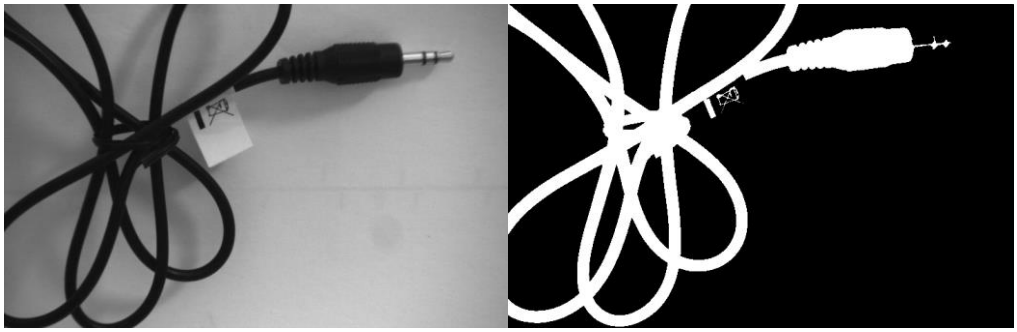
7.3. Prahování

Prahování (Thresholding) je základní operace zpracování obrazu, při které se hodnota každého pixelu porovnává s referenční hodnotou (prahovací úrovní), a v závislosti na výsledku této operace je ji přiřazena buď minimální, nebo maximální hodnota.

Je-li tedy k dispozici obrázek o 256 stupních šedi, pak při zvolené prahovací úrovni (například 72) algoritmus vyhodnocení vypadá následovně:

$$h(x,y) = \begin{cases} 255\ pro\ f(x,y) > 72 \\ 0\ pro\ f(x,y) < 72 \end{cases}$$

Prahovacích úrovní může být i více, výsledkem operace s n prahovacími úrovněmi je pak obrázek s n+1 stupni šedi. V navrhované aplikaci je ale žádoucí binární obraz, tedy pouze jedna prahovací úroveň. Pro lepší ilustraci funkce prahování poslouží následující snímek:



Obr. 18 - Prahování (vlevo originál, vpravo zpracovaný obrázek, inverzní)

7.4. Hledání disjunktních objektů – Connected-component labelling

Metoda Connected-component labellingu představuje metodu pro detekci objektů v obrazu, jejíž rozbor a použití bylo již vyzkoušeno v rámci [1], a která byla dále optimalizována například využitím rozhodovacího stromu. Tuto metodu jsem tedy mohl bez problémů převzít a využít.

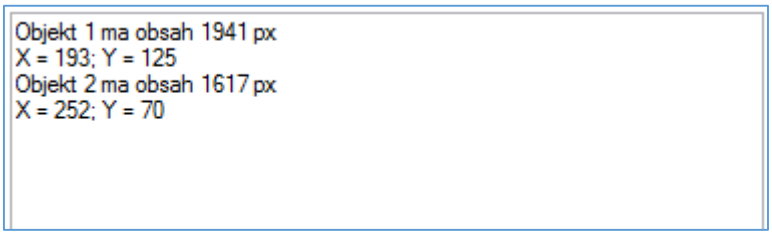
V průběhu vykonávání algoritmu labellingu je rovněž počítána poloha těžiště jednotlivých detekovaných objektů. Výpočet těžiště objektu je počítán s využitím klasického fyzikálního vzorce

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

upraveného na diskrétní podobu, kde hmotnostní element dm je nahrazen jasovou funkcí $f(i,j)$, 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 (algoritmus je aplikován na oprahovaný obraz, jde tedy o binární funkci):

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

Výsledkem použití metody labellingu je detekce a označení jednotlivých objektů v obrazu. V mé 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.



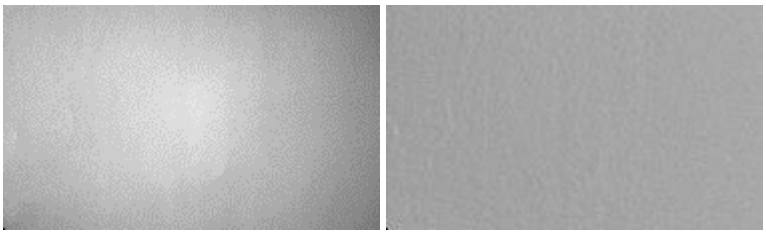
Obr. 19 - Výstup hledání objektů v obraze

Výstup algoritmu je vylepšen použitím Gaussova filtru, který nebyl v rámci bakalářské práce použit kvůli nedostatku výkonu tehdy použitého hardwaru. Vzhledem k přítomnosti dostatečně velké SDRAM paměti na současném hardwaru nejsou již výpočty tak časově kritické, abychom si nemohli použití tohoto algoritmu použít.

7.5. Korekce vinětace

Snímání obrazu je zpravidla ovlivněno různými optickými problémy a vadami snímací soustavy. Jedním z nejzásadnějších problémů při použití kamery k našim účelům je efekt vinětace. Jde o vadu snímací soustavy, která se projevuje nižším jasem v okrajích obrazu. Důvodem je, že světlo dopadá na okrajové pixely senzoru pod menším úhlem než na pixely

uprostřed senzoru. Tím se snižuje hustota fotonů dopadajících na jednotku plochy a tedy i tmavší obraz v okrajích obrazu. Proto, stejně jako v [1] je nutno tuto chybu korigovat. Efekt vinětace objektivu je nepřímo úměrný ohniskové vzdálenosti použitého objektivu – u objektivu s delší ohniskovou vzdáleností dopadají fotony na aktivní plochu senzoru pod větším úhlem, a tedy na menší plochu.



Obr. 20 - Efekt vinětace (vlevo) a efekt její korekce (vpravo) na obrázku sejmutém senzorem MT9V034 s objektivem s f = 3,6 mm

Z obr. 20 je zřejmě, že pokud by byl uprostřed zorného pole tmavý objekt, který by měl být cílem detekce metodou prahování podle jasové úrovně, velmi snadno by se mohlo stát, že kromě objektu uprostřed obrazu budou detekovány i rohy obrazu, které jsou rovněž poměrně tmavé.

7.6. Morfologické operace

Pro vylepšení obrazového výstupu zejména při detekci čar pravítka byly implementovány morfologické operace zpracování obrazu. Tyto operace se mimo jiné aplikují na binární obrazy (získané například prahováním) a slouží především k odstranění šumů v sejmutém obraze. Jejich použití v této práci je žádoucí především kvůli relativně nízkému rozlišení kamery (752x480 px.) vzhledem k velikosti obrazových značek (čárek pravítka).

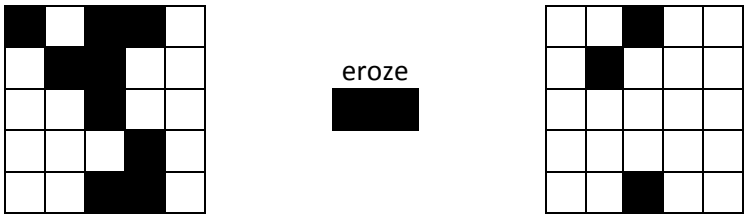
Základní myšlenkou binární morfologie je realizace matematické relace zkoumaného obrazu s jiným, menším, bodovým obrazem, tzv. „strukturním elementem“. Morfologická operace pak představuje opakovanou matematickou operaci mezi zkoumaným obrazem a strukturním elementem přes celou plochu zkoumaného obrazu a následnou editaci zkoumaného obrazu v závislosti na výsledku této operace.

Základními morfologickými operacemi jsou dilatace a eroze. Dilatace skládá zkoumaný obraz a strukturní element pomocí vektorového součtu a jejím výsledkem je, že jsou objekty v obraze zvětšeny (v závislosti na strukturním elementu) na úkor pixelů reprezentujících pozadí. Horizontální dilatace elementem (1,1,1) pak vypadá následovně:

Obr. 21 - Morfologická dilatace (horizontální)

Vertikální dilatace je pak provedena analogicky k horizontální dilataci.

Druhou morfologickou operací je eroze, která skládá zkoumaný obraz a strukturní element pomocí vektorového rozdílu a jejím výsledkem je naopak ztenčení objektů v obraze, opět v závislosti na použitém strukturním elementu. Horizontální eroze elementem (1,1) pak vypadá následovně:



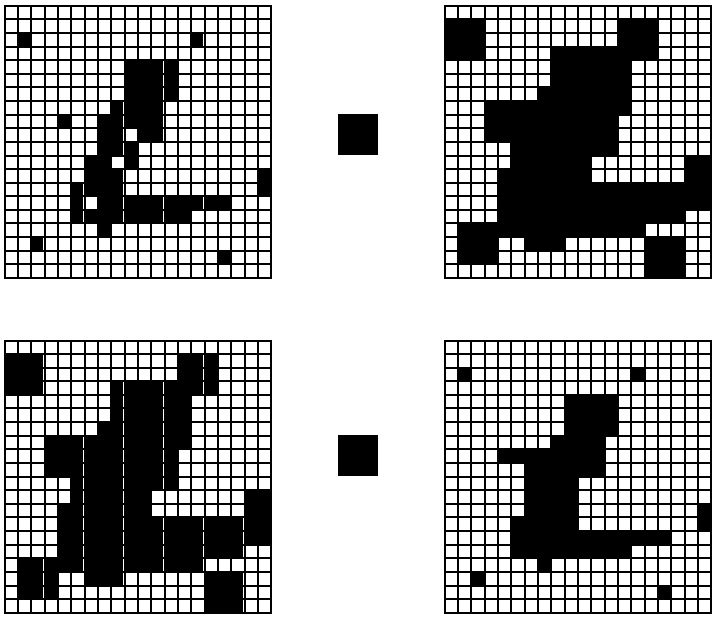
Obr. 22 - Morfologická eroze elementem (1;1)

Jak erozi, tak dilataci lze provést různými elementy podle zamýšleného efektu. Pokud by byl obr. 22 erodován maticí

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix},$$

byl by výsledkem již jen prázdný prostor, neb neexistuje žádná oblast obrazu, která vyhovuje erozní matici, a která by tak zachovala svou hodnotu.

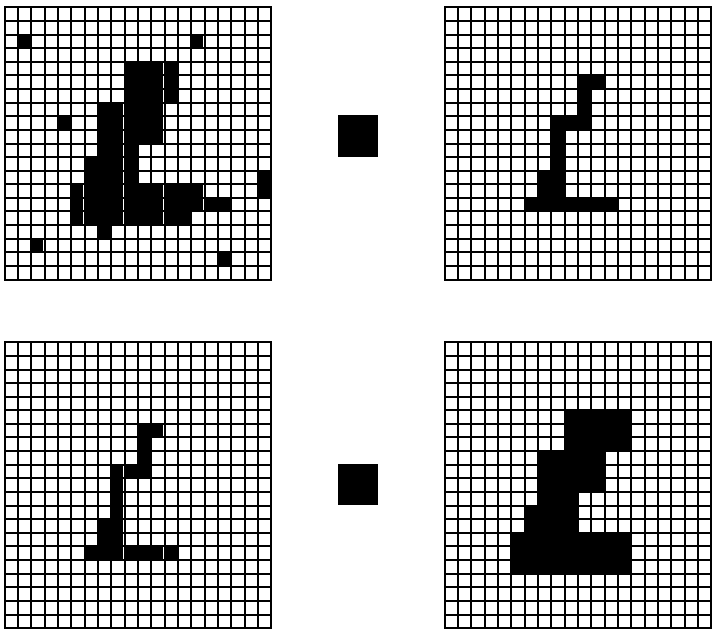
Spojením operací dilatace a eroze v tomto pořadí vznikne operace zvaná uzavření, spojením těchto operací v opačném pořadí vznikne operace otevření. Operace morfologického uzavření spojuje objekty, které jsou blízko u sebe (velikost mezery je daná velikostí strukturního elementu), vyplňuje malé díry v objektu a vyhlazuje obrys.



Obr. 23 - Morfologické uzavření - dilatace následovaná erozí

Operace morfologického otevření naopak odděluje objekty spojené úzkou spojkou (její velikost je opět dána velikostí strukturního elementu) a tím rozdělí objekty v obraze na menší objekty

s jednodušší strukturou a malé objekty odstraňuje úplně. Jde proto o jednu ze základních operací při odstraňování šumu.



Obr. 24 - Morfologické otevření - eroze následovaná dilatací

8. Detekce značek a regulace na cílovou polohu

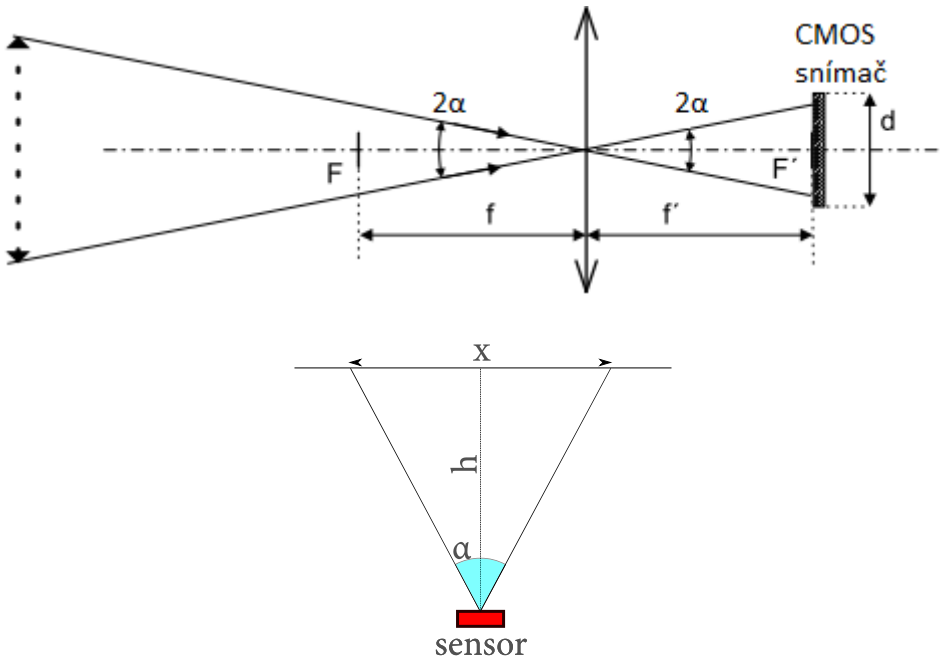
Hlavní funkcí navrhovaného programu obecně je detekce optických značek kamerovým systémem a následná regulace polohovacího systému (např. pojezdu s krokovým motorem) na uživatelem zadanou polohu ve vztahu k optickým značkám. Zamýšlené varianty návrhu optických značek jsou dvě – ta jednodušší počítá s jedním kontrastním bodem, který se bude aplikace snažit snímat a regulací pojezdu umístit například do středu zorného pole. Ta náročnější počítá s využitím nebo vytvořením složitější značky reprezentující měřítko. To může být dáno například určitou soustavou bodů a značek, které budou vzájemně porovnatelné, a podle kterých bude možné určit pozici polohovacího systému. Takovým měřítkem může být například klasické pravítko s centimetrovými a milimetrovými zarážkami a číslicemi nebo libovolné vlastní měřítko, které může být z pohledu snímacího systému jednodušší na analýzu, ale jeho využitelnost bude stejná nebo podobná.

8.1. Volba objektivu

Ještě před návrhem měřítek a implementací regulace polohy je třeba zvolit vhodný objektiv. Je tedy třeba spočítat ohniskovou vzdálenost f pro požadovaný úhel zorného pole (dále jen zorný úhel) kamery a velikost snímané plochy.

- Vzdálenost kamery od pojezdu: ~30cm
- Rozsah pojezdu: ~30cm
- Fyzická velikost snímače: 1/3“
    - Aktivní plocha 4,51mm(H) x 2,88mm(V), 5,35mm(úhlopříčka)

Vhodnou ohniskovou vzdálenost f objektivu lze vypočítat z obr. 25:



Obr. 25 - Proporce snímacího systému

Zorný úhel α lze vypočíst základním výpočtem s využitím tangensu:

$$\tan(\propto) = \frac{b}{c}, \propto = \arctan\left(\frac{b}{c}\right) = \arctan\left(\frac{1}{2}\right) \cong 26{,}57°$$

Zorný úhel kamery je roven dvojnásobku této hodnoty, tedy $\mathbf{53,14°}$. Nyní je třeba vypočítat ideální ohniskovou vzdálenost, tedy takovou ohniskovou vzdálenost, která umožní snímat celé zorné pole kamery. Pokud

$$\propto = 2\arctan\left(\frac{d}{2f}\right)$$

je obecně vztah pro výpočet úhlu zorného pole při znalosti rozměru aktivní plochy senzoru (d) a ohniskové vzdálenosti (f), pak hledaná ohnisková vzdálenost je

$$f = \frac{d}{2\tan\left(\frac{\alpha}{2}\right)} \cong \mathbf{4,51mm}.$$

V době tvorby této práce byly k dispozici objektivy s různými ohniskovými vzdálenostmi (zhruba v rozsahu 1,8mm - 16mm); pro finální aplikaci byl zvolen objektiv s ohniskovou vzdáleností $f = 8mm$, jehož zorné pole (s využitím vzorců výše) je **31,05°** a tedy přibližně **55,6%** snímané plochy. Potřebný výtah tohoto objektivu pro zaostření na požadovanou vzdálenost 30cm je pak

$$z' = \beta' \cdot f' = \frac{d}{x} \cdot f' = \frac{4{,}51mm}{300mm} \cdot 8mm = 0{,}12mm,$$

kde x je velikost snímaného objektu, d je velikost snímače, f' je ohnisková vzdálenost objektivu. Tohoto výtahu lze s použitým objektivem bez problémů docílit.

Důvodem volby tohoto konkrétního objektivu bylo zvýšení prostorového rozlišení - při volbě objektivu s $f = 4{,}5mm$ by při rozlišení kamery 752px připadalo zhruba 0,4mm/px, při volbě objektivu s $f = 8mm$ pak zhruba 0,22mm/px. Použití kratšího objektivu (tedy objektivu s menší ohniskovou vzdáleností) by tedy rozlišení systému pro účely snímání milimetrového pravítka snížilo na hranici použitelnosti.

Omezení zorného pole kamery také snižuje šanci, že se do zorného pole dostanou nežádoucí objekty v pozadí kamery (jinými slovy lze snadno vytvořit homogenní pozadí pro delší objektiv než pro kratší objektiv – stačí menší plocha).

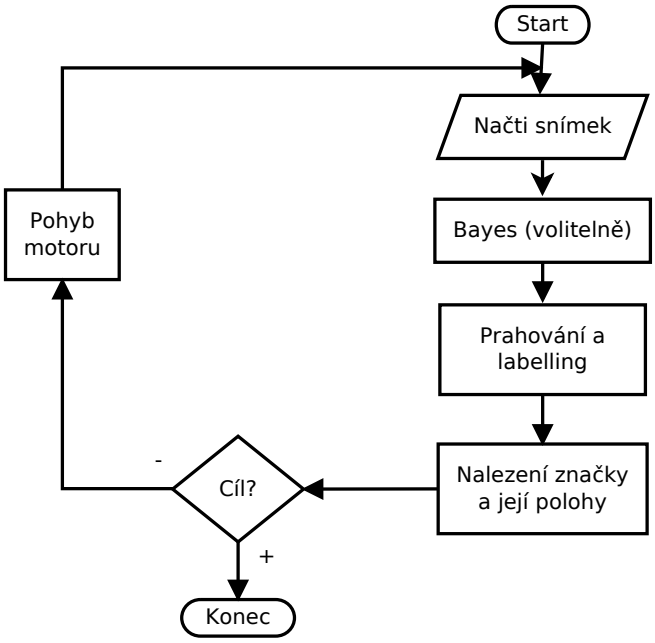
8.2. Detekce značky ve formě jednoho bodu

V první variantě programu bylo započato řešení problému regulace polohy s využitím pouze jednoho kontrastního bodu v obraze. Cílem v tomto bodě je vyzkoušení navržených algoritmů zpracování obrazu a regulace polohy bez nutnosti řešit další výpočty – například vyhodnocování vztahů mezi jednotlivými značkami, pokud je takových značek více.

Objektem detekce je v tomto případě pouze jeden kontrastní bod v obraze (tedy např. černá tečka na bílém podkladu nebo lépe reflexní tečka nasvícená LED diodou a snímaná použitou kamerou). V případě detekce tmavé tečky na světlém pozadí je největším problémem právě zajištění homogenního pozadí.

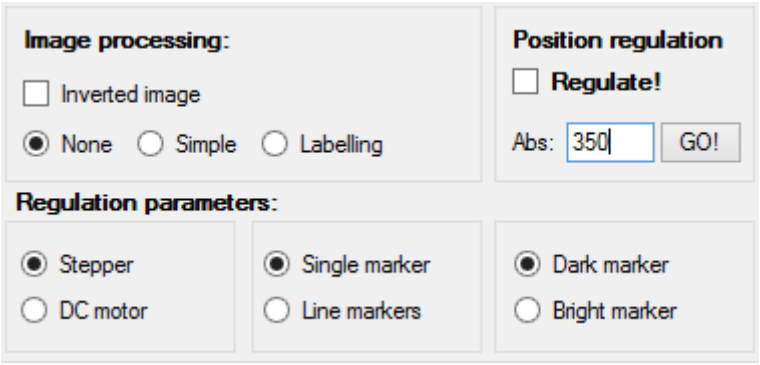
Zajištění homogenity pozadí bylo v této variantě provedeno umístěním homogenního bílého pozadí (bílá tvrdá deska) za snímaný objekt. V případě použití objektivu s ohniskovou vzdáleností f = 3,6mm pak je navíc téměř nutné provést i korekci vinětace, na kterou použité objektivy trpí poměrně výrazně. Při použití objektivů s většími ohniskovými vzdálenostmi f = 8mm a f = 16mm není korekce vinětace nutná, její použití by bylo vhodné, pouze pokud bychom snímali značku, která není příliš kontrastní.

Nejprve je nutno zajistit identifikaci značky pro případ, že by v obraze bylo více kontrastních bodů. To lze zajistit například specifickým tvarem značky nebo jen zajištěním, že je značka jediným objektem v zorném poli kamery. Pokud není značka v zorném poli jediná, je třeba zajistit, aby byla alespoň největší nebo s velkým odstupem nejkontrastnější – například laserová stopa bude prakticky za všech okolností daleko kontrastnější než jakýkoli jiný detekovaný objekt v obraze. Algoritmus zanedbává příliš malé objekty, velikost, pod kterou jsou objekty zanedbávány je ale nutné nastavit experimentálně podle toho, jak velkou značku je potřeba snímat.



Obr. 26 - Vývojový diagram základního algoritmu

Regulace na cílovou polohu je v základu realizována tak, že se systém snaží posouváním značky tuto umístit přesně doprostřed snímaného zorného pole. Nastavením v embedded GUI nebo v PC aplikaci je pak možno volit přesný bod, kam se má systém snažit značku umístit. Algoritmus také implicitně předpokládá, že osa posuvu je (alespoň přibližně) rovnoběžná s horizontální osou obrazu snímaného senzorem.

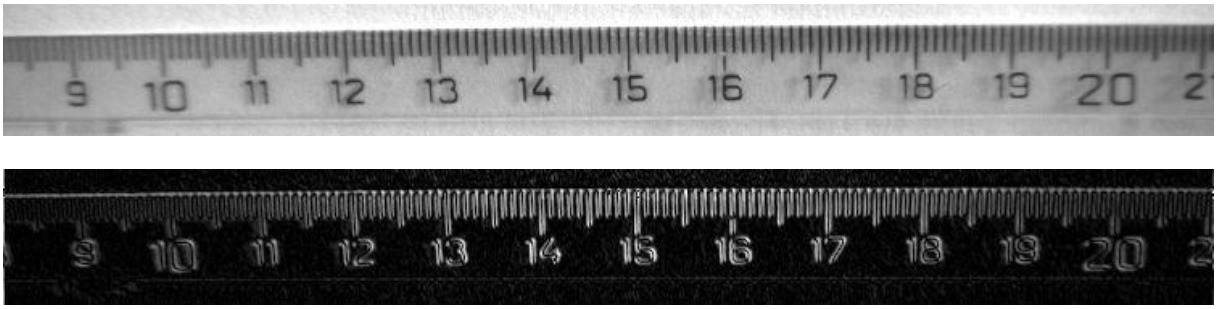


Obr. 27 - Nabídka v PC aplikaci

8.3. Detekce pravítka, detekce vlastního měřítka

Po naprogramování regulace s využitím detekce jedné značky byl program rozvířen o detekci s využitím soustavy značek, které tvoří jakési „pravítko“. Možností řešení této úlohy je více – například využití klasického pravítka s centimetrovými a milimetrovými zarážkami a číslicemi nebo vytvoření vlastního měřítka, které nahradí číslice jinými značkami. Detekce zarážek a identifikace, zda jde o centimetrové nebo milimetrové zarážky (v případě, že jsou alespoň velikostně rozdílné, což zpravidla značky na pravítku bývají) nepředstavuje velký problém –

použitím detekce hran podle Prewittové (viz 7.2), volitelně také s dalším pre/post-processingem lze tyto značky relativně spolehlivě detekovat. Další analýzou vlastností jednotlivých objektů lze detekovat i směr osy pravítka.

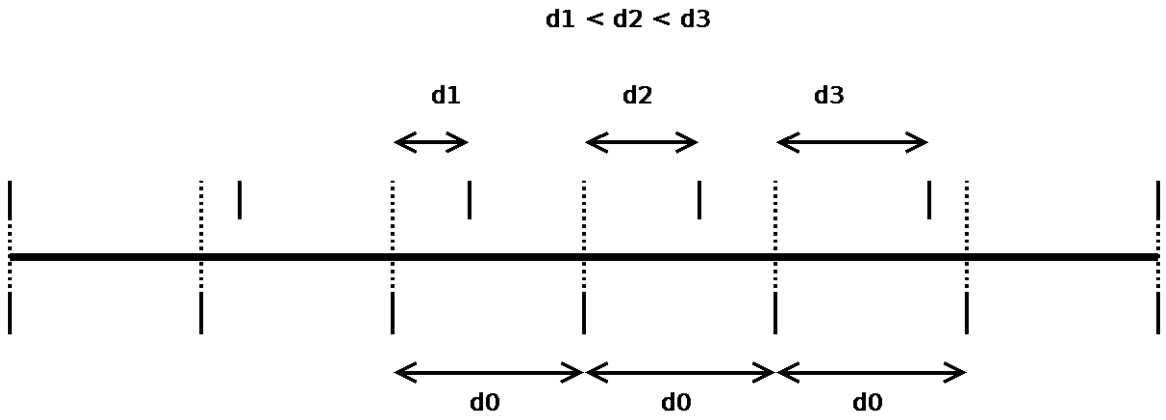


Obr. 28 - Snímek pravítka (nahoře), hledání hran (dole)

Detekce značek na klasickém pravítku bohužel neřeší problém zjišťování, kde se vlastně na pravítku nacházíme. Výjimkou by byla situace, kdy je za každých okolností v zorném poli celé pravítko - na to by byl potřeba objektiv s ohniskovou vzdáleností alespoň f = 4,51mm a i poté by uvedené platilo pouze v případě, že je pravítko umístěno přesně ve středu zorného pole. K vyhodnocení polohy relativně k počátku měřítka (pravítka) samozřejmě slouží číselné značky na pravítku. Tyto značky je ale nutné detekovat a vyhodnotit.

Číselné značky na pravítku je tak nejprve nutné lokalizovat. Pak by bylo možné je detekovat a vyhodnocovat například pomocí Bayesovského klasifikátoru, který by bylo nutno naučit tvary jednotlivých číslic. Pak by bylo třeba vyřešit problém velikosti značek – což lze udělat jednoduchým přepočtem, ale je nutné přesně ohraničit oblast, ve které se číslice nachází (a navíc vyřešit situaci, kdy se číslo skládá z více číslic). Posledním problémem tohoto přístupu je náročnost Bayesovské klasifikace na hrubý výpočetní výkon. Použití takového klasifikátoru tedy předpokládá agresivní optimalizaci celého algoritmu.

Z důvodů uvedených v předchozích odstavcích nebyla zvolena varianta použití klasického pravítka či jiného pásma s číselnými značkami, nýbrž varianta navržení a implementace vlastního měřítka, čímž se bude dát vyhnout potřebě identifikace číslic. Bylo tedy navrženo měřítko s ekvidistantními značkami, mezi kterými jsou umístěny další značky s proměnnou pozicí mezi ekvidistantními značkami. Měřítko má tedy stejný princip jako například PWM signál se zarovnáním na okraj, kdy náběžné hrany jsou ekvidistantní a sestupná hrana je klouzavá. Princip vytvořeného měřítka je znázorněn na následující straně.



Obr. 29 - Vlastní měřítko (přerušované čáry jsou jen pro ilustraci)

Z obr. 29 je zřejmě, že spodní zarážky fungují jako „náběžné hrany“ a horní zarážky jako „sestupně hrany“ pomyslného signálu PWM. Prostřední čára pak slouží jako oddělovač mezi spodními a horními značkami. Je-li zjištěna poloha oddělovače a také polohy jednotlivých značek, pak lze prostým porovnáním vertikální polohy jednotlivých objektů s vertikální polohou oddělovače oddělit ekvidistantní značky od značek klouzavých.

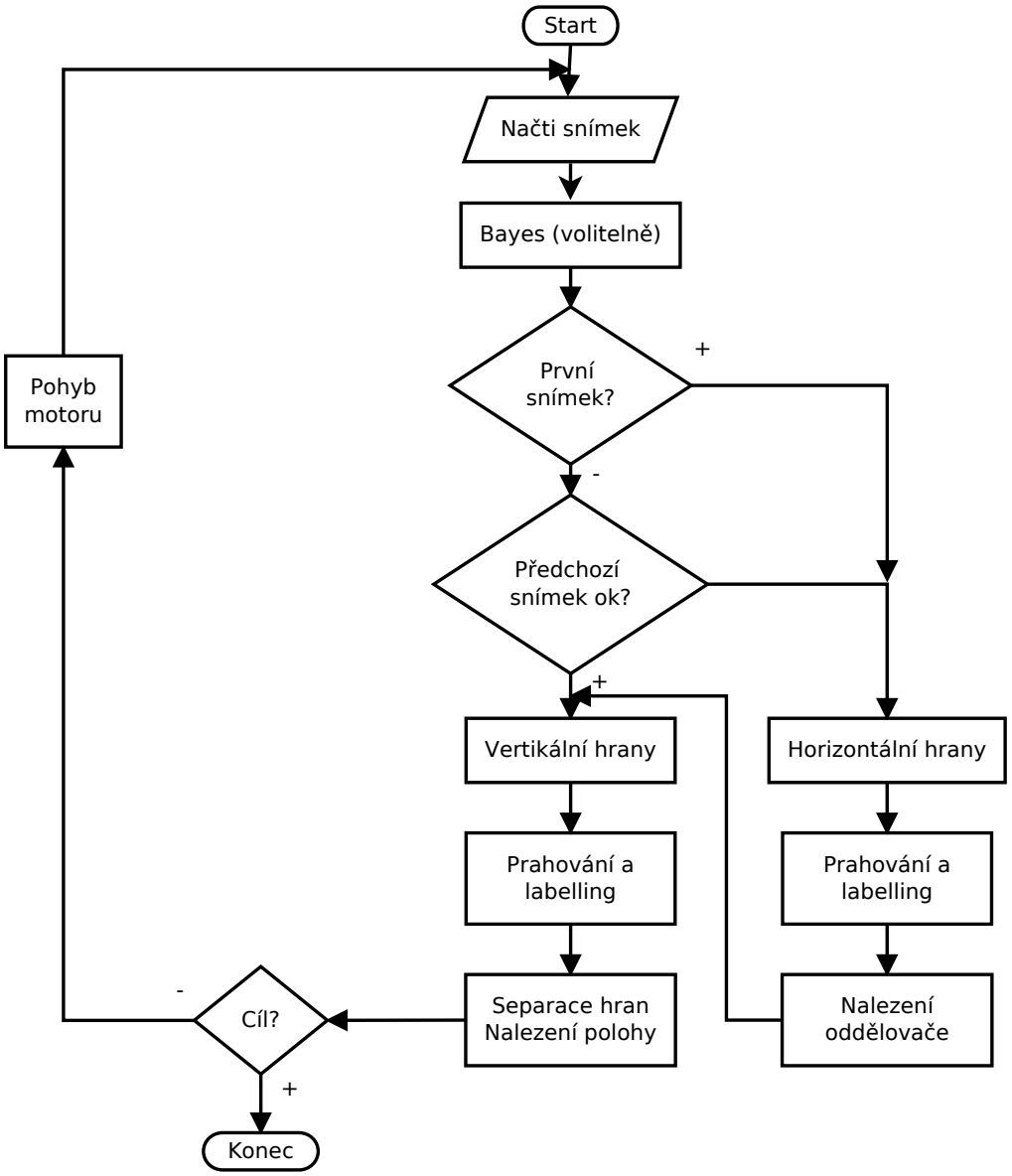
Detekce takového objektu přímo „volá“ po použití detekce hran, například operátorem Prewittové nebo Sobelovým operátorem. Nadto je však ale potřeba také identifikovat jednotlivé hrany a zjistit jejich pozice – k tomuto úkolu se dá s výhodou využít labellingu spojeného s binárním prahováním. Na výsledek operace hledání hran operátorem Prewittové lze poměrně jednoduše aplikovat prahování s velmi nízkou hodnotou, neboť výstupní obraz z hledání hran je na celé své ploše prakticky černý a hrany svým jasem velmi výrazně vystupují nad pozadí. Algoritmus hledání hran je navíc téměř imunní vůči působení vinětace objektivu. Menší problém může představovat šum v obraze, to lze však efektivně vyřešit aplikací rozostření Bayesovým filtrem.

Pozor: Je nutné si uvědomit, že využitím algoritmu hledání hran nejsou ve skutečnosti detekovány středy jednotlivých čar, ale jejich okraje (hrany) – přesněji levé okraje. Číslo vyjadřující vzdálenost od čáry je proto ve skutečnosti šířka čáry plus vzdálenost od čáry. V případě počítání vzdálenosti dvou čar to samozřejmě není problém.

Logika vykonávání programu je následující:

1. Sejmutí snímku, aplikace Bayesovského rozostření (volitelně)
2. hledání horizontálních hran (hran s gradientem ve vertikálním směru) filtrem $\begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$, aplikací prahování a labellingu
3. Nalezení horizontálního oddělovače a zaznamenání jeho vertikální polohy
4. Sejmutí snímku, hledání vertikálních hran („PWM“) stejným způsobem
5. Rozdělení vertikálních hran (na „horní“ a „dolní“)
6. Vyhodnocení polohy na pravítku

Vzhledem k tomu, že aplikací operátoru hledání hran dojde ke znehodnocení zdrojového obrazu, je třeba vstupní obrázek duplikovat ještě před použitím hledání hran (hledání hran je prováděno dvakrát, nejdříve v jednom směru a poté v druhém).



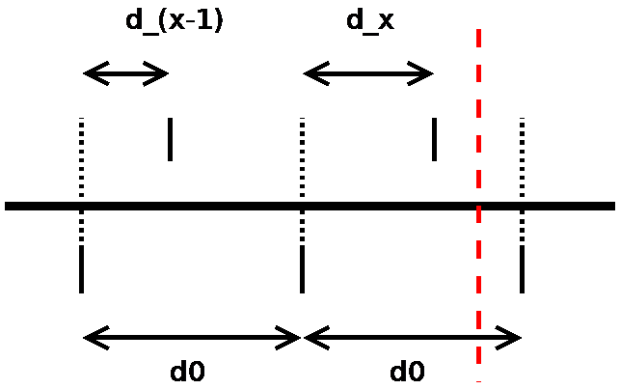
Obr. 30 - Vývojový diagram hlavního algoritmu

8.3.1. Vyhodnocení přesné polohy na měřítku

Určení vzdáleností na měřítku může být dáno různými způsoby – může být absolutní či relativní. Absolutní měřítko musí mít jasně danou vzdálenost d_0 a rovněž vzdálenosti d_n. Ty mohou být buď se známým inkrementem (tedy například 1mm), nebo jich může být známý počet (například 10, pak se vzdálenost d_n vyhodnocuje v procentech vzdálenosti d_0). Relativní měřítko pak nemusí mít jasně danou vzdálenost d_0, počet vzdáleností d_n ale musí být předem známý. V takovém měřítku se pak aktuální poloha vyhodnocuje v procentech z celkové délky měřítka.

Konkrétní měřítko, implementované pro demonstrační aplikaci, která je součástí této práce, má vzdálenost $d_0 = 2cm$ a inkrement $d_n = 1mm$. Rozsah tohoto měřítka je tedy 40cm (do vzdálenosti d_0 se vejde 20 inkrementů d_n, existuje tedy 20 rozlišitelných dvoucentimetrových intervalů), a je tedy dostačující pro použitý pojezd s délkou zhruba 30cm. Tloušťka čáry je přibližně 0,7mm.

V předchozím bodě byl obraz zpracován, byl nalezen horizontální oddělovač a také byly nalezeny jednotlivé značky podél oddělovače. V dalším kroku tedy algoritmus nalezne dvě „dolní" hrany, které jsou nejblíže horizontálnímu středu obrazu, a vypočte jejich obrazovou vzdálenost a určí si konstantu pro přepočet z pixelů na centimetry. V tomto kroku jsou zanedbány vady objektivu – především soudkovité či poduškovité zkreslení (použitý objektiv trpí na zkreslení soudkovité). Díky tomu, že jsou vyhodnocovány jen značky u středu objektivu, je vliv tohoto zkreslení zanedbatelný. Pokud by se ale rozměry měřítka posouvaly až na samotnou hranu rozlišitelnosti senzoru, mohly by vady objektivu mít větší vliv.



Obr. 31 - Výřez měřítka, červeně střed zorného pole

Algoritmus nalezení přesné polohy měřítka je následující:

- Nalezení délky intervalu d_0 mezi dvěma „spodními" značkami nejblíže ke středu obrazu
- Vyhledání „horní" značky ležící v intervalu daném nalezenými „spodními" značkami a určení její vzdálenosti od pravé „spodní" značky d_n (na obrázku výše d_2)
- Vyhodnocení aktuálního intervalu d_0 při znalosti celkového počtu n intervalů d_0 (tedy zda je právě detekovaný interval např. desátým z celkem dvaceti intervalů d_0, tedy těsně před polovinou měřítka)
    - Ze znalosti celkového počtu intervalů n je vypočten očekávaný inkrement
    $$d_x - d_{x-1} = (d_0/n)$$
    - Změřená vzdálenost d_x je vydělena očekávaným inkrementem, čímž je zjištěno kolikátý interval d_0 byl sejmut (se zaokrouhlením), označeno jako n
    $$n = d_x/(d_0/n)$$
- Nakonec je vypočtena vzdálenost středu zorného pole od první „spodní" značky a přičtena k n-násobku intervalu d_0

- Výsledek může být reprezentován buď v procentech z rozsahu měřítka, nebo při znalosti přesné velikosti d_0 může být výsledek přepočten na centimetry (nebo palce či libovolnou jinou jednotku délky).

Aktuální pozice polohovacího systému je pak dána součtem celých intervalů a zbytku daného vzdáleností středu zorného pole od počátku detekovaného intervalu d_0:

$$d = k \cdot n \cdot d_0 + k \cdot (CMOS_{CENTER} - a_n)\ [cm]$$

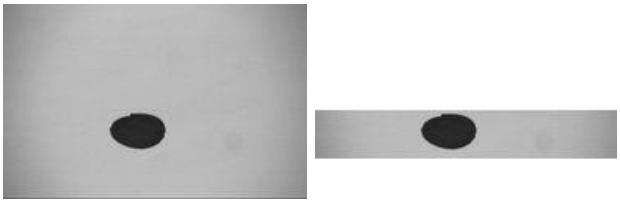
Kde k je převodní konstanta pro převod z pixelů do centimetrů, n je zjištěný počet celých intervalů d_0 a a_n je levý krajní bod detekovaného intervalu. Je tedy spočítán počet dvoucentimetrových intervalů plus vzdálenost středu obrazu od poslední (n) spodní značky převedená opět na centimetry.

8.4. Rychlost a optimalizace použitých algoritmů

V zájmu zrychlení celé aplikace (obzvláště vzhledem k relativně nízkému dostupnému výpočetnímu výkonu) je nanejvýš vhodné pokusit se optimalizovat navržené algoritmy.

8.4.1. Optimalizace detekce značky ve formě jednoho bodu

Možností, jak optimalizovat první algoritmus – detekci jednoho bodu, není mnoho. Použité algoritmy zpracování obrazu (především labelling) jsou již optimalizovány, jednou možností optimalizace tak je omezení zpracovávané části obrazu. Předpokládá-li se posun pojezdu a tedy i detekované značky pouze v horizontální ose, pak je možné velkou část obrazu při zpracovávání zanedbat. Zrychlení algoritmu je pak nepřímo úměrné velikosti zpracovávané oblasti. V případě, že ve zvolené menší oblasti není žádný objekt nalezen, provede se vyhodnocení celé oblasti. Při optimálních podmínkách by ale k tomuto kroku nikdy nemělo dojít. V tabulce níže je uveden vliv volby velikosti vyhodnocované oblasti na rychlost zpracování.



Obr. 32 - ROI - zmenšení vyhodnocovaného pole na 25%

Krok	Rychlost	Změna (proti předch.)	Změna (celkem)
Bez optimalizací	4,17Hz	-	-
Zmenšení ROI	10,87Hz	+260,7%	+260,7%
Optimalizace kompilátoru (-O3)	10,99Hz	+1,1%	+263,5%

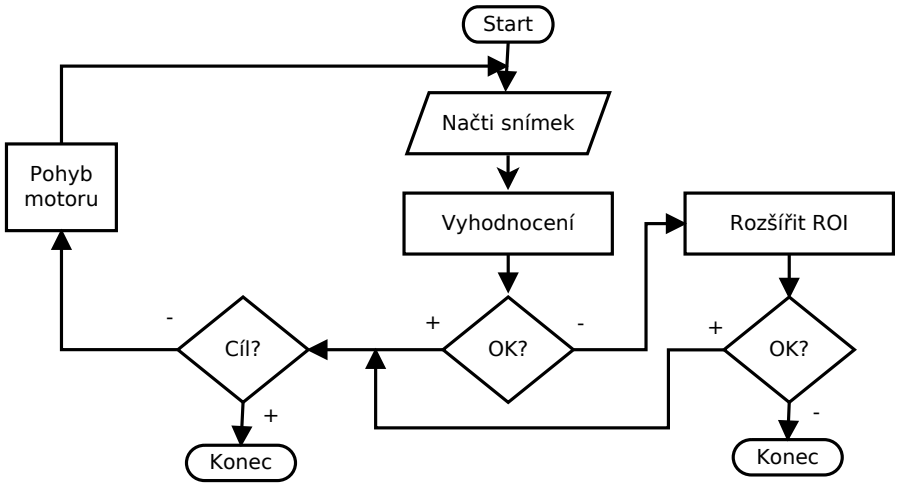
8.4.2. Optimalizace detekce vlastního měřítka

Detekce měřítka je z výpočetního hlediska podstatně náročnější. V základním stavu je vyhodnocován vždy celý snímek, a to jak hledání horizontálního oddělovače, tak vertikálních značek. Rychlost celého algoritmu v tomto případě je v tomto případě necelý jeden snímek za sekundu, což je pro předpokládané použití zcela nevyhovující.

Pro zvýšení rychlosti aplikace se horizontální oddělovač nevyhodnocuje při každém průchodu algoritmem, ale pouze pokud se nedaří nalézt vstupní data v očekávaných oblastech (tedy například všechny značky jsou nad oddělovačem a pod oddělovačem není žádná) a také v případě, že výpočtem zjištěná hodnota je mimo očekávaný rozsah (0-40cm). Z tohoto přístupu je zřejmé, že velmi záleží na správném umístění měřítka v zorném poli kamery. Tato úprava zrychlí vykonávání celého algoritmu na více než dvojnásobek, přesto jde stále přibližně o 1,13 snímků za sekundu.

Detekci hran Prewitt/Sobel operátorem nelze vynechat a také ji není možné dále optimalizovat. Pro potřeby tohoto algoritmu naštěstí stačí používat detekci hran jen v jednom směru (je to dokonce žádoucí), což nároky toho algoritmu poněkud snižuje oproti variantě s detekcí gradientů ve všech čtyřech směrech dle 7.2.

Zásadního zrychlení celé aplikace lze opět docílit omezením zpracovávané části obrazu. Tedy například místo obrazu o velikosti 752x480px může stačit i obraz o velikosti 752x120px nebo i menší. V první fázi algoritmu, ve které se hledá horizontální oddělovač, je sice nutné ponechat plnou výšku obrazu, není ale problém zmenšit obraz zúžením druhého rozměru obrazu. Tato optimalizace tedy zásadně (několikanásobně, v závislosti na zvolené šířce zorného pole) zvýší výkon celého algoritmu. Stejně jako v předchozím případě je algoritmus ošetřen tak, aby v případě selhání detekce dočasně rozšířil vyhodnocovanou oblast na celý snímek a pokusil se o detekci znovu.



Obr. 33 - Přizpůsobení velikosti vyhodnocované oblasti

V dalším kroku byla provedena další optimalizace algoritmů nad rámec „běžné“ optimalizace kódu – tedy například odstraněním složitých vzorců v podmínkách „if“ nebo „while“ nebo

nahrazením některých strukturovaných příkazů sekvenčním zápisem (za cenu mírně většího kódu). Cílem takových operací je „přinutit“ kompilátor k efektivnějšímu překladu do strojových instrukcí. Časová úspora je malá, ale ne zanedbatelná, navíc je účinnost tohoto kroku závislá na použitém kompilátoru. Přesto může být v případech, kdy je požadována skutečně co nejvyšší rychlost, dobrým nápadem podívat se na výsledný strojový kód a pokusit se v tomto duchu mírně experimentovat.

Posledním optimalizačním krokem je nastavení agresivní optimalizace kompilátoru přepínačem -Ox (x=0..3). Tento krok je vhodně provést jako úplně poslední, výsledný kód již prakticky není debugovatelný (krokovatelný). V následující tabulce jsou shrnuty přínosy jednotlivých optimalizačních kroků včetně změn jak od původního stavu, tak od předchozího optimalizačního kroku.

Krok	Rychlost	Změna (proti předch.)	Změna (celkem)
Bez optimalizací	0,56Hz	-	-
Bez oddělovače	1,13Hz	+101,8%	+101,8%
Zmenšení ROI	4,32Hz	+382,3%	+771,4%
Optimalizace algoritmů	4,51Hz	+4,4%	+805,4%
Optimalizace kompilátoru (-O3)	5,49Hz	+21,7%	+980,3%

Tab. 3 - Výsledky optimalizace použitých algoritmů

Z tabulky výše je tedy zřejmé, že všechny tyto metody přispěly k zásadnímu zrychlení aplikace, a to přibližně desetinásobnému. Tím se snímková frekvence dostává na použitelnou hodnotu.

Problematickým prvkem je případné použití Gaussova rozostření, u kterého se každý pixel násobí devíti hodnotami, což zabere opět mnoho procesorového času. Je proto nanejvýš vhodné zvážit přínos využití toho kroku při celkovém zpracování obrazu. Zapnutí tohoto algoritmu sníží rychlost vykonávání na každém optimalizačním kroku přibližně o 40% (jde o výpočetně přibližně stejně náročný algoritmus jako je detekce hran).

Ke zvýšení spolehlivosti labellingu (Tedy po vyhledání hran a převodu do binárního obrazu prahováním) je dále možno využít morfologické dilatace (podle 6.6). Po detekci hran a jejich oprahování mohou v závislosti na prahovací úrovni vznikat nespojitosti v detekovaných hranách, které pak představují problém pro algoritmus labellingu. Těchto se dá poměrně snadno a spolehlivě zbavit právě využitím morfologické dilatace v horizontálním směru v případě oddělovací čáry, nebo ve vertikálním směru v případě samotných značek. Ke stejnému účelu se dá využít i operace morfologického otevření, nicméně na dilataci potřebujeme pouze polovinu procesorového času ve srovnání s operací otevření. Zpracování obrazu se nicméně prokázalo jako dostatečně robustní i bez použití morfologických operací. Přesto však byly tyto operace implementovány a otestovány.

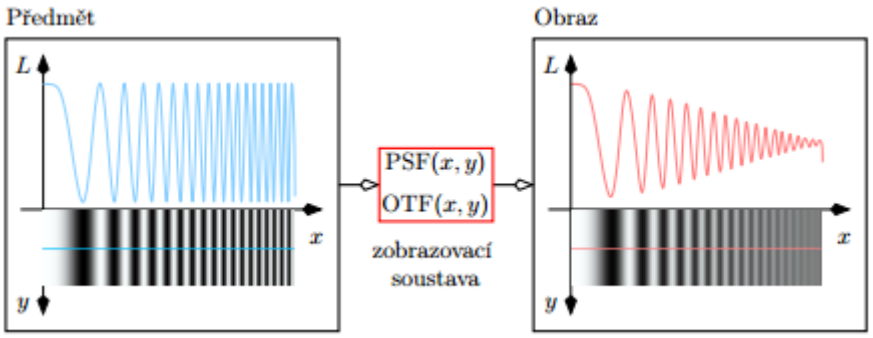
8.5. Ověření chyb měření a použitelnosti zvoleného měřítka

Omezení návrhu vlastního měřítka dle 8.3 se sestávají především z:

- Minimální tloušťky čar
- a maximálního počtu intervalů

Při odhadu minimálních nároků na parametry snímacího systému (minimální velikost značek, minimální vzdálenosti mezi jednotlivými značkami) je vhodné začít zohledněním optické přenosové funkce neboli OTF (jak objektivu, tak senzoru). OTF je objektivním měřítkem kvality optického systému, které vyjadřuje přenos kontrastu v závislosti na prostorové frekvenci optického signálu. Definováno je jako Fourierova transformace prostorové impulsní odezvy optického systému. Jde o komplexní funkci reálné proměnné, jejíž modul zjednodušeně udává podíl kontrastu obrazu v obrazové rovině vůči jeho kontrastu v předmětové rovině (pro sinusový obrazec) a její fáze udává fázový posuv obrazu v obrazové a předmětové rovině.

Prostorová impulsní odezva (PSF) pak udává prostorové rozložení jasu v obrazové rovině soustavy, pokud je na vstup tohoto systému, tedy do předmětové roviny, přiveden dvourozměrný Diracův impuls (=černý pixel). Princip je tedy stejný jako u libovolného jiného lineárního časově invariantního systému. Na obr. 34 je zřejmý vliv OTF na přenos optického signálu typu „sweep“ (také „chirp“) optickou soustavou.



Obr. 34 - Vliv PSF a OTF na přenos signálu typu „sweep“ optickou soustavou (Zdroj: [13])

Experimentálně bylo vyzkoušeno, že minimální vhodná tloušťka značky (z pohledu senzoru) je přibližně 2-3px. Minimální tloušťka čáry pak při použití objektivu s $f = 8mm$ odpovídá úhlově přibližně 0,082-0,123°, při rozměrech scény uvedených na začátku 8.1 pak zhruba 0,44-0,66mm. Zvolená tloušťka čáry 1mm je tedy zcela vyhovující, z hlediska robustnosti celého systému není použití tenčích čar vhodné.

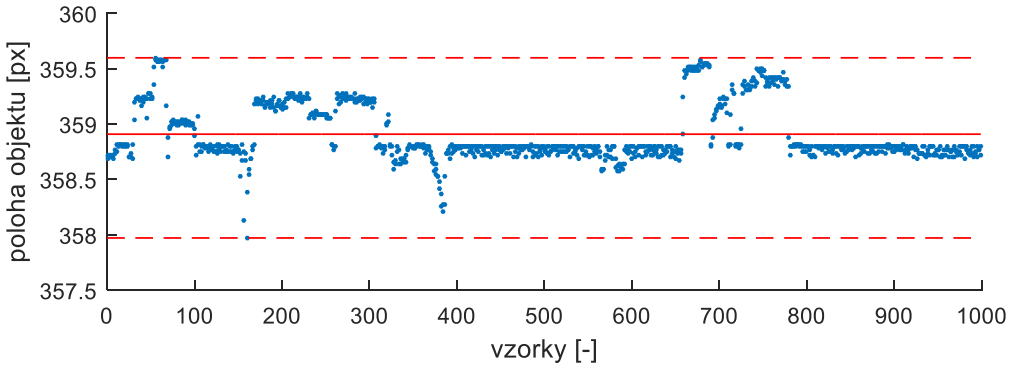
Navržený algoritmus je poměrně dobře obecně použitelný, jedinou hodnotou, kterou je nutné znát, je celkový počet intervalů d_0 pravítka (tedy například 20) – tato hodnota je nastavena přímo ve firmware (ale v principu může být samozřejmě zadávána i uživatelem v rámci inicializace systému). Celkový počet intervalů d_0 a celková délka měřítka $n \cdot d_0$ jsou pak teoreticky neomezené. V praxi je nutné volit počet intervalů tak, aby inkrement $d_n - d_{n-1}$ byl větší, než je rozlišovací schopnost systému.

Pro určení opakovatelnosti měření tak byl vytvořen algoritmus, který opakovaně snímá stejnou scénu a ukládá informace o polohách jednotlivých bodů a ukládá také největší a nejmenší hodnoty souřadnic příslušící jednotlivým bodům. Následující data byla získána snímáním scény s několika čárami velikostí odpovídajícími čarám dle 8.3 (především tloušťkou, která činí přibližně 0,7mm). Měření bylo provedeno 1000x, a to za proměnlivých světelných podmínek. Z naměřených hodnot byla vypočtena statistická nejistota (typ A).

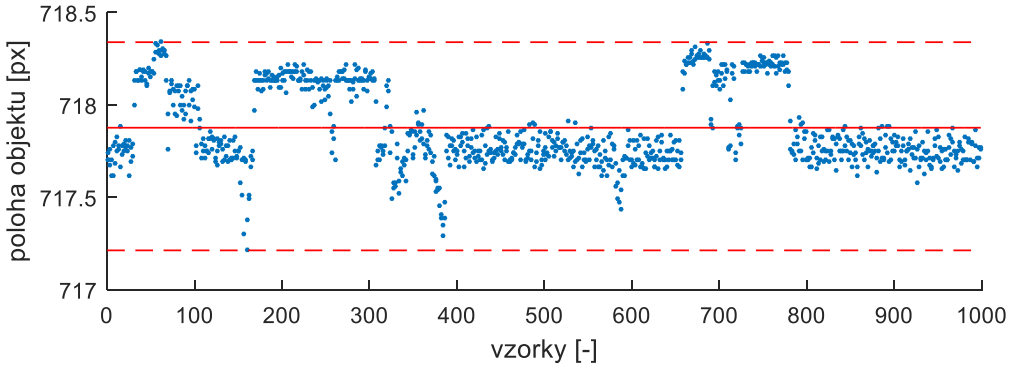
$$u_A = \sqrt{\frac{1}{n(n-1)}\sum_n (x_i - \bar{x})^2}$$

Čára	$\bar{x}$[px]	X_{max} [px]	X_{min} [px]	U_A[px]
1.	116,47	117,21	115,64	**0,0087**
2.	237,10	237,93	236,05	**0,0069**
3.	358,91	359,60	357,97	**0,0082**
4.	480,75	481,55	479,82	**0,0091**
5.	599,53	600,03	598,68	**0,0052**
6.	717,88	718,34	717,21	**0,0067**

Tab. 4 - Nejistoty snímání CMOS senzorem



Obr. 35 - Středy detekovaných objektů u středu obrazu

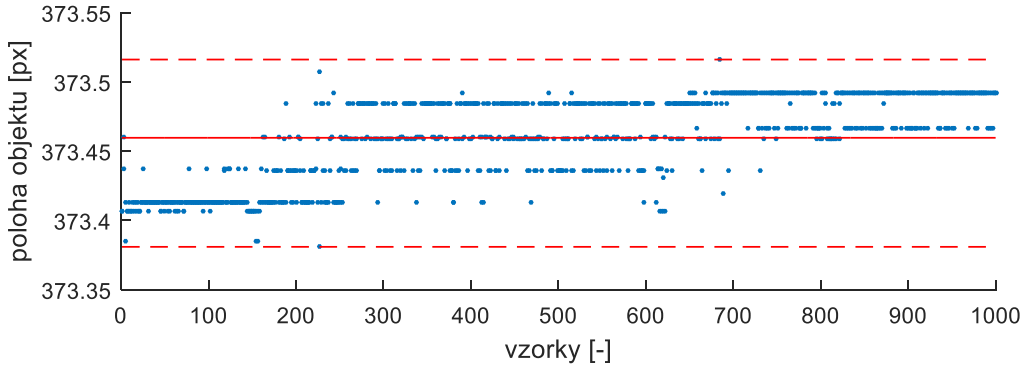


Obr. 36 - Středy detekovaných objektů u okraje obrazu

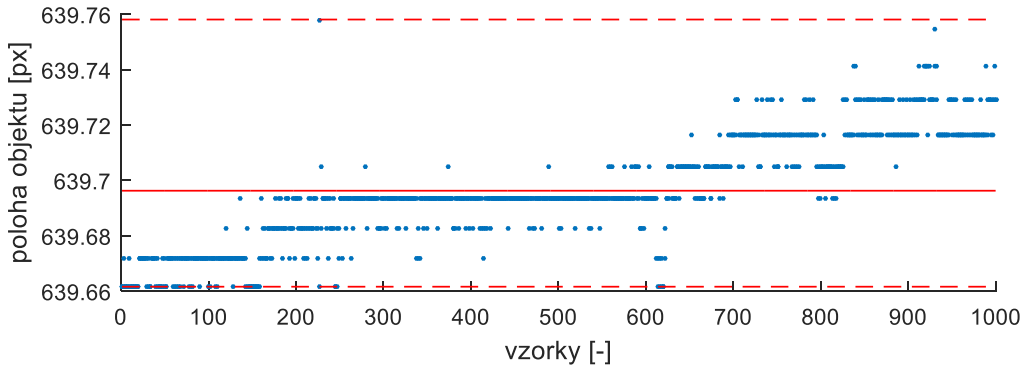
V prvním měření, jehož výsledky jsou vyneseny v obr. 35 a obr. 36, je zřejmá přítomnost společného vnějšího rušení (hrubé chyby) všech měřených hodnot, a to především v intervalech přibližně 180.-320. vzorku a 680.-780. vzorku. Z tohoto důvodu bylo měření opakováno za co nejstabilnějších optických podmínek – z následující tabulky a grafů je patrná absence hrubé chyby, což vede k daleko lepším výsledkům než v prvním případě. Z grafů na obr. 37 a obr. 38 je vidět mírný trend, způsobený pravděpodobně především změnou přirozeného osvětlení v časovém intervalu celého měření (měření bylo prováděno za osvětlení denním světlem) a při použití umělého stacionárního zdroje světla by pravděpodobně i tento trend z naměřených hodnot zmizel.

Čára	$\bar{x}$[px]	X_{max} [px]	X_{min} [px]	U_A[px]
1.	107,57	107,64	107,48	**$7{,}26.10^{-4}$**
2.	239,49	239,53	239,46	**$6{,}25.10^{-4}$**
3.	373,46	373,52	373,38	**$9{,}65.10^{-4}$**
4.	504,31	504,46	504,22	**$9{,}22.10^{-4}$**
5.	639,70	639,76	639,66	**$6{,}08.10^{-4}$**

Tab. 5 - Nejistoty snímání CMOS senzorem



Obr. 37 - Středy detekovaných objektů u středu obrazu

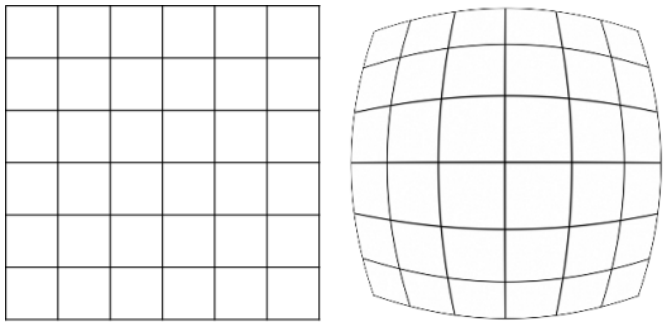


Obr. 38 - Středy detekovaných objektů u okraje obrazu

Pokud z naměřených hodnot odhadneme (z obr. 37, který patří bodu naměřenému s největší nejistotou) hodnoty šumu špička-špička (vzhledem k absenci průměrování ve standardním režimu snímání), dostáváme se na nejistotu určení polohy čáry přibližně 0,2px (tedy 0,045mm při rozlišení 0,22mm/px). Zvolený inkrement vlastního měřítka (1mm) je tedy přibližně 20krát větší, než je maximální šum při detekci polohy značky, a je tedy zvolen bezpečně. Má-li být pro korektní detekci značky hodnota šumu menší než polovina uvažovaného inkrementu, pak tedy může být tento inkrement oproti navrhovanému měřítku ještě několikanásobně menší.

Pokud by bylo snímání průměrováno z více snímků, pak by bylo možné volit i menší inkrementy.

Nejistotu typu B, tedy nejistotu vzniklou jiným než statistickým zpracováním, v tomto případě představuje především chyba linearity objektivu a také minimální rozlišení použitého CMOS senzoru. Pro objektiv s $f = 8mm$ tedy připadá na jeden pixel obrazu přibližně 0,22mm v předmětové rovině (viz 8). Použitý objektiv rovněž trpí na soudkovité zkreslení, kdy se značky (a také jejich vzdálenosti) zmenšují se zvyšující se vzdáleností od středu obrazu. Na následujícím grafu je zřejmý efekt soudkovitého zkreslení na vzdálenosti jednotlivých bodů v sejmutém obraze a zároveň je zřejmé, že při použití takového objektivu pouze v jednotkách procent ze zorného úhlu by mělo být možné toto zkreslení zanedbat:

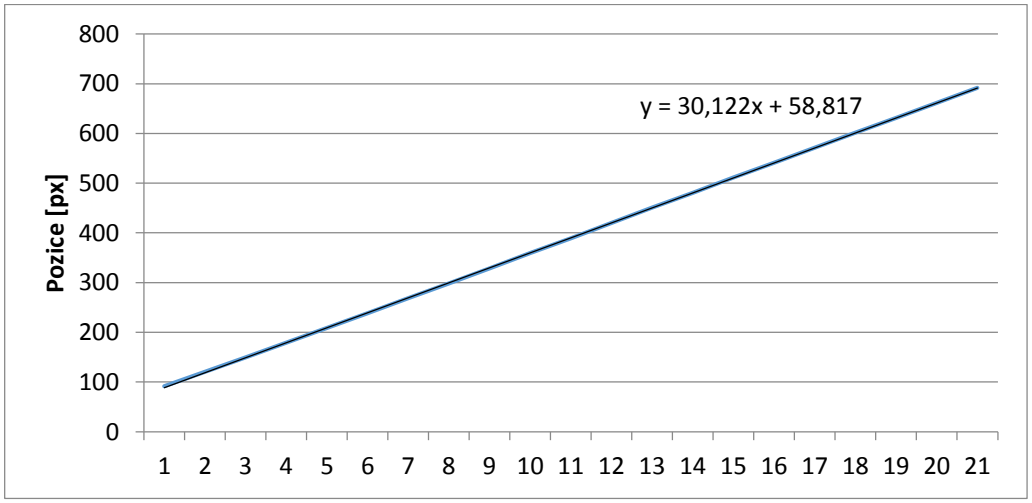


Obr. 39 - Soudkovité zkreslení (vpravo)

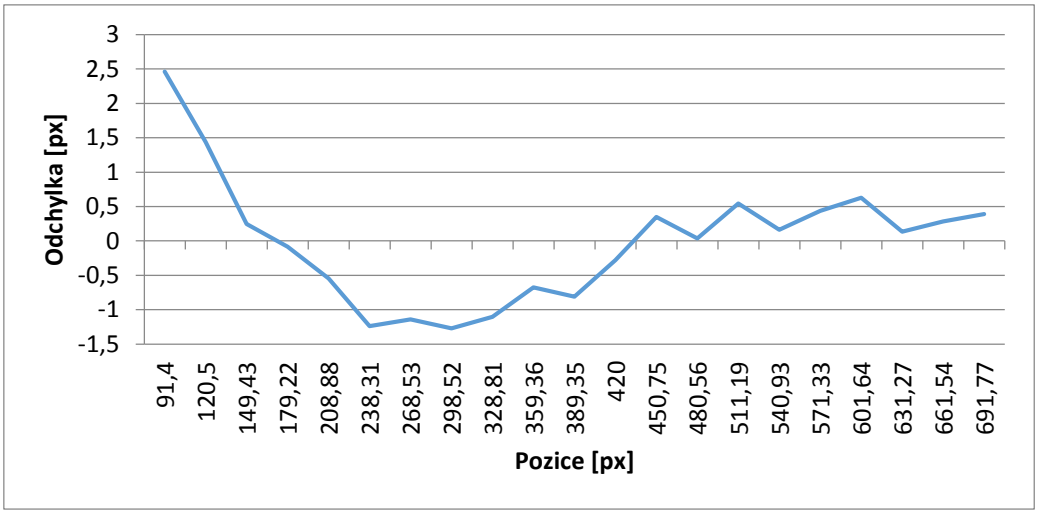
Pro potřeby této úlohy nebyl efekt soudkovitého zkreslení zohledněn – v případě použití kvalitnějšího objektivu za účelem skutečně precizního měření by ale bylo vhodné jej zohlednit. Pokud by objektiv trpěl vysokou nelinearitou zobrazení, mohlo by se při měření vzdálenosti dvou značek blízko okraje zorného pole stát, že by byl detekovaný interval po kvantování vyhodnocen jako kratší, než ve skutečnosti je, což by vedlo ke špatné detekci polohy na měřítku. Efekt soudkovitého zkreslení v navrhované aplikaci je ale minimalizován detekcí objektů pouze blízko středu obrazu. Pro potvrzení bylo provedeno měření linearity zobrazení objektivem v celém zorném poli snímače – k testu posloužil stejný druh značky, jaký byl použit k vytvoření vlastního měřítka, a který byl opakovaně snímán, vyhodnocován a posouván v ekvidistantních krocích ~6,6mm s využitím krokového motoru. Změřené polohy detekované značky byly vyneseny do grafu a byla spočtena jejich odchylka od linearity:

Číslo náměru	Určená pozice[px]	Odchylka od linearity[px]
1	*91,4*	2,461
2	*120,5*	1,439
3	*149,43*	0,247
4	*179,22*	-0,085
5	*208,88*	-0,547
6	*238,31*	-1,239
7	*268,53*	-1,141
8	*298,52*	-1,273
9	*328,81*	-1,105
10	*359,36*	-0,677
11	*389,35*	-0,809
12	*420*	-0,281
13	*450,75*	0,347
14	*480,56*	0,035
15	*511,19*	0,543
16	*540,93*	0,161
17	*571,33*	0,439
18	*601,64*	0,627
19	*631,27*	0,135
20	*661,54*	0,283
21	*691,77*	0,391

Tab. 6 – Změřené polohy objektů a odchylky od linearity



Obr. 40 - Změřené hodnoty proložené spojnicí (metoda nejmenších čtverců)

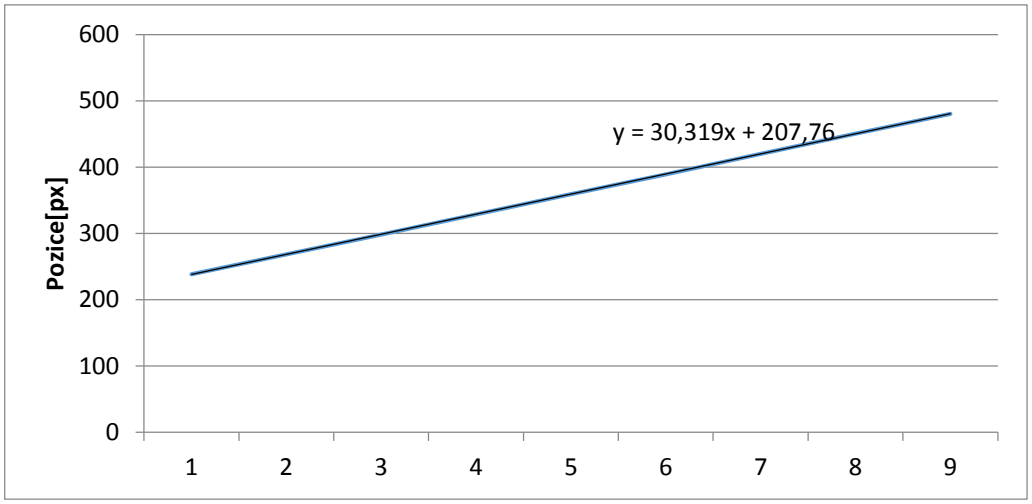


Obr. 41 - Odchylka od linearity přes celý měřicí rozsah

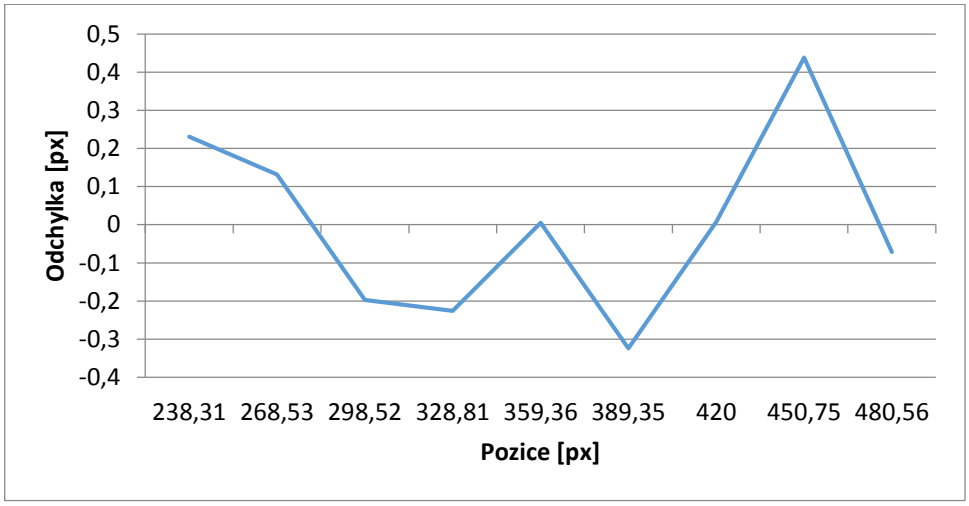
Jak je z naměřených dat zřejmé, chyba linearity zobrazení v rozsahu zorného pole 600px je přibližně 1,2px. To tedy představuje $1{,}2/600 = 0{,}2\%$ chybu linearity v tomto rozsahu (tj. přibližně 0,26mm). Jak již bylo řečeno, v samotné navrhované aplikaci je vždy vyhodnocována jen malá oblast u středu zorného pole snímače. I v nejextrémnějším případě je proto maximální rozsah zorného pole (při použití měřítka se vzdálenostmi bodů 2cm) jen 4cm. Bylo tedy provedeno vyhodnocení chyby linearity zobrazení i pro tento omezený rozsah, u kterého je předpoklad, že je hledaná chyba linearity ještě menší:

Číslo náměru	Určená pozice[px]	Odchylka od linearity[px]
1	*238,31*	0,231
2	*268,53*	0,132
3	*298,52*	-0,197
4	*328,81*	-0,226
5	*359,36*	0,005
6	*389,35*	-0,324
7	*420*	0,007
8	*450,75*	0,438
9	*480,56*	-0,071

Tab. 7 - Změřené polohy objektů a odchylky od linearity v omezeném rozsahu



Obr. 42 - Změřené hodnoty proložené spojnicí (omezený rozsah)



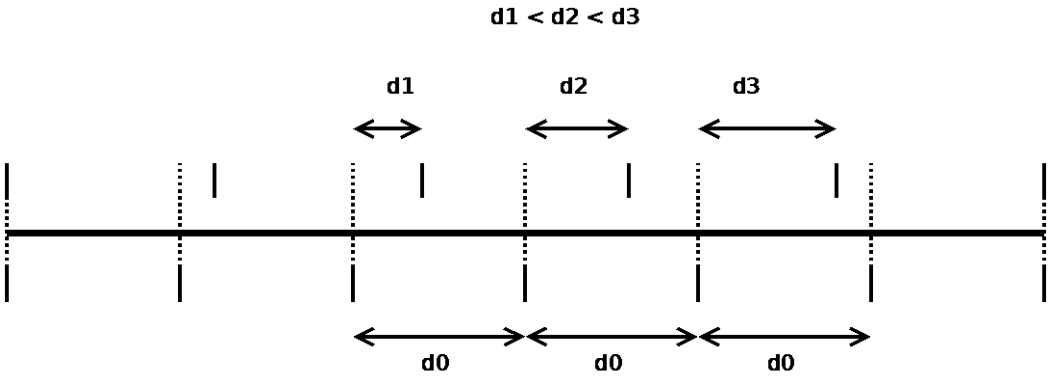
Obr. 43 - Odchylka od linearity přes omezený měřicí rozsah

Dle předpokladu je chyba linearity zobrazení v tomto omezeném rozsahu menší než v celém měřicím rozsahu a představuje přibližně 0,2px v rozsahu zorného pole 240px. To tedy představuje chybu linearity $0{,}2/240 = 0{,}08\%$ (tj. přibližně 0,044mm). To představuje řádové zmenšení chyby linearity objektivu při omezení využívaného zorného pole snímače. Chyba linearity je tedy dostatečně malá pro použití navrhovaného měřítka s minimálním inkrementem o velikosti 1mm.

8.5.1. Limity návrhu vlastního měřítka

V případě potřeby většího měřítka je nutné buď navýšit počet intervalů až k hranici dané chybami měření určenými v předchozí kapitole, nebo je naopak nutné zvětšit velikost intervalu d_0. Velikost intervalu d_0 je ale opět možné zvětšovat pouze do takové míry, aby její velikost představovala maximálně polovinu velikosti zorného pole senzoru (v opačném případě není možné zaručit, že se v zorném poli senzoru bude nacházet vždy alespoň jeden celý interval). V modelové situaci z bodu 8.1 pak při použití objektivu s $f = 8mm$ je maximální velikost d_0

rovna zhruba 8,34cm (zorný úhel **31,05°**, vzdálenost měřítka **30cm,** celková délka měřítka v zorném poli **16,68cm**).



Obr. 44 - Vlastní měřítko
(přerušované čáry jsou jen pro ilustraci)

Minimální tloušťka čáry je dána použitým objektivem a vzdáleností měřítka od objektivu. Jak bylo popsáno v předchozí kapitole, minimální tloušťka čáry z pohledu senzoru je alespoň 2-3 pixely. Při horizontálním rozlišení senzoru 752 pixelů to pak představuje $3/752$ úhlu zorného pole, neboli **0,4%** úhlu zorného pole. Při použití objektivu s $f = 8mm$ a fyzickém rozměru aktivní plochy snímače 4,51mm je zorný úhel přibližně 31,05° a tak je minimální tloušťka čáry úhlově **0,082-0,123°,** lineárně **0,44-0,66mm**. K výpočtu je možné využít následující vztah (více v 8.1):

$$\alpha = \frac{3}{752} \cdot 2 \arctan\left(\frac{d}{2f}\right)$$

V případě použití objektivu s $f = 8mm$ a vzdálenosti měřítka 30cm od objektivu je pak maximální délka $d_0 = 8{,}34cm$ a bezpečný minimální inkrement (dle 398.5) je přibližně 0,5mm. To představuje zhruba 166 intervalů a celkovou délku měřítka 1384,4cm, tedy 13,8m. Při použití delšího ohniska se zmenšuje zorný úhel a tedy i maximální délka intervalu d_0, snižuje se ale také minimální tloušťka čáry. Při použití kratšího ohniska se analogicky zvětšuje zorný úhel a tedy i maximální délka intervalu d_0, zvyšuje se ale také minimální tloušťka čáry.

Posledními faktory, které ovlivňují limity navrženého měřítka, jsou rozlišení a fyzické rozměry senzoru a jeho vlastnosti. Ty se v rámci této diplomové práce nemění, v principu však není problém nahradit použitý senzor lepším. Pro následující shrnutí se předpokládá použití objektivu se stálou ohniskovou vzdáleností:

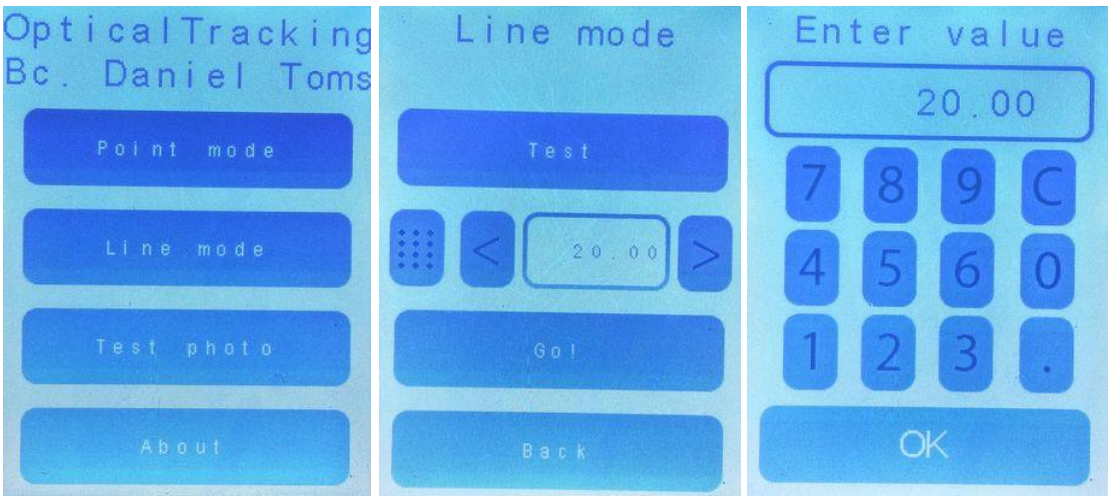
- Senzor s vyšším rozlišením a stejnými rozměry si zachovává velikost zorného pole, má však větší úhlové rozlišení. To dovoluje použití tenčích čar a tím pádem i více intervalů d_0. V praxi však se se zvyšováním počtu pixelů při zachování rozměrů senzoru více projeví přenosová funkce objektivu OTF (více v 8.1), který již není schopen přenést tak velkou změnu kontrastu na prostorovou jednotku, což má za následek nutnost použití tlustších čar. Zmenšením fyzické velikosti jednotlivých pixelů se také zmenšuje aktivní

plocha, na kterou dopadají fotony. To je nutné kompenzovat prodloužením doby expozice, což představuje problém především v horších světelných podmínkách.

- Senzor se stejným rozlišením a většími fyzickými rozměry má širší zorné pole a umožňuje tak zvětšit interval d_0. Také ale vyžaduje tlustší čáry. To tedy znamená zvětšení absolutní velikosti celého měřítka za cenu zmenšení úhlového rozlišení a tedy i snížení přesnosti určení absolutní polohy.
- Senzor s vyšším rozlišením a většími rozměry pak představuje to nejlepší z obou světů, je však nutné použít větší objektiv (na rozdíl od situace, kdy se použije objektiv pro větší senzor s menším senzorem, což pouze změní efektivní ohniskovou vzdálenost).

9. GUI pro embedded LCD

Vzhledem k tomu, že jsou na obou kitech (F429-Discovery a F7-Discovery) osazeny LCD displeje s dotykovými vrstvami, nabízí se naprogramování grafického rozhraní, kterým by bylo možné embedded aplikaci ovládat i bez nadřazeného PC (byť s omezenými možnostmi nastavení). To by bylo výhodou při samostatném (standalone) použití například při demonstracích v laboratořích či při prezentacích na dnech otevřených dveří či veletrzích.



Obr. 45 - Ukázka embedded GUI

Varianty, jak vytvořit embedded GUI, jsou dvě – vytvoření vlastního GUI, nebo využití některého z dostupných GUI frameworků. Výhodou vývoje vlastního GUI je potenciálně menší velikost kódu a výpočetní náročnost především díky tomu, že je GUI psáno pro konkrétní hardware bez jakékoliv hardwarové abstrakce, nevýhodou je náchylnost k programátorským chybám a také časová náročnost celé implementace. Výhodami externího GUI frameworku je jeho relativní univerzálnost, kdy je celá horní vrstva frameworku hardwarově nezávislá (až na samotný layout grafického rozhraní, které je samozřejmě závislé na rozlišení displeje).

Mezi uvažované GUI frameworky patřila knihovna STemWin, což je komerční framework emWin od společnosti Segger, který je zdarma dostupný pro vývoj na kitech ST-Discovery

(jinak jde o placený produkt). Nevýhodami této knihovny jsou značná velikost po kompilaci a také distribuce frameworku ve formě předkompilované knihovny bez přístupu ke zdrojovým kódům.

Druhým uvažovaným je uGFX Framework (ugfx.org), jehož podstatnou výhodou je jeho nenáročnost na systémové prostředky (RAM i Flash), protože se skutečně kompilují jen ty součásti, které uživatel potřebuje, ale také proto, že je celé grafické rozhraní ve srovnání se STemWinem poměrně jednoduché a postrádá některé pokročilejší „widgety" (ty důležité, jako je například checkbox, radiobox, list, slider nebo textbox mu samozřejmě nechybí). V rámci diplomové práce byl otestován a částečně zprovozněn uGFX framework na obou kitech (F429, F746), bohužel ale nebylo dosaženo stoprocentně korektního chování tohoto frameworku (problémy s detekcí dotyku a další menší chyby). Po konzultaci přímo s autory tohoto frameworku bylo zjištěno, že jde o omezení dána specifickým chováním použitého kompilátoru ARMCC (jde o problém v algoritmu přepínání úloh, který je v uGFX používán). Toto omezení by bylo možno obejít použitím jiného kompilátoru (například arm-gcc), nebo implementací RTOS ChibiOS, u kterého je pak možné použít vestavěný scheduler, který tímto problémem netrpí.

Vzhledem k rozsahu celého projektu by bylo kompletní přepsání pro ChibiOS velmi časově náročné, proto bylo nakonec rozhodnuto naprogramovat vlastní jednoduché grafické rozhraní. Kit F429-Discovery je osazen displejem o úhlopříčce 2,4" s rozlišením 240x320px (tedy v režimu portrait), s rezistivní dotykovou vrstvou a ovladačem STMPE811, připojeném skrze rozhraní I^2C. Jak bylo uvedeno v kapitole 5.2, připojení LCD a dotykového ovladače má kvůli nedostačujícímu LQFP144 pouzdru a přítomnosti dalších periferií (jako je například externí SDRAM) mnoho kolizních vodičů s kamerovým rozhraním DCMI, proto není možné zároveň snímat obraz a zobrazovat data na LCD displeji. Toto omezení se netýká kitu F746-Discovery, který používá větší pouzdro BGA216.

9.1. Design grafického rozhraní

Grafické rozhraní na embedded displeji je jednou ze dvou variant ovládání celé aplikace. Druhou variantou je použití počítačové aplikace, která je popsána v kapitole 10. Počítačová aplikace ve srovnání s embedded GUI poskytuje více možností nastavení a také poskytuje výstup z CMOS senzoru v plném rozlišení. Embedded GUI naproti tomu poskytuje možnost rychlého použití celé aplikace v případě, že není k dispozici běžné PC.

Součástmi embedded grafického rozraní jsou:

- Hlavní nabídka
  - volba typu snímané značky
  - zobrazení testovacího snímku (bez zpracování a bez regulace)
  - informace o aplikaci
- Nabídka snímání značky a následné regulace
  - spuštění testovacího snímání (se zpracováním a vyhodnocením, bez regulace)
    - podle zvoleného typu snímané značky
  - nastavení cílu regulace (s tlačítky +/-, případně numerickou klávesnicí)
  - spuštění regulace polohy

GUI tedy podporuje oba navržené režimy snímání a vyhodnocování – režim jednoho bodu i režim měřítka. V režimu jednoho bodu může uživatel zadat cílovou pozici, které má snímaný bod dosáhnout, a to v rozsahu 0-752 (odpovídá počtu pixelů na širší straně použitého CMOS senzoru. V režimu měřítka pak uživatel může zadat cílovou pozici na měřítku (omezení pak vychází z délky pojezdu a použitého měřítka – proto ve finálním softwaru žádné omezení implementováno není, je tedy na uživateli, aby zadal korektní hodnotu) v centimetrech a milimetrech. Pro urychlení celé aplikace byla implementována malá tolerance, do které se regulovaný systém musí dostat – jde o přibližně jeden milimetr.

GUI dále podporuje režim testu, a to buď základního testu, kdy pouze sejme snímek a zobrazí uživateli (aby bylo možné nastavit správně zorné pole kamery a ujistit se, že je snímaný obraz v pořádku), nebo testu snímání měřítka. V tomto režimu program sejme snímek a provede na něm zvolený algoritmus vyhodnocení polohy. Regulace polohy se v tomto režimu neprovádí, výstupem je pouze informace o tom, zda se zvolené měřítko podařilo nalézt a jaká je jeho pozice.

Vzhledem k časové náročnosti programování embedded GUI není možné přes toto GUI nastavovat parametry snímání, jako je například převrácení obrazu, způsob nastavení doby expozice (viz kapitola 6) ani použité algoritmy. Při použití PC aplikace si tak uživatel může prohlédnout a vyzkoušet všechny použité algoritmy zpracování obrazu a také měnit jejich parametry, při použití embedded GUI jsou tyto parametry pevně dány a jsou nastaveny na takové hodnoty, aby byla aplikace funkční ve většině světelných podmínek. Jedinou výjimkou je možnost nastavení prahovací úrovně v režimu snímání jediného bodu, kterou je velmi obtížné (až nemožné) spolehlivě nastavit tak, aby fungoval za všech okolností. Algoritmus pro detekci měřítka, který je hlavní funkcí celé aplikace, je ale podstatně robustnější a tak za běžných světelných podmínek není potřeba měnit jeho nastavení.

Schéma nabídek vytvořeného grafického rozhraní je následující:

- Hlavní nabídka
  - Bodová regulace
    - Test
    - Nastavení cíle
      - Numerická klávesnice
    - Regulace
    - Zpět
  - Regulace s měřítkem
    - Test
    - Nastavení cíle
      - Numerická klávesnice
    - Regulace
    - Zpět
  - Testovací obrázek
  - O aplikaci

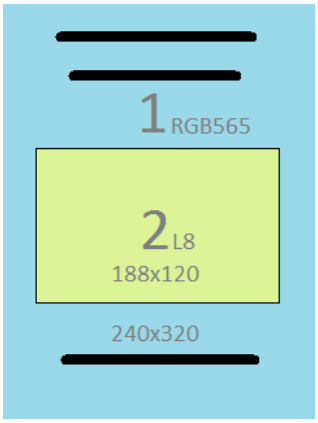
9.2. Programování grafického rozhraní

Po spuštění systému je automaticky spuštěna rutina zobrazení LCD a snímání dotyku stálým dotazováním (touch kontrolér nicméně podporuje také režim přerušení dedikovaným vodičem). Grafické elementy (tlačítka, fonty, …) jsou uloženy jako bitmapy v paměti flash. K tomuto kroku bylo přistoupeno z důvodu šetření časem za cenu větší náročnosti na paměť flash oproti procedurálnímu generování všech grafických prvků.

Zobrazování elementů je prováděno přímým zápisem do vyhrazeného snímkového bufferu v paměti SDRAM. Pro zobrazování snímku z CMOS kamery je využívána schopnost vestavěného LCD kontroléru (LTDC Controller procesoru STM32F429/F746) pracovat se dvěma vrstvami, které mají vlastní snímkové buffery (a u kterých lze dále nastavit jejich prolínání, lze jimi pohybovat v rámci celé viditelné plochy LCD displeje a další funkce, vše bez asistence mikrokontroléru, tzn. bez vlivu na výpočetní výkon).

Zásadní výhodou použití dvou oddělených vrstev je možnost nastavit odlišné barevné režimy. Zatímco první vrstva zobrazuje barevně v režimu RGB565 (16bit), druhá vrstva zobrazuje pouze černobíle v režimu L8 (8bit luminance, černobílá s 256 odstíny šedi). Pokud by se obraz ze senzoru měl zobrazovat do první vrstvy, musel by se každý pixel převést z formátu L8 do formátu RGB565, zatímco při zobrazování ve druhé vrstvě žádné přepočty nejsou potřeba. Použitý senzor MT9V034 má rozlišení 10bpp (bits-per-pixel), lze jej ale nastavit i do režimu 8bpp. Formát výstupních dat pak přesně odpovídá formátu snímkového bufferu LCD kontroléru v režimu L8.

Velikost druhé vrstvy je tak nastavena na přesnou velikost snímaného obrazu a umístěna zhruba doprostřed plochy LCD displeje. DMA kontroléru, který se stará o přenos dat ze senzoru, je pak nastavena cílová adresa na snímkový buffer druhé vrstvy. Zobrazování dat na LCD displeji pak tedy probíhá zcela bez asistence mikrokontroléru a je okamžité. Vzhledem k velikosti LCD displeje je v CMOS senzoru nastaven režim čtyřnásobného slučování (binningu) pixelů v obou rozměrech, efektivně jde tedy o šestnáctkrát zmenšený obraz (752x480px versus 188x120px). To platí pouze pro testovací snímky sloužící k vizuální kontrole uživatelem, zpracování obrazu pro regulaci polohy samozřejmě probíhá v plném rozlišení.

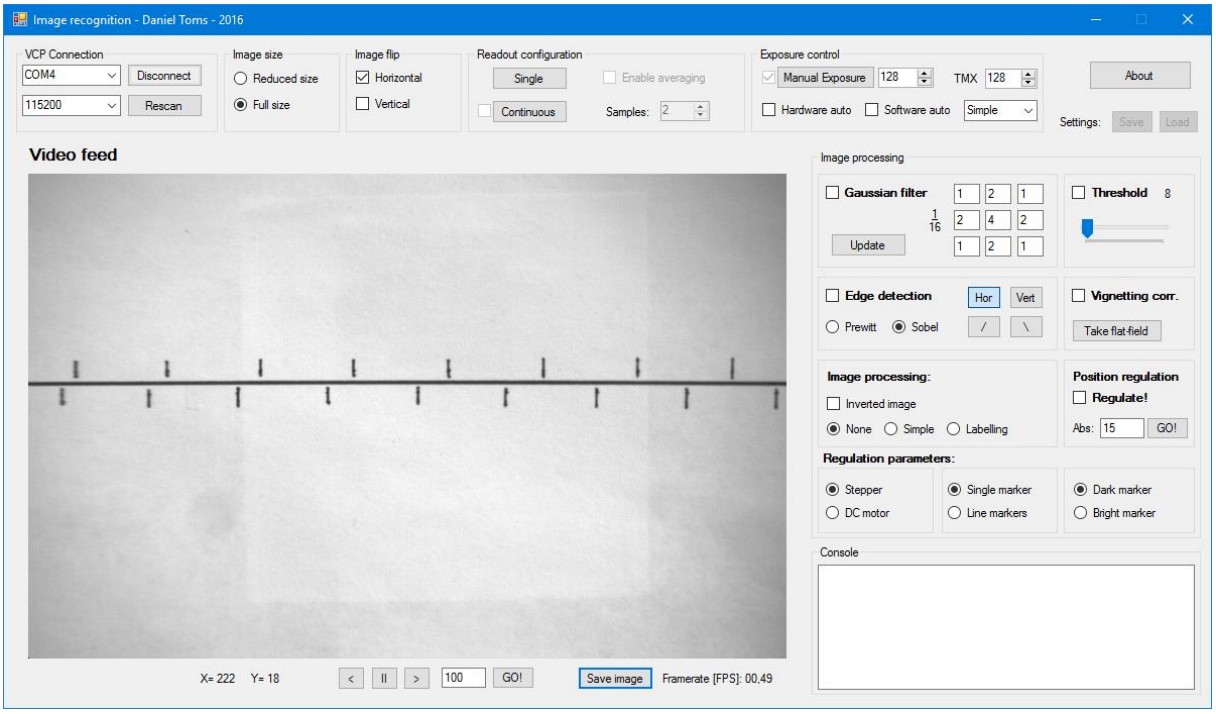


Obr. 46 - Režim dvou vrstev

Jak již bylo řečeno, snímání dotyku je řešeno stálým dotazováním ve smyčce, v případě připojení PC aplikace je smyčka přerušena a program přechází do standardního režimu vyčkávání na příkazy z nadřazeného počítače. V případě odpojení PC aplikace se program opět vrací do režimu ovládání dotykovým panelem.

10. PC aplikace pro komunikaci a ovládání kitu

Vytvořenou aplikaci lze vedle embedded GUI ovládat také pomocí PC aplikace. Tato aplikace umožňuje kompletně nastavit veškeré parametry snímání a vyhodnocení a také zobrazit obraz ze senzoru. Aplikace je psána v jazyce C# s využitím grafického frameworku Windows Forms. Použitým vývojovým prostředí je Microsoft Visual Studio 2015 Community, což je vývojové prostředí poskytované společností Microsoft bezplatně pro všechny samostatné vývojáře, open source projekty, vědu a vzdělávání a malé profesionální týmy do 250 lidí či ročního příjmu do 1.000.000 USD. Projekty vytvořené touto edicí mohou být komerčního i nekomerčního charakteru. Komunikace s mikrokontrolérem probíhá pomocí virtuální sériové linky (Virtual COM port) s volitelnou rychlostí.



Obr. 47 - PC aplikace pro ovládání kitu F429-Discovery (C# WinForms)

Aplikace přijímá obrazová data z mikrokontroléru a zobrazuje je uživateli – to představuje mimo jiné užitečný způsob, jak ladit parametry snímání (nastavení filtrů, expozice a dalších). Obrázek lze uložit tlačítkem „Save Image“ ve formátu BMP (Windows Bitmap). Aplikace dále přijímá také textová data, která vypisuje uživateli (typicky informace o velikostech a polohách předmětů detekovaných v zobrazeném snímku). Mezi nastavované parametry patří:

- Parametry snímání
  - Plný / redukovaný obraz
  - Doba expozice
    - Manuální, automatické, automatické s parametry
  - Vodorovné / svislé převrácení
  - Průměrování – 2-16 vzorků
  - Inverzní obraz
  - Snímání jednotlivě / průběžně
- Parametry zpracování obrazu
  - Použití Gaussova filtru a zadání samotného filtru
  - Použití filtru Prewittové
  - Nastavení prahování a prahovací hodnoty
  - Nastavení korekce vinětace a snímání korekčního snímku
  - Tmavá / světlá značka
  - Jedna značka / měřítko (více značek – např. pravítko)
  - Nastavení image processingu
    - Labelling, jednoduchá metoda (překrývání pruhů, viz [1])
  - Spuštění připojeného motoru pro regulaci polohy

10.1. Formát datové komunikace s vývojovým kitem

10.1.1. Komunikace mezi PC a Discovery

Komunikaci ve směru PC->Discovery obstarává několik krátkých paketů jednoduchého protokolu:

Byte	0	1	2	3+
Snímání	's'			
Stop	't'			
Korekce vinětace	'f'			
Uložit nastavení	'S'			
Nahrát nastavení	'L'			
Konfigurace	'c'	'f' (horizontální flip)	0/1 (on/off)	
		'g' (vertikální flip)	0/1 (on/off)	
		'e' (expozice)	'm' (manual)	0x00-0xFF (Bits 8-15)
		'e'	'h' (hardware auto)	
		'e'	's' (software auto)	0x01-0x04 (typ), 0x00-0xFF (Cílový jas)
		'i' (invertování)	0/1 (on/off)	
Processing	'p'	'g' (Gauss)	0/1 (on/off)	Bytes 3-11: Hodnoty matice 3x3
		'e' (Edge Detection)	1-4 (direction)	0/1/2(off/Prewitt/Sobel)
		't' (thresholding)	0/1 (on/off)	0x00-0xFF (Threshold)
		'v' (korekce vinětace)	0/1 (on/off)	
		'r' (image processing)	0/1/2 (žádné/rezervováno/labelling)	
Tracking + motor	'G'	0/1 (tracking on/off)	1/2 (Krokový /DC motor)	
Marker type	'm'	0/1/2 (žádný/bod/měřítko)	1/2 (tmavý/světlý)	Bytes 3-6: cílová pozice v cm (float)
Move	‚M‘	0/1/2 (stop/doprava/doleva)	1/2 (krokový /stejnosměrný)	0x00-0xFFFFFFFF (kroky)

Tab. 8 - Komunikační protokol ve směru PC -> Discovery

10.1.2. Komunikace mezi Discovery a 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)

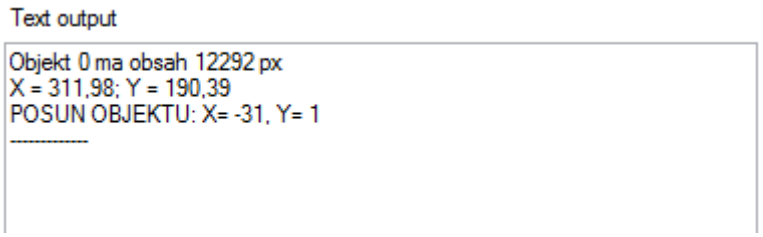
Tab. 9 - Byty paketu DATA

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. 10 - 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. 48 - Ukázka dat přijatých PC aplikací

11. Přizpůsobení projektu pro kit F7-Discovery

Dalším bodem této diplomové práce bylo přizpůsobení celého projektu pro kit STM32F7-Discovery. Tento kit, uvedený v roce 2015, je se svým novým jádrem ARM Cortex-M7 výrazně výkonnější než kity s jádry Cortex-M4 a navíc je zajímavě vybavený – mimo jiné velkým 4,3“ LCD displejem s kapacitní dotykovou vrstvou, 64Mbit SDRAM paměti a 128Mbit Flash paměti.

Kit STM32F7-Discovery bohužel nemá stejný typ rozšiřujících GPIO headerů jako kit F429-Discovery, namísto toho má Arduino konektor a navíc FFC plochý konektor pro kamerový interface. Z tohoto důvodu bylo potřeba navrhnout a vytvořit adaptér, umožňující připojit ke kitu existující kamerový modul.

11.1. Adaptér pro kit STM32F7-Discovery

Pro propojení kitu STM32F7-Discovery a kamerového modulu byl vytvořen adaptér, který využívá jak FFC konektoru pro připojení kamerového modulu, tak všech čtyř jednořadých Arduino konektorů pro vyvedení dodatečných signálů. Schémata jsou k dispozici v příloze 2.

Součástí FFC konektoru je 8bit paralelní sběrnice (DCMI D0-D7), horizontální a vertikální synchronizace (HSYNC, VSYNC) a Pixel clock (PIXCLK), tedy hodinový signál pro samotné posílání dat. Dále je součástí tohoto konektoru signál CMOS_STANDBY a také hodinový signál o frekvenci 24MHz.

Na rozdíl od kitu STM32F429, kde se hodinový signál generuje hardwarovým timerem přímo v mikrokontroléru, je na kitu STM32F7-Discovery využit 24MHz krystalový oscilátor. V rámci zachování konfigurovatelnosti rychlosti CMOS snímače tak byl z Arduino konektoru vyveden navíc ještě výstup jednoho z hardwarových timerů. Uživatel tak může pomocí přepínače P8 volit mezi využitím externího 24MHz oscilátoru nebo generování vlastního hodinového signálu procesorem.

Z Arduino konektorů je dále vyveden jeden USART (který lze využít opět ke komunikaci s deskou EVAL6470H, která ovládá krokový motor), sběrnice SPI a sběrnice I^2C a dále dva vstupní porty AD převodníku.

Na adaptéru je dále osazen další LDO regulátor 3.3V/500mA. Na kitu STM32F7-Discovery je osazen stejný regulátor, nicméně měřením bylo ověřeno, že při využití všech komponent vývojového kitu (především dotykového LCD displeje) je trvalý odběr samotného kitu 0,4A, nárazově až 0,45A. Černobílý senzor MT9V034 odebírá až 100mA, barevný MT9M001 (o jehož budoucím využití v laboratoři videometrie, vzhledem k hardwarové vybavenosti kitu F7-Discovery, nemůže být pochyb) pak odebírá až 250mA. Regulátor na kitu F7-Discovery by tedy nebyl schopen trvale napájet celý kit i s připojeným senzorem.

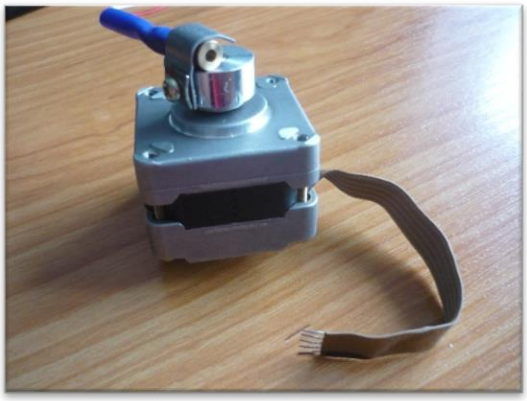
To je tedy důvod přítomnosti dalšího LDO regulátoru na propojovací desce, který má za úkol napájet senzorový modul a případně další zařízení, která jsou připojena na rozšiřující konektor. Z napájecího konektoru na rozšiřující desce vede externí napájení jednak do LDO regulátoru na této desce, ale také do LDO regulátoru na Discovery kitu. Volba zdroje napájení F7-Discovery kitu je prováděna polem jumperů přímo na kitu (napájení z externího zdroje, napájení z ST-Linku či napájení z datových USB), při špatném nastavení tohoto jumperu kit nemusí být napájen i přesto, že je do něj napájení přivedeno. Na to je tedy třeba si dát pozor.

12. Elektromechanická regulace polohy laboratorního systému

Hlavním úkolem této práce je regulační systém využívající výstupu algoritmů zpracování obrazu a lokalizace objektů k řízení polohovacího systému. Tento polohovací systém byl sestaven z kovové konstrukce (původem z tiskárny) s posuvným jezdcem, který je poháněn

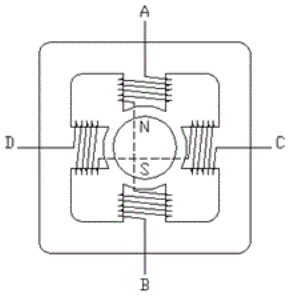
bipolárním krokovým motorem. Ke konstrukci byl rovněž vyroben úchyt pro snímací systém (kit Discovery s CMOS senzorem a kit EVAL6470H pro řízení krokového motoru).

V pojezdu použitý krokový motor je stejnosměrný, bipolární. K jeho řízení je využit inteligentní obvod EVAL6470H-DISC R1, kterému je pomocí sériové linky USART/RS232 předávána informace o tom, kterým směrem a o kolik kroků má otočit motorem (v alternativním režimu pak nikoliv o kolik kroků, ale pouze kterým směrem a jakou rychlostí, volitelně také jak dlouho). Tento režim tedy simuluje řízení stejnosměrného motoru. Informaci o tom, o kolik kroků (případně jak rychle a jakým směrem) se má systém posunout (posouvat), počítá nadřazený mikrokontrolér na desce F429-Discovery. Samotná regulace je ale přenechána podřízenému kitu, čímž je šetřen procesorový čas hlavního mikrokontroléru.



Obr. 49 - Bipolární krokový motor

Krokové motory obecně sestávají ze statoru a rotoru. Rotor je zpravidla prostřední částí (nejen) krokového motoru, a je vyroben z feromagnetického materiálu. Je zpravidla válcového tvaru s výstupky (zuby) po jeho obvodu:

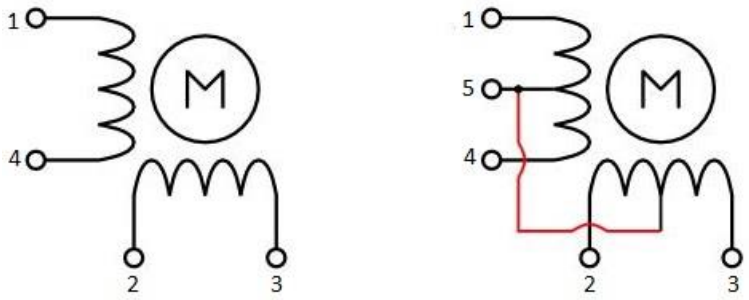


Obr. 50 - Vinutí bipolárního krokového motoru

Rotor je pak obklopen statorem, který má po svém obvodu pravidelně rozmístěné cívky. Pohyb rotoru je pak vyvolán aktivací cívek statoru v daném pořadí přivedením napětí na tyto cívky. To vede ke vzniku magnetického pole kolem těchto cívek. Jednotlivé zuby ozubeného jádra rotoru, vyrobeného z feromagnetického materiálu, jsou pak tímto magnetickým polem přitahovány. Střídáním aktivních cívek v určitém pořadí pak dochází k otáčení rotoru krokového motoru.

Jednou ze základních variant krokových motorů je dvoufázový krokový motor v unipolární a bipolární variantě. Stator krokového motoru může mít (a zpravidla má) větší množství cívek, které jsou ale zpravidla sloučeny do dvou skupin – fází. Všechny cívky v jedné fázi jsou tak aktivovány v ten samý okamžik. Existují i motory vícefázové, těmi se tato práce zabývat nebude, ale princip jejich fungování je stejný.

Jednotlivá vinutí unipolárního motoru mají středové kontakty, které jsou zpravidla vnitřně propojeny a slouží jako společná zem. Každé vinutí má pak i dva krajní kontakty. Běžný unipolární motor má tedy 5 kontaktů. Přivedením napětí na krajní kontakt vinutí tak vzniká magnetické pole okolo dané poloviny vinutí. Přivedením napětí na druhý krajní kontakt vzniká magnetické pole opačné orientace okolo druhé poloviny vinutí. Výhoda tohoto zapojení je zřejmá – je možné měnit orientaci magnetického pole beze změny směru proudu, což snižuje nároky na řídicí obvod (který může v tomto případě být skutečně jednoduchý – v podstatě stačí jeden tranzistor). Největší nevýhoda unipolárního řízení motorů je, že v každý jeden okamžik je napájena jen polovina všech vinutí – čímž se snižuje celkový točivý moment krokového motoru.



Obr. 51 - Schéma bipolárního (vlevo) a unipolárního (vpravo) krokového motoru
Zdroj: http://www.jangeox.be/2013/10/change-unipolar-28byj-48-to-bipolar.html

Vinutí bipolárního motoru na rozdíl od unipolárního motoru nemají středový kontakt. Běžný bipolární motor má tedy 4 kontakty. Pro změnu orientace magnetického pole kolem jednotlivých vinutí je tedy třeba změnit směr proudu protékajícího jednotlivými vinutími. To komplikuje návrh řídicího obvodu, který musí být schopen obracet směr toku elektrického proudu (například využitím H-můstku). Existuje však mnoho řídicích čipů, které tento problém řeší – ať už jde o skutečně triviální jednoúčelový čip, nebo o inteligentní čip (kterým je například v této práci použitý obvod L6470), který umí i další funkce navíc (detekci přeproudování, nastavení držícího proudu, generování microsteppingu a další). Výhodou bipolárního řízení krokových motorů je lepší využití dostupných vinutí a tím pádem vyšší výkon při stejných rozměrech.

Některé bipolární krokové motory mohou mít vyvedeny středové kontakty jednotlivých cívek podobně jako unipolární motory, tyto středy ale nejsou vnitřně propojeny (takový motor má tím pádem 6 kontaktů). Takový motor je pak možné řídit bipolárně (zapojením pouze krajních vinutí) nebo unipolárně (zapojením krajních vinutí a připojením středových vinutí na společnou zem.

Některé motory mají i jiný počet vývodů (7, 8, nebo i více), u takových motorů se pak ale již zpravidla nedá jednoduše určit, jakým způsobem mají být řízeny a je proto vhodně vyhledat manuál výrobce, ve kterém by toto mělo být uvedeno (takový motor může mít například i vestavěnou řídicí logiku pro kterou žádná z výše uvedených zapojení neplatí).

Mezi další výhody krokových motorů 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.

Ať je typ krokového motoru jakýkoliv, vždy je třeba mít nějaký řídicí obvod, který bude generovat sekvenci spínání jednotlivých cívek, která povede k otáčení krokového motoru. Pro všechny krokové motory také platí, že mají konstrukčně daný počet kroků na jednu otáčku (u malých krokových motorů nejčastěji 200, může být ale i více, či méně. Výhoda je jasná – není třeba zpětnovazební řízení, řídicí jednotka vždy ví, v jaké pozici se krokový motor (a na něj navázané posuvné zařízení) nachází. Jedinou výjimkou je, pokud se motor tzv. „utrhne“, tzn. že dojde k překročení mezního zatížení a motor se při kroku neotočí. V tu chvíli ztrácí řídicí jednotka informaci o tom, v jaké pozici motor je.

12.1. Řízení krokového motoru pomocí EVAL6470H

Rotor použitého bipolárního krokového 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

$$\varphi = \frac{360}{200} = 1.8°.$$

Vnitřní odpory vinutí jsou 10Ω, motor je provozován na napětí 12V.

Úhlové rozlišení 1,8° by pro zamýšlenou aplikaci mělo být dostatečné, použitý řídicí obvod L6470 ale umí i režim 1/128 microsteppingu, čímž teoreticky poskytuje úhlové rozlišení až ~0,014°, které je více než dostatečné. Použitím microsteppingu se nicméně snižuje točivý moment motoru, to ale v této aplikaci příliš nevadí.

Pro řízení krokového motoru je využit kit ST EVAL6470H, jehož součástí je inteligentní řídicí obvod L6470 a řídicí mikrokontrolér s jádrem ARM Cortex-M3. K ovládání obvodu L6470 je využita knihovna „dspin.c“, která se stará o inicializaci odpovídajících komponent (pinů, přerušení, …) řídicího procesoru a samotného obvodu L6470. Uživatel musí v programu pouze nastavit limitní hodnoty různých regulačních parametrů, jako je například regulaci napájení (maximální střída při otáčení a při stojícím motoru – holding current, maximální rychlost otáčení, maximální hodnoty akcelerace a decelerace, proudovou ochranu a další).

Poté je provedena standardní inicializace voláním funkcí dSPIN_Peripherals_Init(), Dspin_Reset_And_Standby() a pak nastavení výše zmíněných parametrů pomocí funkce

dSPIN_Registers_Set(struct), které se, podobně jako ve standardních knihovnách ST, předává seznam parametrů ve formě struktury typu „dSPIN RegistersStructure“.

Po inicializaci je provedena kalibrace polohy posuvného systému – k tomu je využit mechanický koncový spínač na posuvném systému. Jezdec se pomalou rychlostí posouvá jedním směrem, dokud nedojde k přerušení vyvolanému sepnutím koncového spínače. Tato poloha je označena jako nulová a jezdec je posunut do poloviny rozsahu, který činí zhruba 570 celých kroků (73000 kroků při 1/128 microsteppingu). V rámci kalibrační rutiny je naprogramována výjimka – a to pro případ, kdy je koncový spínač při zahájení kalibrace již sepnutý. Při snímání koncového spínače je také třeba vyřešit ochranu proti zákmitům koncového spínače, které jsou u použitého pojezdu poměrně časté.

Podřazený mikrokontrolér na desce EVAL6470H dále provádí kontrolu, zda není požadovaná cílová pozice mimo rozsah (v případě krokového režimu). Vzhledem k tomu, že je režim stejnosměrného motoru pouze emulován, je interní počítadlo absolutní polohy (které by u stejnosměrného motoru existovat nemohlo, pokud by motor neměl nějakou jinou možnost hlídání polohy, tedy například inkrementální snímač) používáno i v režimu stejnosměrného pohybu, aby zamezilo pohybu mimo rozsah pojezdu. V případě použití skutečného stejnosměrného motoru by stačilo snímat proud tekoucí motorem (který je při zastavení motoru – dojetí do koncové polohy – vyšší, než při pohybu) nebo použít již zmíněný inkrementální senzor.

Součástí firmwaru je rovněž konfigurace a obsluha rozhraní UART, pomocí kterého kit EVAL6470H dostává instrukce k otáčení motoru od kitu F429-Discovery – pro naše účely jde o postačující řešení, hlavně je to ale jediné komunikační rozhraní, které je na desce EVAL6470H vyvedeno. Spolehlivost tohoto komunikačního kanálu je rovněž dostatečná. Komunikační protokol je jednoduchý: paket má 6B a vypadá následovně:

Paket DATA – velikost 5B:

BYTE	VÝZNAM	PŘÍPUSTNÉ HODNOTY
0	Směr otáčení	1/2/3 (doprava, doleva, stůj)
1	Režim (krokový, stejnosměrný)	„s“ (krokový), „a“ (stejnosměrný)
2-5	Posun v mikrokrocích (krokový režim) Doba pohybu v ms (stejnosměrný režim)	0 – (2^{32}-1)

Tab. 11 - Byty paketu DATA

EVAL6470H pak odpovídá jednobytovými zprávami: „K“(0x4B) jako potvrzení příjmu instrukce a „F“(0x46) jako oznámení o dokončení instrukce. Dále je možné vyslat příkaz „S“(0x53) pro RESET, zedy opětovnou kalibraci posuvného systému.

12.2. Řízení na úrovni hardwaru

O generování řídicích signálů přímo pro krokový motor se stará čip L6470, není třeba je tedy implementovat. Přesto je vhodné stručně shrnout základní způsoby řízení krokových motorů.

12.2.1. Plný krok a poloviční krok

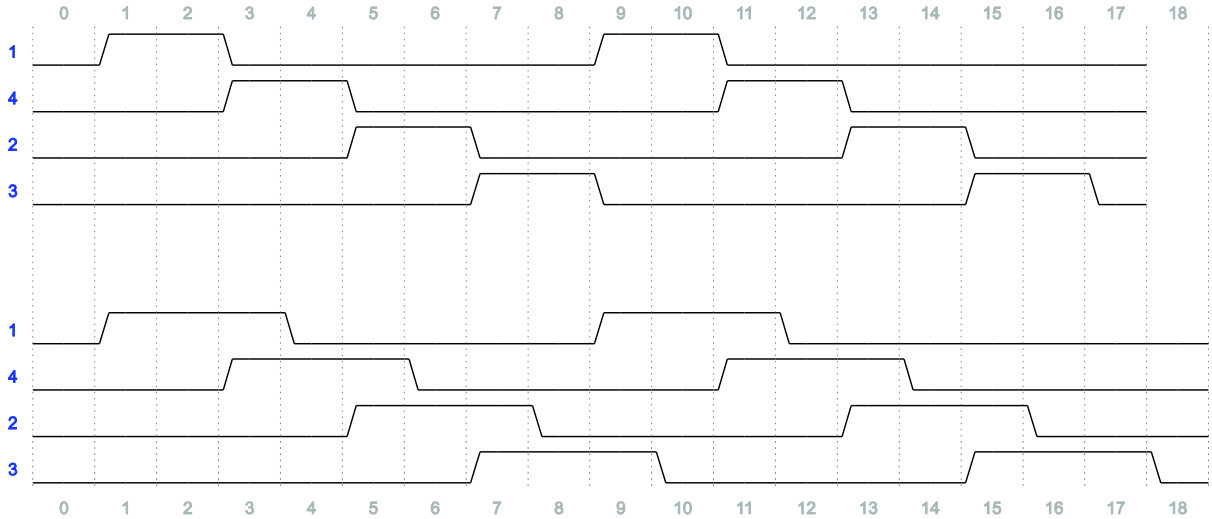
Základními způsoby řízení unipolárního krokového motoru jsou řízení plnými kroky a polovičními kroky. Jak již bylo řečeno, bipolární krokový motor má čtyři vstupy. Motorem lze tedy otáčet vhodnou sekvencí spínání těchto vstupů. V režimech řízení plným a polovičním krokem jde tedy pouze o to, posílat na tyto vstupy správnou sekvenci řídicích půlslabik (1 půlslabika = 4 bity) následujícími způsoby:

Krok:	1.	2.	3.	4.	5.	6.	7.	8.
Vinutí 1-5	1	0	0	1	1	0	0	1
Vinutí 5-4	1	1	0	0	1	1	0	0
Vinutí 2-5	0	1	1	0	0	1	1	0
Vinutí 5-3	0	0	1	1	0	0	1	1

Tab. 12 - Řízení unipolárního krokového motoru po celých krocích

Krok:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.
Vinutí 1-5	1	1	0	0	0	0	0	1	1	1	0	0	0	0	0	1
Vinutí 5-4	0	1	1	1	0	0	0	0	0	1	1	1	0	0	0	0
Vinutí 2-5	0	0	0	1	1	1	0	0	0	0	0	1	1	1	0	0
Vinutí 5-3	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1

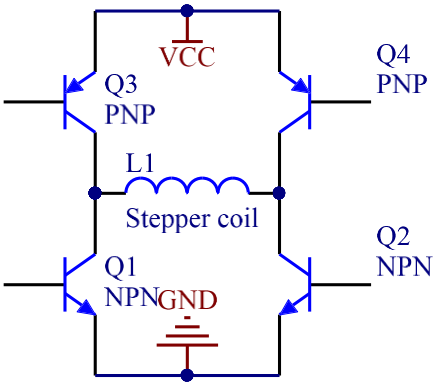
Tab. 13 - Řízení unipolárního krokového motoru po půlkrocích



Obr. 52 - Řídicí signály unipolárního krokového motoru, nahoře plný krok, dole půlkrok

Tyto signály lze jednoduše generovat na čtyřech pinech řídicího obvodu, nebo (v případě nedostatku pinů procesoru) například v kombinaci s obvodem 74595, což je 8bitový posuvný registr s paralelním výstupem ve formě záchytného registru. To ale platí pouze pro unipolární krokové motory.

Řízení bipolárních krokových motorů je mírně komplikovanější díky tomu, že je nutné měnit směr proudu v cívkách 1-4 a 2-3 (dle tab. 14 a tab. 15). Pravděpodobně nejjednodušším řešením je využití dvojitého H-můstku (pro každou fázi jeden), což je jednoduchý obvod sestávající z osmi spínačů (tranzistorů - 4x NPN, 4x PNP), který je možné vytvořit jak pomocí jednotlivých součástek, tak zakoupením hotového integrovaného obvodu, který má ve své nabídce mnoho výrobců za ceny nižší, než by byla cena osmi samotných tranzistorů.



Obr. 53 - H-můstek

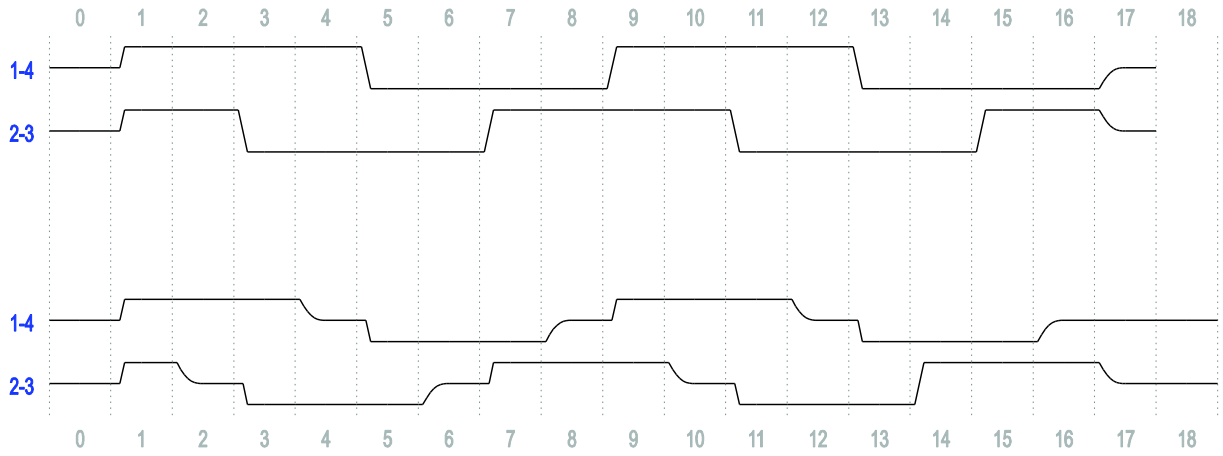
Řídicí obvod musí zajistit, aby nikdy nedošlo k otevření dvou tranzistorů „nad sebou“, neboť by došlo ke zkratu. K tomu může dojít nejen nesprávným otevíráním tranzistorů (vadná ovládací logika) ale i časovými prodlevami při otevírání a zavírání tranzistorů. Řídicí logika pak vypadá následovně (plné a poloviční kroky, 1 a -1 vyznačují směr toku el. proudu):

Krok:	1.	2.	3.	4.	5.	6.	7.	8.
Vinutí 1-4	1	1	-1	-1	1	1	-1	-1
Vinutí 2-3	1	-1	-1	1	1	-1	-1	1

Tab. 14 - Řízení bipolárního krokového motoru po celých krocích

Krok:	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.
Vinutí 1-4	1	1	1	0	-1	-1	-1	0	1	1	1	0	-1	-1	-1	0
Vinutí 2-3	1	0	-1	-1	-1	0	1	1	1	0	-1	-1	-1	0	1	1

Tab. 15 - Řízení bipolárního krokového motoru po půlkrocích

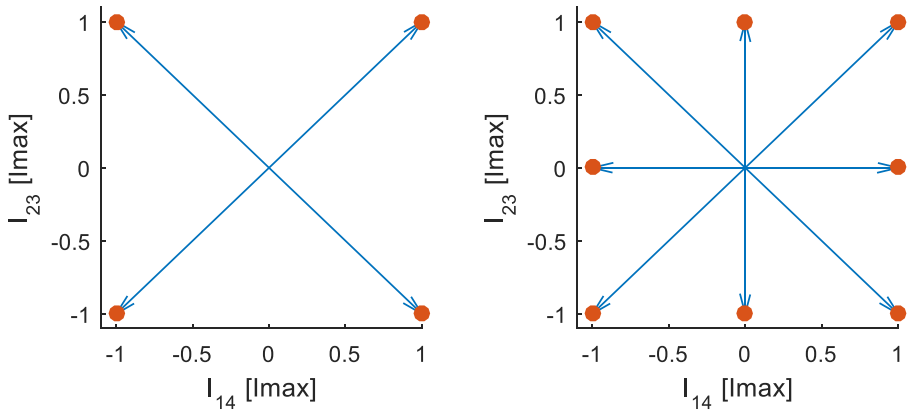


Obr. 54 - Řídicí signály bipolárního krokového motoru, nahoře plný krok, dole půlkrok

12.3. Sinový-kosinový Mikrokrok (Sine-Cosine Microstepping)

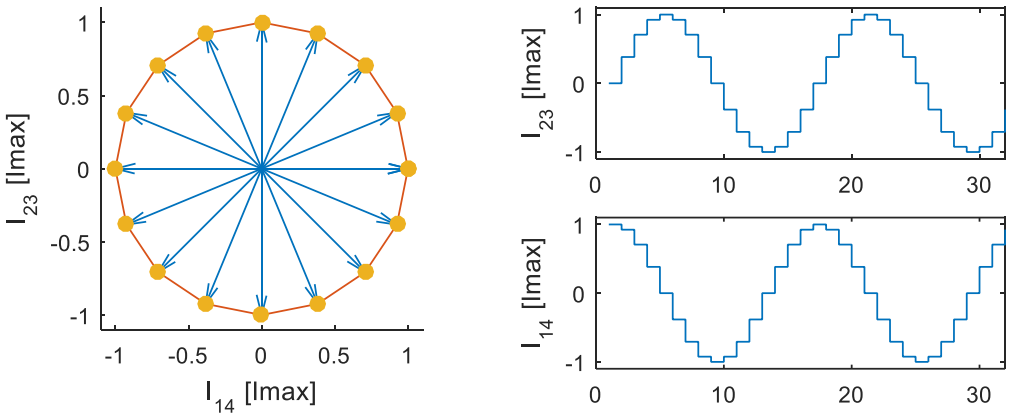
Při řízení plným krokem byl do vinutí bipolárního motoru vždy pouštěn proud o velikostech I_{max} a $-I_{max}$. Řízení polovičním krokem bylo prováděno pomocí proudů o velikostech I_{max}, 0 a I_{min}. Rozšířením o další proudové úrovně a vytvořením dalších kroků řídicí posloupnosti pak lze docílit řízení tzv. „micro-steppingem“. Počet takových kroků je teoreticky neomezený, prakticky je pak omezen ze dvou stran – jednak počtem úrovní, které dokáže řídicí obvod vygenerovat (použitý L6470 jich dokáže vygenerovat 128 na jeden krok) a pak také tím, že se zvyšujícím se počtem mikrokroků výrazně klesá točivý moment (až do stavu, kdy je příliš malý i na roztočení samotného rotoru a rotor se netočí vůbec nebo vynechává). S rostoucím počtem mikrokroků také klesá „opakovatelnost krokování“, tedy spolehlivost určení aktuální pozice motoru počítáním vykonaných (mikro)kroků.

Pro lepší vizualizaci problému je možné si vykreslit 2D graf – fázový diagram - s jednou osou představující proud jedním vinutím a druhou osou představující proud druhým vinutím:



Obr. 55 - Vizualizace prudů tekoucích jednotlivými vinutími v režimech plného kroku (vlevo), polovičního kroku (vpravo)

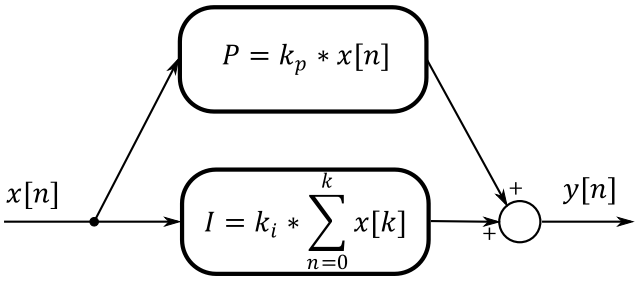
Microstepping je pak jen rozšířením o další body – v reálném světě ale existují především dvě varianty microsteppingu – první vytvoří v grafu čtverec, podobně jako v grafech výše, druhá vytvoří kružnici (v žádném bodě řízení tedy nedochází k tomu, že by oběma vinutími procházel maximální proud). V reálných aplikacích se více aplikuje právě druhá, „kruhová“ varianta, neboli sinově-kosinový microstepping, a to i přesto, že takto řízený motor dosahuje nižšího špičkového točivého momentu. Jak je vidět z grafu níže, proud vinutími přibližně aproximuje sinusový střídavý AC průběh. Vizualizace sinově-kosinového microsteppingu se čtyřmi mikrokroky je na obr. 56:



Obr. 56 - Vizualizace proudů tekoucích jednotlivými vinutími v režimu microsteppingu

12.4. PID (PSD) regulátor pro řízení polohy polohovacího systému

Pro řízení polohy polohovacího systému byl zvolen PID regulátor, respektive jeho diskrétní, proporcionálně-sumačně-diferenční (PSD) varianta. Oproti použití čistě proporcionálního regulátoru PSD regulátor eliminuje trvalou regulační odchylku a zlepšuje stabilitu regulátoru. Regulátor PSD vznikne paralelním spojením proporcionálního, sumačního a diferenčního regulátoru. Základní schéma PSD regulátoru je znázorněno na obr. 57, přičemž $x[n]$ regulační odchylka (vzdálenost bodů), a $y[n]$ je žádaný počet kroků.



Obr. 57 - Základní schéma PS regulátoru

Pro realizaci regulátoru jsou v programu implementovány vedle funkce move(millimeters), která posune polohovací systém o daný počet miliemtrů, dále funkce findDifference(), která

spočte vzdálenost bodů, a funkce adjust(), která provede samotnou regulaci a volá funkci move(). Funkce adjust() vypadá následovně:

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

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.

13. Dosažené výsledky

V rámci této práce jsem navrhl demonstrační polohovací systém, jehož základem se stal posuvný mechanismus tiskové hlavy z vyřazené jehličkové tiskárny. Ten je osazen výkonným bipolárním krokovým motorem a také koncovým spínačem. Tento mechanismus jsem očistil o všechny součásti, které bezprostředně nesouvisí s jeho novou funkcí (tedy především o tiskové válce a mechanismus jejich posuvu). Vytvořil jsem jednoduchou mechanickou konstrukci, kterou jsem připevnil ke zmíněnému posuvu, a na kterou jsem připevnil kamerový a regulační systém.

Kamerový a regulační systém je sestaven z kitu STM32F429-Discovery, z monochromatického CMOS senzoru Aptina MT9V032 a kitu EVAL6470H, určeného k řízení bipolárních krokových motorů. K použití dedikovaného kitu pro řízení krokových motorů jsem se rozhodl proto, že je jeho součástí inteligentní obvod L6470, který se automaticky stará o generování řídicích sekvencí pro bipolární krokové motory včetně otáčení směru toku proudu jednotlivými vinutími (součástí obvodu je duální DMOS můstek). Další výhodou tohoto řešení je uvolnění procesorového času kitu F429 pro operace zpracování obrazu.

Pro komunikaci mezi těmito nezávislými jednotkami jsem zvolil potvrzovanou komunikaci po rozhraní UART (jde o jediné komunikační rozhraní dostupné na kitu EVAL6470H kromě USB). Součástí vytvořeného firmwaru pro kit EVAL6470H je rovněž počáteční kalibrace s využitím koncového spínače a také kontrola vstupních dat a aktuální polohy pojezdu tak, aby nedocházelo k regulaci mimo rozsah použitého pojezdu.

Jednou ze základních vlastností navrhované demonstrační aplikace je robustnost. Ta je při použití optického senzoru dána především odolností vůči nepříznivým světelným podmínkám. Základním parametrem při snímání obrazu je doba expozice snímku. Množství aktuálně dostupných senzorů disponuje funkcí automatického nastavení doby expozice, přesto jsem ale implementoval vlastní algoritmy pro nastavení doby expozice. K tomuto kroku existují dva důvody. Prvním důvodem je možná situace, kdy bude potřeba použít senzor, který

automatickým nastavením doby expozice nedisponuje. Druhým důvodem jsou specifické požadavky na scénu snímanou pro účely zpracování obrazu – pro detekci jasného předmětu, kterým může být například laserové ukazovátko, je žádoucí mít podexponovaný snímek, kde jediným jasnějším objektem je právě sledovaná laserová stopa. Toho lze použitím automatického nastavení doby expozice v senzoru docílit jen obtížně nebo vůbec.

V rámci této práce jsem navrhl zpracování dvou druhů značek – jedné značky a soustavy značek. První varianta, tedy regulace polohy ve spojení s detekcí jedné značky, spočívá ve vyhodnocení polohy detekované značky (která je připevněna k pojezdu) v zorném poli kamery a poté posuvu pojezdu tak, aby tato značka byla na předem určené pozici v zorném poli (tedy například uprostřed). Cílová pozice je zadávána uživatelem.

Pro druhou variantu jsem vytvořil vlastní soustavu značek, která je z funkčního hlediska schopná nahradit klasické pravítko s číselnými značkami. K této variantě jsem přistoupil po zvážení náročnosti určení polohy na klasickém pravítku (včetně rozpoznávání čísel). Vytvořené měřítko v podstatě kopíruje tvar PWM signálu s lineárně se zvyšující střídou, kde aktuální střída v právě detekovaném segmentu udává aktuální polohu (tedy v polovině měřítka je 50% střída).

Ke zpracování obrazu jsem využil známých a vyzkoušených algoritmů zpracování obrazu, mezi které patří rozostření Gaussovým filtrem, prahování, labelling a detekce hran konvolucí obrazu se Sobelovým operátorem. Tyto algoritmy jsem optimalizoval vyřazením zbytečných větví jednotlivých algoritmů a také omezením rozsahu zpracovávaného obrazu. Ve většině případů je totiž zcela zbytečné zpracovávat celý obraz, ale stačí zpracovat jen jeho malou část. Výsledná rychlost algoritmu v případě detekce jedné značky je 11 snímků za sekundu, v případě vlastního měřítka pak přibližně 5,5 snímku za sekundu.

Vytvořenou demonstrační aplikaci je možné ovládat pomocí grafického rozhraní přímo na dotykovém displeji, který je součástí kitu F429-Discovery a také pomocí počítačové aplikace psané v jazyce C# a komunikující po USB pomocí emulovaného sériového rozhraní UART (Virtual COM Port). V rámci počítačové aplikace je také možné nastavit pokročilé parametry snímání a zobrazit i uložit snímek ze senzoru.

V posledním bodě této práce jsem navrhl schéma zapojení CMOS senzoru pro kit STM32F7-Discovery, který využívá nového jádra ARM Cortex-M7. Pro připojení CMOS senzoru k tomuto kitu nelze bohužel použít již vytvořeného hardwaru, a tak bylo nutné vytvořit nový. Výhodou použití tohoto kitu je oproti kitu F429-Discovery dvojnásobný výkon na jednotku frekvence a také bezproblémové připojení všech používaných periferií zároveň (není tedy nutné řešit problém, zda chceme snímat obraz, nebo zobrazovat data na vestavěném displeji – není problém mít obojí). Navrhnul jsem nové zapojení hardwaru, z časových důvodů bohužel již nebyl celý projekt převeden na tento nový hardware.

14. Závěr

V rámci této diplomové práce byla vyvinuta embedded aplikace, která na základě obrazových informací z obrazového senzoru a s využitím algoritmů zpracování obrazu sleduje optickou značku (marker) připevněnou k posuvnému systému (pojezdu) a reguluje polohu této značky posouváním pojezdu. Byly vytvořeny algoritmy pro dva druhy značek – pro jednotlivou značku, která je regulována na zadanou cílovou polohu v zorném poli kamery, a pro soustavu značek tvořící měřítko, u které je regulace provedena tak, aby ve středu zorného pole byla uživatelem zadaná poloha na měřítku.

Byly využity algoritmy zpracování obrazu, které byly vyvinuty v rámci mé bakalářské práce, a které byly doplněny o několik algoritmů předzpracování obrazu (Gaussův filtr a další) a optimalizovány pro co nejvyšší rychlost. Vzhledem k využití vestavěné SDRAM paměti o dostatečné kapacitě nebylo již nutné zpracovávat obraz průběžně, ale bylo možno zpracovávat jej offline a využít tedy i výpočetně náročnější operace.

Ovládání vytvořené aplikace je možné buď skrze vytvořenou PC aplikaci nebo pomocí embedded GUI na vestavěném dotykovém displeji, který je součástí použitého hardwaru. V rámci práce pro laboratoř videometrie byla vytvořena univerzální demo aplikace pro potřeby laboratoře videometrie, jejímž vypracováním jsem se sice mírně odchýlil od cíle této diplomové práce, ale v rámci zpracování této aplikace jsem zprovoznil LCD displej a dotykovou vrstvu a napsal jsem jednoduchou grafickou aplikaci, na jejíchž základech jsem vystavěl i embedded GUI pro finální aplikaci, která je výsledkem této diplomové práce.

V rámci diplomové práce byl rovněž vytvořen algoritmus pro robustnější nastavení doby expozice, které je možno nastavit podle konkrétních potřeb aplikace (například nižší či vyšší průměrný jas obrazu) – to by v případě využití automatické expozice vestavěné přímo v senzoru nebylo možné.

Literatura

[1] TOMS, Daniel. Zpracování obrazu pro sledování optické stopy. Praha, 2015. Bakalářská práce. ČVUT FEL.

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

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

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

[5] DOKOUPIL, Vojtěch. STM32F429 Based CMOS Camera for Teaching Purposes. Praha, 2015. Diplomová práce. ČVUT FEL.

[6] STM32F429 Reference Manual. In: STM32F429/439 - STMicroelectronics [online]. 2016 [cit. 2016-05-22]. Dostupné z: *http://www.st.com/content/ccc/resource/technical/document/reference_manual/3d/6d/5a/66/b4/99/40/d4/DM00031020.pdf/files/DM00031020.pdf/jcr:content/translations/en.DM00031020.pdf*

[7] STM32F429 Discovery Kit User Manual. In: 32F429IDISCOVERY - STMicroelectronics [online]. 2016 [cit. 2016-05-22]. Dostupné z: *http://www.st.com/content/ccc/resource/technical/document/user_manual/6b/25/05/23/a9/45/4d/6a/DM00093903.pdf/files/DM00093903.pdf/jcr:content/translations/en.DM00093903.pdf*

[8] 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*

[9] MT9V034 Datasheet. In: MT9V034 Datasheet [online]. 2015 [cit. 2016-05-22]. Dostupné z: *http://www.onsemi.com/pub_link/Collateral/MT9V034-D.PDF*

[10] STM32F746 Reference Manual. In: STM32F746NG - STMicroelectronics [online]. 2016 [cit. 2016-05-22]. Dostupné z: h*ttp://www.st.com/content/ccc/resource/technical/document/reference_manual/c5/cf/ef/52/c0/f1/4b/fa/DM00124865.pdf/files/DM00124865.pdf/jcr:content/translations/en.DM00124865.pdf*

[11] STM32F746 Discovery Kit User Manual. In: 32F746GDISCOVERY - STMicroelectronics [online]. 2016 [cit. 2016-05-22]. Dostupné z: *http://www.st.com/content/ccc/resource/technical/document/user_manual/f0/14/c1/b9/95/6d/40/4d/DM00190424.pdf/files/DM00190424.pdf/jcr:content/translations/en.DM00190424.pdf*

[12] BATTIATO, S., MESSINA, G., CASTORINA, A.. Exposure Correction for Imaging Devices [online]. 2008 [cit. 2016-01-16]. Dostupné z: *http://www.dmi.unict.it/~battiato/download/Exposure2008.pdf*

[13] Modulační přenosová funkce digitálního fotoaparátu. In: Modulační přenosová funkce digitálního fotoaparátu [online]. 2007 [cit. 2016-05-22]. Dostupné z: *http://radio.feld.cvut.cz/courses/D37LBR/materialy.php?akce=dlf&zdroj=vpm&fkey=7&xtgt=2f686f6d652f53657276696365732f7777772f68746d6c2f6564755f6465706f742f2f583334464f54*

Příloha 1 – Dodatečná aplikace – F429 Universal demo

Dříve, než bylo možno přistoupit k realizaci samotného projektu, vyvstala potřeba vytvoření univerzálního dema pro kit F429-Discovery. Z bakalářských a diplomových prací na katedře měření (a především v laboratoři videometrie) vzniklo v průběhu času více různých aplikací, mezi něž patří například aplikace vyhodnocující hodnoty hozených kostek, jednoduchá kamera, aplikace sledující laserový paprsek, …

Pro potřeby prezentace laboratoře na veletrzích či jen před jednotlivými studenty by tedy bylo vhodné mít jednu aplikaci (jeden firmware), který by obsahoval všechny tyto aplikace a aby mezi nimi bylo možno volit, nejlépe přímo za běhu programu. Tato aplikace by také měla být jednoduše rozšiřitelná o případně další „dema“, která by byla vytvořena později.

Pro realizaci této aplikace byl zvolen rovněž kit F429-Discovery, který přes svou příznivou cenu (cca 580kč, farnell.com) obsahuje mnoho (2MB) vnitřní pamětí, do které se vejde více aplikací najednou, a navíc LCD displej s dotykovou vrstvou, na kterém by bylo možné volit mezi jednotlivými aplikacemi a nastavovat jejich parametry.

Realizace programu proběhla v jazyce C, ve vývojovém prostředí Keil uVision 4. Program se v tuto chvíli sestává ze dvou modulů, a to z „Dices“ dema (které počítá součet teček ze dvou kostek, a je popsáno v práci [4]) a z jednoduchého kamerového programu, který sejme snímek při stisku tlačítka a zobrazí jej na LCD displej (tato aplikace vznikla jako vedlejší produkt práce [1]). Uživatel, který chce přidat k aplikaci další demo, se může inspirovat souborem „unimplemented_demo.c“.

Obr. 58 - Univerzální demo pro kit STM32F429-Discovery

Při řešení tohoto projektu bylo potřeba vyřešit problém kolize pinů, sdílených napříč periferiemi. V rámci mé bakalářské práce [1] jsem již řešil problém kolize pinů LCD zobrazovače a CMOS sensoru, který nebylo možné vyřešit jinak, než střídavým zapínáním a vypínáním těchto periferií. Proto nebylo a není možné zároveň snímat obraz a zobrazovat data na LCD displeji. Z praktického hlediska to však při předpokládaném využití nevadí, displej jednoduše po tu kratičkou dobu snímání nezobrazuje žádná data.

Druhým problémem byla kolize CMOS sensoru a I2C sběrnice, ke které je připojena dotyková vrstva LCD displeje. Řešení problému je podobné, není možné využívat obě periferie najednou, navíc je třeba vypnout signál CMOS_ENABLE, jinak bude kolizní pin vždy „přetlačen“ LCD modulem a komunikace přes I2C neproběhne.

Pro přidání dalšího dema je třeba:

1) Přidat jeden nebo více zdrojových souborů
2) Zdrojový soubor musí mít svoji hlavní funkci (například camera_demo_main())
3) Zdrojový soubor musí mít svoji funkci pro zobrazení menu (například camera_show_menu())
4) V souboru main.c deklarovat externí funkci z bodu 2) (tedy např. extern void camera_demo_main(void))
5) V souboru main.c ve funkci goToDemo() na novou pozici umístit volání funkce z bodu 4)
6) Zkompilovat a nahrát celou aplikaci do mikrokontroléru

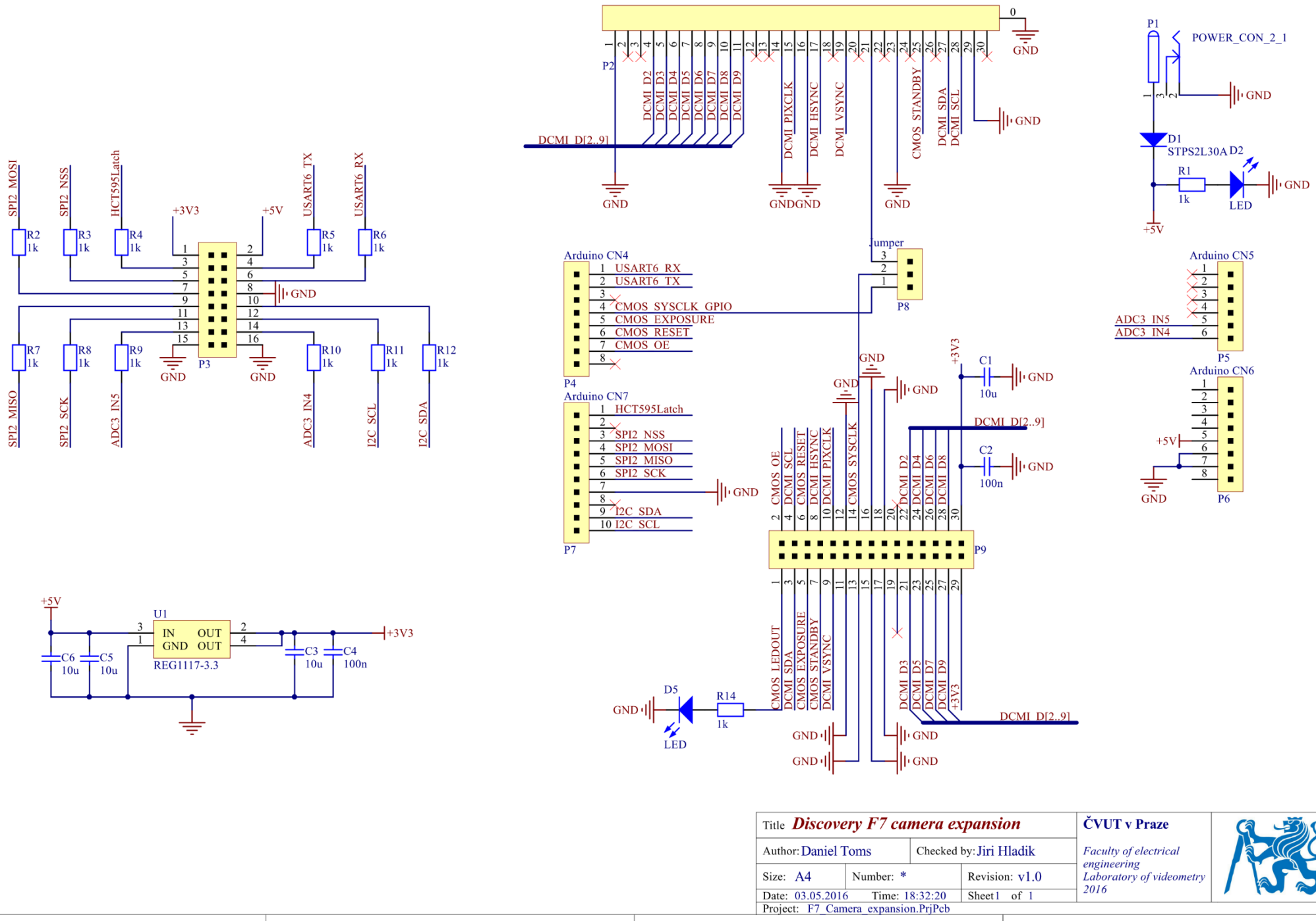
Pro kreslení GUI je možno využít několika připravených grafických prvků. Jejich kresbu lze provést voláním funkce

```
LCD_DrawBitmap(0, 256, 240, 64, (uint8_t *)&button_background[0]);
```

Mezi připravené prvky patří:

- button_background – jednoduché tlačítko pod text o rozměru 240x64 (text je třeba zapsat zvlášť standardní funkcí LCD_DisplayStringAt())
- checkbox_unchecked – nezaškrtnutý checkbox
- checkbox_checked – zaškrtnutý checkbox
- radio_unchecked – nezaškrtnutý radio button
- radio_checked – zaškrtnutý radio button

Tato připravená grafika bude využita dále i v diplomové práci, bude však vyzkoušena i varianta s využitím dostupného STemWin frameworku nebo jiných GUI knihoven. Je ale možné, že tyto knihovny budou spotřebovávat příliš mnoho procesorového výkonu, který bude potřeba pro zpracování obrazu.


SPI2_MOSI
SPI2_NSS
HCT595Latch
+3V3
+5V
USART6_TX
USART6_RX
R2 1k
R3 1k
R4 1k
R5 1k
R6 1k
GND
P3
R7 1k
R8 1k
R9 1k
R10 1k
R11 1k
R12 1k
SPI2_MISO
SPI2_SCK
ADC3_IN5
ADC3_IN4
I2C_SCL
I2C_SDA
P2
DCMI_D[2..9]
DCMI_D2
DCMI_D3
DCMI_D4
DCMI_D5
DCMI_D6
DCMI_D7
DCMI_D8
DCMI_D9
DCMI_PIXCLK
DCMI_HSYNC
DCMI_VSYNC
CMOS_STANDBY
DCMI_SDA
DCMI_SCL
GNDGND
P1
POWER_CON_2_1
D1
STPS2L30A
D2
R1 1k
LED
+5V
Arduino CN4
1 USART6_RX
2 USART6_TX
3
4 CMOS_SYSCLK_GPIO
5 CMOS_EXPOSURE
6 CMOS_RESET
7 CMOS_OE
8
P4
Jumper
P8
Arduino CN5
ADC3_IN5
ADC3_IN4
P5
Arduino CN6
P6
Arduino CN7
1 HCT595Latch
2
3 SPI2_NSS
4 SPI2_MOSI
5 SPI2_MISO
6 SPI2_SCK
7
8
9 I2C_SDA
10 I2C_SCL
P7
C1 10u
C2 100n
CMOS_OE
DCMI_SCL
CMOS_RESET
DCMI_HSYNC
DCMI_PIXCLK
CMOS_SYSCLK
DCMI_D2
DCMI_D4
DCMI_D6
DCMI_D8
P9
CMOS_LEDOUT
DCMI_SDA
CMOS_EXPOSURE
CMOS_STANDBY
DCMI_VSYNC
DCMI_D3
DCMI_D5
DCMI_D7
DCMI_D9
D5
R14 1k
LED
U1
IN
OUT
GND
REGI117-3.3
C6 10u
C5 10u
C3 10u
C4 100n
Title Discovery F7 camera expansion
Author: Daniel Toms
Checked by: Jiri Hladik
Size: A4
Number: *
Revision: v1.0
Date: 03.05.2016
Time: 18:32:20
Sheet 1 of 1
Project: F7_Camera_expansion.PrjPcb
ČVUT v Praze
Faculty of electrical engineering
Laboratory of videometry
2016

Příloha 3 – Obsah přiloženého CD

- Diplomová práce ve formátu PDF
- Aplikace a programy
    - Programy pro MCU
        - STM32F429I Hlavní program – detekce polohy
        - EVAL6470 Firmware pro řízení krokových motorů
    - PC Aplikace
        - F429 Image Processing a Regulace
- Citované zdroje diplomové práce
- Další materiály
    - Schéma propojovacího modulu pro STM32F7-Discovery