TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: B2612 – Elektrotechnika a informatika
Studijní obor: 1234R567 – Název studijního oboru

Modul pro vyhodnocování parametrů laserového svazku kamerou pro LabView

Module parameters for evaluation of the laser beam camera for LabView

Bakalářská práce

Autor: **Jakub Šmejkal**

Vedoucí práce: Ing. Lenka Kretschmerová Ph.D.

Konzultant: Jan Hřebíček

V Liberci 2012

PROHLÁŠENÍ

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

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

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

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

Datum

Podpis

PODĚKOVÁNÍ

Rád bych poděkoval vedoucí práce Ing. Lence Kretschmerové Ph.D. V první řadě za zprostředkování tohoto velmi zajímavého tématu a dále za cenné náměty, rady a především usměrnění mé práce během uplynulých semestrů. Dále bych rád poděkoval panu Ing. Jiřímu Horčičkovi z laboratoře počítačového zpracování obrazu za pomoc při pronikání do problematiky, zejména v počátcích.

ABSTRAKT

Tato práce seznamuje s použitím počítačového zpracování obrazu. V první části je vytvořen přehled, co to jsou kamerové systémy strojového vidění. Seznamuje se strukturou a stručně popisuje některé části těchto systémů. Jsou zde uvedeny základní technologie obrazových snímačů, vysvětlen pojem elektronické závěrky a zmíněny možnosti komunikačního rozhraní. Další část je věnována konkrétnímu typu kamery, kterou používám. Zde jsou uvedeny základní parametry a možnosti kamery. V následující části je stručně popsáno vývojové prostředí LabVIEW. A poslední část je věnována popisu programu. Zde se zabývám strukturou programu, popisem uživatelského rozhraní a možnostmi aplikace. Nechybí ani popis mého postupu při vývoji aplikace a zmínění některých problémů na které jsem během práce narazil.

ABSTRACT

This thesis addresses the use of computer image processing. The first part describes camera systems of machine vision. The structure of these systems is explained and some of their parts are briefly described. Basic image sensor technologies are listed, followed by an explanation of the term „electronic shutter“. Communication interface options are mentioned. The second part concerns the specific type of camera that has been used while creating this thesis. Basic parameters and options of the camera are listed. The next part briefly describes the LabVIEW development environment. The final part addresses the program itself. The structure of the program, user interface and various option of the application are all described. The development process is also mentioned, as well as certain problems I have encountered during the task.

OBSAH

Seznam obrázků ... 9
Seznam symbolů, zkratek a termínů ... 10
1 Kamerové systémy strojového vidění ... 11
1.1 Snímač obrazu ... 11
1.1.1 CCD ... 11
1.1.2 CMOS (APS) ... 12
1.1.3 Hot Pixels ... 12
1.2 Závěrka a rychlost snímání ... 13
1.3 Komunikační rozhraní ... 13
1.3.1 FireWire (IEEE 1394) ... 13
1.3.2 GigE (IEEE 802.3ab) [1] ... 14
1.3.3 USB ... 14
2 Kamera uEye 154XLE-M ... 16
2.1 Typ uEye UI-154XLE-M ... 16
2.2 Možnosti provozu (Operating modes) ... 17
3 LabVIEW ... 19
3.1 Dataflow programming ... 20
3.2 Vision Development module (VDM) ... 20
3.3 Vytváření aplikace ... 21
4 Program ... 22
4.1 Vision Acquisition Software ... 22
4.2 Struktura programu ... 23
4.3 Uživatelské rozhraní ... 24
4.4 Alokace paměti a počátečních podmínek ... 26
4.5 Nastavení parametrů kamery ... 27
4.6 Zobrazení snímku a obsluha výřezu ... 28
4.7 Ukládání snímku ... 29
4.8 Centroid ... 32
4.9 Výpočet šířky svazku ... 32
4.9.1 Definice šířky svazku ... 34

4.9.2 Vytvoření vlastního typu sdílené proměnné 35
Závěr 36
Použitá literatura 37

SEZNAM OBRÁZKŮ

Obr. 1 - Kamera uEye 1548XLE-M 16
Obr. 2 - Live mode 17
Obr. 3 - Snap Mode 17
Obr. 4 - Software trigger 18
Obr. 5 - Hardware trigger 18
Obr. 6 - Načtení obrazu ze souboru - IMAQ 21
Obr. 7 - Struktura programu 23
Obr. 8 - Blokový diagram - LabVIEW 24
Obr. 9 - Uživatelské rozhraní - Hlavní panel 24
Obr. 10 - Uživatelské rozhraní – řez osou Y a šířka svazku 25
Obr. 11 - Alokace paměti a inicializace počátečních podmínek 26
Obr. 12 - Nastavení parametrů kamery 27
Obr. 13 - Nastavení doby expozice 27
Obr. 14 - Zobrazení snímku z kamery 28
Obr. 15 - Zobrazení výřezu 29
Obr. 16 - Menu pro ukládání snímků 29
Obr. 17 - Výzva k určení složky pro ukládání snímků 30
Obr. 18 - Uložení snímku pomocí dialogového okna 31
Obr. 19 - Uložení snímku do připravené složky 31
Obr. 20 - Určení polohy centroidu a vyznačení do obrazu 32
Obr. 21 - Vytvoření jednorozměrného pole (poloha bodu na přímce, intenzita) 33
Obr. 22 - Sekvence pro otevření VI z jiného VI 33
Obr. 23 - Zobrazení grafu intenzity v ose y a zobrazení šířky svazku 34

SEZNAM SYMBOLŮ, ZKRATEK A TERMÍNŮ

LabVIEW - Laboratory Virtual Instrumentation Engineering Workbench

CCD - Charged Coupled Device

CMOS - Complementary Metal–Oxide–Semiconductor

USB – Universal Seriál Bus – Univerzální Seriová Sběrnice

VI – Virtual Instrument – program v prostředí LabVIEW

VDM – Vision Development Module – knihovny pro zpracování obrazu

VAS – Vision Acquisition Software – univerzální ovladač periferií

FPS – Frames per Second – počet snímků za sekundu

1 KAMEROVÉ SYSTÉMY STROJOVÉHO VIDĚNÍ

Kamera je přístroj, jenž vytvářející dvojrozměrný jasový obraz trojrozměrné reality, poskytující své výstupy dnes již výhradně v digitální formě. Vždy v kameře nalezneme následující části: snímač obrazu, obvody pro obsluhu snímače obrazu a interface. Některé kamery digitalizují obraz už přímo v kameře (uEye u1548-LE) [1] [2] [3].

1.1 SNÍMAČ OBRAZU

Snímač obrazu (obrazový senzor) je tvořen maticí světlocitlivých buněk, skládajících se v podstatě z fotodiody a kondenzátoru. Po dobu vystavení buňky osvětlení se v kondenzátoru akumuluje elektrický náboj, který je tomuto osvětlení úměrný. Výrobní postup a způsob sběru náboje z buněk určují typ a vlastnosti snímače obrazu. V dnešní době jsou nejrozšířenější dvě technologie výroby těchto snímačů a to CCD a CMOS.

1.1.1 CCD

CCD čip funguje jako analogový posuvný registr. Každý kondenzátor předává svůj nahromaděný náboj do sousedního kondenzátoru. Poslední je pak připojen na výstupní zesilovací prvek, který náboj převádí na elektrické napětí. To je pak pomocí A/D převodníku je toto převedeno do digitální podoby. U kamer pro strojové vidění se dnes již nepoužívá prokládané řádkování (interlaced), známé z televizní normy. Současná technologie progressive scan „vysouvá“ akumulovaný náboj v buňkách všech obrazových řádků za sebou. Mezi výhody CCD lze zmínit velkou citlivost na osvětlení (kvalitní obraz) a malý šum (jehož hodnota je stejná v celém snímku). Nevýhodami jsou např. menší rozsah intenzit, vzájemné ovlivňování sousedních pixelů, nemožnost adresovat jednotlivé pixely. Snímače CCD jsou také poměrně nákladné a navíc k činnosti potřebují několik různých napájecích napětí. Kromě toho je nutné jejich videosignál digitalizovat v následných obvodech, v současné době většinou vyrobených technologií CMOS.

1.1.2 CMOS (APS)

Objevila se myšlenka integrovat snímač obrazu a A/D převodník na jednom čipu vyrobeném technologií CMOS. Oproti CCD technologii spočívá rozdíl v tom, že jednotlivé buňky snímače mají každá vlastní obvody pro odvedení a měření naakumulovaného náboje. Snímače CMOS tedy poskytují již digitalizovaný signál, což umožňuje zjednodušit konstrukci kamery. Odlišný způsob vybírání náboje z obrazových buněk také dovoluje vybírat obraz jen z určité části plochy snímače. CMOS čipy oproti CCD mají menší spotřebu energie (až 10x). A s nízkou spotřebou souvisí i menší zahřívání čipu. CMOS má jednodušší výrobní postup, umožňuje vyrábět vice kusů najednou, s čímž souvisí levnější výrobní cena. Další výhodou je velká rychlost vyčítání a větší rozsah intenzit (oproti CCD cca 2x). Nevýhodami jsou větší šum a menší citlivost. Tento typ kamer se díky možnosti integrace na jeden čip společně s procesorem používá u malých kamer pro embedded systémy, v průmyslových kamerách apod.

1.1.3 HOT PIXELS

Hot pixel (nebo v širším slova smyslu, vadný pixel), je pixel, který nereaguje lineárně vzhledem k „množství“ dopadajícího světla, nebo nereaguje vůbec. (reakcí rozumíme, že buňka snímače vykazuje náboj). Vyskytují se z různých důvodů, např. znečištěním během výroby snímače nebo vlivem stáří snímače (jak u CCD tak CMOS). Při stejných provozních podmínkách mají CCD snímače oproti CMOS obvykle méně hot pixelů. S tmavou scénou a dlouhou dobou expozice jsou hot pixely na snímku vidět jako jasné body. Tento jev je podporován dlouhou dobou expozice a vysokou pracovní teplotou snímače.

Při výrobě se snímače testují právě na tmavých snímcích s dlouhým časem. Pixely s jasem nad určitou hodnotou jsou klasifikovány jako hot pixels. Seznam souřadnic jednotlivých hot pixelů je uložen v paměti kamery a opravu této vady provádí ovladač kamery.

1.2 ZÁVĚRKA A RYCHLOST SNÍMÁNÍ

Elektronická závěrka na rozdíl od mechanické závěrky neřídí množství světla, které má během snímku dopadnout na snímač (mechanické závěrky se používají u DSLR). Snímač je osvětlen nepřetržitě a elektronická závěrka pouze řídí domu akumulace náboje. Snímače CCD zaručují jednotnou dobu akumulace všech buněk snímače díky synchronnímu presunu akumulovaného náboje do analogového posuvného registru. Elektronická závěrka CCD je vždy typu *global shutter* (centrální závěrka). U standardního snímače CMOS, kde je náboj z buňky vybírán postupně pomocí adresace buněk , je v principu každý řádek exponován v jiném okamžiku. Tím vzniká efekt zvaný *rolling shutter* (štěrbinová závěrka). Důsledkem toho je, že při pohybu objektu v době expozice se původně svislé hrany mění v šikmé. Moderní snímače CMOS již také využívají centrální závěrku, ale jsou dražší vzhledem ke složitější technologii výroby.

Elektronické závěrky dovolují používat i velmi krátké doby expozice. Mohou tedy dosahovat velký rychlostí snímaní (počet snímků za sekundu). Problémem je však velký tok dat, proto je pro rychlost snímání kamery rozhodující rychlost komunikačního rozhraní.

1.3 KOMUNIKAČNÍ ROZHRANÍ

Toto rozhraní zprostředkovává přenos obrazových dat mezi kamerou a systémem. Rozhraní, která kamery používají, vycházejí většinou z komerční rozhraní používaných u osobních počítačů.

1.3.1 FIREWIRE (IEEE 1394)

Je standard sériové sběrnice pro připojení periferií k počítači zavedený firmou Apple v roce 1995. Rozhraní je dnes především rozšířeno v komerčních videokamerách, v průmyslu je pozvolna nahrazováno vysokorychlostními modifikacemi USB.

1.3.2 GIGE (IEEE 802.3AB) [1]

Pracuje na stejných principech jako standardní Ethernet. Pro přenos obrazových dat byl vyvinut široce akceptovaný přenosový standard GigE Vision. Začátek vývojových prací na protokolu GigE Vision se datuje na začátek roku 2004. Podílelo se na nich mnoho předních výrobců kamer a společností zabývajících se digitálním zpracováním obrazu. Standard je spravován společností Automated Imaging Association. Protokol obsahuje:

- Detekci zařízení a přiřazování IP adres;
- Kontrolní protokol;
- Sdělovací kanál
- Vysoce výkonný datový tok

Pro detekci zařízení jsou kamery v síti LAN uloženy do seznamu. Vzhledem ke kompatibilitě s moduly Plug and Play oznámí kamery zanedlouho po připojení základní informace (výrobce, model atd.). Zařízením GigE Vision je přidělena IP adresa běžným postupem prostřednictvím ethernetového protokolu DHCP. Kontrolní protokol má dva základní úkoly: číst a zapisovat registry v kameře a implementovat mechanismus, který kontroluje přístup ke kameře a monitoruje spojení mezi aplikací a kamerou. Sdělovací kanál umožňuje asynchronně posílat zprávy od kamery hostovi. Protokol pro tok dat umožňuje rychlý přenos dat (nejen obrazových) z kamery k hostovi. Ve spojení kontrolním protokolem umožňuje mechanismus znovuzaslání paketů získat ztracené pakety. Mapa samozaváděcího registru (bootstrap map) definuje pouze funkce a adresy registrů, které jsou pro komunikaci důležité (IP adresy, datový tok). Navíc jsou zde uvedeny základní informace o zařízení.

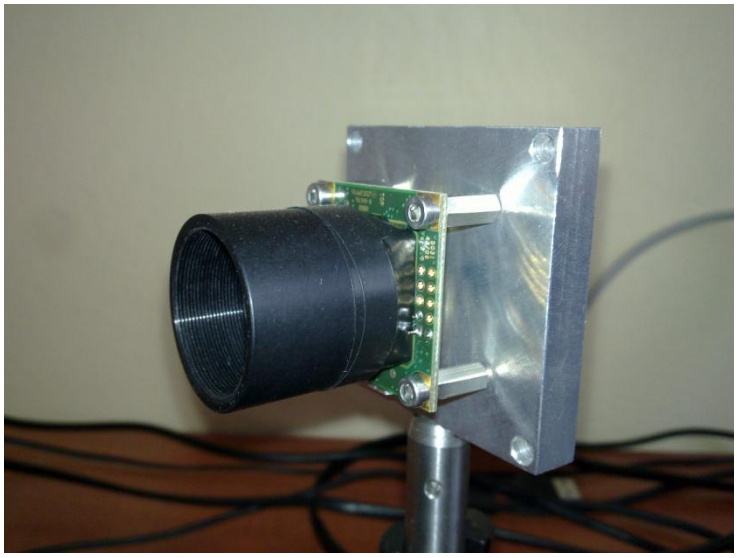
1.3.3 USB

V současné době nahrazuje všechny starší porty používané u PC (COM, LPT, GamePort, PS/2), ale navíc umožňuje snadno připojovat další zařízení (diskové jednotky, CD/DVD mechaniky, fotoaparáty apod.). Ale využití USB pro přenos obrazových dat bylo možné až po zavedení standardu USB 2 s přenosovou rychlostí 480 Mb/s.

Základem je jednoduchost pro uživatele – zařízení lze u USB i napájet (5V, max. 500mA) a je zde implementováno Plug & Play. První specifikace tohoto standardu je objevili v roce 1995 (definováno ve spolupráci Compaq, HP, Intel, NEC, Microsoft a Philips). Počítač je jediný master, tudíž zařízení nemůže posílat data samo. Jsou definované speciální protokoly pro určité druhy komunikace (např. Audio, Video, Printer, Human Interface Device, Mass Storage), nebo lze přenášet čistá data a vytvořit vlastní ovladač. Přenos probíhá v rámcích s délkou 1ms a uvnitř rámce mohou být data pro více zařízení. Pro připojení více zařízení je nutný HUB (až 127 zařízení)-

Nevýhodou je maximální délka kabelu, který by pro garantovanou funkci neměl být delší než 5m. Problémem někdy bývá i kompatibilita softwaru PC s řadičem USB v kameře. Přesto se rozhraní používá pro průmyslové kamery stále častěji a tento trend určitě zesílí po rozšíření standardu USB 3, který startuje v letošním roce.

2 KAMERA UEYE 154XLE-M



Obr. 1 - Kamera uEye 1548XLE-M

uEye je označení pro kamery firmy IDS, která se specializuje na výrobu profesionálních průmyslových digitálních kamer. Jsou vybaveny široce používaným rozhraním USB a Ethernet portem, je tedy možné je snadno propojit s velkým množstvím nejrůznějších systémů. Snímky jsou digitalizovány přímo v kameře a poté přenášeny do počítače. Další zpracování snímku již není nutné. Kamery jsou vybaveny buďto CCD nebo CMOS snímači. Rozlišení kamer se pohybuje od 640x480 pixelů (VGA) do 2560x1920 pixelů (QSXGA), v závislosti na použitém snímači.

2.1 TYP UEYE UI-154XLE-M

Jedná se o kameru ze série LE (light edition), vyznačují se extrémně malými rozměry a vysokorychlostním snímačem typu CMOS s plochou 6,4 x 4,8 mm (rozlišení 1280 x 1024; 1,3Mpx). Kamera je monochromatická s 8 bitovou hloubkou.

Kamera bude pracovat ve vyčerpaném prostředí. A tento typ kamery nemá v tomto prostředí problém s přehříváním.

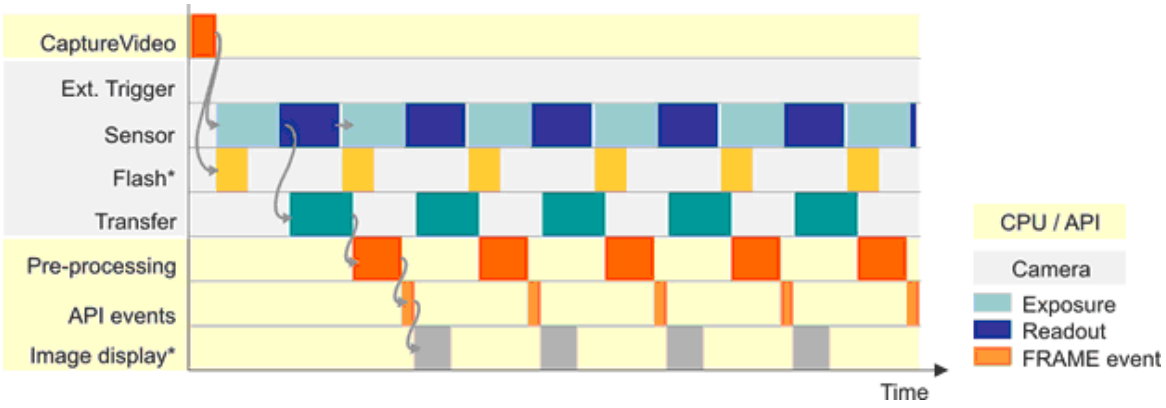
2.2 MOŽNOSTI PROVOZU (OPERATING MODES)

Kamery podporují dva režimy provozu (free, trigger). V režimu trigger i free je možnost použití výstupní funkce pro blesk, touto funkcí se ovšem zabývat nebudu.

Freerun režim

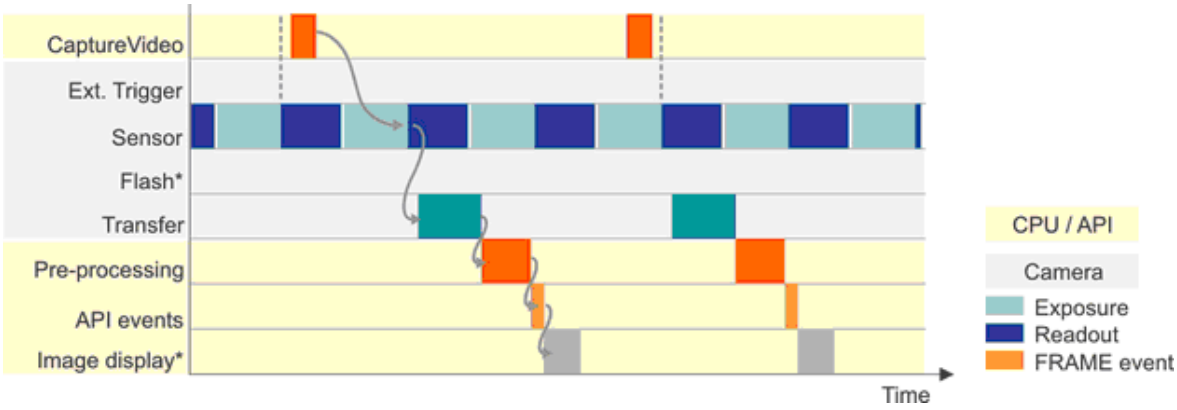
V tomto režimu kamera snímá jeden snímek za druhým dle nastavení počtu snímků za vteřinu. Expozice a zpracování/odesílání obrazových dat probíhá souběžně (paralelně). Počet snímků za sekundu a dobu expozice lze nastavovat samostatně. Pořízené snímky mohou být do počítače přenášeny jeden po druhém nebo průběžně. Pokud je aktivovaný trigger, je potřeba jej vypnout před aktivací režimu freerun.

Live mode (continuous mode)
Snímky jsou pořizovány a odesílány průběžně.



Obr. 2 - Live mode

Snap mode (snímkový režim)



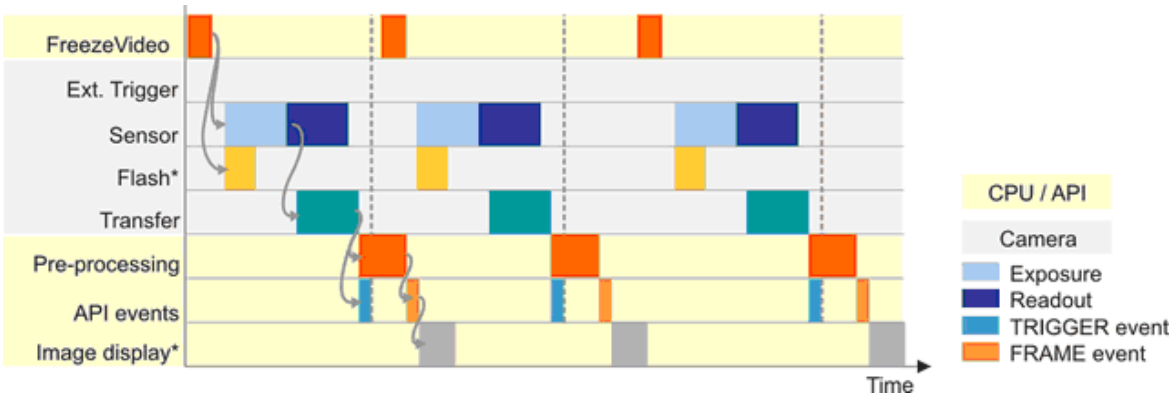
Obr. 3 - Snap Mode

Trigger režim (spouštění)

V módu trigger, je snímač ve stavu standby a je připraven na příjem spouštěcího signálu. Samotné spuštění může být iniciováno softwarovým příkazem (software trigger) nebo elektrickým signálem do vstupu kamery (hardware trigger).

Software trigger

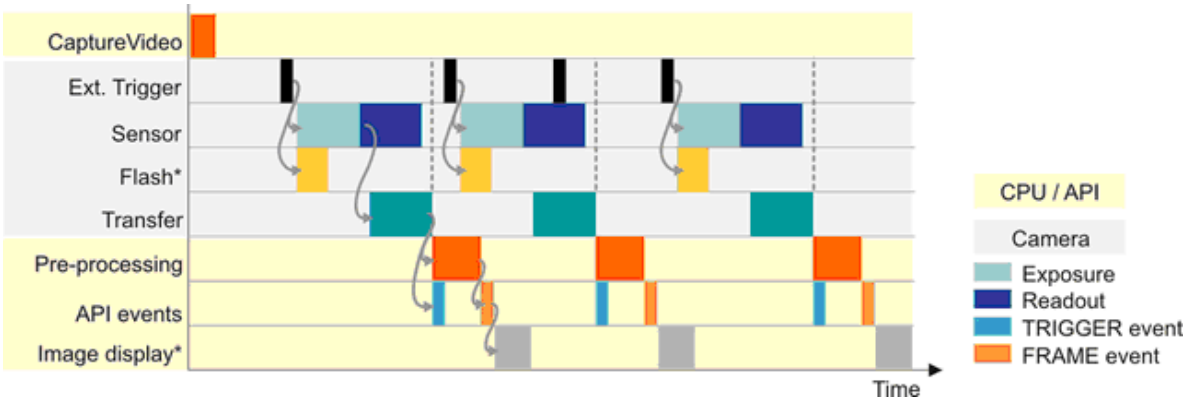
Když je tento mód aktivní, volání funkce snap spouští zachycení obrazu, který je pak přenesen do PC. Volání funkce live jsou snímky pořizovány a přenášeny průběžně.



Obr. 4 - Software trigger

Hardware trigger

Volání snap funkce připraví kameru k pořízení jednoho snímku. Po obdržení externího signálu kamera zachytí jeden snímek a ten pošle. Je- li volána funkce live obraz je zachycen a poslán při každém přijetí elektrického signálu.



Obr. 5 - Hardware trigger

3 LABVIEW

LabVIEW (z anglického Laboratory Virtual Instrumentation Engineering Workbench) je vývojové prostředí pro tvorbu měřících, testovacích a řídicích systémů od firmy National Instruments). Prostředí používá grafický programovací jazyk (G – language), ve kterém propojujeme pomocí „drátů“ jednotlivé bloky (kterými jsou reprezentovány jednotlivé funkce či procedury), podobně jako ve vývojových diagramech. Dnes, je již možné v LabVIEW kombinovat grafický jazyk, jazyk C a MathScript (MATLAB). Vývojové prostředí má dvě hlavní části (okna): Block Diagram a Front Panel. První slouží k samotnému programování, ve formě vývojového „grafu“. Každá komponenta typu vstup/výstup má svou grafickou podobu, právě na zmíněném Front Panelu. Mohou to být různá tlačítka, přepínače, editační okna či třeba nějaký grafický výstup ve formě grafu. Front Panel tedy představuje jakýsi virtuální panel připomínající čelní panel skutečného přístroje. Programy vytvořené v LabVIEW se nazývají Virtual Instruments (VIs). Právě proto, že napodobují svou koncepcí skutečné přístroje. LabVIEW obsahuje množství nejrůznějších knihoven obsahujících velké množství funkcí sběr dat, generování signálů, zpracování signálů, matematické a statistické funkce atd.. Ale největší síla tkví v obrovském množství hardwaru, se kterým je LabVIEW schopno komunikovat.

Grafický programovací jazyk, obsahuje stejné principy, jako většina ostatních (textových) programovacích jazyků. Obsahuje všechny standardní konstrukce jako datové typy, proměnné, smyčky, zpracování událostí, objektově orientované programování atd.

3.1 DATAFLOW PROGRAMMING

Způsob běhu programu je určen strukturou grafického blokového diagramu kde programátor zapojuje jednotlivé funkce-bloky pomocí tažení „drátů“ (wires). Tyto „dráty“ představují proměnné a každý blok se může spustit, jakmile má data na všech vstupech. Jelikož k tomu může dojít u více bloků současně, je G-jazyk samozřejmě schopen paralelního zpracování. Vestavěný plánovač automaticky využívá hardware pro multi-processing a multi-threading.

Program vytvořený pomocí LabVIEW tedy vykonává svoji funkci podle pravidel „toku dat“ místo aby uplatňoval tradičnější procedurální přístup (série jednotlivých příkazů, které budou vykonány), který používá většina programovacích jazyků založených na textovém prostředí, jako C a C++. Jazyky využívající dataflow, (G-jazyk, stejně jako Agilent VEE, Microsoft Visual Programming Language a Apple Quartz Composer) považují data za hlavní koncept každého programu. Dataflow vykonávání je daty řízeno nebo ovlivněno. Pořadí vykonávání příkazů je řízeno tokem dat mezi bloky, nikoliv pořadím řádků v textu.

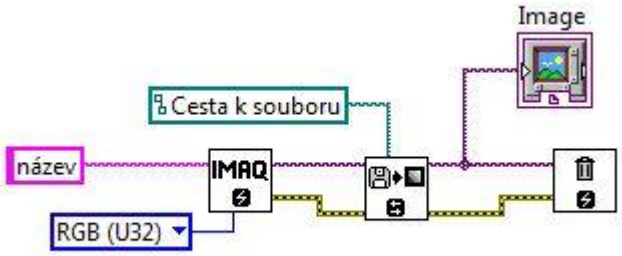
3.2 VISION DEVELOPMENT MODULE (VDM)

Je soubor funkcí a nástrojů pro vývoj aplikací strojového vidění a zpracování obrazu. Je k dispozici pro LabVIEW, C/C++, Visual Basic a .NET. Knihovna zavádí obrazový datový typ *IMAQ Image*, který lze zobrazit pomocí kontrolu *Image Display*, který knihovna zavádí také.

Knihovna je rozdělena do tří palet funkcí: Vision Utilities, Image Processing a Machine Vision. Paleta Vision Utilities umožňuje manipulaci a zobrazení obrazu. Image Processing poskytuje funkce analýze, filtrování a zpracování obrazu. Machine Vision je paleta vysokoúrovňových VI, které umožňují řešit základní úlohy strojového vidění: sledování a analýza pohybu částic, detekce hran, klasifikace objektů, měření barevných odstínů, prostorová kalibrace, měření rozměrů (vzdálenosti, úhly, průměry, obsahy atd.).

3.3 VYTVÁŘENÍ APLIKACE

Nejprve je třeba vytvořit obraz jako takový, což je v podstatě deklarace proměnné IMAQ Image. Zde je hlavně potřeba definovat typ obrazu (jestli se jedná o barevný či černobílý obraz a jaká je jeho bitová hloubka). Poté můžeme do připravené „proměnné“ vložit obraz, ten získáme buďto z nějakého záznamového zařízení (kamera) nebo ze souboru. Dále už je možné obraz zobrazit, upravovat, ukládat do souboru atd. Takto by vypadal program pro načtení obrazu ze souboru a zobrazení na čelním panelu.



Obr. 6 - Načtení obrazu ze souboru - IMAQ

4 PROGRAM

Obdržel jsem kameru a ovladače pro LabVIEW od výrobce kamery. Začal jsem studovat jednotlivé VI, které knihovna obsahuje. Bohužel manuál je dost strohý a funkce bloků je popsána jen velmi stručně. Ovladače neobsahují nápovědu a taktéž nejsou přítomny ani příklady použití jednotlivých bloků. Jak jsem zjistil, tak výstupní bloky (GetSingeImage, GetActImage) neposkytují přímý obrazový výstup, ale pouze pole hodnot a je tedy třeba dalšího zpracování. Typicky pomocí *Reshape Array* kde se zadají rozměry obrazu v pixelech a blok tedy převede 1D pole na 2D, které už je možné zobrazit v grafu intenzit. Možností je samozřejmě víc. Začal jsem tedy pracovat s takovýmto polem hodnot. Pomocí transpozic se mi podařilo obraz otáčet o 90° a překlápět. Dále intenzitu překreslovat do barev, dle definovaných parametrů (úroveň = barva). Ale rychle jsem pochopil, že práce s polem hodnot je velmi pracná. Usoudil jsem, že budu potřebovat softwarový balík NI Vision Development Software pro LabVIEW. Jehož součástí je Vision Acquisition Software.

4.1 VISION ACQUISITION SOFTWARE

Vision Acquisition Software (VAS) je balík ovladačů, pomocí kterého se lze připojit k průmyslovým kamerám řady výrobců. Licence je v rámci VDM nebo ji lze pořídit i samostatně. VAS není závislý na kameře konkrétního výrobce a podporuje základní důležité standardy (Camera Link, USB, FireWire, GigE atd.). Pro testování kamery lze použít aplikaci NI Measurement & Automation Explorer (MAX), která je součástí instalace.

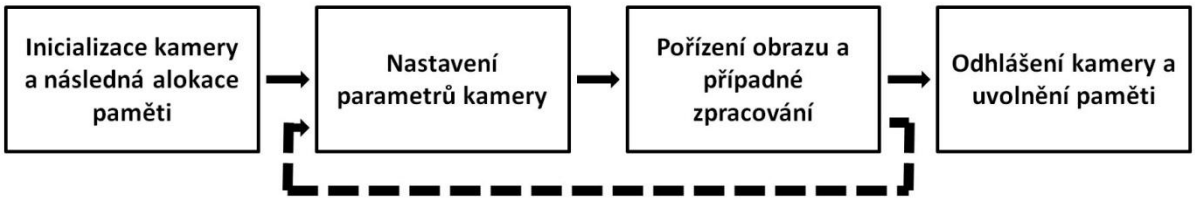
Tento balík obsahuje bloky pro spuštění kamery, pořizování snímků/videa, ukončení kamery. Dále poskytuje jednoduché nastavení parametrů kamery, parametrů výstupního obrazu a možnosti ovládání více kamer současně. Celek již pracuje s obrazovými daty IMAQ Image. Tudíž ideální by bylo využít Vision modul k získání snímků z kamery a následně jej využít ke zpracování.

Na internetových diskusích, kde se řeší problematika počítačového zpracování obrazu, jsem se dočetl, že některé kamery uEye připojené pomocí USB mají problém s komunikací přes VAS a na vině jsou právě zmíněné univerzální

ovladače. Bohužel typ kamery, který používám, patří mezi ty, které přes VAS nekomunikují.

4.2 STRUKTURA PROGRAMU

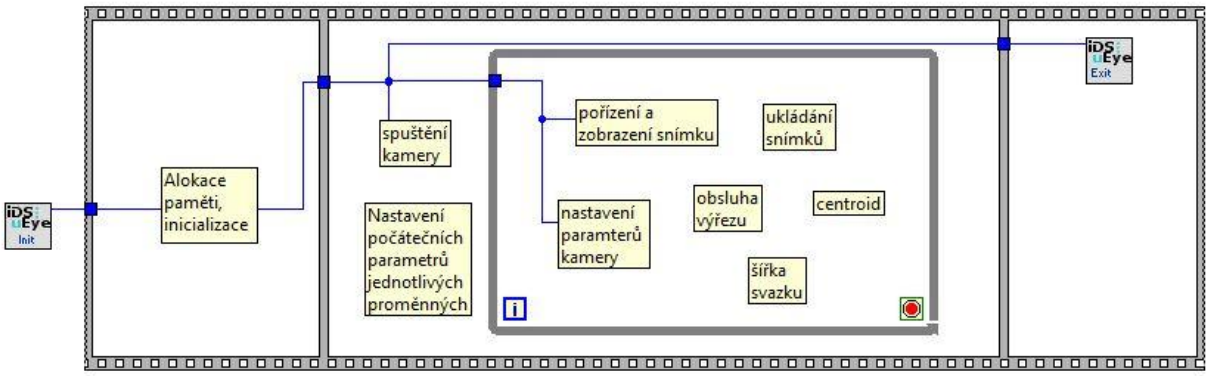
Jak vyplývá z předchozích poznatků, bylo potřeba zkombinovat ovladače dodané od výrobce a VDM. Struktura programu je následující:



Obr. 7 - Struktura programu

Řídící část programu, tj. inicializace, alokace, nastavení parametrů, pořízení snímku a ukončení kamery, je řešena pomocí VI (ovladačů) od výrobce. Po získání snímku jsou obrazová data převedena na typ IMAQ Image. K následnému zpracování a zobrazení obrazu jsou použity VI z knihovny Vision.

Tuto strukturu programu je třeba bezpodmínečně dodržet. Když jsem při vývoji aplikace potřeboval program přerušit dříve, než došlo k odhlášení kamery, nebylo již možné program znovu spustit, protože kamera zůstala v systému zavedená a blok, který kameru inicializuje, zahlásí chybu. Použít samostatně blok pro odhlášení kamery ze systému nelze použít, protože signál z VI s inicializací je potřebný pro fungovaní všech ostatních. Při přerušení programu je potřeba restartovat celé vývojové prostředí LabVIEW. Elegantnější řešení odebrání kamery z paměti se mi nalézt nepodařilo.

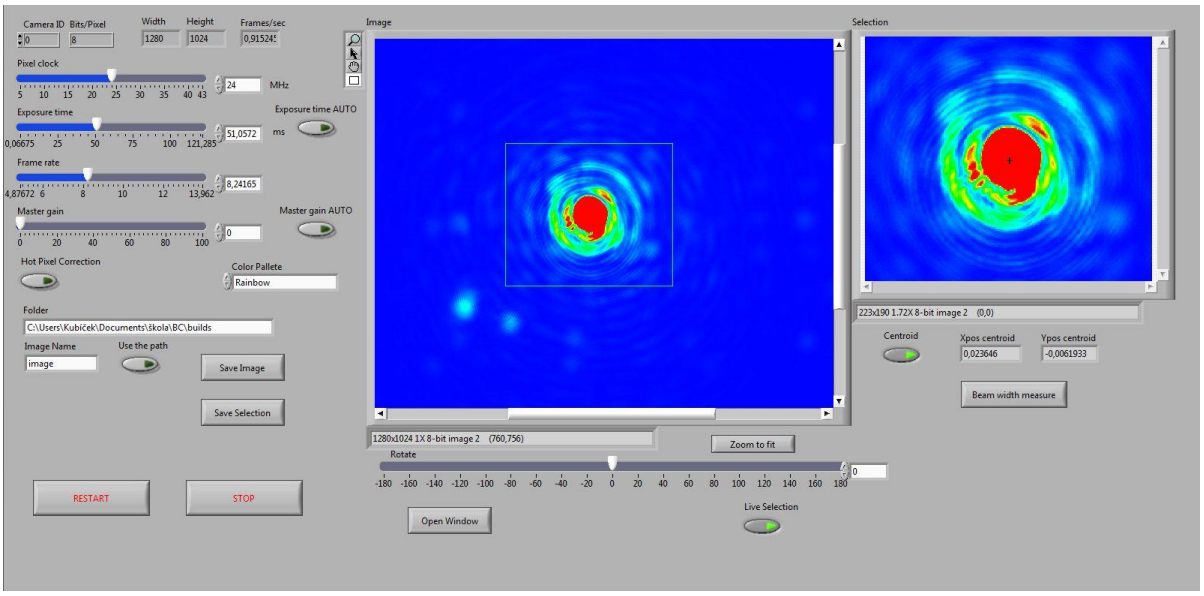


Obr. 8 - Blokový diagram - LabVIEW

Obr. 8 ukazuje, jak by vypadalo blokové schéma přiblížené do prostředí LabVIEW. Hlavní část programu je umístěna do smyčky *while*, jejíž perioda opakování je dána počtem snímků za vteřinu. Ukončením běhu smyčky se ukončí celý program. Před tuto smyčku je možné umístit deklarace a parametry použitých proměnných.

4.3 UŽIVATELSKÉ ROZHRANÍ

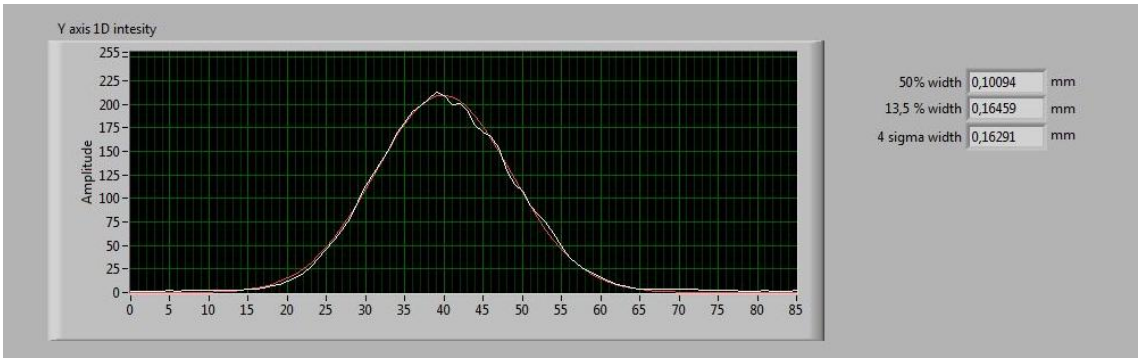
Úvodem bych rád upozornil, že konečná podoba uživatelského rozhraní velmi záleží na počítači, kde bude systém provozován, resp. na jeho zobrazovacích zařízeních (velikost, rozlišení displejů, jejich počet). Uvedená varianta je optimální pro rozlišení 1600x900.



Obr. 9 - Uživatelské rozhraní - Hlavní panel

V levém horním rohu jsou základní informace: číslo kamery, barevná hloubka obrazu (v bitech), šířka a výška obrazu (v pixelech), a reálná hodnota počtu snímků za vteřinu. Pod tímto info blokem se po levé straně nachází posuvníky, které umožňují nastavení parametrů kamery. Doba expozice, počet snímků za sekundu, zisk a frekvence vyčítání dat z kamery (pixel clock). Vedle posuvníků jsou editovatelná okna pro zadání či zobrazení přesné hodnoty. Dále směrem dolů je možnost volby barevné palety (pořízený snímek je černobílý) a možnosti ukládání snímku jak původního tak výřezu. Ve spodní části jsou tlačítka pro ukončení a restart programu.

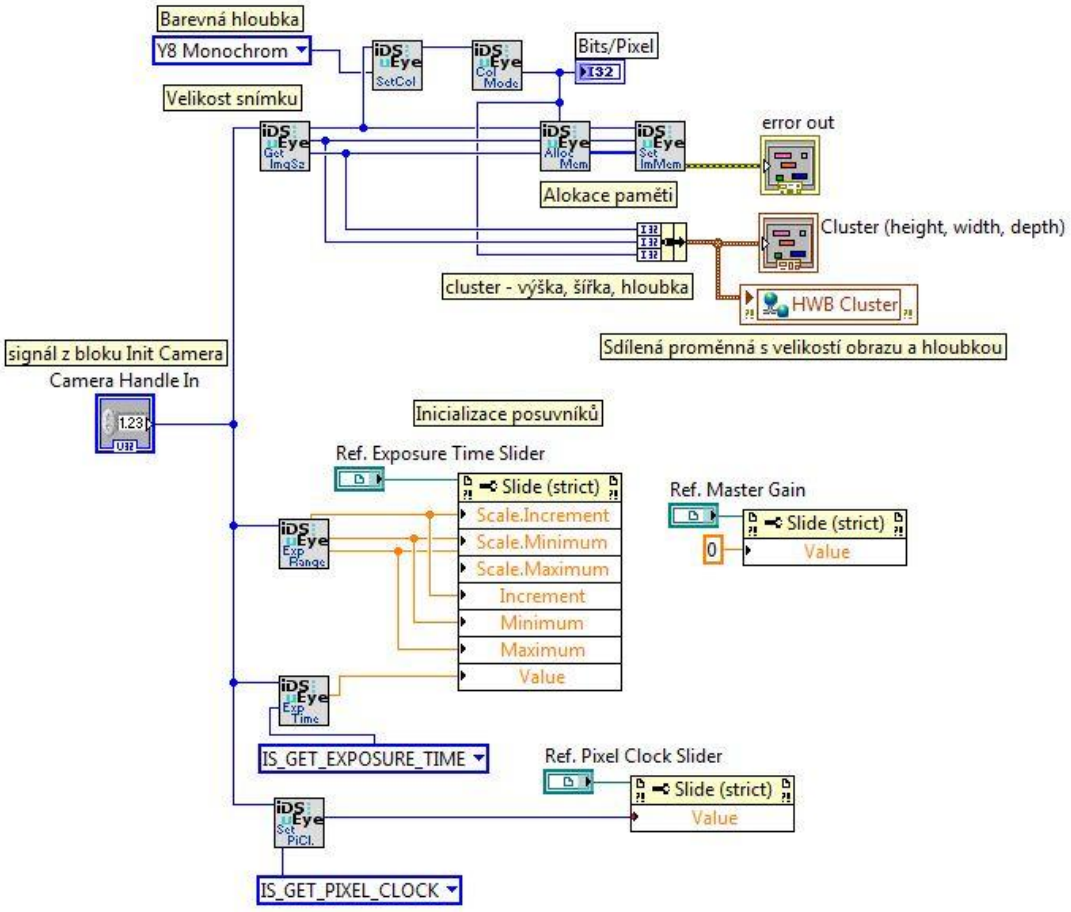
Zhruba uprostřed výchozí obrazovky je umístěn hlavní zobrazovací nástroj (panel Image), kde vidíme video z kamery, v levém horním rohu jsou nástroje (lupa, posun, výřez). Pod obrazem je okno se základními informacemi, např. hodnota intenzity v místě daném polohou kurzoru. Dále je zde umístěn posuvník, který umožňuje otáčení celého náhledu a tlačítko s funkcí oddálení obrazu (zoom to fit). Na pravé straně od hlavního náhledu je okno, kde se zobrazuje výřez (panel Selection), který můžeme definovat interaktivně na hlavním panelu. Náhled může být buďto živý, nebo pouze snímek. To zajišťuje přepínač umístěný ve spodní části (Live Selection). U okna výřezu je přepínač zajišťující výpočet polohy centroidu a jeho zobrazení do okna výřezu. Poloha se zobrazuje v milimetrech jako odchylka od středu snímače v osách x a y. A nakonec se zde nalézá tlačítko Beam width measure, které otvírá okno s měřením šířky svazku. Zobrazí se zde řez intenzitou v obou osách se středem v centroidu. Tímto průběhem se proloží ideální Gaussova křivka a určí se směrodatná odchylka průběhu. Šířka svazku se zobrazuje ve třech definovaných hodnotách intenzity (50%, 13,5%, D4σ). Výsledky se zobrazují v milimetrech.



Obr. 10 - Uživatelské rozhraní – řez osou Y a šířka svazku

4.4 Alokace paměti a počátečních podmínek

Jak již bylo zmíněno, program začíná blokem *Init Camera* a jeho výstupní veličina je potřeba pro fungování ostatních bloků (VI od výrobce). Vstup a výstup této veličiny na jednotlivých VI se nazývá *Camera Handle In/Out* a lze ji chápat jako jakousi deklaraci proměnné „kamera“. Dále v textu budu výraz Handle používat pro tuto proměnnou.



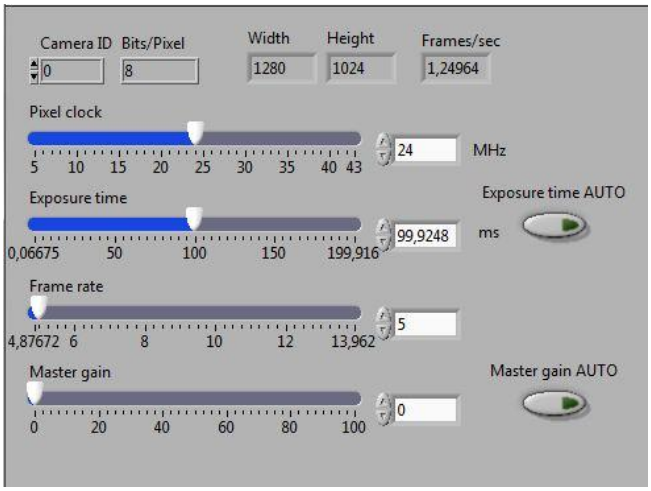
Obr. 11 - Alokace paměti a inicializace počátečních podmínek

Horní část blokového diagramu se stará o alokaci paměti. Handle je přiveden na blok *Get Image Size*, který určí šířku a výšku obrazu. Dále je zde blok *Set Color Mode* do kterého je přiveden údaj o barevném profilu kamery a blok *Get Color Mode*, který z předchozího signálu určí barevnou hloubku (v bitech na pixel). Údaje o šířce, výšce a hloubce jsou poté přivedeny do bloku *Alloc Mem*, který již provádí alokaci samotnou. Hodnotu barevné hloubky přivádím na indikátor. A z hodnot hloubka, šířka, výška vytvářím cluster, který mimo běžného výstupu definuji jako sdílenou proměnnou pro budoucí použití.

Dolní část definuje slider, který nastavuje expoziční dobu. Jeho počáteční hodnotu a hlavně rozsahy. Dále je zde nastavena defaultní hodnota frekvence vyčítání dat z kamery a nulová hodnota zisku.

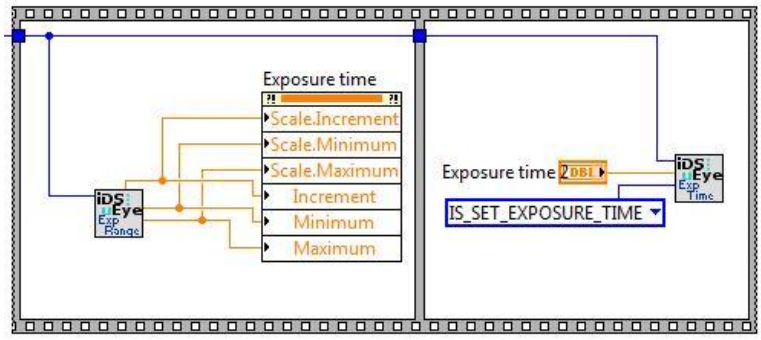
4.5 NASTAVENÍ PARAMETRŮ KAMERY

Nastavení parametrů, je možné buď prostřednictvím posuvníku, nebo zadáním hodnoty do okna u něj.



Obr. 12 - Nastavení parametrů kamery

Není však možné nastavit libovolnou kombinaci doby expozice, počtu snímků za vteřinu a pixel clock. Tyto veličiny se navzájem ovlivňují resp. ovlivňují své rozsahy hodnot. Programově je to řešeno bloky *Get Frame Rate, Get Exposure Time* atd., které vrací rozsah své veličiny vzhledem k veličinám ostatním. To se pak aplikuje jako meze jednotlivých posuvníků. Pokud byla původní hodnota mimo nový rozsah, posune se na bližší kraj platného intervalu. Po nastavení mezí se pomocí bloků *Set HW Gain, Set Exposure Time, Set FPS* atd. pošle aktuální hodnota posuvníku. Na obrázku uvádím jednoduchý příklad základního nastavení pro dobu expozice.

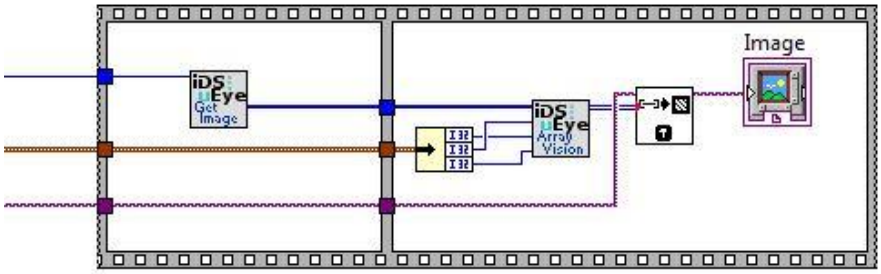


Obr. 13 - Nastavení doby expozice

Můj program umožňuje zvolit si automatické nastavení doby expozice a zisku. K tomu je potřeba pouze správně volaný blok *AutoParameter*. Ten má vstup *Enable* a *Parametr*. *Enable* je logický vstup a parametr je funkce která bude volána např. IS_SET_ENABLE_AUTO_GAIN. Pokud by v tomto případě byla na *Enable* vstupu logická 1, hodnota zisku se bude nastavovat automaticky. Zároveň je třeba slider *Master gain* změnit z ovladače na indikátor. V případě logické 0 opačně. To lze jednoduše zařídit vhodně použitou strukturou *case*.

4.6 ZOBRAZENÍ SNÍMKU A OBSLUHA VÝŘEZU

K zobrazení snímku jsem použil blok *GetActImage*, který vrací jednorozměrné pole hodnot a to jsou již hodnoty intenzit pro jednotlivé pixely. Tento výstup poté přivedeme na vstup bloku *ArrayToVision*, který převádí jednorozměrné pole na dvourozměrné dle vstupních parametrů (šířka, výška, hloubka). Toto pole již lze snadno pomocí *ArrayToImage* převést na proměnnou typu *IMAQ Image.ctl*, tedy pro potřeby modulu Vision. Následně už stačí snímek pouze zobrazit na vhodném indikátoru.

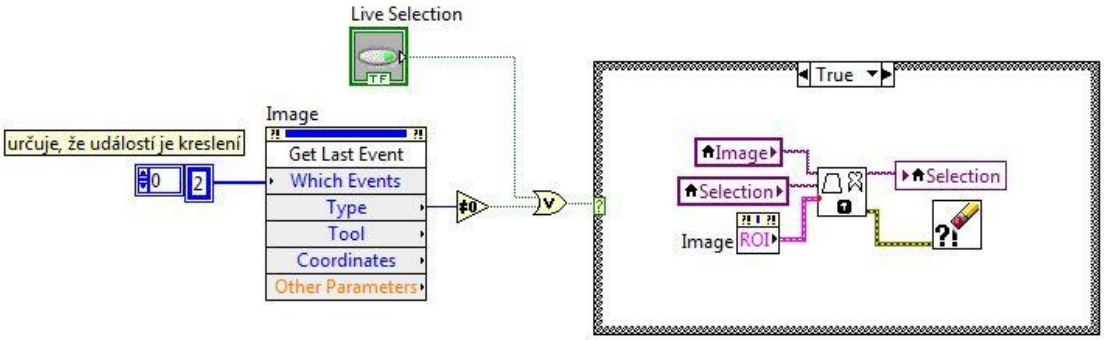


Obr. 14 - Zobrazení snímku z kamery

Na obr je část blokového diagramu, která zajišťuje zobrazení pořízeného snímku. Do bloku *GetActImage* vstupuje Handle z bloku *InitCamera*. Pro vstupní parametry bloku *ArrayToImage* použijeme cluster vytvořený při alokaci paměti. Výstup zobrazíme na panelu Image

Pomocí kurzoru myši lze na panelu Image vytvořit obdélníkový výřez dále (ROI – Region Of Interest) který se automaticky zobrazí na panelu Selection a zůstává neměnný. Obraz se aktualizuje při změně rozměrů či polohy ROI.

K tomuto účelu jsem si pomocí Invoke Node vytvořil událost Get Last Event a jako událost jsem zvolil kreslení. Tento blok při dokončení události pošle signál o tom, jaká událost to byla. Z toho se dá snadným porovnáním získat logická podmínka. Výřez samotný se provádí pomocí bloku Extract Tetragon. Ten požaduje zadání zdrojového/cílového obrazu a popis oblasti, kterou chceme extrahovat. K tomu jsem použil přímo vlastnost (property) proměnné Image. V případě sepnutého přepínače Live Selection se na panel s výřezem zobrazuje každý sejmutý snímek.

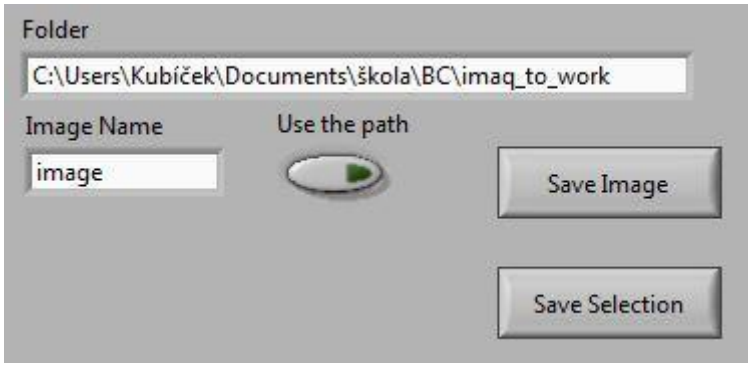


Obr. 15 - Zobrazení výřezu

Po levé straně pod panelem Image je tlačítko s nápisem Open Window. Po jeho stisknutí se poslední zachycený snímek otevře v externím okně v poměru 1:1 (v pixelech). Při opakovaném stisknutí se otevírají další okna, která jsou číslována. Lze nadefinovat maximální počet otevřených oken, číslování však pokračuje dál.

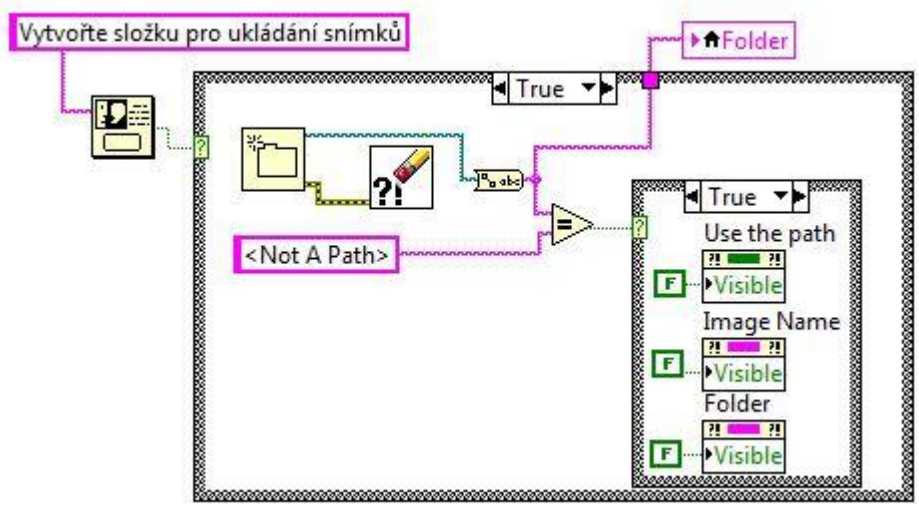
4.7 UKLÁDÁNÍ SNÍMKU

Bezprostředně po spuštění programu se objeví výzva k určení složky pro ukládání snímků. Pokud složku zvolíme, část čelního panelu zprostředkovávající ukládání snímků vypadá následovně



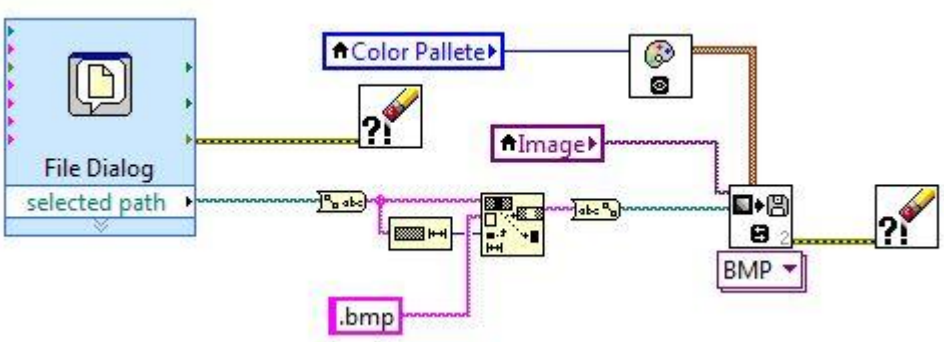
Obr. 16 - Menu pro ukládání snímků

Vidíme pole, kde je cesta k námi zvolené složce a pole s názvem souboru. Pokud je přepínač *Use the Path* aktivní, tak se při kliknutí na tlačítko *Save* obraz uloží do nadefinované složky s uvedeným názvem a pořadovým číslem snímku, které je umístěno bezprostředně za ním. Pokud přepínač *Use the Path* aktivní není, zobrazí se klasické okno s výzvou uložení souboru. Pokud při startu programu složku nezvolíme, okna (složka, soubor) a přepínač se na čelním panelu vůbec nezobrazí. A zůstává pouze dvojice tlačítek *Save*. Snímky se ukládají ve formátu BMP.



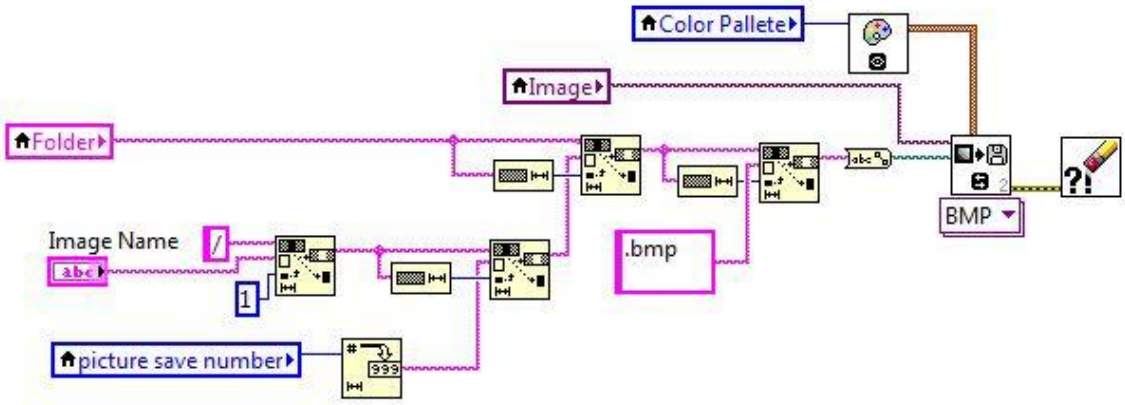
Obr. 17 - Výzva k určení složky pro ukládání snímků

Na obr je část blokového diagramu, která má na starost výzvu k určení složky pro ukládání při spuštění programu. Tento blok se provede po tom, co proběhne v pořádku inicializace kamery. První blok je *One Button Dialog*, ten pouze zobrazí správu a čeká na „odkliknutí“, poté může program pokračovat. Následuje blok *Create Folder*. Pokud není zvolena adresa, tak se na výstupu objeví hláška <Not A Path>. Přítomnost tohoto řetězce testuji v dalším kroku. Pakliže adresa zadána není a přítomnost řetězce se potvrdí, jsou prvky adresa, název a přepínač nastaveny jako neviditelné. Adresa se ukládá do proměnné *Folder.*



Obr. 18 - Uložení snímku pomocí dialogového okna

Tato část programu se volá, jestliže ukládáme obraz, jehož adresa a název není znám. Na výzvu k uložení souboru jsem použil expresní VI *File Dialog,* které po zadání vrací cestu a název souboru, ovšem bez přípony. Převedu tedy cestu na řetězec znaků (Path to String) a na konec tohoto řetězce přidám *.bmp* a zpět převedu na typ Path (cesta). Ke „sčítání" řetězců jsem použil blok *Replace Substring* kam se přivedou dva řetězce, druhý řetězec se zařadí do prvního v místě zadaném indexem. V mém případě délka prvního řetězce zjištěná blokem *Array Size*, takže vlastně zařadím druhý řetězec za první. Do bloku *Write File* je tedy přiveden název souboru i s cestou a příponou (File Path), dále obraz, který chceme uložit a barevná paleta, o té se zmíním později.

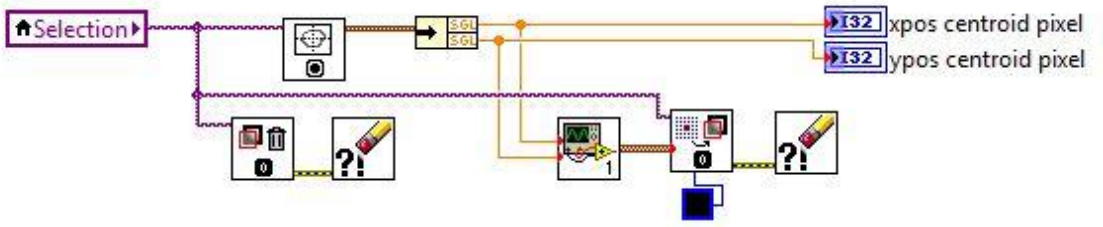


Obr. 19 - Uložení snímku do připravené složky

Obr ukazuje, jak vypadá ukládací procedura, když máme danou cestu a název souboru. Je potřeba poskládat cestu ke složce, název souboru a číslo (opět pomocí *Replace Substring*). Pro číslování jsem vytvořil proceduru, která přičítá proměnnou *picture save number* při **úspěšném** uložení snímku.

4.8 CENTROID

Knihovny Vision modulu obsahují VI s názvem Centroid, který počítá pozici energetického středu.



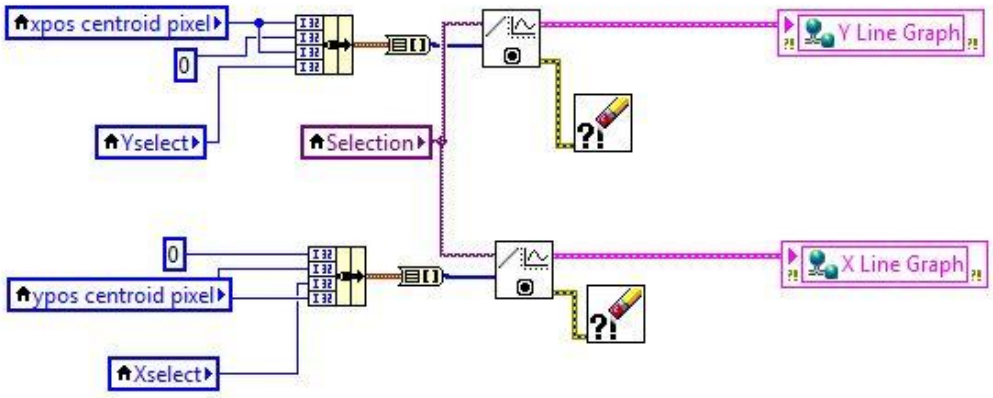
Obr. 20 - Určení polohy centroidu a vyznačení do obrazu

Zde je podprogram pro zjištění polohy centroidu v pixelech a zobrazení tohoto bodu na panel Selection. Vykreslení do grafu provádím pomocí knihovny IMAQ Overlay. Pomocí *Overlay Points* jsem vykreslil tmavý bod v místě centroidu. Byl to jeden pixel a nebyl téměř vidět. Proto jsem si vytvořil proceduru, která vytváří pole dvouprvkových clusterů, což jsou x a y souřadnice bodu. Výsledkem je, že se na panel s výřezem vytvoří kříž o velikosti 5x5 pixelů se středem v místě centroidu.

Výpočet centroidu je velmi náchylný na rušení proto se počítá pouze z výřezu. Protože znám fyzické rozměry snímacího čipu byl jsem schopen přepočítat polohu centroidu, tak aby se zobrazovala v milimetrech jako odchylka od středu snímače v osách x a y.

4.9 VÝPOČET ŠÍŘKY SVAZKU

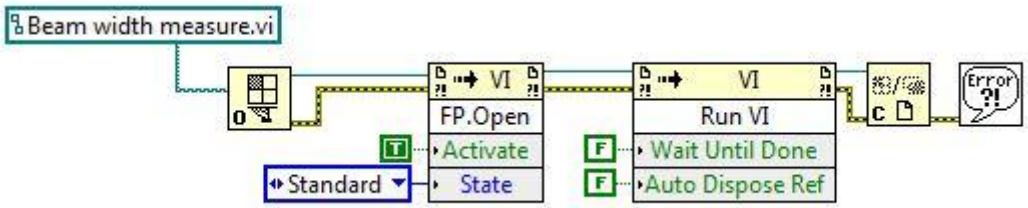
Výpočet šířky svazku probíhá ve třech úrovních intenzity. V 50% maximální intenzity, této šířka se označuje jako FWHM (z anglického Full width at half maximum). V 13,5% intenzity (také jako $1/e^2$). A ve 4 sigma (viz odstavec: definice šířky). Bylo potřeba zjistit hodnoty intenzit v osách x a y, tak aby procházeli centroidem. Jinak řečeno proložit svislou a vodorovnou přímku centroidem a vytvořit graf, kde na ose x bude pořadové číslo pixelu přímky a na ose y hodnota intenzity právě tohoto pixelu. Pak stačí použít nějakou funkci, která vyhledá příslušné hodnoty intenzit a určí vzdálenost mězi nimi.



Obr. 21 - Vytvoření jednorozměrného pole (poloha bodu na přímce, intenzita)

K tomuto účelu jsem použil blok *LineProfile*, což je funkce, jejíž vstupními parametry jsou obraz a poloha úsečky. Na výstupu obdržíme cluster, kompatibilní s LabVIEW grafem. Součástí tohoto clusteru je naše chtěné 1D pole intenzit. Úsečka se v bloku LineProfile definuje 1D polem 4 hodnot, kde první dvě jsou souřadnice počátečního bodu úsečky, a druhá dvojice určuje souřadnici koncového bodu. Toto pole snadno vytvoříme viz. obrázek. V proměnné *xypos centroid pixel* je uložena pozice centroidu a v *XYselect* jsou rozměry výřezu.

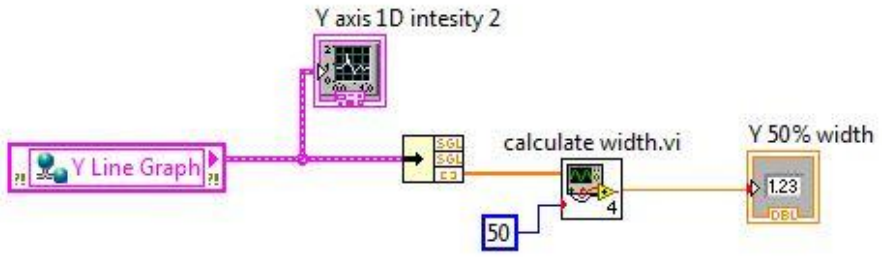
Protože grafy intenzit v osách se zobrazují na jiném čelním panelu, bylo potřeba zajistit přenos informací mezi dvěma VI. Použil jsem strukturu, která je v LabVIEW označena jako sdílená proměnná (shared variable). Otevření VI s grafy intenzit a výpočtem šířky svazku se provede následujícím způsobem.



Obr. 22 - Sekvence pro otevření VI z jiného VI

Sekvence začíná blokem Open VI Reference, zde se zadá cesta k VI, které chceme otevřít. Když je volaný soubor ve stejné složce jako spuštěný, stačí zadat název. Následuje *Invoke Node* a volba Select Class → VI Server → VI →VI a jako *Method* zvolit Front Panel → Open. Tím se otevře námi požadovaný VI. Následuje blok spuštění volaného VI. Vytvoříme stejný *Invoke Node*, ale jako *Method* zvolíme

Run VI. A ukončení reference. Tuto sekvenci mám umístěnou ve struktuře *case* s podmínkou na stisk tlačítka *Beam Measure Width*.



Obr. 23 - Zobrazení grafu intenzity v ose y a zobrazení šířky svazku

Na Obr. 23 je uveden příklad zobrazení grafu intenzit v ose y. Následně použijeme z clusteru Y Line Graph položku Pixels Line, ze které spočítáme šířku. Pro výpočet šířky jsem si vytvořil proceduru *calculate width,* jejímiž vstupy jsou Pixel Line a „výška" v procentech maximální intenzity ve které budeme šířku měřit.

Abychom mohli určit šířku 4 sigma, je potřeba proložit naměřeným profilem ideální Gaussovu křivku. A určit směrodatnou odchylku průběhu (sigma). Šířka je pak přímo rovna čtyřnásobku směrodatné odchylky. Pro ideální Gaussový profil svazky by měli šířky $1/e^2$ a D4σ vycházet stejně.

4.9.1 DEFINICE ŠÍŘKY SVAZKU

FWHM

Nejjednodušší definice šířky svazku, v polovině maximální intenzity odečteme šířku (průměr) svazku. Také se lze setkat s označením HPBW (z anglického half-power beam width).

$1/e^2$

Tato hodnota odpovídá zhruba 13,5% maximální intenzity. Toto kritérium obvykle nemá velký význam, pokud měříme svazek s více vrcholy (multimodální).

D4σ

D4σ je zkratka pro šířku svazku v bodě 4 krát σ, kde σ je směrodatná odchylka rozdělení. Při měření hodnotu D4σ více ovlivňují „křídla" Gaussovy křivky, než její střed. Je to způsobeno tím, že hodnoty jsou vážené druhou mocninou vzdálenosti

od středu paprsku. Pokud paprsek nevyplní více než třetinu plochy, tak okolní pixely, které mají nenulovou hodnotu, mohou významně ovlivnit výsledek (kvůli vlivu druhé mocniny vzdálenosti od středu, která se při výpočtu používá).

D4σ má, na rozdíl od šířek FWHM a $1/e^2$, smysl i pro multimodální rozdělení – tj. pro paprsky s více než jedním vrcholem. Vyžaduje ovšem pečlivý odečet základové hodnoty, aby výsledky byly přesné. Měření šířky D4σ je dle norem ISO považováno za standardní způsob určování šířky paprsku.

4.9.2 VYTVOŘENÍ VLASTNÍHO TYPU SDÍLENÉ PROMĚNNÉ

Sdílená proměnná se vytváří v Project Exploreru. Pokud vytváříme první sdílenou proměnnou v projektu, klikneme pravým tlačítkem na My Computer → New → Variable. Otevře se okno, kde se nastaví jméno proměnné, její typ (Network-Published) a typ dat, která bude proměnná obsahovat. Defaultně LabVIEW nabízí základní sadu typů, ale my potřebujeme typ vlastní a bude to cluster s obrazovými daty. Postup vytvoření vlastní typu pro sdílené proměnné je následující. Vrátíme se zpět do našeho VI a blokovém diagramu vytvoříme indikátor té proměnné, kterou budeme chtít jako sdílenou. Přepneme na čelní panel a na indikátor kliknemě pravým a zvolíme Advanced → Customize. Následující okno uložíme jako soubor typu *Control* (.ctl). A je hotovo. Již můžeme při volbě typu dat v proměnné zvolit From Custom Control a vybrat námi vytvořený typ. Proměnná se nám zobrazí v Project Exploreru ve složce Library1.lvlib. Při vkládání do blokového diagramu je sdílená proměnná na paletě Structures. Zda bude proměnná vstupní nebo výstupní, určuje položka Access Mode.

ZÁVĚR

Mým úkolem bylo v první řadě seznámit se s vývojovým prostředím LabVIEW. Nějaké zkušenosti z minulých jsem již měl, ale zpracování obrazu pro mě bylo něčím zcela novým. Nicméně během vytváření této aplikace jsem si osvojil mnoho programových konstrukcí, které LabVIEW nabízí a jsem rozhodnutý v práci v tomto prostředí pokračovat.

Co se týče programu samotného, vytvořil jsem aplikaci, která je schopna zobrazit data z kamery, zjistit základní informace o obrazu, nastavit parametry sejmutí obrazu (doba expozice, zisk). Provézt výřez obrazu, který nás bude zajímat pro další zpracování (ROI – Region Of Interest). Dále program umí určit polohu energetického středu svazku (centroid) a změřit šířku paprsku v definovaných hodnotách intenzity (FWHM, $1/e^2$, $D4\sigma$).

Tento projekt pro laserové centrum Eli-Beams je však mnohem obsáhlejší a budu na něm pracovat i nadále v rámci diplomového projektu případně diplomové práce. Je plán rozšířit aplikaci. Aby byla schopna počítat celkový výkon svazku, hustotu výkonu, umožnit na jednom počítači provozovat více kamer. Dále předávání základních údajů automaticky pro potřeby možného řízení a komunikace s nadřazeným systémem (který vyvíjí kolega).

POUŽITÁ LITERATURA

[1] HAVLE, Otto. Strojové vidění I: Principy a charakteristiky. *Automa*: *časopis pro automatizační techniku*. Praha: FCC Public, 2011, č. 1. Dostupné z: http://www.odbornecasopisy.cz/index.php?id_document=36550

[2] HAVLE, Otto. Digitální kamery pro systémy strojového vidění. Automa: časopis pro automatizační techniku. Praha: FCC Public, 2011, č. 11, s. 25-26. Dostupné z: http://www.odbornecasopisy.cz/res/pdf/44851.pdf

[3] DENEMARK, Tomáš. GigE Vision: nový standard pro průmyslové kamery. *Automatizace*: *Odborný časopis pro měření a inženýrskou informatiku*. 2007, roč. 50, č. 10. Dostupné z: http://www.automatizace.cz/article.php?a=1907

[4] VLACH, Jaroslav, Josef HAVLÍČEK a Martin VLACH. *Začínáme s LabVIEW*. 1. vyd. Ilustrace Viktorie Vlachová. Praha: BEN - technická literatura, 2008, 247 s. ISBN 978-80-7300-245-9.

[5] NATIONAL INSTRUMENTS. *LabVIEW Core 2*: *Course Manual*. 2010.

[6] IDS Imaging Development Systems GmbH. UEye Camera Manual, Version 4.0. 2012.

[7] IDS Imaging Development Systems GmbH. *Manual uEye LAbVIEW*: *Version 4.0*. 2012.