ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
Katedra měření

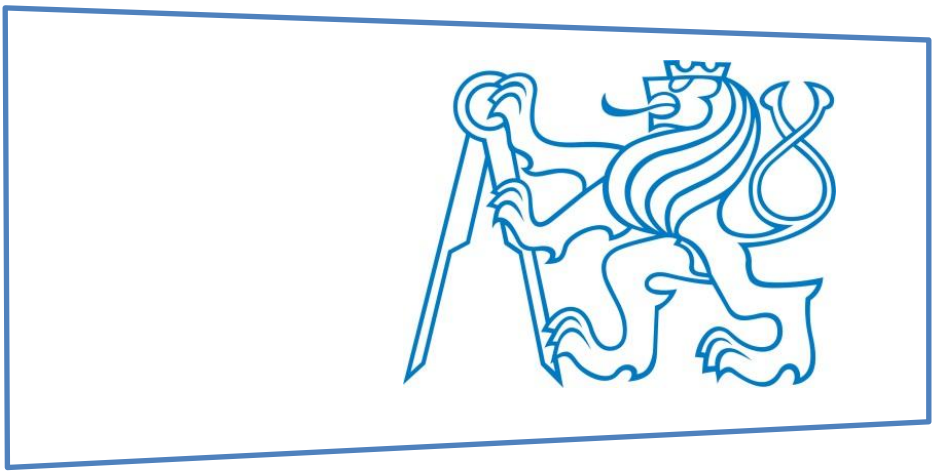
Bakalářská práce

Zpracování obrazu vestavěným mikrořadičem

Image Processing Using Embedded Microcontroller

Vojtěch Dokoupil

Kybernetika a robotika
Senzory a přístrojová technika



2013

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

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
Katedra měření

Akademický rok **2012-13**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student:	**Vojtěch Dokoupil**
Studijní program:	**Kybernetika a robotika**
Obor:	**Senzory a přístrojová technika**
Název tématu česky:	**Zpracování obrazu vestavným mikrořadičem**
Název tématu anglicky:	**Image Processing Using Embedded Microcontroller**

Pokyny pro vypracování:

Analyzujte dostupné, případně navrhněte nové metody zpracování obrazu pro vyhodnocení stavu a změny scény, které vystačí s omezenou vnitřní pamětí vestavných mikrořadičů s jádrem ARM Cortex – M4. Realizujte funkci porovnání snímků dvou scén a funkci parametrizace snímku a vyhodnocení jeho změn. Navržené metody implementujte a vytvořte potřebné programové vybavení pro vlastní vestavný mikrořadič STM32F407 i řídicí program pro nadřazené PC.

Seznam odborné literatury:

[1] ARM Limited: Cortex-M4 Revision r0p1, Technical Reference Manual, ARM DDI 0439C
[2] STMicroelectronics: Reference Manual, RM0090, Doc ID 018909 Rev3
[3] Yiu J.: The definitive Guide to the ARM Cortex- M3. Elsevier, 2007

Vedoucí bakalářské práce:	doc. Ing. Jan Fischer, CSc.
Datum zadání bakalářské práce:	7. prosince 2012
Platnost zadání do[1]:	31. ledna 2014

Prof. Ing. Vladimír Haasz, CSc.
vedoucí katedry

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta elektrotechnická

Prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 7. 12. 2012

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

Prohlášení

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

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

V Praze dne: 23.5.2013 Podpis: 

Poděkování

Zde bych rád poděkoval rodině a přátelům za podporu, které se mi v průběhu studia a vytváření této práce dostalo.

Dále bych rád poděkoval vedoucímu bakalářské práce doc. Ing. Janu Fischerovi, CSc. za cenné rady, podklady a připomínky, které mi pomohly při vypracování práce.

Anotace (CZ)

Práce se soustředí na metody zpracování obrazu vestavěným mikrořadičem s jádrem ARM Cortex - M4 a ukazuje, že se v případě vhodného použití DMA řadiče dá pracovat efektivně s obrazovými daty, i když má mikrořadič omezenou paměť. Součástí práce je ukázka několika metod zpracování obrazu. Byla vyvinuta metoda pro rozpoznání teček na vrhacích kostkách a určení jejich počtu a metody pro detekci obecné změny ve scéně a popis této změny. Součástí práce je i vytvoření několika aplikací pro uživatelský přístup k senzoru z PC pomocí USB. Také je prezentován vyvinutý hardware pro propojení existujících částí senzorového a mikroprocesorového modulu.

Annotation (EN)

This bachelor thesis is focused on embedded microcontroller image processing using ARM Cortex - M4 microcontroller. The problem of bounded memory and how can be overcome by appropriate use of DMA peripheral is shown. A part of work shows several image processing methods. There is an algorithm which is able to count sum of dots on two dices under the sensor. The other one is a change detection algorithm for scene change recognition and description. PC application for access to sensor and microcontroller through USB is presented. The developed hardware for connecting already existing parts of sensor and microcontroller modules is shown as well.

Slovník použitých pojmů

AEC	*Auto Exposure Control* – funkce automatické délky expozice CMOS senzoru
AGC	*Auto Gain Control* – funkce automatického zesílení CMOS senzoru
ALU	*Arithmetic Logic Unit* – výpočetní jednotka mikrořadiče
ARM	*Advanced RISC Machine* – typ mikrořadiče
Buffer	*např. vyrovnávací paměť, obecně blok paměti* – pole hodnot užívané například při komunikaci pomocí sériové linky
CAN	*Controller Area Network* – typ sběrnice
CMOS senzor	použitý senzor – *Aptina MT9V034*
DCMI	*Digital Camera Interface* – paralelní sběrnice pro komunikaci s obr. senzorem
Discovery kit	*STM32F4-Discovery* – vývojová deska, kterou užívám v rámci této práce
DMA	*Direct Memory Access* – řadič umožňující přesun dat (například ze senzoru do paměti) bez vytížení jádra procesoru
Freeware	volně šiřitelný a dostupný software
FRB	typ konektoru
I^2C	*Inter Integrated Circuit* – typ sběrnice
LCD	*Liquid Crystal Display* – technologie zobrazování, dnes se používá jako výraz pro samotný zobrazovač založený na této technologii
MCU	*Microcontroller Unit* – používáno jako alternativa pro výraz mikrořadič
PC	*Personal Computer* – nadřazený počítač
PCB	*Printed Circuit Board* – deska tištěného spoje
ROI	*Region of Interest* – název pro část obrazu (výřez), který je předmětem zájmu
S_{AD}	*Sum of Absolute Differences* – kritérium rozhodování v algoritmu template matching
SPI	*Serial Peripheral Interface* – typ sběrnice
Temp. Matching	metoda lokace objektů v obraze porovnáváním vzoru a obrazu
USART	*Universal Synchronous Asynchronous Receiver Transmitter* – typ sběrnice
USB	*Universal Serial Bus* – typ sběrnice

Obsah

Seznam obrázků

Seznam tabulek

Seznam rovnic

1. Úvod

S rostoucím výkonem mikrořadičů určených prvotně pro vestavěné či řídící aplikace se nabízí možnost jejich využití i pro účely zpracování obrazu. Tato myšlenka vychází z toho, že výrobci přidávají na MCU obrazová rozhraní a přitom neumožňují připojení externí paměti RAM. Z toho plyne, že se počítá i s možností zpracovávat obraz s omezenými paměťovými prostředky. Cílem práce je tyto možnosti analyzovat a otestovat.

Práce se soustředí na zpracování obrazu vestavěným mikrořadičem s jádrem ARM Cortex - M4. Jádrem práce je návrh a implementace konkrétních metod pro parametrizaci snímku a porovnání snímků dvou scén na mikrořadiči ve spojení s obrazovým senzorem.

Náplní práce bude analýza či vytváření jednoduchých metod zpracování obrazu a aplikace těchto metod na data z obrazového senzoru přímo v mikrořadiči ARM s omezenou velikostí vnitřní paměti RAM. Bude navržena a implementována metoda pro rozpoznání teček na vrhacích kostkách pod objektivem senzorového modulu a určení jejich součtu. Dále budou navrženy a naprogramovány metody pro porovnání snímků dvou scén. To bude probíhat několika způsoby, mezi jinými i pomocí přesunu obrazových dat do dvou paměťových lokací užitím DMA řadiče (double buffer mód). Tato metoda se dá označit jako průběžné zpracování obrazu.

Dále se předpokládá vytvoření uživatelské aplikace na PC pro jednoduchou obsluhu senzoru, která bude umožňovat přístup k registrům senzoru a případně k obrazovým datům, či jejich upravené (prahované, invertované nebo jiné) podobě.

Práce si dále klade za cíl navrhnout a nechat vyrobit desku plošného spoje pro propojení existujícího modulu obrazového senzoru a STM32F4 - Discovery kitu (obsahujícího MCU). Neplyne to přímo ze zadání, ale zvýší to spolehlivost vyvíjeného systému pro zpracování obrazu. Deska má umožňovat připojení LCD zobrazovače pro zobrazování výpočetních a naměřených dat. To se bude týkat hlavně finálních aplikací zpracování obrazu bez použití PC. Tato deska bude navržena pro mechanické uchycení na prototypový držák, který bude také v případě dostatku času navržen a vyroben.

Výsledkem práce by měl být přehled aplikovaných metod, více či méně úspěšných. Z toho by mělo vyplynout, zda je zpracování obrazu vestavěným mikrořadičem perspektivní a použitelné.

Navržený hardware by měl usnadnit práci budoucím zájemcům díky jednoduchému propojení se sériově vyráběným STM32F4 - Discovery kitem a již vyvinutým modulem obrazového senzoru. Předpokládá se i použitelnost ostatních výsledků práce pro další studenty zabývající se v budoucnu tímto problémem.

2. Současný stav problematiky

Mikrořadiče nabývají v dnešním světě na důležitosti a to v oblasti komunikace, řízení a různých vestavěných aplikací. Množství technických prostředků je často součástí čipu a ten tak umožňuje komunikovat například po velkém množství standardizovaných sběrnic, mezi jinými například USB, I^2C, SPI, USART, CAN, DCMI. Ačkoliv jsou na mikrořadičích dostupná výše zmíněná komunikační rozhraní, připojení externí paměti RAM pro bezproblémové zpracování obrazu není standardní. Z toho plyne, že se počítá s možností zpracovávat obraz takovým způsobem, který nevyžaduje velký paměťový prostor a vystačí s vnitřní pamětí RAM. V následujících odstavcích je prezentován dostupný hardware pro zpracování obrazu, a to jak samotné senzory, tak sestavy s mikroprocesory.

2.1. Existující senzory a systémy pro účely strojového zpracování obrazu

2.1.1. Dělení senzorů

Problematiku je potřeba rozdělit na dvě větve. Při procházení webových stránek výrobců obrazových senzorů a systémů jsem dospěl k závěru, že musím rozlišit existující systémy, které obsahují standardizované komunikační rozhraní (USB, Ethernet) či jistou vrstvu inteligence (hledána hesla jako: "smart cameras", "vision sensors", atp.), a existující čistý hardware - obrazové senzory, které jsou základem strojového vidění.

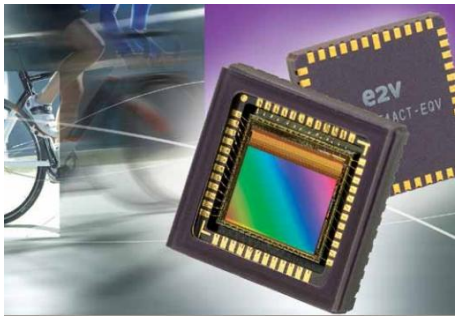
2.1.2. Obrazové maticové senzory - stručný přehled nabídky

Senzory pro aplikace strojového vidění a příjmu digitálního obrazu se dále dělí dle technologie na **CMOS** (Complementary Metal - Oxide Semiconductor) senzory a **CCD** (Charged - Coupled Device) senzory. Porovnání těchto senzorů a jejich principů není předmětem této práce, a tak jen ve zkratce uvedu několik informací. CMOS senzory jsou vývojově mladší, technologicky ne tak náročné na výrobu (stejně jako integrované obvody) a jejich cena je tedy obecně nižší. Čipy CMOS senzorů obsahují zpravidla rovnou A/D převodníky i datová či komunikační rozhraní nižší úrovně (pro komunikaci s mikrokontroléry, ne s PC). CCD senzory jsou vyzrálejší, používají se díky tomu v dražších přístrojích a jejich pořizovací cena je vyšší. Dnes jsou již obrazové možnosti obou typů senzorů téměř srovnatelné. Co se týče ceny, vede CMOS technologie. Dalším aspektem, který by mohl být zajímavý, je ten, že CMOS senzory umožňují rychlejší komunikaci a mají nižší spotřebu. CMOS senzory netrpí problémem nazývaným v angličtině smear (stěr), který je typický pro většinu

neprofesionálních CCD senzorů (příklad tohoto efektu: slunce vám zasvítí do fotoaparátu a uvidíte vertikální čáru). Vzhledem k předpokládanému zaměření práce poskytuji informaci o dvou CMOS senzorech a jednom CCD senzoru. Smyslem bylo uvést několik klíčových informací, které by umožnily porovnat jednotlivé senzory. Ne vždy je však tento přístup možný, informace poskytované zdarma jednotlivými společnostmi jsou různé a někdy i nepřístupné.

Tab. 1 - CMOS senzor EV76C454 (Jade)

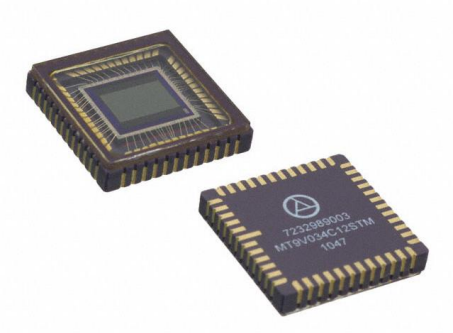
CMOS senzor EV76C454 (Jade)	
typ senzoru	barevný i monochromatický
maximální rozlišení	860 x 640 px
max. rychlost čtení při max. rozlišení	60 fps
rozlišení - datový výstup	8 bitů
velikost pixelu	5.8 µm x 5.8 µm
napájecí napětí	1.8 - 3.3V
max. hodinová frekvence vyčítání	48 MHz



Obr. 1 - CMOS EV76C454 senzor od firmy e2v, převzato z [4]

Tab. 2 - CMOS senzor MT9V034

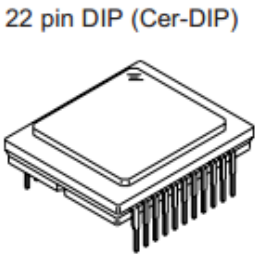
CMOS senzor MT9V034	
typ senzoru	barevný i monochromatický
maximální rozlišení	752 x 480 px
max. rychlost čtení při max. rozlišení	60 fps
rozlišení - datový výstup	10 bitů
velikost pixelu	6 µm x 6 µm
napájecí napětí	3.0 - 3.6 V
max. hodinová frekvence vyčítání	27 MHz



Obr. 2 - CMOS MT9V034 senzor od firmy Aptina, převzato z [5]

Tab. 3 - CCD senzor ICX414AL

CCD senzor ICX414AL	
typ senzoru	monochromatický
maximální rozlišení	640 x 480 px
max. rychlost čtení při max. rozlišení	přibližně 60 fps
velikost pixelu	9.9 µm x 9.9 µm
napájecí napětí	14.55 - 15.45 V
max. hodinová frekvence vyčítání	24.54 MHz



Obr. 3 - CCD senzor ICX414AL od firmy Sony, převzato z [6]

Více senzorů a popisů naleznete v přiloženém dokumentu ve formátu Excel obsahujícím seznam webových stránek výrobců systému strojového vidění a senzorů pro ně (adresář Další materiály na přiloženém CD).

2.1.3. Systémy strojového vidění pro komerční použití - chytré kamery

Pokud se v první části vyskytly problémy s nalezením vhodných senzorů, v této části je problém opačný. Nabídka hotových systému pro účel identifikace předmětů, měření rozměrů

součástek ve strojové výrobě, kontrolu kvality, regulaci, kontrolu scény, rozpoznávání RZ automobilů, rozpoznávání obličejů pro fotoaparáty atp., je opravdu široká. Systém je ve valné většině případů realizován jako "smart camera" připojitelná k PC přes USB, Ethernet, či jiné adekvátní rozhraní. Výpočetní inteligence pak může být soustředěna v počítači. Druhý případ je takový, že kamera posílá už finální data včetně "inteligentního" zpracování signálu. Uvádím zde opět tři příklady jako v případě obrazových maticových senzorů. Řazeny jsou podle následujících kritérií: od "nejméně příslušenství na inteligentním senzoru a nejvíce v PC" po "nejvíce příslušenství a funkcí v senzoru a nejméně v delegovaném PC".

Tab. 4 - Lumenera Lm085 VGA mini USB 2.0 camera

Lumenera Lm085 VGA mini USB 2.0 camera	
technologie a typ senzoru	CMOS - Aptina MT9V032
maximální rozlišení	640 x 480 px
max. rychlost čtení při max. rozlišení	60 fps
rozlišení - datový výstup	8 - 10 bitů
rozhraní pro komunikaci s PC	USB 2.0
napájení	USB 2.0
inteligence v senzoru	Ne
osvětlení scény	Ne
software pro zpracování obrazu	Ano

Obr. 4 - Lumenera Lm085 VGA mini USB 2.0 camera, převzato z [7]

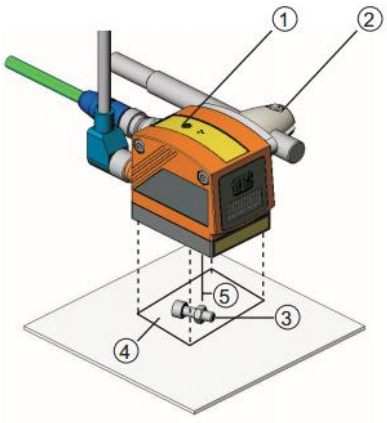
Tab. 5 - Vision Components VISICUBE

Vision Components VISICUBE	
technologie senzoru	CCD
maximální rozlišení	640 x 480 px
max. rychlost čtení při max. rozlišení	32 fps
rozlišení - datový výstup	10 bitů
rozhraní pro komunikaci s PC	Ethernet TCP/IP, RS422
napájení	24 V
procesor	Texas instruments TMS320C64 , 400 MHz, 3200 MIPS
paměťová specifikace senzoru	4 MB Flash EPROM pro program a data programovatelná v hotovém systému (ISP)
osvětlení scény	LED bílé a červené
software pro zpracování obrazu	Ano

Obr. 5 - Vision-Components VISICUBE, převzato z [8]

Tab. 6 - IFM O2D222 senzor pro rozpoznávání objektů

IFM O2D222 senzor pro rozpoznávání objektů	
technologie senzoru	CMOS
maximální rozlišení	640 x 480 px
rychlost detekce	20 Hz
rozhraní pro komunikaci s PC	Ethernet TCP/IP
napájení	24 V
inteligence v senzoru	sledování přítomnosti a polohy, kontrola kvality, úlohy spojené s tříděním a počítáním
osvětlení scény	Infračervené
software pro PC	Ano



Obr. 6 - IFM O2D222 senzor pro rozpoznávání objektů, převzato z [9]

Více senzorů a popisů naleznete v přiloženém dokumentu ve formátu Excel obsahujícím seznam webových stránek výrobců systému strojového vidění a senzorů pro ně (adresář Další materiály na přiloženém CD).

3. Vstupní podmínky a stanovení cílů práce

Motivací práce je využití mikroprocesoru typu ARM. Takovéto MCU dnes již disponují dostatečným výkonem i větší pamětí než tomu bývalo dříve. Navíc mají velice často již implementované rozhraní pro komunikaci se senzorem (DCMI, PPI, I^2C, CAN) i nadřazeným PC (USB, USART) přímo na čipu. V rámci práce se bude pracovat s MCU obsahujícím jádro ARM Cortex - M4 a disponujícím omezenou velikostí vnitřní paměti RAM (maximálně 192 KB). Konkrétní typ požitého MCU bude STM32F407VG popsaný v další části práce.

Vstupním podmínkám se částečně věnovala již část zaměřená na současný stav problematiky. Ta se soustředila pouze na průzkum trhu v oblasti obrazových senzorů a systému pro zpracování obrazu z nich. Podstatnou částí tohoto přehledu by však měly být i dostupné metody zpracování obrazu používané pro mikroprocesory s paměťovým omezením. Avšak, jak poučený čtenář očekává, tento trend je celkem nový a neobvyklý. Z toho, že firmy vyrábějící MCU doplňují možnosti užití rozhraní jako DCMI, lze však soudit, že se s možností zpracování obrazu na této platformě počítá.

Velikost výpočetního výkonu a paměti dnes není trnem v oku vývojových oddělení a mnoho firem se tak soustředí na víceúčelové procesory se zbytečně velkým výkonem a pamětí. Firmy, které se problémem omezené paměti mohou zabývat, ať už z důvodu spotřeby, rychlosti, či jakéhokoliv jiného, rozhodně své know-how nevystaví zdarma na webové stránky. Převážně se tedy bude jednat spíše o výsledek mé práce a experimenty na konkrétním procesoru s konkrétní velikostí vnitřní paměti.

Metod zpracování obrazu je mnoho, a to zdali vystačí s omezenou vnitřní pamětí požadovaného mikrořadiče, nezjistím jinak než jejich zkušební implementací. Cílem práce je vyzkoušet co nejvíce metod pro rozpoznání jednoduchých objektů a implementovat funkci porovnání dvou scén, detekce změny ve scéně. Dále pokusit se složitější metody zjednodušit, či použít jejich výhod při implementaci na paměťově omezeném mikrořadiči. S velkou pravděpodobností půjde o redukci 2D metod, jelikož jednou z možností, jak se zbavit problému omezené paměti, je **načítat obrázek po částech**. Tyto globální metody pak nelze použít v nezměněné podobě. Jinou obecně známou metodou je **načítat obraz ve sníženém rozlišení**. Stejně tak je potřeba uvažovat nad metodami, které nevyžadují uchování původního obrázku při aplikaci konkrétní metody filtrace/lokace. Poté se totiž paměťová náročnost zdvojnásobuje.

4. Volba a testování algoritmů pro účel parametrizace snímku a porovnání snímků dvou scén

4.1. „Image Preprocessing“ – předzpracování signálu

Jedná se o operace prováděné na obrazových datech před jejich samotným inteligentním zpracováním. Tato problematika je rozvedena v mém projektu, na nějž navazuje tato BP. I přesto neuškodí lehká rekapitulace.

4.1.1. Hledání hran pomocí diskrétní konvoluce

Kvůli zjednodušení problému se zde budu věnovat pouze černobílému obrazu (stupně šedi). Jasová informace je pro potřeby rozpoznávání dostatečná.

Za předpokladu spojitosti jasové funkce (představme si jako 3D mapu hor, jejichž podstavou jsou rozměry obrázku a výška daného bodu je hodnota jasu, která může nabývat jakékoliv reálné hodnoty - vlastně se jedná o obrázek s nekonečným rozlišením a nekonečnou jasovou hloubkou) můžeme místa, kde se vyskytují hrany, označit jako místa, kde je nejvyšší gradient v oné vícedimenzionální funkci – nejstrmější stoupání. Pokud se omezíme jen na 1D signál (řádek), bavili bychom se o derivaci. Jádro problému detekce tkví tedy v hledání nejvyšších derivací v jednotlivých bodech signálu. Nesmíme ale zapomenout na to, že obrázek má konečné rozlišení a konečnou jasovou hloubku, je tedy diskrétní. Derivace nebo gradient ze spojitého světa nelze použít.

Derivaci lze ale nejjednodušeji převést do diskrétního světa pomocí diference. V podstatě jde o hledání nejvyšších rozdílů jasů pixelů v určitém okolí daného pixelu. To, jakým způsobem se s jasovou informací daných bodů nakládá, záleží na účelu dané operace. Používá se k tomu takzvaných **konvolučních masek**. Tyto masky se používají pro mnoho obrazových filtrů, je to prostředek, jakým uskutečníme požadovanou obrazovou operaci. Proč konvoluce? Signál je zpravidla brán pixel po pixelu, a tedy v jistém diskrétním čase je jeden pixel předchozí, jeden aktuální, další následující atp. Takže je ke zpracování této diference vhodné použít konvoluci.

Obr. 7 - Příklad diskrétní konvoluce s jádrem (1 ,0,-1)

Nutno podotknout, že stejně jak je činěno v uvedeném 1D příkladu na obr. 7 lze hrany detekovat i ve 2D například pomocí *operátoru Prewittové*, viz [12], slide č. 18. Vyvstává ale problém *detekce ve směru* (podobně jako v případě rozdílu mezi gradientem a derivací). Detekce směru je opět nějak kvantována (jen několik směrů ze všech existujících). S operátorem výše zmíněným, který tvoří matice 3x3 prvky, dosáhneme snadno 4 orientací hran (respektive 8 směrů). Zpravidla však postačuje svislý a vodorovný směr. Každá hrana je pak kombinací těchto směrů. Masky pro tři různé směry vidíme na obr. 8. 2D zpracování mi může již dělat problémy s paměťovým prostorem (v případě zvoleného MCU), takže jednoduchost bude mít pravděpodobně přednost.

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix} \begin{pmatrix} 0 & -1 & -1 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{pmatrix}$$

Obr. 8 - Konvoluční jádra po vzoru Prewittové

4.1.2. Prahování pro převod do binárního obrazu i jako metoda zúžení hran

Prahování je nejjednodušší metoda pro převod obrázku ve stupních jasu do obrázku černobílého o dvou jasových hodnotách (v případě jednoho prahu - rozhodovací meze), nebo více hodnotách (v případě několika prahů). Abych nebyl tak obecný, odkážu se opět k jednoduchému příkladu na obr. 9, který navazuje na příklad diskrétní konvoluce. Z popisu jasně plyne smysl prahování.

Obr. 9 - Příklad prahování s dvěma různými prahy

4.1.3. Prahování s hysterezí

Pokud máme obrázek, který po aplikaci detekce hran výše popsanými metodami a po prahování dává stále neuspokojivý výsledek, je jednou z možností použít takzvané prahování s hysterezí. Obrázek po předzpracování prahujeme dvěma mezemi. Oba výsledky zpracujeme následujícím způsobem: Pokud je intenzita v bodě menší než menší z mezí, poté je bod zahozen. Pokud je hodnota intenzity vyšší než vyšší mez, je zachován. Pokud je mezi oběma mezemi, je přidán pouze v případě, že jeho okolní bod je nad vyšší mezí nebo už přidaný tímto způsobem. Zaniknou tak nevýznamné řetězce bodů, které nejsou součástí významnějších řetězců.

4.1.4. Gaussovský filtr a průměrování (rozostření Gaussiánem)

Zpravidla se aplikuje jako první před hledáním první derivace konvolucí a prahováním. Slouží k odstranění takzvaného Gaussova šumu, který mírně pozměňuje intenzitu jednotlivých pixelů. Výsledkem této operace je mírné rozostření obrázku, které komplikuje částečně hledání hran, nicméně potlačuje hrany nevýznamné. Jde o to najít rovnováhu. Pro níže demonstrované příklady používám druhou konvoluční masku z obr. 10. První je prosté průměrování jasu okolních pixelů. Druhá konvoluční maska už je Gaussián v pravém slova smyslu, který zdůrazňuje střed – vážené průměrování.

$$\frac{1}{9}\begin{pmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{pmatrix} \qquad \frac{1}{16}\begin{pmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{pmatrix}$$

Obr. 10 - Možné masky vyhlazovacích filtrů pro zbavení se šumu

4.1.5. Laplacián (metoda druhé derivace jasové funkce) - alternativa hledání hran

V praktických příkladech níže pracuji především s metodami vycházející z první derivace jasové funkce a poté hledám jejich maxima, která říkají, kde je největší jasový rozdíl, tedy hrana. Jedná se o naivní použití gradientu, protože na základě použité masky využíváme i informaci o směru derivace. Laplacián využívá toho, že druhá derivace jakékoli (1D, 2D) funkce prochází nulou tam, kde má první derivace maximum (extrém). Nevýhodou a výhodou Laplaciánu v jednom je, že jeho použitím ztrácím informaci o směru hrany, respektive jasovém spádu v daném pixelu. Problémem Laplaciánu je citlivost na nechtěný šum a zněkolikanásobování hran. Výpočetně by měl být méně náročný, protože aplikujeme konvoluci jen s jednou maskou, a ne ve všech směrech s maskami různými.

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Obr. 11 - Konvoluční maska pro aplikaci Laplaciánu (druhé derivace) pro hledání průchodů nulou

Hledání hran takzvanou metodou průchodů nulou je ale veskrze jednoduché, nejsložitější je nastavit parametry Gaussovského filtru, který je vhodné použít tak, abychom metodu byli vůbec schopni aplikovat. Pokud se nám to povede, tak aplikujeme Laplacián například konvoluční maskou jakou vidíme na obr. 11. Poté použijeme jednoduchou masku 2x2 na detekci toho, jestli se v okolí daného bodu mění znaménko druhé derivace. Referenční bod si zvolím v levém horním rohu, ale mohl by být kdekoliv jinde. Pokud se v tomto malém okolí nachází body s odlišnými znaménky, tak je referenční bod prohlášen za 1. Jinak za 0.

4.2. „Template matching“ (Pattern matching) - hledání vzoru v obrázku

Jednou ze zajímavých metod použitelnou pro porovnání dvou obrázků, respektive takzvaného „template“ nebo „pattern“, jakožto vzoru, k němuž hledáme podobnost v celém obrázku přijímaném například z CMOS senzoru, je takzvaný „template matching“. Principiálně se jedná o to, že se prochází obrázek a v jednotlivých pixelech obrázku a vzoru se porovnávají hodnoty jasu. Pokud toto porovnání splňuje požadované kritérium (definované programátorem), lze prohlásit hledaný objekt za alokovaný.

$$S_{AD}(x,y) = \sum_{i=0}^{T_{\text{řádek}}} \sum_{j=0}^{T_{\text{sloupec}}} \left| B\big((x+i),(y+j)\big) - B(i,j) \right|$$

$\boldsymbol{S_{AD}(x,y)}$ … sum of absolute differences (součet rozdílů jasů v abs. hodnotě)

$\boldsymbol{x,y}$ … souřadnice v obrázku

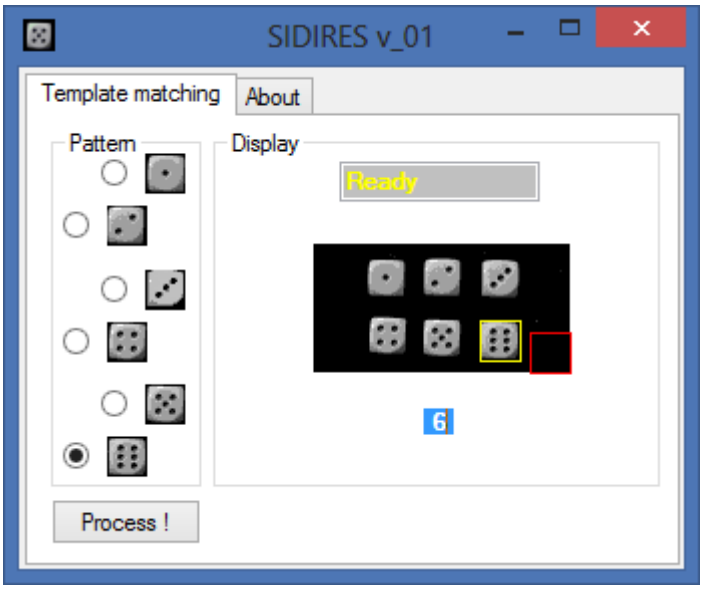
$\boldsymbol{i,j}$ … souřadnice ve výřezu

$\boldsymbol{T_{\text{řádek}}, T_{\text{sloupec}}}$ … rozměry porovnávaného výřezu

$\boldsymbol{B}\ (\boldsymbol{a,b})$ … hodnota intenzity jasu na pozici a , b $(0-255, 8$ bitová jasová škála)

Rce. 1 - Template matching – S_{AD} kritérium

Jak popisuje rce. 1 výše, S_{AD} je velmi jednoduché obecně známé kritérium pro popis jasové podobnosti části obrázku a hledaného vzoru. Nyní stačí hledat souřadnice - (x, y), pro které je toto kritérium minimální, popřípadě splňuje námi stanovené požadavky. Stačí prohledávat do rozměru obrázku mínus velikost vzoru (jak výška, tak šířka), pokud chceme, aby byl vzor celý v mezích obrázku. Pro demonstraci tohoto jednoduchého algoritmu jsem napsal aplikaci ve Visual Studio 2012 C# Express. Aplikace je přiložena v materiálech a je pojmenována SIDIRES (Simple Dices Recognition System). Na výběr máme ze šesti vzorů 20x20 pixelů (strany hrací kostky), které mají být hledány v obrázku 128 x 64 pixelů. Nutno podotknout, že jde o nejjednodušší verzi algoritmu bez schopnosti poznat vzor, který je v obrázku rotovaný. To je vzhledem k budoucí implementaci na STM32F4 bezpředmětné.



Obr. 12 - Template matching - demonstrační aplikace

Obr. 12 výše ukazuje výsledek po hledání vzoru šestky. Žlutý čtverec se v průběhu aplikace přesouvá podle aktuální nejlepší shody. Červený čtverec ukazuje aktuální pozici v algoritmu, kde zrovna počítá S_{AD}. Protože pokud by se zobrazovala každá iterace, tak by překreslovací metoda grafiky nestíhala a zpomalilo by to celou aplikaci, červený čtverec se vykresluje každých 10 iterací. Do značné míry bude také záviset na rychlosti procesoru, na kterém se aplikace spustí. To jsem si vzhledem k účelu aplikace dovolil ponechat nevyřešené.

4.3. Metody porovnání snímků dvou scén

Metodika pro porovnání po sobě jdoucích snímků a na základě jejich vzájemné změny stojí u počátků počítačového a strojového vidění obecně. V padesátých letech byl americkým psychologem J. J. Gibsonem zaveden pojem "Optical flow", ze kterého se později stal i technický pojem. Tento optický tok lze vyjádřit jako vektorové pole ukazující velikosti a směry změny jasu v jednotlivých snímcích (resp. mezi snímky). Pokud vynechám směr změny, u nějž algoritmy musí nutně pracovat s okolím bodu a mezi snímky tak potřebují uchovávat značné množství informací, získávám celkem jednoduché kritérium pro detekci změny v obraze, avšak ne úplně spolehlivé. Na základě velikosti jasové změny v pixelech, potažmo ve větších segmentech, je možno získat informaci o tom, zdali došlo k přesunu objektu v obraze. Tato metoda se nazývá diferenční, neboli jednoduchá diferenční metoda, viz [13]. Metoda má své neduhy, ty jsou popsány v části věnované samotné implementaci pro MCU.

5. Hardware a software pro demonstrační systém

5.1. Hardware pro systém zpracování obrazu

5.1.1. Mikrořadič a vývojový kit od firmy ST Microelectronics

Vzhledem k zadání práce jsem se v prvním přiblížení rozhodl pracovat s STM32F4 - Discovery kitem (obr. 13), popis v [10], jelikož osazovat si desku s procesorem samotným by bylo ve fázi experimentů kontraproduktivní.

Obr. 13 - Vývojový kit STM32F4 - Discovery, převzato z [10]

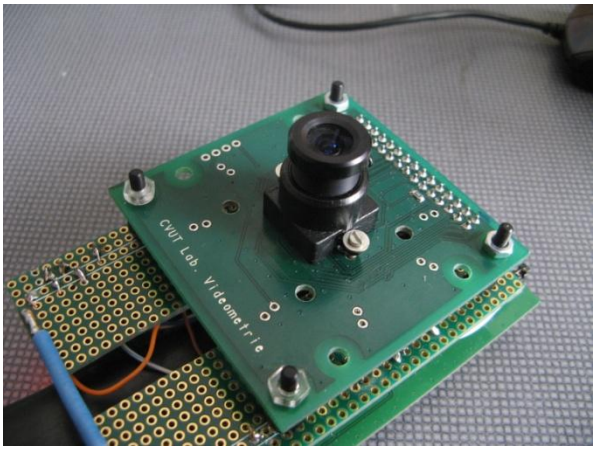
Všechny potřebné informace jsou uvedeny v datasheetu [10], ale uvedu zde přesto jednoduchou přehledovou tabulku (tab. 7) se základními parametry procesoru, který je na tomto kitu osazen, vyčerpávající popis je možno nalézt v referenčním manuálu [1].

Tab. 7 - STM32F407VGT6 informace o procesoru a použitých funkcích

STM32F407VGT6	
jádro	32-bit ARM Cortex-M4F core
paměť RAM	192 kB
paměť FLASH	1 MB
rozhraní použité pro komunikaci s PC	USB 2.0 OTG (On The Go) pouze Full Speed
rozhraní pro komunikaci s obrazovým senzorem	DCMI, I^2C
pouzdro	LQFP 100

5.1.2. Modul obrazového senzoru CMOS

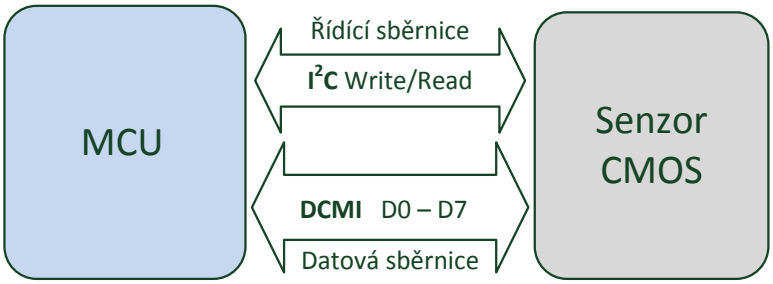
Vzhledem k tomu, že na katedře měření byl již vyvinut modul včetně PCB pro senzor MT9V032/034 - popsaný také v první části (Současný stav problematiky) - tak byla volba jednoduchá. Senzor jsem použil v monochromatické verzi. Pro účel, jímž je snížit paměťovou náročnost algoritmů pro vyhledávání ve scéně/rozpoznávání objektů, není potřeba zvyšovat tuto náročnost barevnými daty. Druhý, čistě praktický důvod byl, že tyto senzory jsem dostal na katedře k dispozici. Co se týče způsobu vyčítání, tak je značně zjednodušen právě u CMOS senzoru v kombinaci se zvoleným MCU - díky zabudovanému komunikačnímu rozhraní DCMI. Desku se senzorem jsem si vlastnoručně osadil a výsledek vidíte na obr. 14. Senzor samotný není možno vidět, jelikož se nad ním nachází objektiv.



Obr. 14 - Osazená deska CMOS senzoru

5.1.3. Propojení mikrořadiče a senzoru pro první testování - prototypová verze HW

Blokové schéma propojení MCU s touto periferií vidíme na obr. 15. Jelikož tento hardware - stejný kit i senzor - byl použit i v bakalářské práci pana Sara [16] z minulého roku, převzal jsem propojení vývodů od něj.

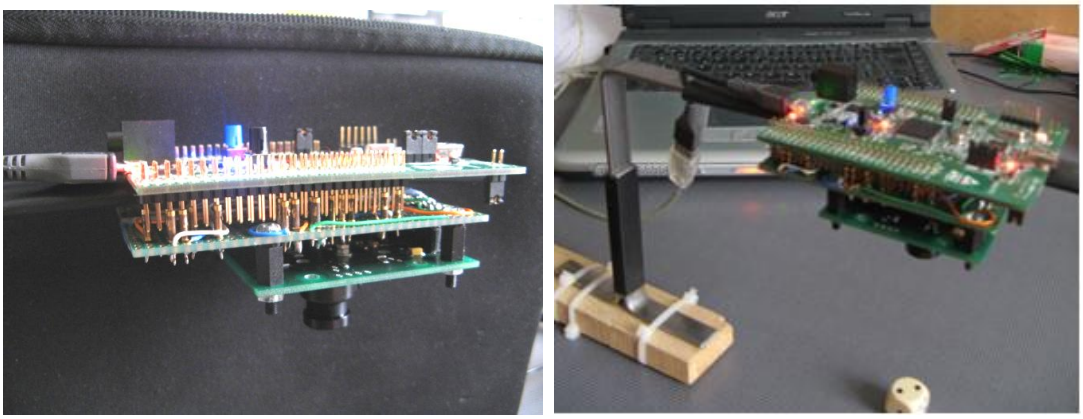


Obr. 15 - Blokové schéma propojení MCU a CMOS senzoru

DCMI, u kterého využíváme 8 datových vodičů, lze zapojit několika způsoby; po kontrole alternativních funkcí dalších pinů jsem pozměnil jen datové vodiče D0 a D1. Dále jsem se setkal s problémem, že CMOS má datové vodiče D0 - D9. **Je potřeba tedy dát pozor a nezapojit vodiče označené jako D0 a D1 (CMOS) a zapojit vrchních osm tedy D2 - D9**. **Pro kit jsou však označeny jako D0 - D7!**

I^2C, o kterém jsem se ještě nezmiňoval, je použito ke konfiguraci senzoru a lze použít jednu z několika linek procesoru. Použil jsem I2C2, po vzoru mého předchůdce.

Vztahy mezi signály vidíme v tab. 8. Snažil jsem se o co nejmenší a nejkompaktnější a současně spolehlivé propojení. Ukázka první verze propojení je zobrazena na obr. 16. Použil jsem univerzální PCB dodávanou k jinému Discovery kitu a rozpůlil ji tak, aby vyšla na obě konektorové strany kitu. Poté jsem na potřebných pozicích napájel FRB konektory a konektor pro propojení se senzorovou deskou, jednotlivé piny jsem pak podle schémat (připojeny v přílohách) propojil izolovanými drátky ze síťového kabelu. Pro první pokusy jsem sestavil ještě jednoduchý výškově polohovatelný držák (obr. 16) na celý senzorový systém, tak abych si mohl ideálně nastavit ohniskovou vzdálenost a velikost scény.



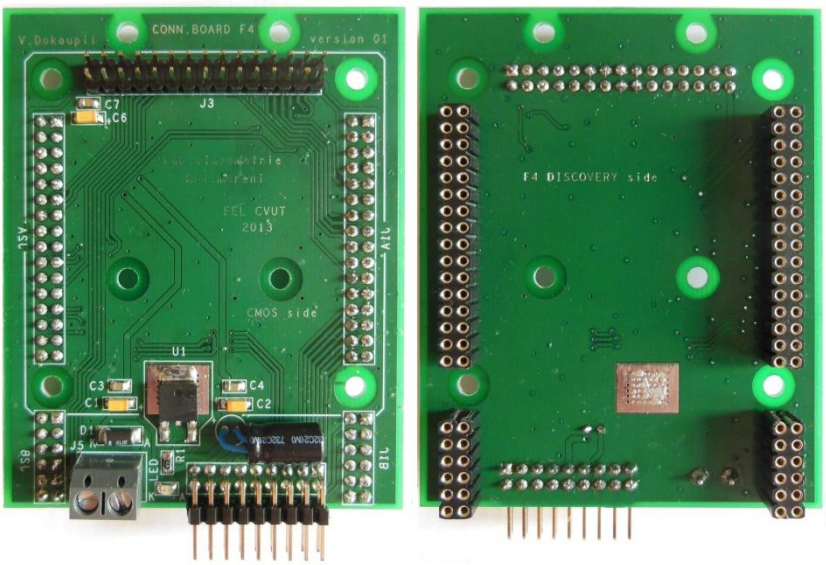
Obr. 16 - Modul senzoru propojený s Discovery kitem / Polohovatelný držák s celou sestavou

Tab. 8 - Propojení modulu CMOS senzoru a Discovery kitu - první verze

Signál v notaci schématu modulu CMOS senzoru	Pin na Discovery kitu
D0	Nezapojeno !
D1	Nezapojeno !
D2	PC6
D3	PC7
D4	PC8
D5	PC9
D6	PC11
D7	PB6
D8	PB8
D9	PB9
PIXCLK	PA6
HSYNC	PA4
VSYNC	PB7
SYSCLK	PB4
RESET	PD5
STANDBY	PD6
OE	PD3
EXPOSURE	PA5
I2C_SDA	PB11
I2C_SCLK	PB10
LEDOUT	PE6

5.1.4. Propojení mikrořadiče a senzoru - finální hardware

Protože jsem měl v průběhu testů stále problém s dvojím napájením Discovery kitu a CMOS senzoru, tak jsem pro budoucí práce a cvičení navrhl plošný spoj místo univerzálního PCB v první verzi propojení. Tento tištěný spoj zaručuje propojení mezi již hotovou deskou modulu CMOS senzoru a deskou Discovery kitu. Návrh počítá s pevným mechanickým propojením s deskou senzoru pomocí čtyř distančních sloupků. Discovery kit bude držet na samotných konektorech, protože jak již předchozí experimenty ukázaly, je to naprosto dostačující spojení. Deska je navržena tak, aby ji bylo možno uchytit opět pomocí čtyř bodů - krátkých distančních sloupků - k držáku, který je popsán v **Příloze 2**.

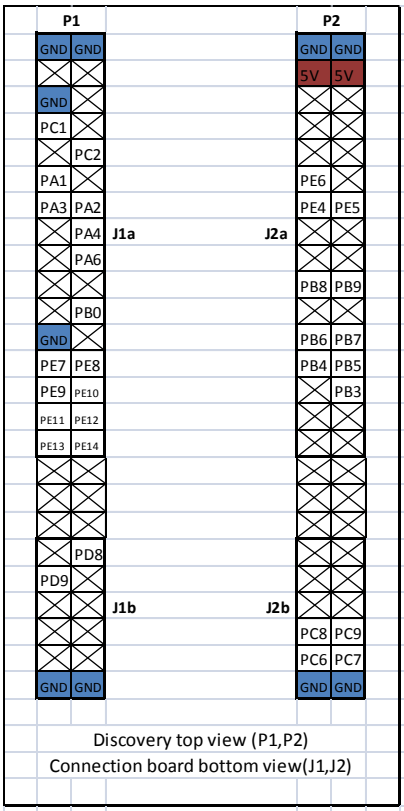


Obr. 17 - Obě strany propojovacího tištěného spoje, po řadě TOP/BOTTOM

Kromě mechanického návrhu je podstatné zmínit stabilizátor umístěný na desce. Ten je určený pro napájení senzorové desky napětím 3,3 V. Zbytek desky je napájen přímo napětím 5 V přivedeným na svorkovnici; stabilizace napětí 5 V se na desce již neprovádí. Předpokládá se připojení stabilizovaného zdroje.

Jak lze vidět na obr. 17, tištěný spoj je oboustranný a kromě konektorů pro CMOS senzor (TOP) a pro Discovery kit (BOTTOM) obsahuje ještě šroubovací svorkovnici pro napájení z 5 V zdroje a konektor, kde jsou vyvedeny rozhraní SPI, UART a I/O piny brány E. Dále je na tomto konektoru vyvedeno napájení 5 V a 3,3 V. Konektor je úhlový, aby připojené vodiče nemohly případně zasáhnout do zorného pole objektivu CMOS senzoru. V případě použití této desky a nedostatku potřebných periferií je možno v kritickém případě napojit piny z vrchní strany Discovery kitu - kratší strana oboustranných kolíků.

Je nutno podotknout, že vzhledem k HW požadavkům - konstrukčním možnostem, jsem musel přeskupit použité piny Discovery kitu a stejně tak používám jinou I^2C periferii. Konkrétní použití je uvedeno na obr. 18. Podrobnější technické informace k této desce naleznete v **Příloze 1** připojené na konci této práce.

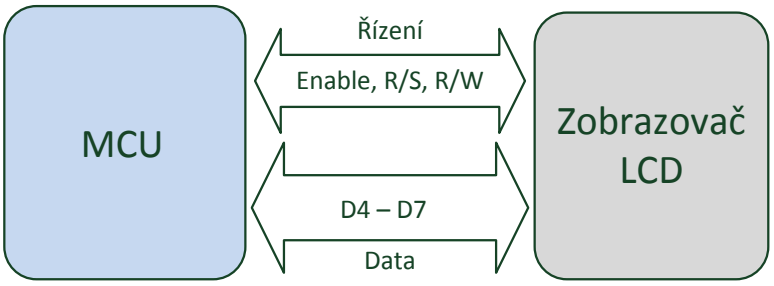


	DISCOVERY SILKSCREEN	function CMOS	other connector
P1	PC1	DCMI_LEDOUT	
	PC2	DCMI_OE	
	PA1	DCMI_EXPOSURE	
	PA3	DCMI_STANDBY	
	PA2	DCMI_RESET	
	PA4	**DCMI_HSYNC**	
	PA6	**DCMI_PIXCLK**	
	PB0	DCMI_SYSCLK	
	PE7		LCD_D4
	PE8		LCD_D5
	PE9		LCD_D6
	PE10		LCD_D7
	PE11		LCD_RS
	PE12		LCD_E
	PE13		LCD_RW
	PE14		LCD_BACKGLIGHT
	PD8		USART3_TX
	PD9		USART3_RX
P2	PE6	**DCMI_D7**	
	PE4	**DCMI_D4**	
	PE5	**DCMI_D6**	
	PB8	I2C_SCL	
	PB9	I2C_SDA	
	PB6	**DCMI_D5**	
	PB7	**DCMI_VSYNC**	
	PB4		SPI3_MISO
	PB5		SPI3_MOSI
	PB3		SPI3_SCK
	PC8	**DCMI_D2**	
	PC9	**DCMI_D3**	
	PC6	**DCMI_D0**	
	PC7	**DCMI_D1**	

Obr. 18 - Použité piny Discovery kitu a jejich funkce - odlišné od prvního zapojení

5.1.5. Propojení mikrořadiče a LCD znakového displeje pro kompaktní verzi

Pro účely zobrazení jednoduchých výstupních informací jsem se rozhodl připojit obyčejný dvouřádkový znakový displej 2x8 znaků, který vidíme na obr. 20, s obecným řadičem S6A0069, viz [11]. Tento řadič odpovídá nejznámějšímu řadiči displejů označovanému jako HD44780. Protože displej umožňuje sérioparalelní mód posílání dat, tak použijeme právě tento pro úsporu vodičů a případných cest na vyvíjené propojovací desce. Ve zkratce lze říci, že se použijí jen datové vodiče D7-D4 a pošlou se dvě čtveřice bitů po sobě místo použití všech osmi datových vodičů a posílání dat v jednom kroku. Blokové schéma propojení MCU a LCD vidíme na obr. 19.



Obr. 19 - Blokové schéma propojení MCU a zobrazovače LCD

Tab. 9 níže uvádí propojení Discovery kitu a LCD zobrazovače s ohledem na volné výstupy zbývající po připojení ostatních periferií. Mimo čtyř datových signálů potřebujeme **R/S** signál pro volbu mezi registrem řídících instrukcí /datovým registrem a **R/W** signál, který v našem případě bude trvale na logické úrovni 0, protože se na displej bude pouze zapisovat bez zpětné kontroly a čtení dat. Dále je podstatný signál **Enable**, s jehož náběžnou hranou se data zapíší z datové sběrnice, kde musí být vystaveny, na displej. Dále jsem vyvedl také signál na ovládání podsvícení displeje, které se bude případně ovládat přes transistor, protože trvalé podsvícení je ve finální aplikaci zbytečné.

Tab. 9 - Připojení LCD zobrazovače

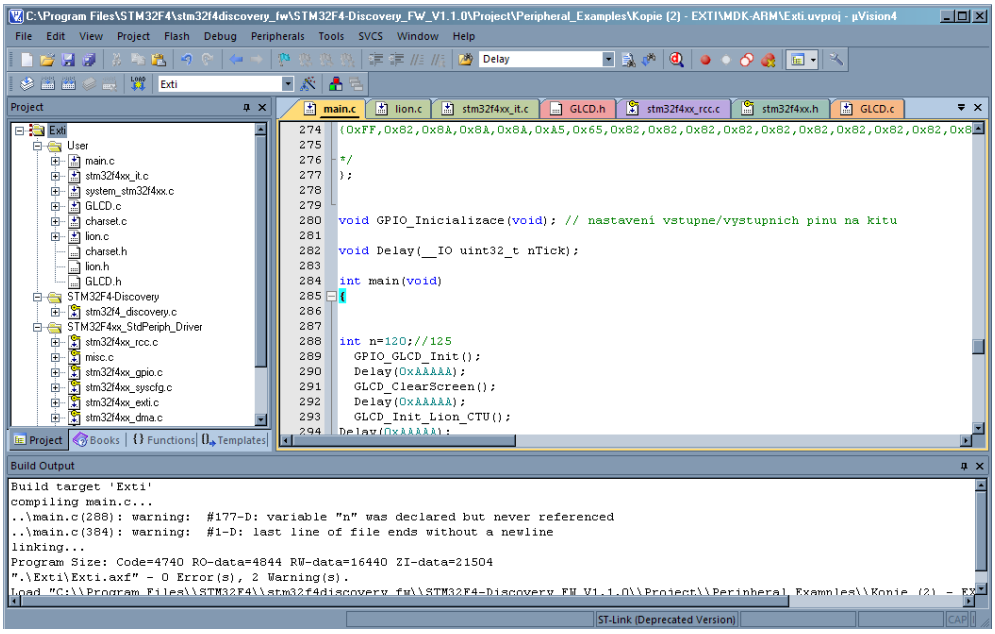
Signál v notaci datasheetu k LCD zobrazovači	Pin na Discovery kitu
D4	PE7
D5	PE8
D6	PE9
D7	PE10
R/S	PE11
E	PE12
R/W	PE13
BACKLIGHT	PE14

Podle výše uvedeného popisu by neměl být problém LCD připojit. Dále se v aplikaci používá knihovna LCD.c, v jejímž hlavičkovém souboru lze jednoduše nastavit piny v případě odlišné konfigurace. Knihovna je upravenou verzí původně vytvořenou pro STM32F0 Discovery kit, kde jsem vycházel z manuálu na webové stránce [15]. Převod pro aktuální vyšší vývojový kit byl otázkou chvilky.

Obr. 20 - Použitý znakový LCD zobrazovač - 8x2 znaků

5.2. Vývojové prostředí pro MCU

Výrobce mikrořadiče informuje o několika dostupných možnostech pro programování MCU. Jsou to jak komerční, tak freeware vývojová prostředí, která umožňují po nainstalování ovladače STLink programovat a debuggovat pomocí USB. Vybral jsem si Keil MicroVision 4, protože jsem s ním pracoval již dříve na 8051 procesoru a jistá redukovaná verze (omezení velikosti programu) je volně šiřitelná. Nicméně pro jisté pokusy s USB jsem také používal Atollic Studio, což je další možnost. Vzápětí jsem se ale vrátil k MicroVision. Obr. 21. ukazuje náhled tohoto vývojového prostředí.

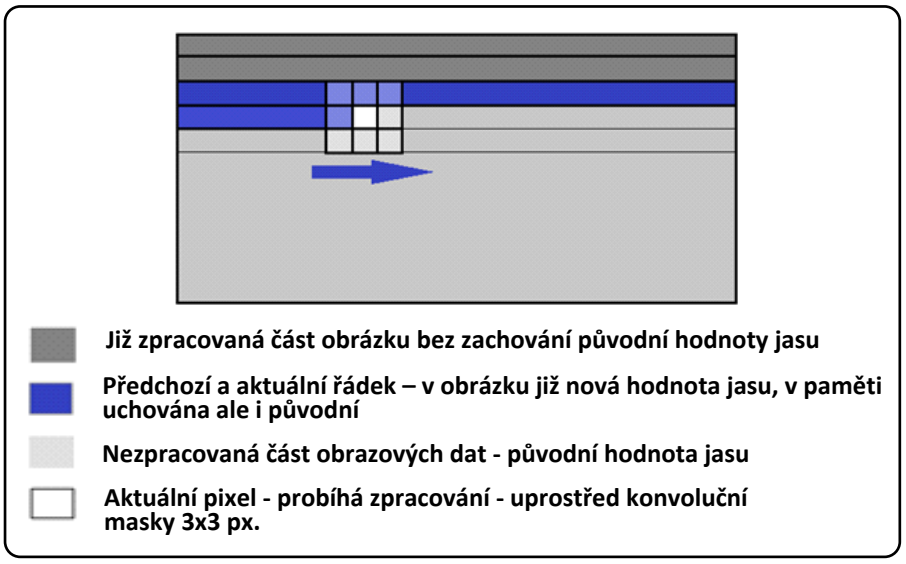


Obr. 21 - Vývojové prostředí Keil MicroVision 4

6. Úprava algoritmů pro omezený paměťový prostor MCU

6.1. Optimalizace procesu předzpracování (filtrace) pro omezený paměťový prostor MCU

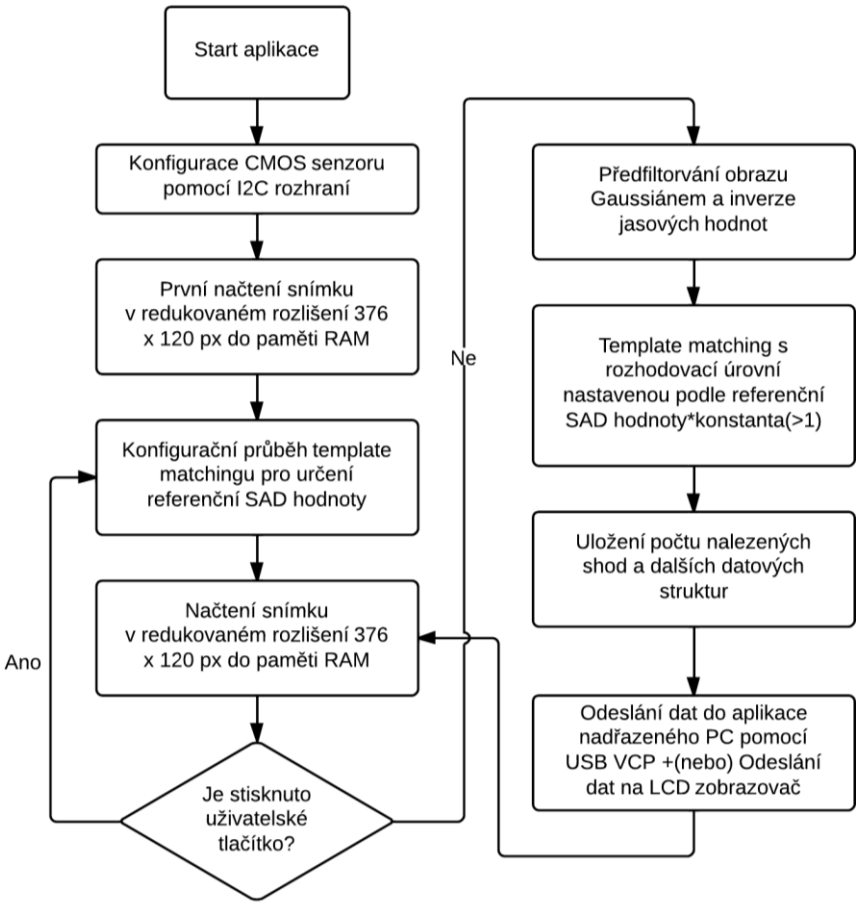
Protože mikrořadič má omezenou velikost vnitřní paměti RAM (konkrétně 192 KB) je nutné metody předzpracování upravit. Tyto jsou v principu paralelními procesy (podle [14], str. 34), jelikož hodnoty zpracované danou metodou ukládáme do jiného pole jasu, protože původní hodnoty musí být uchovány pro zpracování dalších obrazových bodů. Za předpokladu užití osmibitové jasové hloubky a obrázku ze senzoru redukovaného na rozlišení 376 x 120 px je velikost v paměti 44 KB. Pokud bychom ho předzpracovávali čistě "paralelně", zabral by až 90 KB. To je vzhledem k dalším potřebným proměnným a datovým strukturám naprosto zbytečné. Pokud proces předzpracování jednoduše upravíme na částečně "sekvenční", jak je popsáno na následujícím obr. 22, tak potřebujeme uchovávat navíc jen dva řádky. Velikost paměti potřebná pro předzpracování bude tedy 45 120 Byte + 240 Byte. Což zaokrouhlíme stále na 44KB. Touto jednoduchou úvahou jsme ušetřili paměťový prostor pro případné další zpracování a struktury.



Obr. 22 - Princip paměťové úspory pro konvoluční filtrační algoritmy

6.2. Redukovaný "template matching" pro hledání součtu kostek (SUMONDI)

Při implementaci na PC jsem ve své vzorové úloze hledal obraz dle vzoru celé kostky, ideálně umístěné a nerotované vůči rámu. Protože jsem se při implementaci na MCU chtěl vyhnout nutnosti "template matchingu" s rotací (paměťově náročná) a změnou velikosti, tak jsem vymyslel následující. Jelikož jako první cíl jsem si stanovil rozpoznání sumy na dvou kostkách, což je kombinace užívaná ve spoustě deskových her, řekl jsem si, že vzor bude jen jedna tečka na kostce. Tvar tečky zaručuje, že velikost se v rámci zakřivení objektivu bude měnit jen minimálně a rotace tečky je díky jejímu tvaru neexistující problém. Na obr. 23 je pomocí vývojového diagramu nastíněn princip kompaktní verze s LCD displejem. PC verze ještě posílá obraz do PC, což proces mírně zpomalí (v diagramu také naznačeno - odesílání dat do PC). To je však pro mě nepodstatné, jelikož smyslem by mělo být udělat aplikaci, která je schopna fungovat bez PC, potažmo zpracování obrazu v PC. Aplikace nese pracovní název SUMONDI (Sum on Dices).

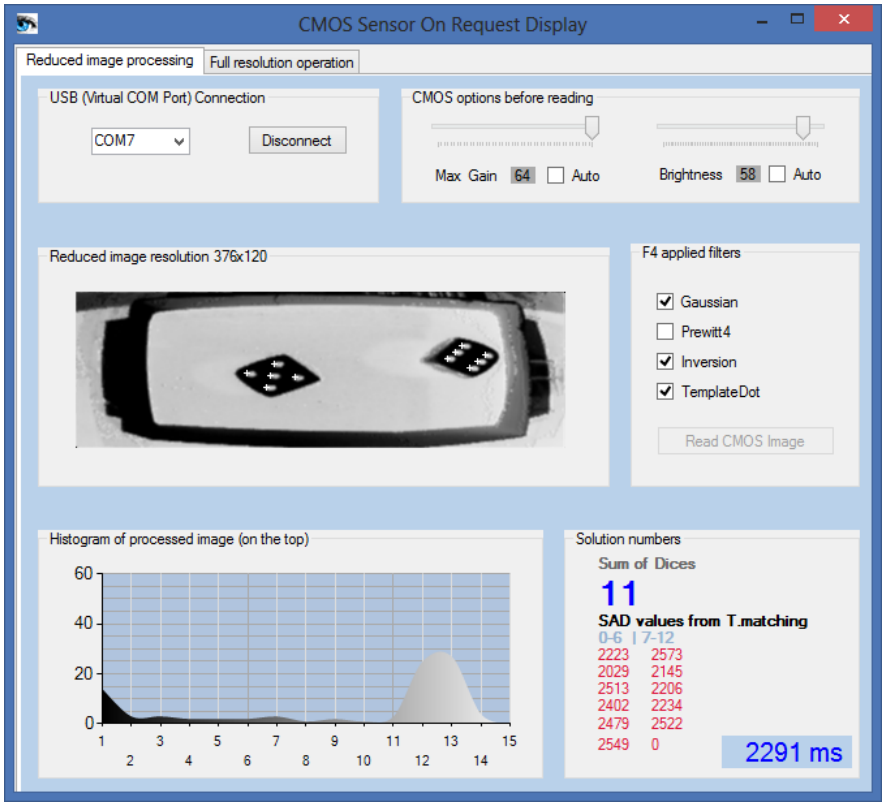


Obr. 23 - Vývojový diagram aplikace hledání sumy kostek pomocí "template matchingu"

Nutno dodat, že pokud požadujeme plynulost aplikace, myšleno ve smyslu neustálého opakovaného házení kostek a zobrazování na displej, "template matching" je aplikace pomalá. Původně jsem měl referenční hledání S_{AD} hodnoty v každém průchodu, což čas výpočtu logicky dvakrát prodloužilo. Požadavek větší plynulosti ale vedl ke změně aplikace. Referenční hodnota S_{AD} záleží na okolních světelných podmínkách. Za předpokladu konstantních podmínek není nutné ji počítat po každém průchodu. Jelikož ale může dojít k jakékoliv změně vnějších podmínek, je zde možnost podržením modrého USER buttonu (uživatelské tlačítko, viz [10]) na Discovery kitu tuto hodnotu v rámci běhu aplikace překonfigurovat. Pokud máme nastaven jednoduchý mód, kdy je výpočet proveden na požadavek uživatele z PC a data se odesílají do PC, rychlost není nutno řešit.

6.2.1. PC aplikace pro první metody hledání v obraze

Na začátku vývoje aplikace bylo potřeba sledovat chování obrazu, tak abych mohl jednotlivé metody optimalizovat. Z této nutnosti vyplynulo vytvoření jednoduché PC aplikace, ze které si uživatel vyžádá požadovanou operaci a poté pomocí tlačítka provede - v nejjednodušším případě pouze vyčtení senzoru a zobrazení dat v bitmapě.



Obr. 24 - PC aplikace pro základní nastavení a zobrazení dat z CMOS senzoru v.01

Aplikace je naprogramována v programovacím jazyku C#. Podobné aplikace a rady k jejich vytvoření se nacházejí na webu [15] a pak na samotných stránkách Microsoftu. Někteří mí předchůdci také pracovali s tímto jazykem, a tak nebyla volba příliš komplikovaná. Důležitým pozitivem je, že v grafickém prostředí Microsoft Visual Studio Expres 2010/2012 si nejen rychle rozvrhnete vzhled aplikace, ale že obsahuje také komunikační třídy a potřebný objekt ***serial port*.** Tento objekt společně s metodou ***serialPortDataReceived()*** zajišťují obsluhu komunikace na straně PC**.** Vzhledem k tomu, že na Discovery kitu používáme USB ve VCP (Virtual COM Port) módu, na PC se jednoduše aplikace tváří tak, že se komunikuje po sériové lince. Výše zmíněná metoda se vyvolá, jak již název napovídá, pokaždé když se v přijímacím bufferu objeví data vyslaná z mikrořadiče.

Základní funkce PC aplikace na obr. 24 byly zpočátku navrhovány tak, aby byly uživatelsky nastavitelné i hodnoty maximálního zesílení senzoru v AGC módu (auto gain control) a celkové světlosti obrazu (desired bin). Po delších experimentech bylo nakonec od auto - módu upuštěno a v aplikaci na MCU je doba expozice nastavena na pevnou hodnotu 300 (nejedná se o časové jednotky SI, ale o čas, který trvá vyčíst daný počet řádků obrazového senzoru) v manual - módu CMOS senzoru. Přepočet na milisekundy, v případě kdy je frekvence hodinového signálu jdoucího do senzoru 21 MHz, ukazuje rce. 2. 752 pixelů je délka řádku a 500 pixelů je aktuálně nastavený Horizontal Blanking (prodleva po dokončení čtení aktivních pixelů v řádku a před dalším horizontálním synchronizačním pulsem).

$$t_{ex} = 300 \cdot \frac{752\ px + 500\ px}{21 \cdot 10^6 Hz \cdot px} = 25\ ms$$

Rce. 2 - Výpočet doby expozice v ms

Dále zde lze nalézt zobrazení invertované scény v redukovaném rozlišení, tak jak ji zpracovává CMOS senzor. Bílé křížky pak v obraze ukazují nalezené tečky na kostkách (vše se děje v MCU aplikaci). MCU aplikace také odešle data o výpočtu - počet nalezených teček (suma kostek) a pak průběžné hodnoty S_{AD} pro potřeby ladění. Tato verze aplikace byla soustředěna na "template matching", nicméně do PC aplikace si můžeme poslat jakákoliv provozní data, jelikož klíčovým problémem, co se týče rychlosti USB komunikace, je odesílání dat obrazových.

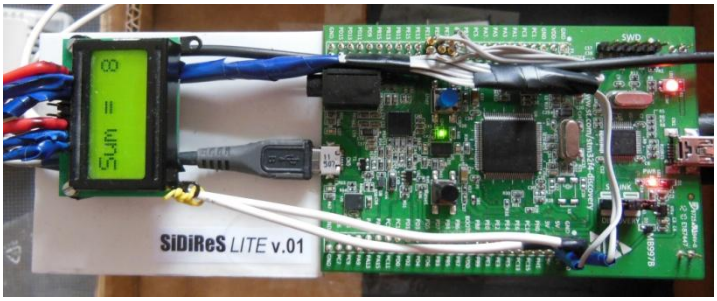
Pro informaci je ještě v pravém dolním rohu aplikace zobrazen čas výpočtu (od okamžiku zaslání požadavku z PC do MCU do okamžiku vykreslení a vypsání požadovaných dat). Zde konkrétně je číslo ještě vysoké, protože v nekontinuálním módu vyčítání je prováděn výpočet

referenční S_{AD} hodnoty v každém průchodu - pokaždé když je metoda uživatelem vyžádána. Jako informační doplněk pak lze ještě v levém dolním rohu aplikace vidět histogram, který je však spočtený až v PC z aktuálně zobrazovaných dat.

6.2.2. Kompaktní verze aplikace pro čtení sumy kostek (SUMONDI) - LCD zobrazovač

Pro snadnou a efektivní demonstraci zobrazení výsledků měření jsem připojil LCD zobrazovač, jak uvádím v HW sekci práce. Na tento displej poté vypisuji sumu teček na dvou kostkách pokaždé, když uživatel kostky hodí pod senzor. Pokud je pod senzorem nedefinovaný počet kostek, více nebo méně než dvě, systém automaticky vyzve k novému vrhu kostek. Základní zobrazené stavy lze vidět v obr. 25. Obr. 26 pak ukazuje tuto verzi systému v celkovém horním pohledu.

Obr. 25 - LCD zobrazení - kalibrace/čekání/pod senzorem kostky



Obr. 26 - Pohled na systém SUMONDI seshora - senzor pod Discovery kitem

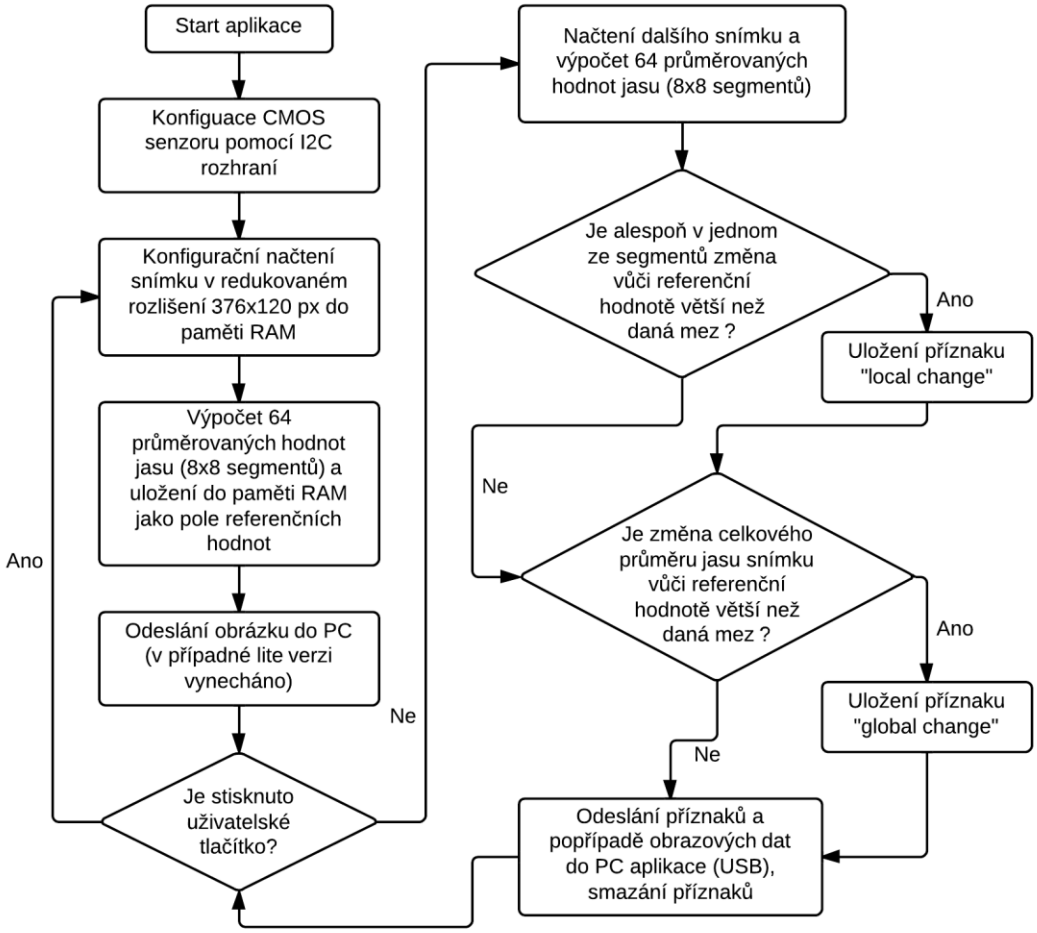
6.3. Porovnání snímků dvou scén (COMPACT GUARD)

Následující metody jsou označeny pracovním názvem COMPACT GUARD.

6.3.1. Verze 1 – Porovnání na základě 64 průměrovaných hodnot po sobě jdoucích snímků (z redukovaného obrazu)

Protože "template matching" je účinná, avšak výpočetně náročná metoda, jak jsem zjistil při implementaci programu SUMONDI na hledání kostek, bylo nutno vymyslet další postupy za

účelem detekce změny ve scéně. Je nutno si uvědomit, že v případě kostek se navíc hledá jen relativně malý objekt (tečka) dle daného vzoru v paměti a stejně se aplikace pohybuje na mezi plynulosti. S velikostí vzoru roste i nutnost porovnávat více pixelů (kvadraticky – s plochou). V případě porovnávání dvou obrázků bychom museli mít tyto dva celé snímky v paměti RAM a další paměť by zabralo pole vypočtených "korelací" - podobností, jak bychom mohli zobecnit název S_{AD} hodnoty. Druhý problém je rotace, což prostor možných korelací znásobuje. Takže tato cesta je s tímto mikrořadičem slepá, v případě požadavku na rychlé nalezení změny v obraze (příchod osoby do místnosti, špatné natočení etikety).



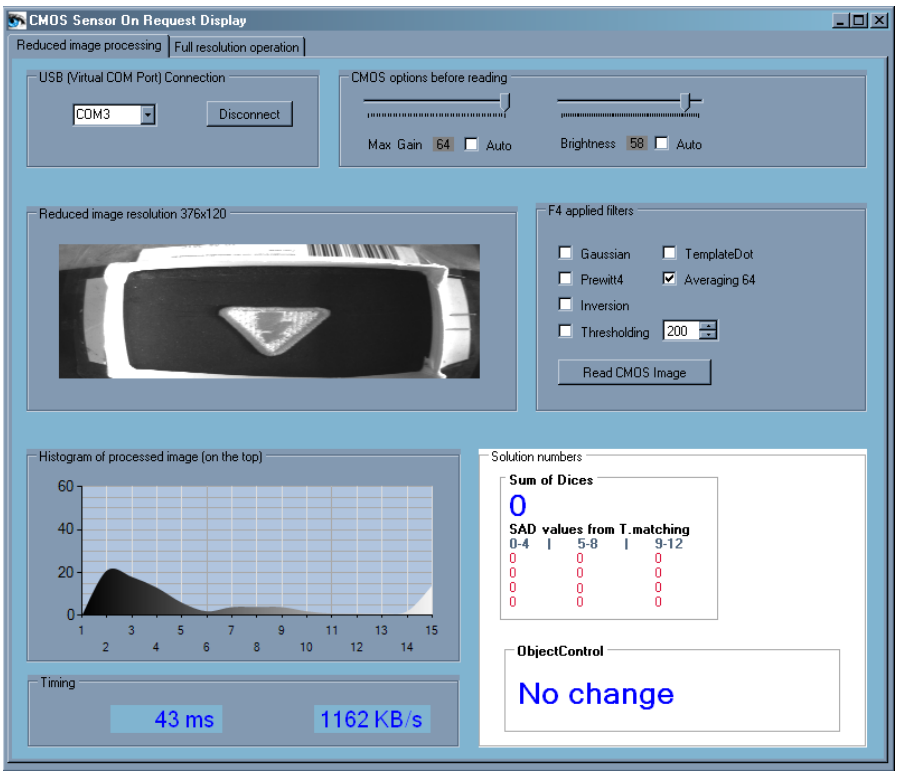
Obr. 27 - Vývojový diagram funkce porovnání snímku dvou scén

Implementoval jsem tedy metodu, kterou zkráceně popisuje následující vývojový diagram na obr. 27. Je to metoda "sníženého" rozlišení, která reprezentuje referenční obrazový prostor pomocí 64 hodnot, které jsou průměrnými hodnotami jasu v segmentech rozdělujících pravidelně scénu v počtu 8x8. Objem dat uchovávaný v mezičase mezi snímky se tak značně sníží. Tyto hodnoty jsou pak vypočteny v každém snímku a porovnány s referenčními. Na

základě toho je vyvolána příslušná reakce v obslužném programu na PC, případně na displeji. Demonstrační vykreslení uchovávaných hodnot je vidět na obr. 28.

Obr. 28 - Uchovávaná informace - demonstrační vykreslení

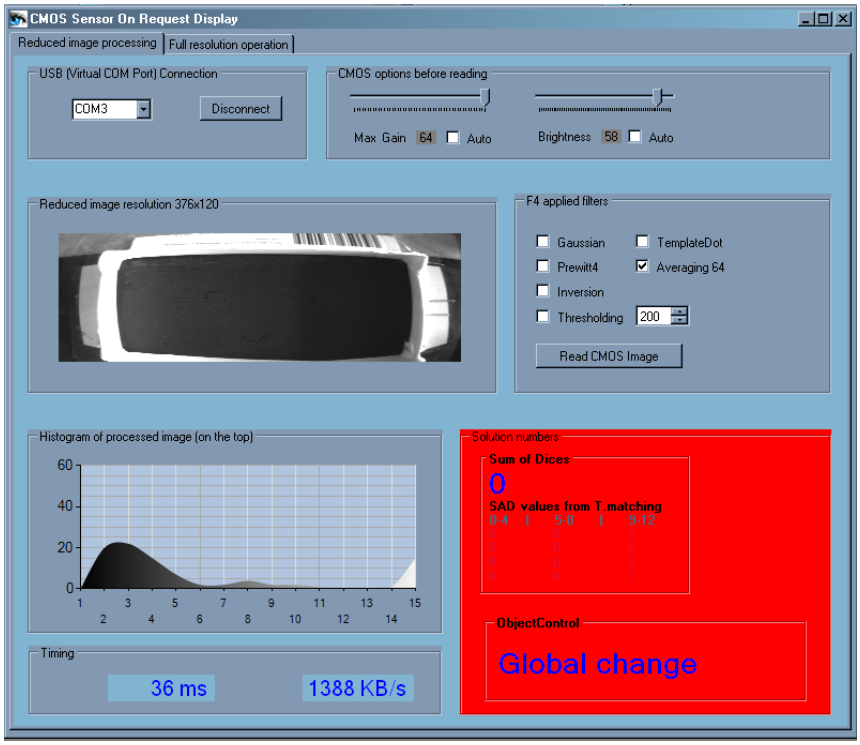
Jelikož implementovaných funkcí přibývalo, přišlo mi v této fázi nesmyslné, pro každou tuto funkci dělat zvláštní PC aplikaci. Proto jsem se rozhodl rozšířit aplikaci, použitou již pro "template matching" v aplikaci SUMONDI s hracími kostkami. Sice to nebude tak uživatelsky komfortní, ale pro mé účely to postačí. V budoucnu se případně dá udělat jedna specifická aplikace. V řadě případů je přenášení obrazu do PC naprosto zbytečné, protože zásadní je pro mě rozhodnutí MCU, zdali je ve scéně něco špatně, nebo zdali je vše v pořádku. Obraz už je jakási přidaná hodnota pro komfort uživatele.



Obr. 29 - Detekce objektu v obraze - výchozí scéna

Obr. 30 - Detekce objektu v obraze - lokální změna - natočení

Obr. 31 - Detekce objektu v obraze - lokální změna - posun



Obr. 32 - Detekce objektu v obraze - globální změna - nepřítomnost

Výše zobrazené snímky obr. 29, obr. 30, obr. 31 a obr. 32 popisují chování aplikace COMPACT GUARD (verze 1) v různých testovaných situacích. Jedná se o výchozí situaci, otočení objektu, přesunutí v rámci zorného pole CMOS senzoru a odstranění ze zorného pole CMOS senzoru.

V průběhu testování bylo dosaženo následujících poznatků. Metoda detekce je velmi závislá na nastavení mezí rozhodujících, zdali je daná změna v obraze (lokální či globální) uznána či zdali je tolerována jako dostatečně malá. Tento problém vyvstává v případě změny nasvětlení scény. Jak bylo totiž výše zmíněno, používá se manuální mód senzoru a nastavuje se expozice. Je to vhodné v případě, že není žádoucí, aby kolísal jas ve scéně, jak se může stát v automatickém módu, nicméně pomalá změna osvětlení scény činí problém. Vhodné řešení by mohlo být následující. V případě, že změna obrazu není uznána jako dostatečně velká, uloží se nové zprůměrované hodnoty, protože se mohly mírně změnit a za nějaký čas to zopakují. Když se drobné nedetekované změny sečtou, tak systém detekuje globální změnu vůči původnímu referenčnímu obrazu. Toto by byla cesta, jak se tomu vyhnout.

6.3.2. Verze 2 – Porovnání na základě 256 průměrovaných hodnot z postupného vyčítání senzoru v plném rozlišení

Primární snaha je navrhnout metody, které jsou implementovány v MCU. Obraz vysílaný do PC je jen jakási nadstavba - například pro kontrolu scény. Zatím nebylo v žádné metodě využito celé rozlišení senzoru, jelikož posílání tohoto obrazu pro kontrolu do PC by celou metodu znehodnotilo. Když se obraz neposílá do PC, dá se také otestovat takzvané postupné vyčítání z CMOS senzoru, a to v plném rozlišení. Senzor se používá v módu, který se někdy odborně nazývá ROI (Region of Interest), z čehož plyne zaměření pouze na část obrazu, která je v daném kroku důležitá. V jednom průchodu dojde k načtení jednoho řádkově orientovaného segmentu **752x30 pixelů** (parametry je možné změnit v hlavičkovém souboru knihovny image_process.h). Segmentů je v tomto případě 480 px/30 px = **16**. Velikost takovéhoto segmentu v paměti je vidět z rce. 3 .

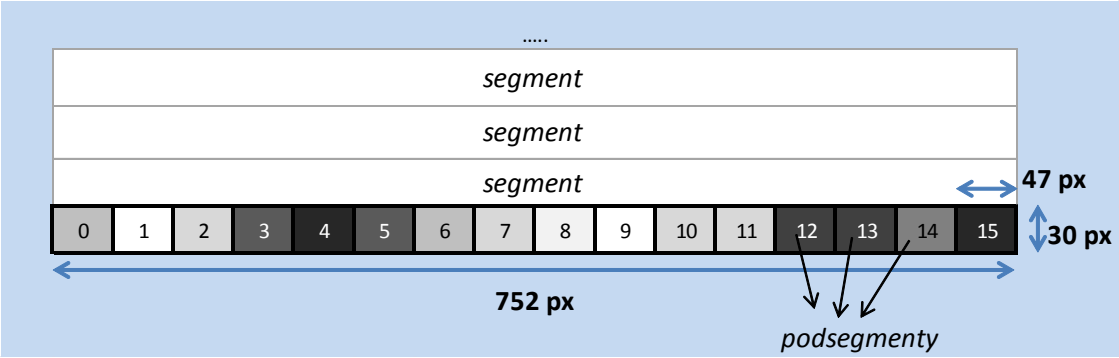
$$752 \cdot 30 \cdot 1\ B = 21960\ B = \frac{21960}{1024} KB = \mathbf{21,4\ KB}$$

Rce. 3 - Výpočet velikosti načteného segmentu

Poté, co je tento segment načten do paměťového prostoru ***CMOS_data[],*** se do pole 32 bitových integerů vytvořeného příslušnou metodou nazvanou ***ApplyAveraging64_ROI()*** nasčítají hodnoty jasu v příslušných podsegmentech, jak je demonstrováno na obr. 33. Proč musí být integery 32- bitové, je vidět z rce. 4. Toto pole bude před redukcí zabírat **64 Byte.**

$$\frac{752}{16} \cdot 30 \cdot 255 = 359\,550 \Rightarrow\ 0x57C7E \Rightarrow \mathbf{0b10101111100011111110} > 16\ bit$$

Rce. 4 - Objem sečtených dat podsegmentu v nejnepříznivějším případě



Obr. 33 - Segment a podsegment - vysvětlení

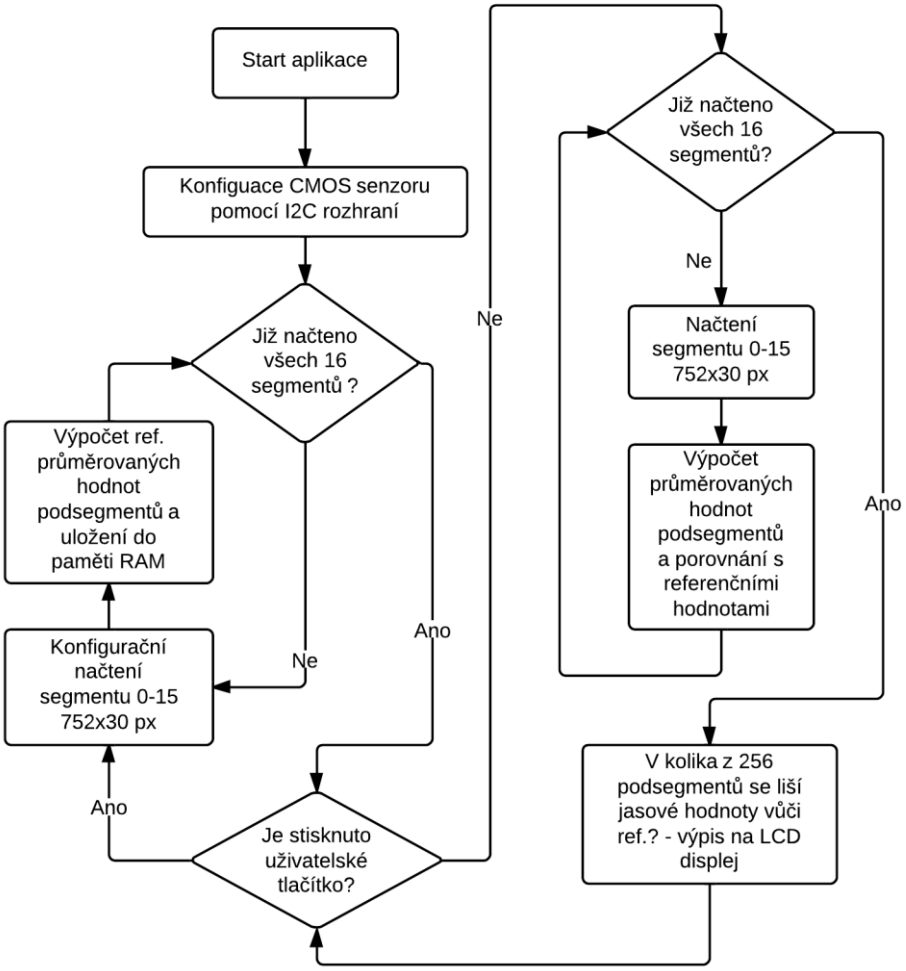
Jakmile celý segment projde tímto algoritmem, podělí se jednotlivé nasčítané hodnoty počtem pixelů v podsegmentech pro převod na 8-bitový rozsah. Toto dělení se musí dělat nakonec celé početní operace, jinak by došlo ke ztrátě informace způsobené celočíselným dělením malého čísla velkým a výsledek by byl nula. Tyto přepočtené hodnoty zabírají pro jeden segment pouze malý paměťový prostor, v tomto případě konkrétně **16 Byte**. Tato data jsou obsažena v jednom segmentu složeném z 16 podsegmentů. Segmentů je celkem 16 a proto se mezi snímky uchovává **256 Byte** obrazové informace, což je vůči velikosti celkového rozlišení snímku (rce. 5) značná komprese. Nyní musíme zvážit využitelnost této informace.

$$752 \cdot 480 \cdot 1\ B = 360\ 960\ B = \frac{360\ 960}{1024} KB = \mathbf{352\ KB}$$

Rce. 5 - Výpočet velikosti celého snímku - více než dvojnásobek velikosti RAM

Celý algoritmus využívající výše zmíněnou funkci pro výpočet průměrovaných hodnot, včetně kalibrace, je popsán vývojovým diagramem na obr. 35. Stojí za to zmínit, že porovnání se provádí pouze lokální (podsegmenty), i když není problém to rozšířit o další možnosti. Porovnání je součástí stále stejné funkce a provádí se z aktuálně vypočtených průměrů a těch uchovaných od předchozího snímku. Nakonec se vypíšou na displej informace o tom, v kolika podsegmentech došlo ke změně oproti referenční hodnotě (první řádek). V druhém řádku se ukazuje doba trvání operace vyčítání a zpracování celého snímku. To je vidět na obr. 34.

Obr. 34 - Zobrazení dat popisujících změnu scény

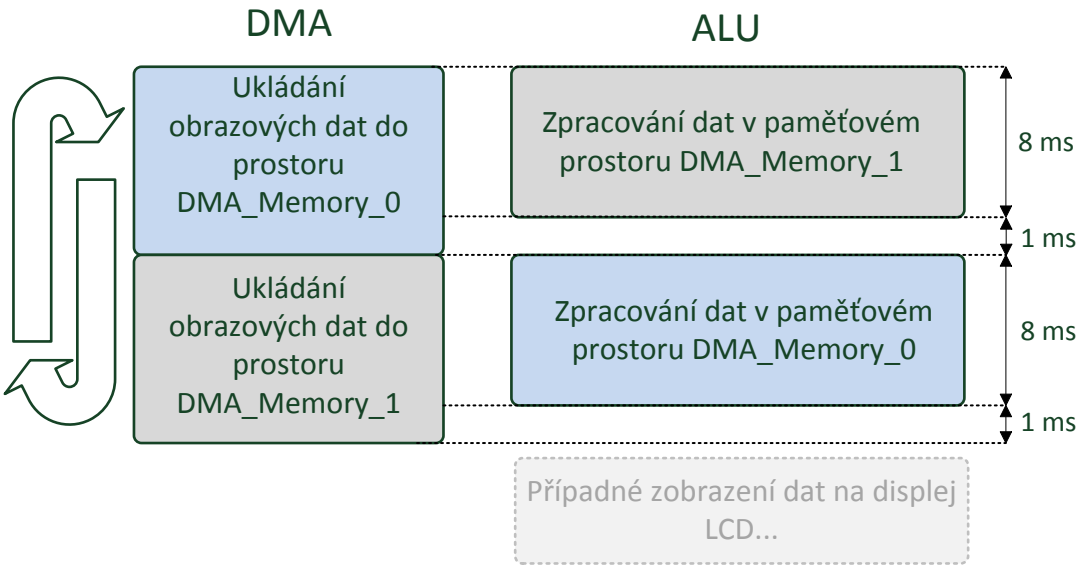


Obr. 35 - Vývojový diagram aplikace postupného vyčítání snímku

6.3.3. Verze 3 – Ukládání dat do dvou paměťových lokací užitím DMA řadiče (v double buffer módu) pro čtení obrazu ze senzoru

Procesor ARM osazený na Discovery kitu disponuje periferií DMA, která je používána pro vyčítání z CMOS senzoru přímo do paměti. Pokud bychom chtěli předchozí aplikace zrychlit, což je logický požadavek (konkrétní časy operací budou uvedeny v části Zhodnocení dosažených výsledků), lze nastavit "double buffer" mód (mód ukládání do dvou paměťových lokací). V principu to znamená, že se na straně senzoru data tváří jako jeden celek, který je postupně vyčítán od počáteční adresy (po DCMI rozhraní). Na straně Discovery kitu je však

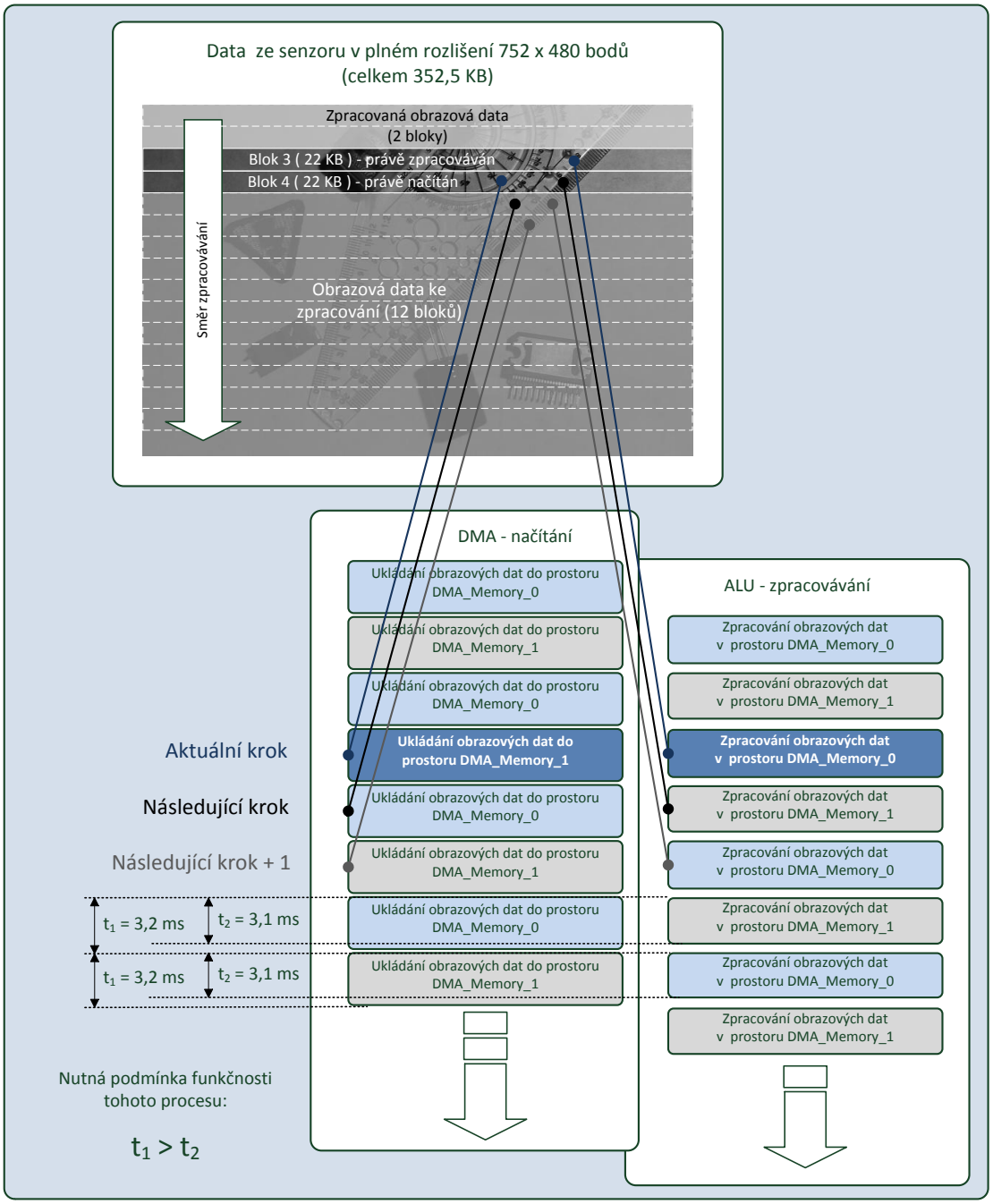
možno tato data v průběhu přenosu rozdělit do dvou bloků - například o poloviční velikosti původního vyčítaného snímku. Tato funkce je užitečná, protože jakmile se naplní první pole hodnot, lze data z něj rovnou zpracovávat v průběhu plnění druhého pole hodnot, a tak stále dokola, jak je zjednodušeně naznačeno na obr. 36. Samotný proces v tomto případě (modifikace Verze 2 - rozdělení vstupních dat na dva bloky 376x60 px) trvá 16 ms. Zpomalení pak způsobí zobrazování na LCD displej, záleží na implementaci zobrazovacích funkcí, která není momentálně optimální. Pokud si představíme tuto aplikaci bez zobrazovače, tak platí časování z obr. 36.



Obr. 36 - Proces "double buffer" funkce na příkladu vyčítání obrazových dat

6.3.4. Verze 4 – Průběžné vyčítání a zpracování snímku v plném rozlišení s užitím přesunu do dvou paměťových lokací (double buffer mód) řadičem DMA

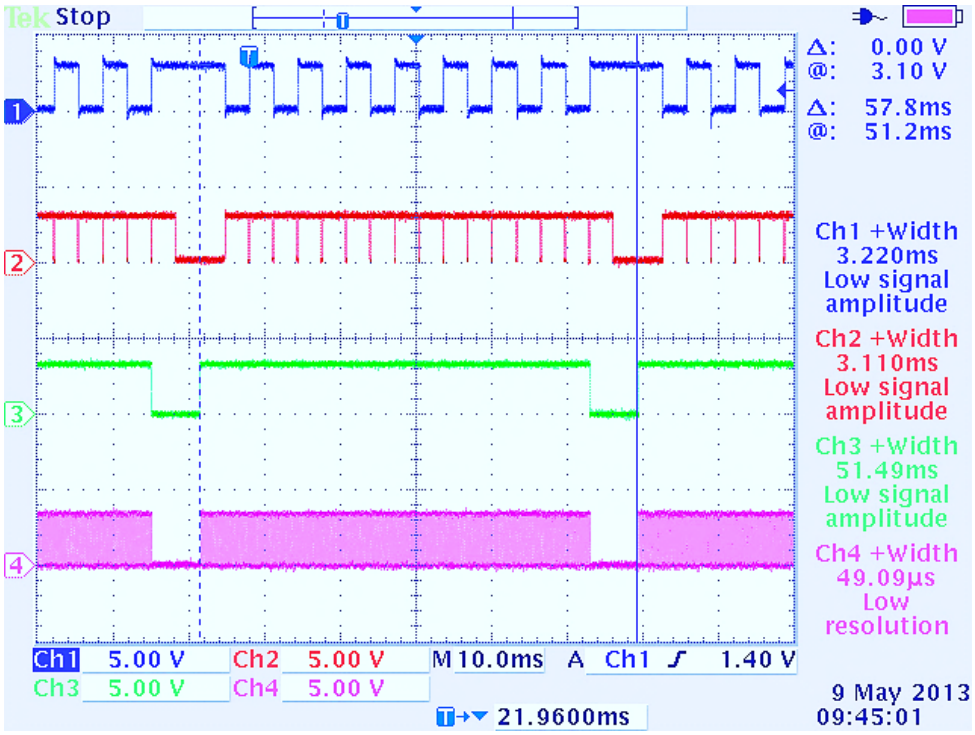
Poslední a zároveň nejdokonalejší metodou ze skupiny COMPACT GUARD, která byla implementována, je průběžné zpracování obrazových dat v plném rozlišení. Obr. 37 nastiňuje princip celého procesu. Obrázek se podobně jako ve verzi 2 vyčítá postupně v celém rozlišení. Rozdílem je, že se nepřepisují řídící registry módu ROI, což je pomalé. Obrázek se pro senzor vyčítá jako celek, takže se na začátku provádí jen jedna konfigurace senzoru. Ostatní čtení a zpracování segmentů už je automaticky řízeno překlápěním DMA v "double buffer" módu nakonfigurované na neustálou obsluhu příjmu dat v případě aktivity VSYNC signálu (také nazývaný Frame Valid - signál aktivní při vyčítání snímku, signál vertikální synchronizace).



Obr. 37 - Princip průběžného vyčítání a zpracování obrazu v plném rozlišení

Proces byl také kontrolován pomocí osciloskopu a na obr. 38 je uveden příklad průběhu jednotlivých signálů popisujících proces čtení a zpracování obrazových dat. První kanál (modrá barva) ukazuje přepínání mezi paměťovými lokacemi DMA_Memory_0 a DMA_Memory_1. Pokud se signál nachází v logické úrovni high a zároveň je v logické úrovni high i zelený signál (Kanál 3 - VSYNC), tak se zapisuje do první lokace. Pokud je modrý signál na úrovni low, tak se zapisuje do druhé lokace. Kanál 2 (červená barva) ukazuje zpracovávání dat. Pokud je signál high, zpracovává se blok dat, který byl v předchozím kroku přijat (buď

v první, nebo druhé paměťové lokaci). Pokud je signál low, nezpracovávají se žádná data. Tato situace nastává mezi sestupnou hranou zeleného signálu (VSYNC) a načtením prvního bloku dat - délku ovlivníme parametrem Vertical Blanking senzoru CMOS. Prodleva také nastává mezi zpracováními jednotlivých segmentů – zde je její délka závislá na parametru Horizontal Blanking senzoru CMOS. Jak je uvedeno na obr. 37, tak čas zpracovávání jednotlivého bloku dat musí být kratší než čas vyčítání tohoto bloku. Na datech z osciloskopu je to ukázáno právě na červeném signálu, kde jsou tyto krátké skoky do logické úrovně low. V případě, že by nebyla tato časová podmínka dodržena, data by se jednoduše nestíhala zpracovat. Pro informaci je zde zobrazen i Kanál 4 (fialová) popisující signál HSYNC (horizontální synchronizace). Tento signál mění svou hodnotu s každým řádkem, takže je jeho frekvence mnohem vyšší než frekvence změn ostatních signálů. Důležité je, že se fialový signál kryje se signálem zeleným (VSYNC).



Obr. 38 - Naměřená data z osciloskopu Tektronix TDS3034B

6.4. Popis knihovny pro operaci s obrazovými daty - IMAGE_PROCESS.c

Jelikož funkcí aplikovaných na obrazová data různých velikostí ze senzoru je mnoho, již od počátku byly všechny sdružovány do samostatné knihovny. V závislosti na typu funkce je umožněna změna velikosti vstupního snímku v hlavičkovém souboru (IMAGE_PROCESS.h), nebo změna velikosti segmentu - v případě postupného zpracování, například při postupném čtení několika výřezů. Většina metod byla implementována pro redukovaný obraz o rozměrech 376 na 120 pixelů. Některé jsou určeny jen pro segmenty, zpravidla je funkčnost dané funkce omezena velikostí vstupních dat (pole hodnot jasu ze senzoru, parametry). V hlavičkovém souboru jdou upravovat například také velikosti průměrovaných segmentů používaných v metodě rozpoznání snímků dvou scén. Všechny tyto vstupní parametry jsou definovány pomocí maker, protože na jejich základě se pak při překladu programu alokují velikosti potřebných datových struktur (zpravidla jednorozměrných polí datového typu integer (celé číslo) délky od 8 do 32 bitů). Měnit tak například v průběhu programu velikost vyčítaného snímku se primárně nepředpokládá. Kromě základního hlavičkového souboru se vkládá ještě soubor math.h, primárně kvůli použití funkce absolutní hodnota (abs).

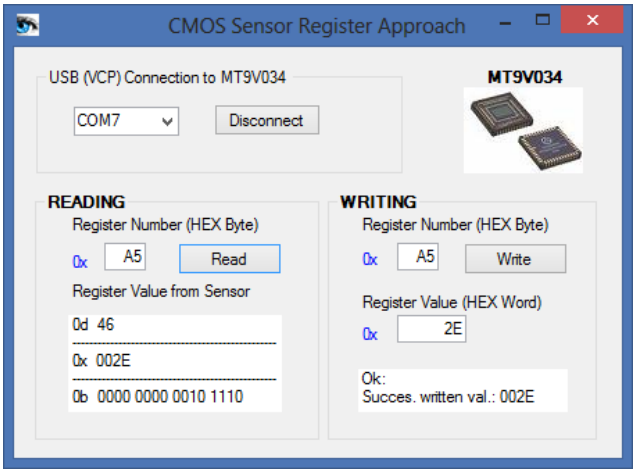
6.5. Kontrolní aplikace pro čtení a zápis do registrů obrazového CMOS senzoru MT9V034

Protože je těžké při vývoji a snaze testovat obrazový senzor získat zpětnou vazbu, zdali je senzor nastavený správně či ne, po konzultaci s vedoucím práce jsem se rozhodl vytvořit jednoduchou PC aplikaci využitelnou i v budoucnosti. Tato aplikace je určena převážně ke čtení, popřípadě k zápisu dat do registrů obrazového senzoru. Funkce jednotlivých registrů lze nalézt v datasheetu [5].

Jelikož adresy registrů jsou osmibitové, tak teoreticky existuje až 256 funkčních registrů k ovládání senzoru nebo informování uživatele. Ve skutečnosti se používá registrů přibližně polovina. Velké množství registrů je určeno jen ke čtení a zápis do nich může senzor v nejhorším případě i poškodit. Proto byla provedena v části kódu pro tuto aplikaci implementovanou na MCU základní ochrana proti zápisu do registrů značených jako RO (read-only), určených pouze ke čtení. Jedná se pouze o kontrolu registrů, kde může být zápis nebezpečný z hlediska poškození senzoru CMOS.

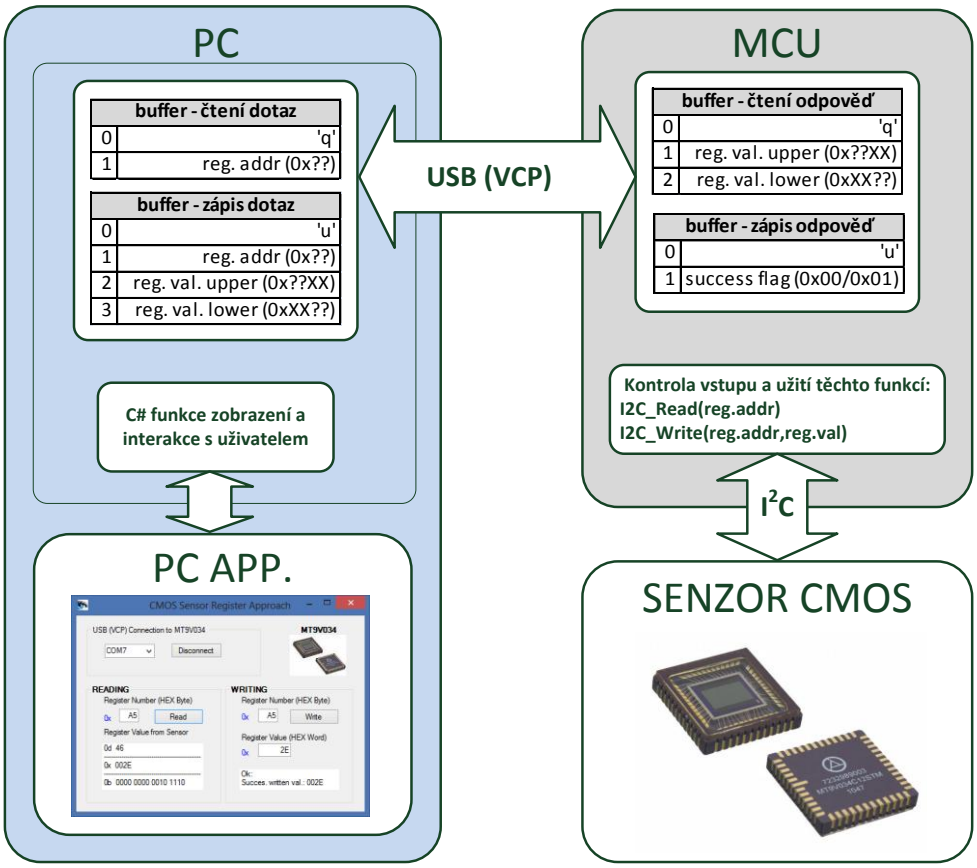
Registry, do kterých se smí zapisovat, pak mají různé rozsahy povolených hodnot, až šestnáctibitové. Kontrola těchto rozsahů by byla náročná a zápis špatné hodnoty není fatální, pokud se týče nebezpečí zničení senzoru. Tyto rozsahy nejsou ošetřeny a závisí pouze na programátorovi, který aplikaci použije, zdali si dobře přečte seznam povolených hodnot.

Aplikace má část implementovanou na MCU a část implementovanou na PC. MCU část je jednoduše přenositelná do všech aplikací, které používají soubor ***usb_cdc_vcp.c*** a jeho komunikační metody. Lze tak poté i v jejich průběhu kontrolovat stavy senzoru. Možnost zápisu závisí na implementaci problému. Například pokud v každém průchodu konfigurujeme nějakou hodnotu registru přímo v aplikaci na MCU, stěží ji přepíšeme trvale z PC. Náhled PC aplikace je vidět na obr. 39.



Obr. 39 - CMOS Sensor Register Approach – okno aplikace

Komunikace mezi senzorem a PC prostřednictvím MCU jako rozhraní mezi nimi probíhá pomocí sběrnice USB a I^2C. Nepoužijeme ale DCMI rozhraní, jelikož data nenačítáme, zajímají nás pouze hodnoty registrů (V této aplikaci). Na obr. 40 je nastíněn přibližně princip komunikace včetně popisu bufferů (bloků) osmibitových hodnot, jaké posílám. Buffer pro čtení a zápis se odlišuje znakem na první pozici a poté následují specifická data. Podobně fungovala i většina komunikace v předchozích aplikacích.

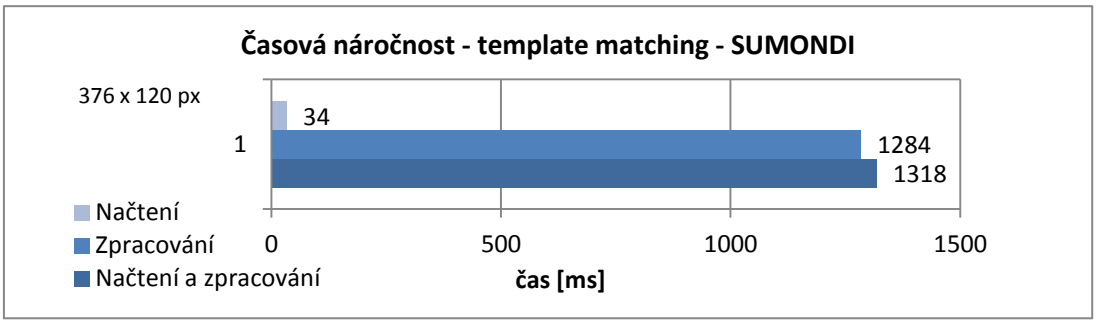


Obr. 40 - Princip aplikace pro čtení a zápis do registrů obrazového senzoru MT9V034

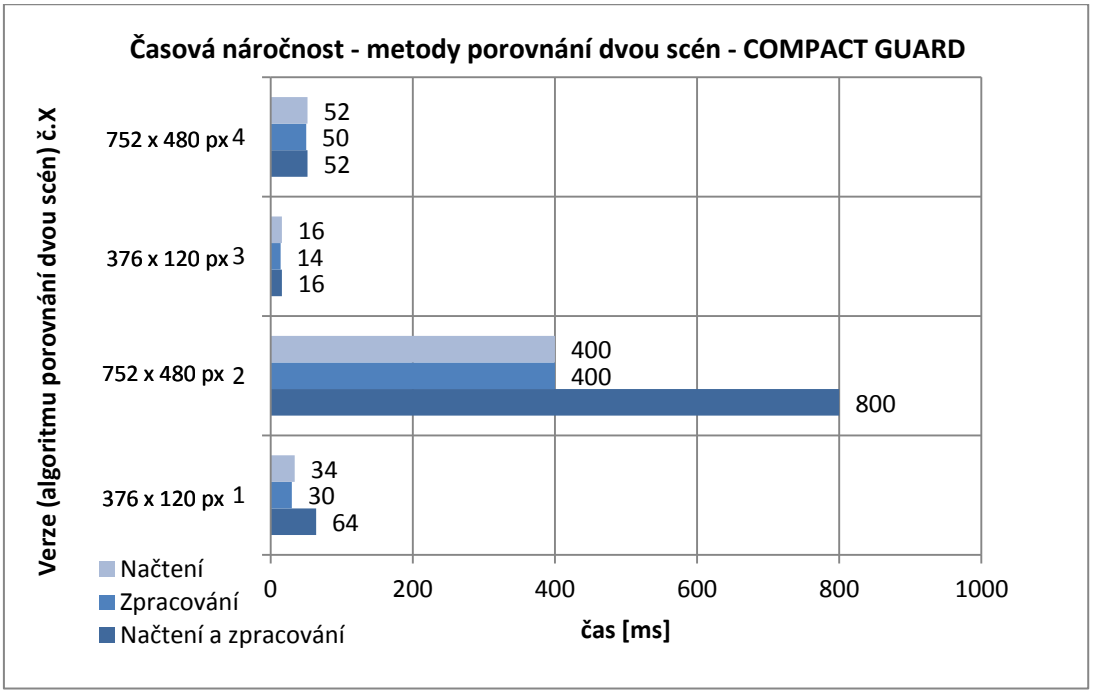
7. Zhodnocení dosažených výsledků

7.1. Porovnání metod zpracování obrazu

Hlavní částí zadání bakalářské práce byla implementace metod porovnání snímků dvou scén a parametrizace snímku. Pro demonstrační úlohu byla nejprve implementována metoda „template matching“, což je jeden ze způsobů parametrizace snímku. Hledaným parametrem je zde počet teček na kostkách pod senzorem. Obr. 41 ukazuje časovou náročnost procesu vyčítání obrazových dat a jejich zpracování touto metodou v kompaktní verzi. Ukázalo se, že pro potřeby demonstrační úlohy je rychlost metody dostatečná, pro plynulé zpracování obrazu (například v průmyslové výrobě) se v této verzi nehodí.



Obr. 41 - Časová náročnost - template matching - SUMONDI



Obr. 42 - Časová náročnost - metody porovnání snímků dvou scén - COMPACT GUARD

Metody porovnaní snímků dvou scén, které byly implementovány, pracují všechny na principu uchování jistého množství redukovaného objemu dat z předchozího snímku, která se porovnají s adekvátními daty ze snímku aktuálního. Důležité je, jakým způsobem se to provádí. Na způsobu značně závisí i rychlost celé operace. Pro tento účel byl nastaven v MCU timer (TIM5) a změřena rychlost trvání čtyř metod, které jsou popsány v práci. Výsledek a porovnání časů lze vidět na obr. 42. Celková doba trvání není u všech metod součtem doby zpracování a načtení, jelikož se tyto dvě činnosti mohou částečně překrývat - při průběžném zpracování. Především vyzkoušení možnosti průběžného zpracování obrazu a možnosti užití přesunu do dvou paměťových lokací pomocí řadiče DMA (v double buffer módu) považuji za úspěch této práce.

Analýza dostupných metod předcházející samotnému programování byla nejtěžší částí. Kromě metod jako „template matching“ a filtračních metod aplikovaných na statický obraz (Gaussián, prahování, inverze, atp.) popsaných výše, byly uvažovány možnosti MCU a metody porovnání snímků dvou scén jsem navrhl sám (založené na teorii diference mezi snímky).

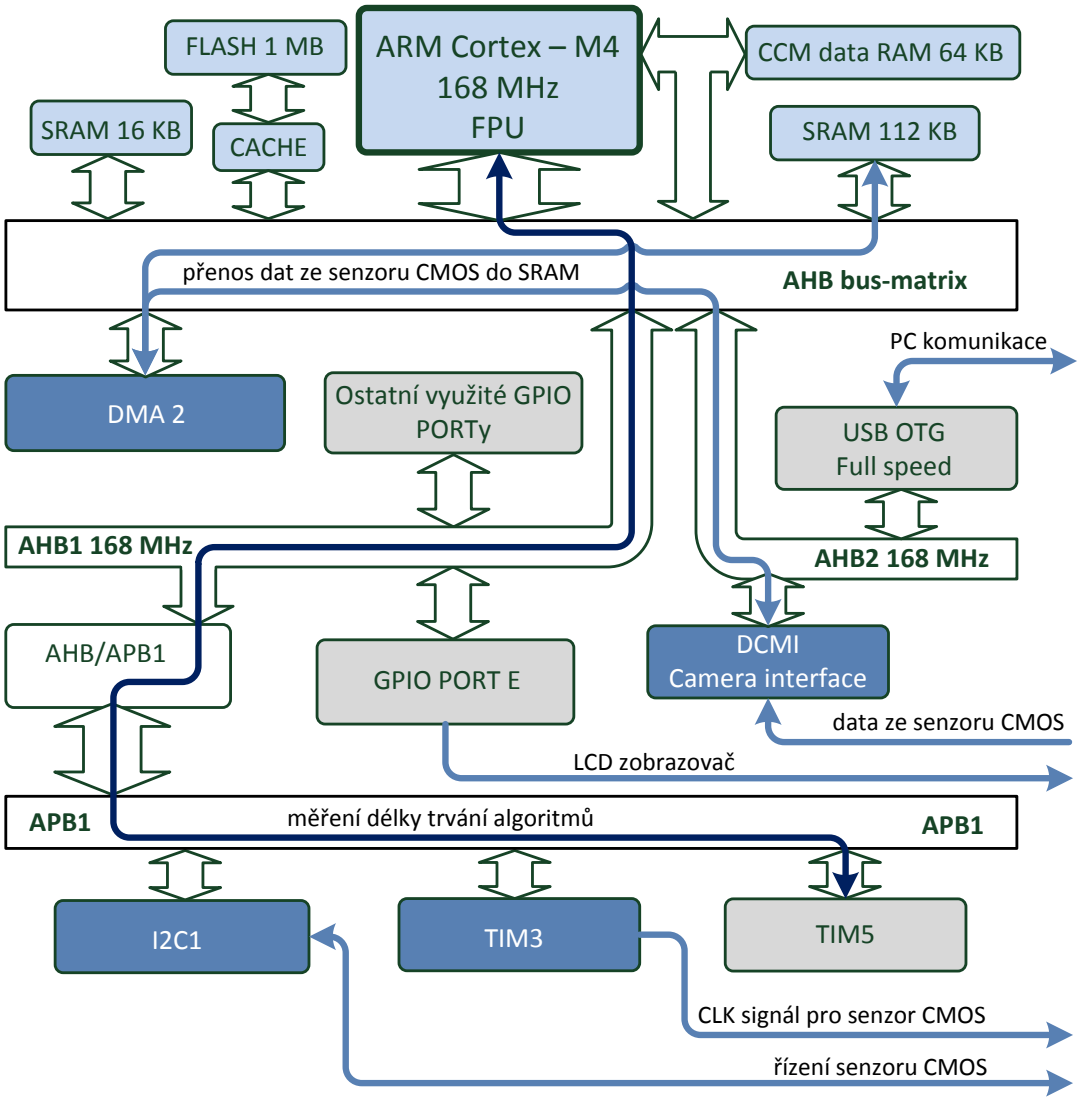
Ostatní části zadání, například vytvoření uživatelské aplikace pro obsluhu senzoru, byly také splněny. Aplikací bylo navrženo více. Sloučení všech navržených aplikací do jedné by znamenalo další úpravy a mohlo by být tématem další práce. Tvorba aplikací si vyžádala naučit se pracovat v programovacím jazyce C#. Jazyk je podobný většině dnešních objektových programovacích jazyků a je uživatelsky příjemný, takže to nebylo příliš náročné.

7.2. Přehled využívaných technických prostředků mikrořadiče

Vzhledem k tomu, že se v průběhu programování MCU využilo mnoho jeho funkcí a dostupných technických prostředků, bylo vytvořeno blokové schéma ve snaze ukázat využívané funkce přehlednou formou.

Obr. 43 ukazuje hlavní využité funkční bloky mikrořadiče v rámci celé bakalářské práce. Komunikace mezi bloky se provádí přes sběrnice, jejichž názvy jsou také uvedeny. Tmavě modrá barva ukazuje bloky určené pro ovládání a vyčítání senzoru CMOS. Ostatní bloky jsou šedé. Jádro mikrokontroléru a paměti byly odlišeny světle modrou barvou (bloky úplně nahoře obrázku).

Porovnáním s dokumentací Discovery kitu tak lze snadno zjistit, která komunikační rozhraní, čítače či ostatní bloky mikrořadiče, se dají použít při současném aplikaci metod z této bakalářské práce.



Obr. 43 - Hlavní použité funkční bloky mikrořadiče STM32F407VG

8. Závěr

V bakalářské práci na téma „Zpracování obrazu vestavěným mikrořadičem“ byly zpracovány všechny body zadání a dokonce i další úkoly, které vyplynuly z postupného řešení zadané práce.

V rámci zpracování zadaného tématu byla provedena analýza dostupných obecných metod zpracování obrazu, avšak nakonec se práce více soustředila na návrh metod vlastních – především z hlediska optimalizace výpočetního procesu na MCU. Nejdůležitějším poznatkem z této části je, že značné zpomalení procesů vyčítání a zpracování může být způsobeno procesem posílání řídících dat do senzoru CMOS. To může způsobit čekání na následující vertikální synchronizační puls rozhraní DCMI, poté co jsou tyto instrukce odeslány. Proto je lepší senzor nastavovat jen jednou (při inicializaci senzoru CMOS) a velikost vyčítaných dat, aktuální souřadnice segmentu atp., je vhodnější upravovat až v MCU. Přínosem je také možnost zpracovávání dat v průběhu vyčítání dat následujících, které probíhá pomocí DMA řadiče (nejlépe v módu přesunu do dvou paměťových lokací - double buffer mód). Tento postup nezatěžuje jádro MCU, protože probíhá přímý přesun dat ze senzoru do vnitřní paměti mikrořadiče.

Byla navržena aplikace SUMONDI pro rozpoznání teček na vrhacích kostkách a určení jejich počtu – pomocí hledání vzoru tečky ve scéně. Dále byly navrženy čtyři verze aplikace COMPACT GUARD pro detekci změny ve scéně a popis velikosti této změny – pomocí počtu segmentů obrazu, kde ke změně došlo. V rámci testování metod lokace objektů ve scéně byla také vyvinuta PC aplikace SIDIRES hledající vzor dané vrhací kostky na základě uživatelského požadavku.

Výběr a implementace metod na mikrořadiči ukázaly, že se lze vypořádat s problémem omezené paměti celkem obstojně, pokud se vyzkouší a plně využijí dostupné možnosti MCU – především přímý přenos dat do paměti pomocí DMA. Pokud je zpracováván obraz redukovaný, dají se aplikovat i takové operace, jako je hledání hran, a to s přijatelnou obnovovací frekvencí zpracovaného obrazu. Pro obraz v plném rozlišení plynule fungují všechny průměrovací programy označované jako COMPACT GUARD.

Uživatelských aplikací pro obsluhu senzoru bylo vytvořeno několik. Byla vytvořena například aplikace pro čtení a zápis do registrů nebo aplikace umožňující jednoduché úpravy obrazových dat takřka v reálném čase s možností jejich zobrazení.

Navržen byl také hardware (deska plošného spoje) pro jednoduchou možnost sestavení „smart kamery“ s STM32F4-Discovery kitem (obsahujícím MCU) a modulem CMOS senzoru. Následně byl vyroben v prototypové výrobě. Testování neukázalo žádné zásadní nedostatky a prototyp tak může být použit v dalších aplikacích.

Užití procesoru s jádrem ARM Cortex-M4 jako samostatné jednotky pro jednoduché zpracování obrazu se ukázalo jako případná levnější a přesto funkční alternativa k složitým obrazovým systémům s výkonnějšími procesory. Především s vhodným využitím DMA se dá dosáhnout požadovaných výsledků v jednoduchých úlohách, například otočení etikety nebo přemístění objektu ve scéně, a to v přijatelném čase.

9. Zdroje

[1] ARM Limited: Cortex-M4 Revision r0p1, Technical Reference Manual,ARM DDI 0439C

[2] ST Microelectronics: Reference Manual, RM0090, Doc ID 018909 Rev.3

[3] Yiu J.: *The definitive Guide to the ARM Cortex-M3*, Elsevier, 2007

[4] E2V: CMOS SENSOR Jade family: technical list [online]. 2012, 2 2/12, s. 2 [cit. 2013-02-14]. Dostupné z: http://www.e2v.com/e2v/assets/File/documents/imaging-space-and-scientific-sensors/CMOS%20Datasheets/20968_JADE%20Flyer%20v5_AW_LR.pdf

[5] APTINA: 1/3-Inch Wide-VGA CMOS Digital Image Sensor MT9V034: datasheet. [online]. 2008, 10/08 EN, s. 82 [cit. 2013-02-14]. Dostupné z: http://www.aptina.com/support/documentation.jsp?t=0&q=137

[6] SONY: ICX414AL: datasheet. [online]. E02115E8Y, s. 31 [cit. 2013-02-14]. Dostupné z: http://www.1stvision.com/cameras/sensor_specs/ICX414.pdf

[7] LUMENERA: Lm085: datasheet. [online]. 2008, č. 06122008, s. 3 [cit. 2013-02-14]. Dostupné z: http://www.lumenera.com/resources/documents/datasheets/industrial/Lm085-datasheet.pdf

[8] VC Vision Sensors: VISICUBE: datasheet. [online]. s. 1 [cit. 2013-02-14]. Dostupné z: http://www.vision-components.com/en/products/smart-cameras/vc-vision-sensors/

[9] I FM: O2D222: datasheet. [online]. 2009, s. 3 [cit. 2013-02-15]. Dostupné z: http://www.ifm.com/products/cz/ds/O2D222.htm

[10] ST Microelectronics: UM1472: STM32F4DISCOVERY STM32F4 high-performance discovery board. [online]. roč. 2012, 022256 Rev 2, s. 38 [cit. 2013-02-15]. Dostupné z: http://www.st.com/web/catalog/tools/FM116/SC959/SS1532/PF252419

[11] S6A0069: 40 SEG / 16 COM DRIVER & CONTROLLER FOR DOT MATRIX LCD: datasheet. [online]. 2000, Ver 0.0, s. 35 [cit. 2013-03-16]. Dostupné z: http://www.cloverdisplay.com/pdf/S6A0069.pdf

[12] HLAVÁČ, Václav. Hledání hran: prezentace. [online]. s. 43 [cit. 2013-03-17]. Dostupné z: http://cmp.felk.cvut.cz/~hlavac/TeachPresCz/11DigZprObr/22EdgeDetectionCz.pdf

[13] RADKE, Richard J. Image Change Detection Algorithms: A Systematic Survey. [online]. 2004, s. 32 [cit. 2013-04-18]. Dostupné z: http://www.ecse.rpi.edu/~rjradke/papers/radketip04.pdf

[14] DAVIES, *E. Computer and machine vision: theory, algorithms, practicalities*. 4th ed. Amsterdam: Academic Press, 2012, xxxvi, 871 s. ISBN 978-0-12-386908-1.

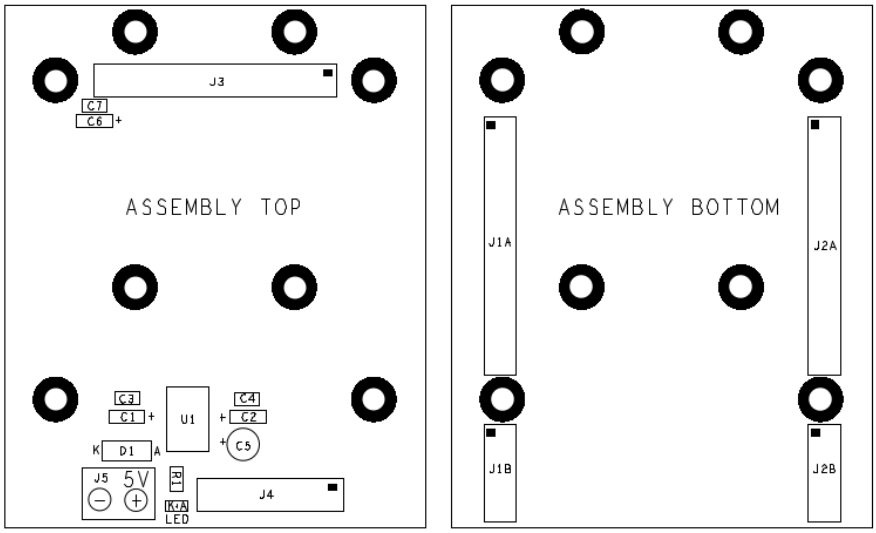
[15] MCU.cz. [online]. [cit. 2013-05-02]. Dostupné z: http://www.mcu.cz

[16] SARO, Jan. *Spolupráce STM32F4xx s optoelektronickými senzory* [online]. Praha, 2012 [cit. 2013-04-05]. Bakalářská práce. ČVUT. Vedoucí práce doc. Ing. Jan Fischer, CSc.

[17] NOVÁK, Tomáš. *Knihovna pro práci se senzorem CMOS* [online]. ČVUT. Praha

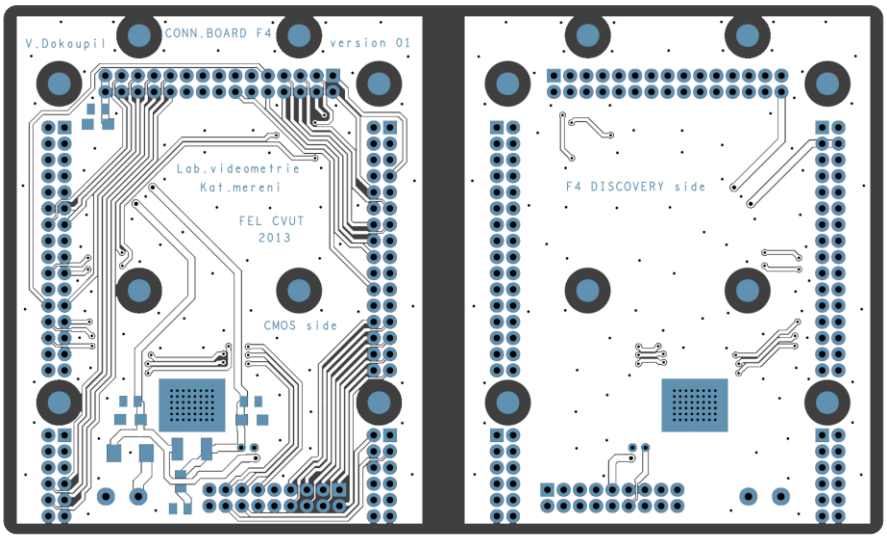
Příloha 1 – Osazovací manuál PCB pro propojení CMOS senzoru a Discovery kitu

V této části naleznete podrobnější informace o tom, jak osadit a zapojit propojovací desku mezi senzor CMOS a Discovery kit. Úvod k této desce naleznete v kapitole 5. Na obr. 44 níže vidíte rozložení součástek.



Obr. 44 - Osazovací plán PCB

Jednotlivé hodnoty součástek užité na této propojovací desce jsou uvedeny v tab. 10. V případě nutnosti lze osadit jinými alternativními součástkami ve vhodných pouzdrech. Návrh desky samotné lze vidět na obr. 45.



Obr. 45 - PCB - pohled z obou stran (vrstvy: soldermask, drill, etch)

Tab. 10 - Seznam součástek

Seznam součástek		
označení ve schématu	**hodnota**	**pouzdro**
C1	10 µF/16 V	SMD A
C2	100 nF	0805
C3	10 µF/16 V	SMD A
C4	100 nF	0805
C5	100 µF/16 V	elyt radiální, RM=2mm
C6	10 µF/16 V	SMD A
C7	100 nF	0805
D1	SUF 4007	MELF
J1A	16x2	header female -přímý
J2A	16x2	header female -přímý
J1B	6x2	header female -přímý
J2B	6x2	header female -přímý
J3	15x2	header male - přímý
J4	9x2	header male - úhlový 90°
J5	1x2	svorkovnice 5 mm
LED	RED	0805
R1	470 Ω	0805
U1	LF33	DPAK TO-252

Pro zjištění funkcí vyvedených na jednotlivé konektory a případně jiné informace jsem připojil i schéma celé propojovací desky. Lze jej vidět na obr. 46. Obsahuje v přehledné formě popis vedení jednotlivých signálů na header - konektory a zapojení napájecího bloku.

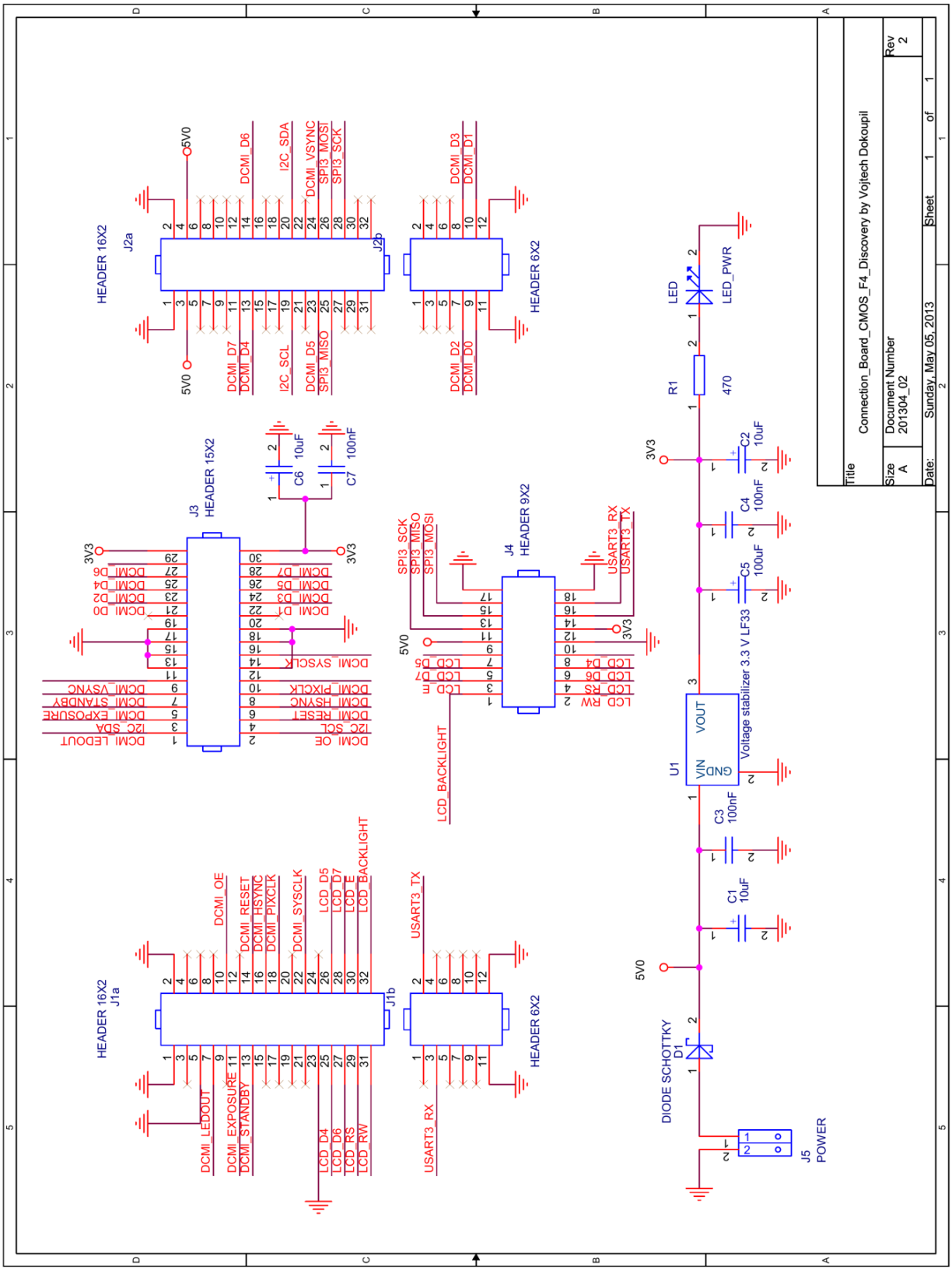
UPOZORNĚNÍ:

Napájecí napětí přivedené na konektor J5 musí být 5 V, vyšší napětí by mohlo způsobit destrukci Discovery kitu a ostatních 5 V periferií, které mohou být připojeny!

DOPORUČENÍ pro osazování:

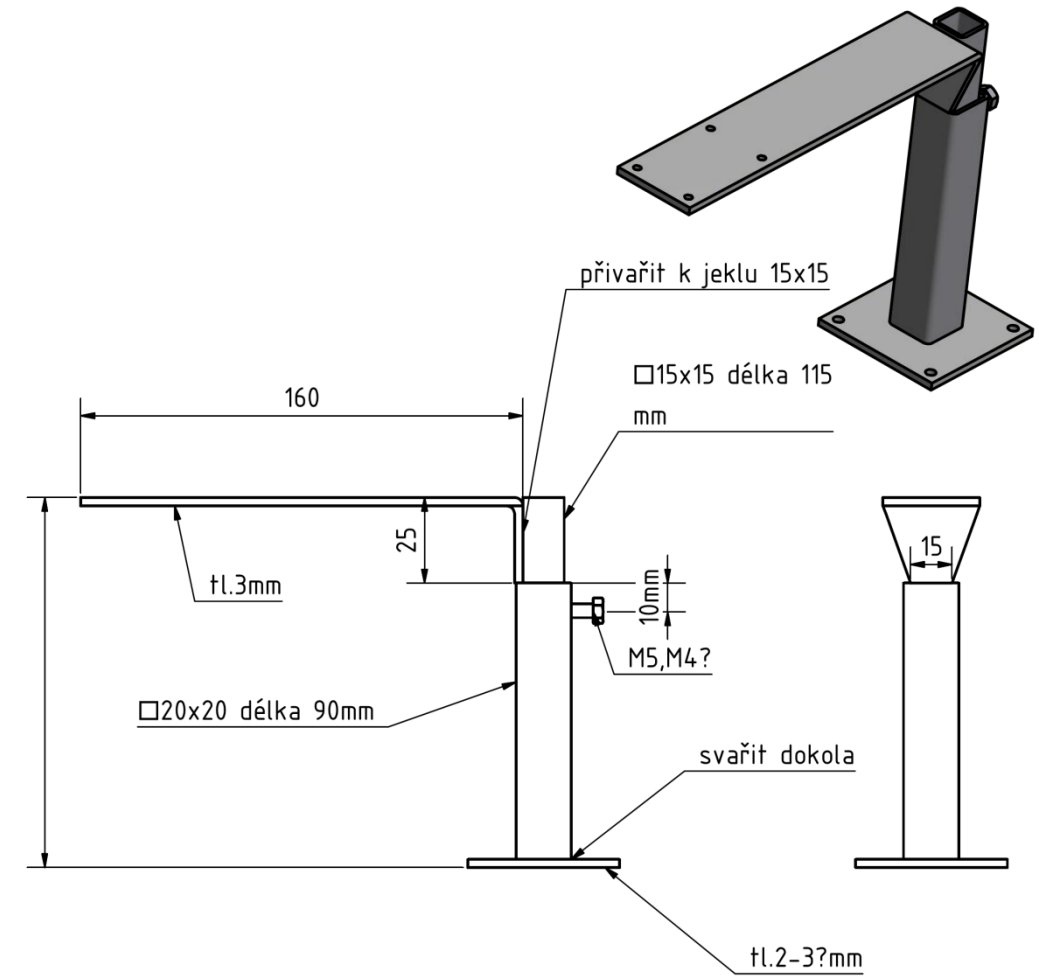
Doporučuji nejprve osadit stabilizátor napětí U1, jelikož jeho velká pájecí plocha potřebuje velké množství tepla na zahřátí a rozpuštění cínu. To by mohlo způsobit přemístění již připájených drobných součástek. Dále doporučuji osadit konektory J3 a J4, protože se pájí z druhé strany a nehrozí pak odpadnutí součástek ve vrstvě TOP v případě dlouhého pájení těchto konektorů. Poslední věc, která je vhodná, je si konektory J1A, J1B, J2A, J2B před

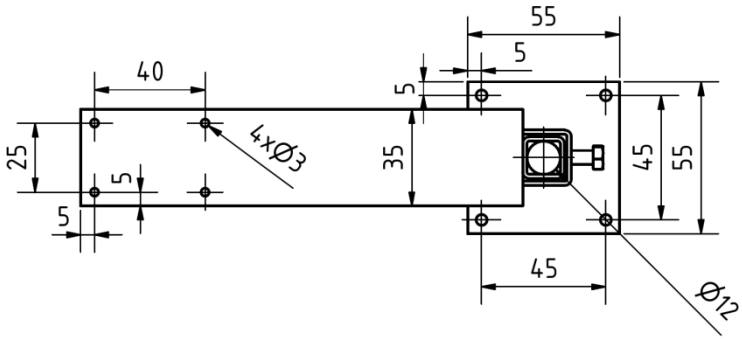
napájením sesadit s konkrétním Discovery kitem, na který se poté bude deska připojovat. Eliminujete tak možnost mechanického napětí a povyskakování konektorů.



Obr. 46 - Schéma PCB

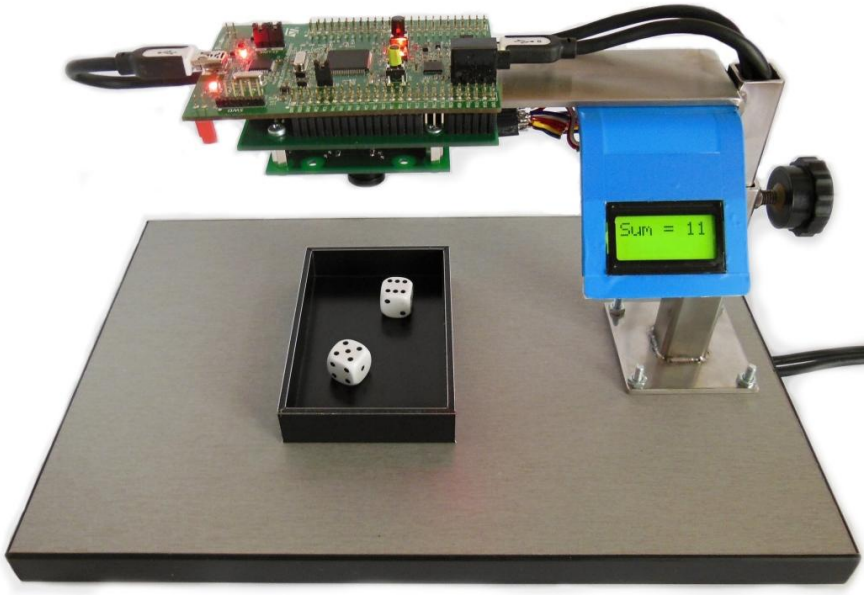
Příloha 2 – Univerzální polohovatelný držák - finální verze



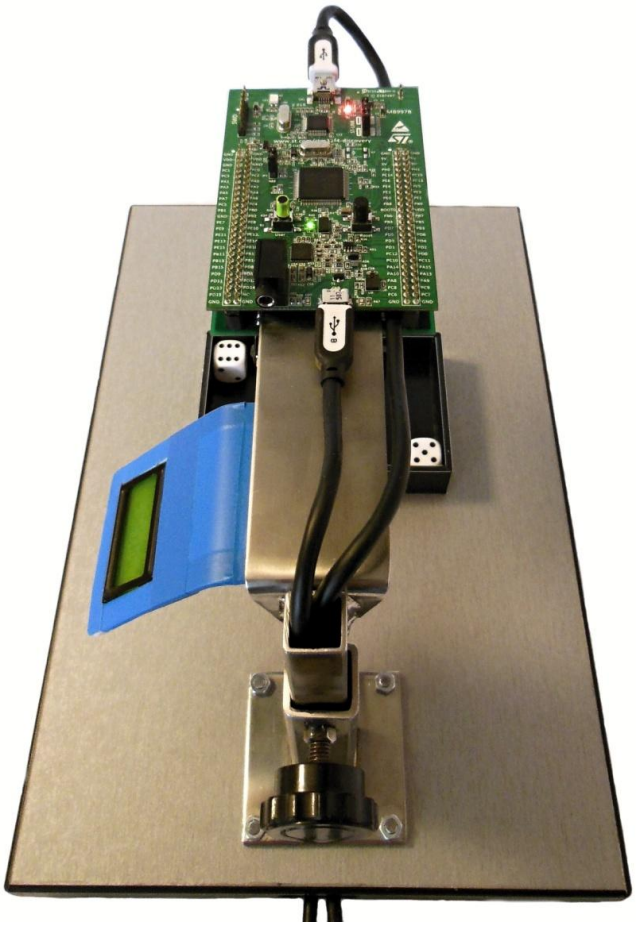


Obr. 47 - Výkres finálního držáku senzoru

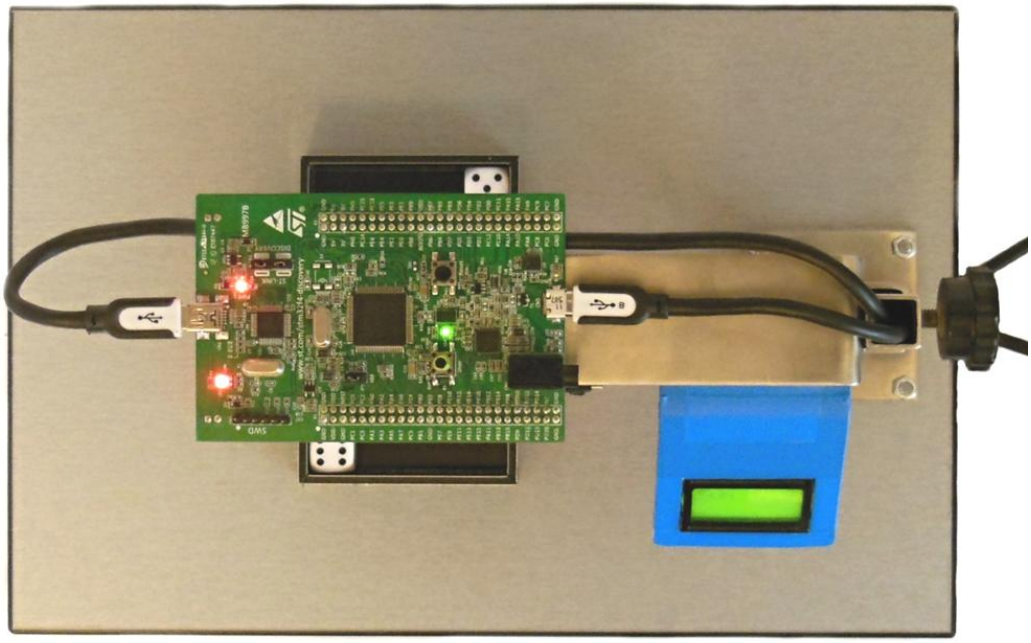
Příloha 3 – Fotodokumentace - demonstrační systém



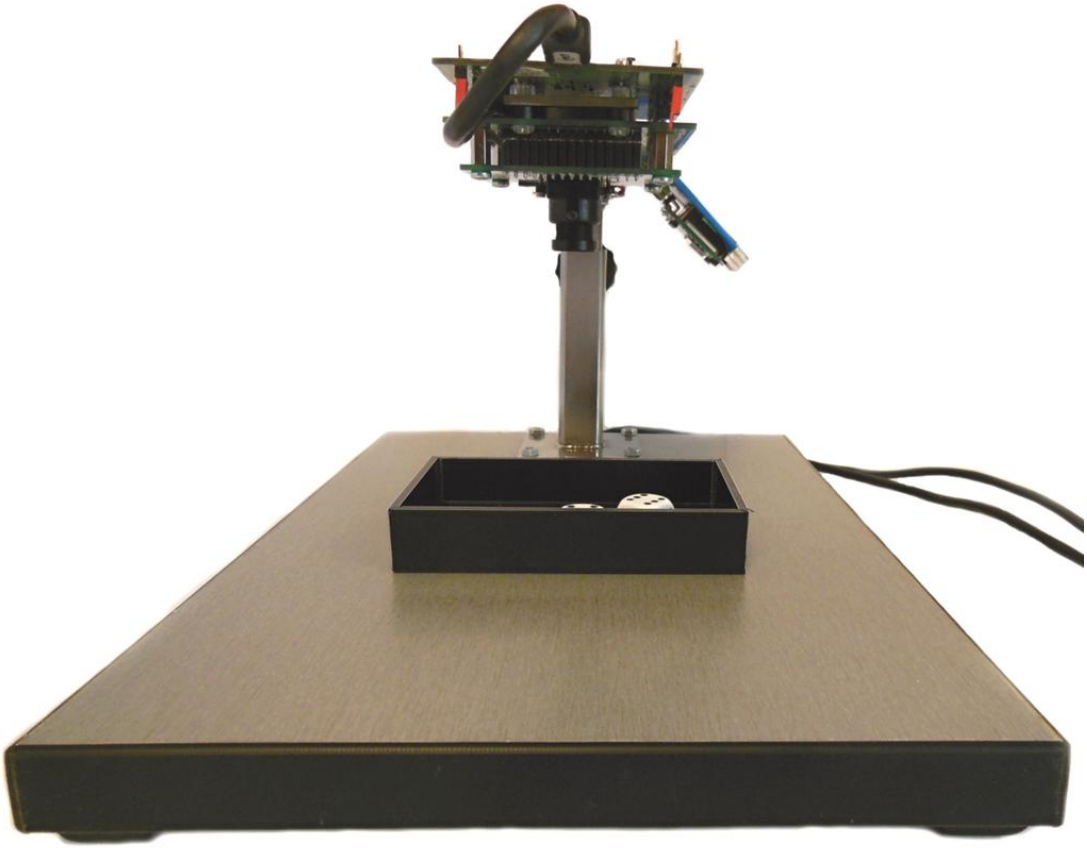
Obr. 48 - Boční pohled na demonstrační systém SUMONDI



Obr. 49 - Zadní pohled na demonstrační systém SUMONDI



Obr. 50 - Horní pohled na demonstrační systém SUMONDI



Obr. 51 - Přední pohled na demonstrační systém SUMONDI

Příloha 4 – Obsah přiloženého CD