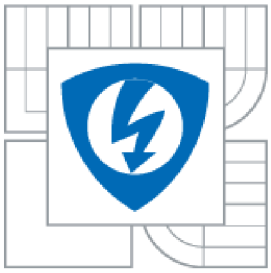
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV ELEKTROTECHNOLOGIE

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF ELECTRICAL AND ELECTRONIC TECHNOLOGY

OBSLUŽNÝ SOFTWARE PRO CCD KAMERU POUŽÍVANOU PRO ELEKTROLUMINESCENCI

SOFTWARE FOR CCD CAMERA USED FOR ELECTROLUMINESCENCE

DIPLOMOVÁ PRÁCE
MASTER´S THESIS

AUTOR PRÁCE — Bc. Pavel Kuttelwascher
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. Jiří Vaněk, Ph.D.
SUPERVISOR

BRNO 2013

VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

Ústav elektrotechnologie

Diplomová práce

magisterský navazující studijní obor
Elektrotechnická výroba a management

Student: Bc. Pavel Kuttelwascher
Ročník: 2

ID: 141930
Akademický rok: 2012/13

NÁZEV TÉMATU:

Obslužný software pro CCD kameru používanou pro elektroluminiscenci

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s principy snímání obrazu a jeho digitalizací. Zaměřte se na strukturu a konstrukci CCD snímačů. V rámci řešení projektu se seznamte s komunikačními protokoly CCD kamery používané pro detekci elektroluminiscenčního záření při analýze defektů solárních článků. Navrhněte softwarové řešení inovace tohoto zařízení tak, aby bylo možné sledovat pozorovaný článek v živém náhledu.

DOPORUČENÁ LITERATURA:

Podle pokynů vedoucího práce.

Termín zadání: 11.2.2013

Termín odevzdání: 30.5.2013

Vedoucí práce: doc. Ing. Jiří Vaněk, Ph.D.
Konzultanti diplomové práce:

prof. Ing. Jiří Kazelle, CSc.
předseda oborové rady

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta elektrotechniky a komunikačních technologií
Ústav elektrotechnologie
1

UPOZORNĚNÍ:

Autor diplomové práce nesmí při vytváření diplomové práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Anotace

Práce se zabývá optimalizací diagnostiky defektů fotovoltaických článků v temné komoře pomocí nového ovládacího softwaru pro použitou CCD kameru. V úvodních kapitolách je rozebrána současná situace s diskuzí možných řešení. Dále je věnována pozornost teorii světlocitlivých CCD snímačů, jejich konstrukci, principem činnosti a teorií snímání obrazu. Také se zaměřuje na analýzu funkcí ovládacích knihoven poskytnutých výrobcem zařízení. Stěžejní částí, je vývoj samotné aplikace pro kameru G2-3200 firmy Moravské přístroje, která podstatně zrychlí práci s kamerou v temné komoře.

Klíčová slova

kamera, CCD, živý náhled, expozice, aplikace, třída, metoda, bitmapa

Annotation

This thesis studies the optimization of diagnosis defects of the photovoltaic cells in the darkroom by the help of new operating software for the used CCD camera. First the present situation and the possible solutions are analyzed. Then the attention is paid to the theory of light-sensitive CCD sensors, their construction, their principles of operation and the theory of scanning images. The thesis also focuses on the analysis of the function of the operating libraries which were provided by the equipment producer. The key part is the development of the application for the camera G2-3200 of the "Moravske pristroje" company which significantly accelerates the work with the camera in the darkroom.

Keywords

camera, CCD, life view, exposure, application, class, method, bitmap

Bibliografická citace práce:

KUTTELWASCHER, P. *Obslužný software pro CCD kameru používanou pro elektroluminiscenci.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2013. 54 s. Vedoucí diplomové práce doc. Ing. Jiří Vaněk, Ph.D..

Prohlášení

Prohlašuji, že svou diplomovou práci na téma Obslužný software pro CCD kameru používanou pro elektroluminiscenci jsem vypracoval samostatně pod vedením vedoucího semestrálního projektu a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedeného semestrálního projektu dále prohlašuji, že v souvislosti s vytvořením tohoto projektu jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne 30. května 2013 .....

podpis autora

Poděkování

Děkuji vedoucímu diplomové práce doc. Ing. Jiřímu Vaňkovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování práce. Dále bych rád poděkoval panu Mgr. Janu Žídkovi za obětavou pomoc při řešení praktické části, a firmě Moravské přístroje, konkrétně panu Ing. Pavlu Cagašovi za poskytnutou dokumentaci a rady při vypracovávání práce.

V Brně dne 30. května 2013 .....

podpis autora

Obsah

Úvod

V dnešní době lidstvo prudce zvyšuje spotřebu energie. Podle prognóz bude dostačovat výroba energií konvenčními způsoby už jen několik desítek let, proto je potřeba hledat nové způsoby jak energii získávat a přitom nezatěžovat životní prostředí. Jedním z řešení je využívání slunečního záření pomocí fotovoltaických panelů.

Při vývoji nových a efektivnějších fotovoltaických článků, nebo i během sériové výroby je velmi důležitá analýza vlastností těchto článků. Jednak to může být zjišťování defektů, které mohou vzniknout při výrobě, ale i zjišťování vlastností článků při vývoji nových typů. Existují různé metody pro analýzu fotovoltaických článků, jako například metoda elektroluminiscence, metoda mikroplasmy, LBIC, nebo metoda fotoluminiscence. Jedná se o nedestruktivní metody, které povětšinou používají fotografické zařízení a temnou komoru. Pomocí nich lze optickými metodami provést detekce vad článků. Tyto metody ovšem nejsou za použití stávajících zařízení příliš efektivní, proto je třeba provést analýzu stavu a vypracovat inovační řešení, pro zefektivnění a zrychlení celého procesu diagnostiky.

1. Zadání práce

Zadáním práce je seznámit se s komunikačním protokolem, který používá CCD (Charge-Coupled Device - zařízení s vázanými náboji) kamera G2-3200 firmy Moravské přístroje, používaná pro diagnostiku výrobních defektů fotovoltaických článků metodou elektroluminescence. Je potřeba inovovat stávající řešení, usnadnit a urychlit celý proces diagnostiky fotovoltaických článků v temné komoře.

1.1 Stanovení cílů práce

Cíle práce spočívají ve vytvoření nového, nebo v úpravě a doplnění stávajícího softwaru, pro ovládání kamery G2-3200 firmy Moravské přístroje, která je původně určena k jiným účelům. Jedná se o hvězdářské zařízení, které nemá určité funkce, jež by značně usnadnily a zrychlily celý proces diagnostiky fotovoltaických článků.

Hlavním problémem práce s kamerou, je absence řešení, které by usnadnilo ostření obrazu v temné komoře. Kamera nedisponuje automatickým ostřením, ani funkcí živého náhledu, která by manuální ostření usnadnila, proto je za současného stavu nutné použít objektiv s manuálním ostřením, a postupně vytvářet jednotlivé expozice pro kontrolu správného zaostření. To znamená, pokaždé vstupovat do komory, přeostřit obraz a znovu exponovat. Toto řešení je velmi neefektivní, vytvoření kvalitního (ostrého) snímku takto trvá až desítky minut. Hlavním cílem práce, je tedy vyřešit tento problém.

1.2 Možná řešení

Možným řešením by bylo opatřit kameru automatickým ostřicím mechanismem. Vzhledem k tomu, že kamera pracuje v temné komoře, nelze použít pasivní autofokus (automatické ostření) z důvodu nedostatku světla dopadajícího na snímač. Použitý přístroj neumí komunikovat s moderními objektivy, které mají obsažen ostřicí motor, ale ani s objektivy starší generace, protože nedisponuje zabudovaným motorem v těle. Proto není možné použít automatické ostření.

Další možností je použití aktivního autofokusu. Toto řešení by spočívalo v doplnění kamery příslušným elektronickým zařízením, které by zajišťovalo měření vzdálenosti objektu a posunutí ostřícího mechanismu do odpovídající polohy. Zde je ovšem stejný problém s akčním prvkem, jako v předešlém případě.

Nejjednodušší řešení spočívá v použití živého náhledu v kombinaci s manuálním ostřením obrazu. To spočívá v naprogramování funkce *life view* (živý náhled) do obslužného softwaru kamery a dále případného vyřešení ručního přeostřování objektivu z vnějšku temné komory.

1.3 Postup řešení

Nejprve je potřeba seznámit se s komunikačním protokolem kamery G2-3200 a získat potřebné materiály, ovladače zařízení od výrobce a pokud možno knihovny, nebo zdrojové kódy pro další analýzu. V dalších krocích naprogramovat a otestovat aplikaci s funkcí *life view*. V závěru otestovat celý systém v praxi a zhodnotit dosažené výsledky.

2. Světlocitlivé CCD snímače

Tato kapitola se věnuje elektronickým světlocitlivým snímačům, jejich konstrukci, principem činnosti a získáním obrazu. Podrobně se zabývá snímači typu CCD (Charge-Coupled Device - zařízení s vázanými náboji).

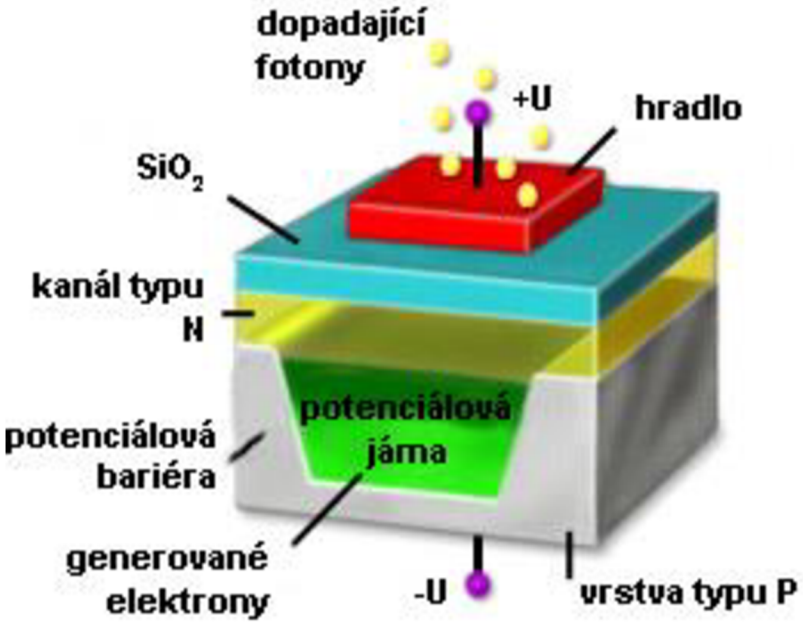
2.1 Úvod do CCD snímačů

CCD snímače patří dnes k nejpoužívanějším snímačům obrazu. Jedná se o součástky využívající tzv. fotoefektu, jevu, při kterém částice světla (foton), při nárazu na atom, dokáže převést některý z jeho elektronů do excitovaného stavu. CCD snímače byly vynalezeny Willardem Boylem a Georgem E. Smithem v roce 1969. Za tento vynález obdrželi Nobelovu cenu. Tyto snímače se používají v digitálních fotoaparátech, digitálních kamerách, čtečkách čárových kódů, skenerech, faxech i v různých měřících přístrojích, jako jsou například expozimetry.

V dnešní době začínají být CCD snímače pomalu nahrazovány snímači vyrobené technologií CMOS (Complementary Metal–Oxide–Semiconductor - doplňující se kov-oxid-polovodič). Ty mají některé výhody, jako je levnější výroba, vyšší rychlost, jakou lze přenést signál na A/D převodník a spotřeba el. energie. Se spotřebou totiž roste množství zbytkového tepla produkovaného snímačem. Rostoucí teplota snímače má negativní vliv na produkci šumu v obraze a tím pádem snižování kvality obrazu. Další výhodu oproti CCD snímačům mají v absenci bloomingu, obrazové vadě, která bude popsána v dalších kapitolách. Oproti CCD snímačům mají ale i své nevýhody. CCD snímače mají vyšší citlivost, díky které lze exponovat při nižším osvětlení, aniž by produkoval takové množství šumu, jako snímač CMOS.

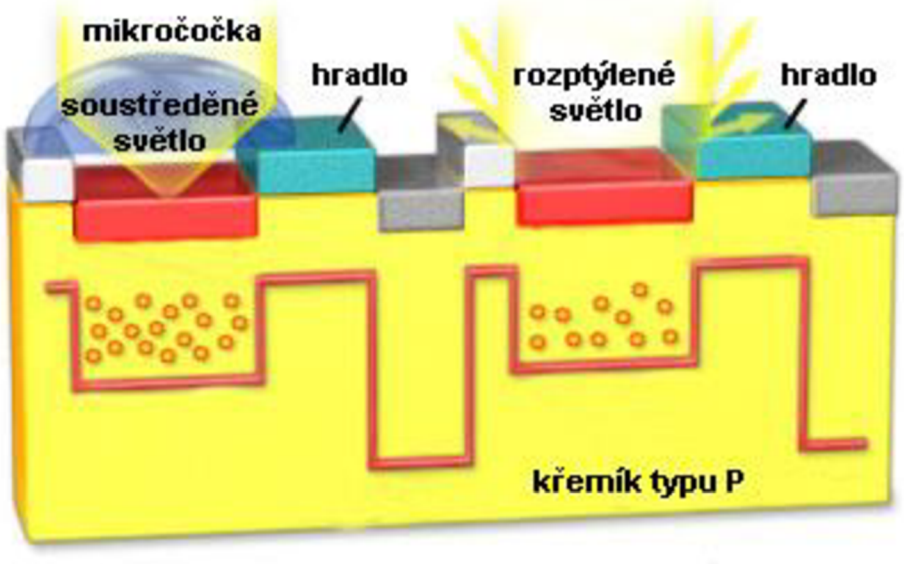
2.2 Konstrukce CCD snímačů

Struktura světlocitlivé buňky CCD snímače je znázorněna na obrázku1. Princip je stejný jako u fotodiody, tzn. dopadem fotonu se uvolní elektron (vznikne pár elektron-díra), který je přitahován ke kladně nabité elektrodě. Ovšem u této struktury je použita ještě tenká vrstva SiO_2, plnící funkci izolace. Použitím této izolační vrstvy se znemožní odvedení uvolněného náboje elektrodou. Ty pak zůstanou v křemíkovém substrátu a mohou se použít jako zdroj informace.



Obrázek 1: Struktura CCD snímače. [5]

Vzhledem k tomu, že hradla se vyrábějí z napařovaného hliníku, který velice dobře odráží světlo, stává se efektivní plocha snímače menší (hliník odráží světlo směrem od snímače) a citlivost snímače se snižuje. Proto se při výrobě na snímači vytvoří mikroskopické čočky, které soustředí většinu světla na efektivní plochu viz obrázek 2.

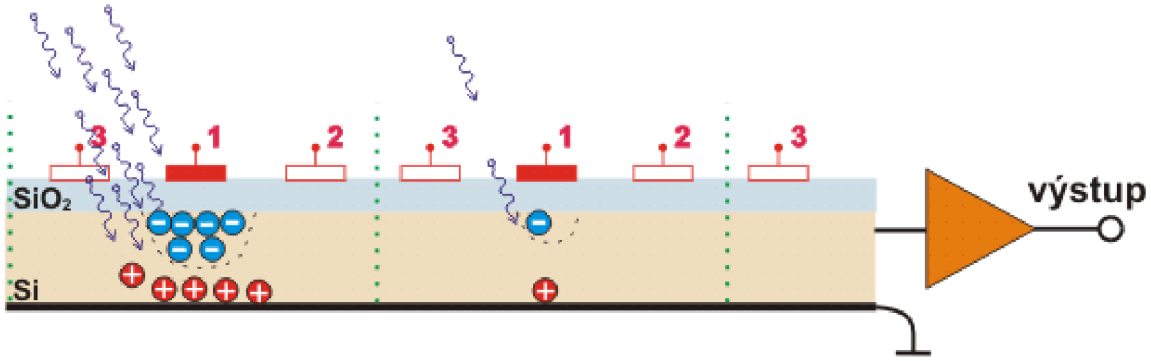


Obrázek 2: Funkce mikročoček používaných u CCD snímačů. [5]

2.3 Princip snímání obrazu

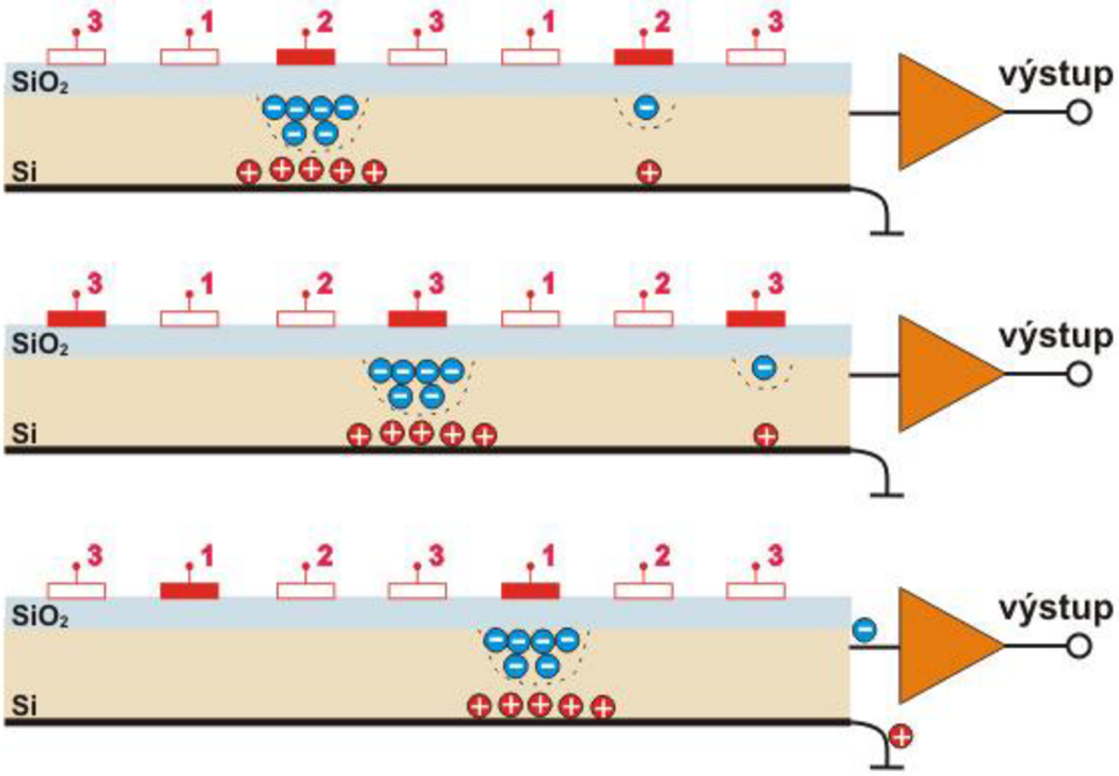
Snímání obrazu má několik fází. První (předpřípravná) fáze, slouží k vyprázdnění nábojů, které ve snímači zbyly z předchozí činnosti, nebo se uvolnily např. působením tepla. Veškeré volné elektrony jsou odebrány bez přístupu světla.

Následně se snímač nechá vystavený působení světla (provede se expozice). Během expozice fotony excitují elektrony v polovodiči. Ty jsou přitahovány ke kladně nabitým elektrodám. Zároveň dojde ke vzniku děr, které jsou přitahovány na záporně nabitou elektrodu na zadní straně snímače. Každá buňka (obrazový bod) snímače má tři elektrody a je oddělena od ostatních.



Obrázek 3: Hromadění uvolněných elektronů pod hradlem během expozice. Na obrázku je patrné oddělení jednotlivých buněk snímače. [6]

Po expozici je nutno odvést obrazový signál (náboj) na posuvný registr. K tomu se využívá třífázový hodinový signál, kdy se na jedno z hradel přivede kladné napětí. Pod tímto hradlem se vytvoří shluk elektronů, protože jsou přitahovány jeho kladným potenciálem. V dalším kroku se začne pomalu zvyšovat napětí na sousední elektrodě ve směru posunu náboje, a současně se snižuje napětí na elektrodě, u které byl náboj soustředěn původně. Tento děj se provádí postupně, dokud není celý řádek snímače odveden na posuvný registr a zpracován. Obrazová informace se tedy čte po řádcích, což může zapříčinit určité obrazové vady (viz další kapitoly).



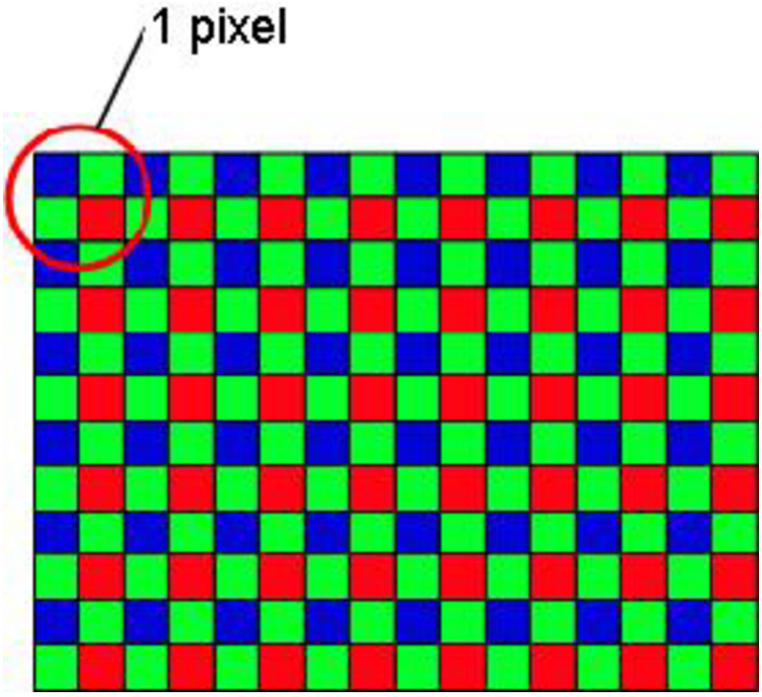
Obrázek 4: Na obrázku je znázorněno čtení řádku snímače posunem náboje shromážděným pod elektrodami jednotlivých buněk. [6]

2.4 Snímání barevného obrazu

Světlocitlivé elektronické snímače jsou z principu schopny získávat obraz pouze v odstínech šedi. Pro získání barevného obrazu se využívá princip skládání barev, kdy každá barva se získá součtem tří barevných složek: červené, zelené a modré (takzvaný RGB model - Red, Green, Blue).

Jeden obrazový bod snímače (pixel) je složen z menších elementů (subpixelů). Každý subpixel má před sebou barevný filtr, který propustí pouze odpovídající barevnou složku. Pole těchto malých filtrů se nazývá Bayerova maska (obrázek 5).

Tvar pixelu je čtvercový, proto jej bylo nutné rozdělit na čtyři segmenty (subpixely). Obsahuje tedy čtyři subpixely pro tři barevné složky. Čtvrtý segment je použit pro snímání zelené složky. Zelených segmentů je více, protože lidské oko je citlivější na tuto barvu a je schopno rozlišit více jejích odstínů. Složení jednoho pixelu je vidět na následujícím obrázku.

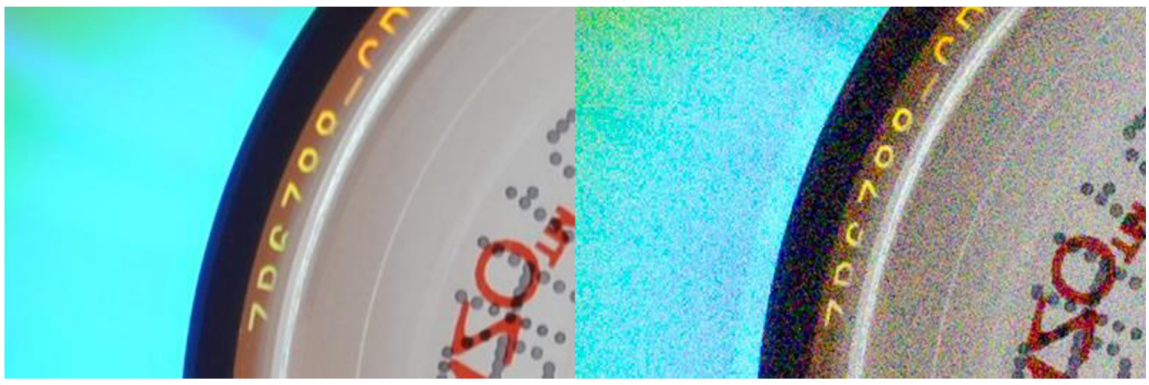


Obrázek 5: Bayerova maska. Zde je znázorněno složení jednoho obrazového bodu ze čtyř segmentů.

2.5 Obrazové vady elektronických snímačů

2.5.1 Šum

Šum je obrazová vada, kdy se na snímači uvolňují elektrony jinou příčinou (teplo, jiné záření), než je dopad fotonu na povrch snímače, tyto náboje neodpovídají dopadajícímu světlu, proto způsobují obrazovou chybu.



Obrázek 6: Ukázka vlivu šumu na kvalitu obrazu. Vlevo je fotografie bez znatelné úrovně šumu, vpravo je fotografie s vysokým množstvím šumu. [3]

Snahou, pro odbourání tohoto efektu je zvětšování plochy obrazových bodů snímače, kdy množství elektronů uvolněných dopadajícími fotony je mnohem větší v poměru s množstvím „chybových elektronů“ generovaných např. teplem. Tím se zvýší odstup signálu od šumu, což způsobí, že je šum v obraze méně patrný. Další možností jak tuto chybu eliminovat je chlazení snímače. U hvězdářských, nebo průmyslových zařízení se pro odvedení tepla používají např. peltierovy články. Přímé vzduchové chlazení snímače se nepoužívá, protože by se mohlo systémem přívodu vzduchu dostat na snímač nežádoucí světlo. Další možností je použití systému vodního chlazení.

Šum se také projevuje při expozici za nedostatečného osvětlení. Vychází z toho, že světlocitlivý snímač není schopen za daný expoziční čas „nasbírat“ dostatečné množství světla, tudíž je náboj každého obrazového bodu nutno zesílit. Spolu s obrazovým signálem se zesílí i šum, takže je více patrný, navíc zesilovač zanáší také určitou chybu, která zvýší šum v obraze.

Ukázka šumu je na obrázku 6, kde je patrné zvýšení šumu, při použití různých stupňů zesílení obrazové informace snímače (ISO-International Standards Organization).

2.5.2 Blooming

Blooming je obrazová chyba vyskytující se u CCD snímačů. Je dána jejich konstrukcí a principem činnosti. Pokud na část obrazových bodů dopadá větší množství světla, nežli odpovídá rozsahu snímače, vytvoří se velké množství náboje, který má tendenci se přesouvat na místa vedlejších obrazových bodů. Protože se u CCD snímačů ukládají data po řádcích, tyto náboje se přemisťují ve směru odpovídajícímu řádku snímače. To způsobuje typické přesvětlené pruhy. Tento efekt je znázorněn na obrázku 7.

Obrázek 7: Na obrázku je patrný blooming způsobený fotografováním silného zdroje světla. Uvolněné elektrony pak mají tendenci přetékat do vedlejších buněk. [4]

Existují technologie, které se snaží tento nežádoucí jev odbourat, například použitím anti-bloomingových hradel, které přebytečné elektrony odvádí pryč tak, aby se nemohly přemístit do vedlejšího obrazového bodu. Tato technologie má ale nevýhodu v tom, že se z důvodu výskytu těchto hradel snižuje užitečná plocha snímače až o 30%. Tím se zvětšuje i generovaný šum.

Další možností jak blooming odbourat, je použití série kratších expozic namísto jedné dlouhé. Díky tomu nemůže daná světlocitlivá buňka „přetéci“. Výsledný obraz se získá sečtením těchto jednotlivých obrazů. Metoda se využívá například v astronomické fotografii.

3. Získání snímku

3.1 Expozice

Expozice znamená vystavení snímače, nebo i například světlocitlivého filmu, působení světla. Na vlastnosti finálního obrazu mají vliv tři aspekty: použité clonové číslo, expoziční čas a citlivost ISO. Expozice má svou jednotku, a sice EV (exposure value). Čím větší je hodnota EV, tím více světlejší je fotografie (tím více světla-náboje se na snímači nahromadí).

3.1.1 Clona

Clona je součást objektivu fotoaparátů. Jedná se o soustavu lamel, pomocí kterých lze zužovat (rozšiřovat) kruhový otvor, kterým se přivádí světlo na snímač. Clona tedy ovlivňuje množství světla, které dopadá na snímač během expozice. Čím větší je clonové číslo, tím menší je otvor pro průchod světla (podobně jako u zornice lidského oka). Tvar otvoru v praxi není nikdy dokonale kruhový, ilustrace clony je na obrázku 8.

Obrázek 8: Ukázka clony fotoaparátu. Na obrázku je patrné, že clona je složena z většího množství lamel, které se překrývají.

3.1.2 Expoziční čas

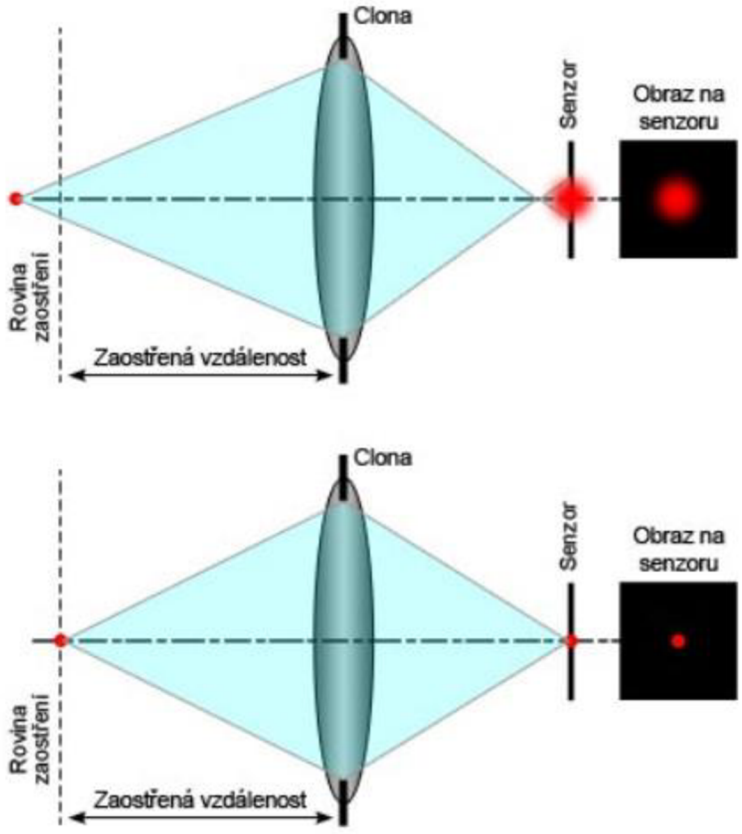
Expoziční čas, je doba, po kterou je světlocitlivý snímač, nebo film, či jiné médium, vystaven světelnému záření. Za tuto dobu se v jednotlivých světlocitlivých buňkách hromadí elektrické náboje způsobené dopadem fotonů. Čím delší je expoziční čas, tím více světla se dostane na snímač. Proto je při fotografování za špatných světelných podmínek potřeba exponovat na delší čas, aby se na snímači mohlo nahromadit dostatečné množství náboje. Expoziční doba se udává ve zlomcích, nebo jednotkách sekund.

3.1.3 ISO

ISO je veličina, která se používala již v klasické fotografii pro udání míry citlivost filmu na dopadající světlo. Dnes se používá i v přístrojích s digitálními světlocitlivými snímači. ISO však neudává citlivost samotného elektronického snímače, ale míru zesílení obrazového signálu, který se na snímač dostal během expozice. Zvýšením citlivosti ISO tedy lze exponovat na kratší časy i při horších světelných podmínkách. Bohužel zesilovač zesílí nejen obrazový signál, ale i šum, který snímač vytvořil, a proto je na výsledné fotografii více patrný. V praxi je snaha používat co nejnižší hodnotu ISO, aby nedošlo ke ztrátě obrazové informace šumem.

3.2 Ostření

Obraz je ostrý v případě, kdy bod v reálné scéně, který se převede na snímač, zůstane bodem. Pokud by se zobrazil jako kruhová ploška určitého průměru, není zaostřený (je rozostřený) viz obrázek 9. Rovina bodů, které jsou správně zaostřeny, se nazývá rovina zaostření. Lidské oko má však menší rozlišovací schopnost, a proto můžeme vidět ostré objekty v jiné vzdálenosti, než je rovina ostření. Rozsah vzdáleností, ve kterých vidíme objekty ostře, se nazývá hloubka ostrosti. Ta je závislá především na použité ohniskové vzdálenosti objektivu, vzdálenosti objektu, na který je zaostřeno, dále na použitém clonovém čísle a na velikosti optického snímače. [2]



Obrázek 9: Ukázka rozostřeného bodu (nahoře) a správně zaostřeného bodu (dole) [2]

V praxi existují různé způsoby ostření obrazu a to manuální nebo automatické. Pro automatické (autofokus) se využívá dvou principů: aktivní autofokus, nebo pasivní autofokus.

Aktivní autofokus využívá infračervený, nebo ultrazvukový signál, který pošle na fotografovaný objekt a měří se doba, za kterou se odrazí zpět. Podle doby se vypočítá vzdálenost fotografovaného objektu a automaticky se nastaví poloha ostřicího mechanismu. Tento systém má výhodu v tom, že nepotřebuje vnější osvětlení objektu, a je rychlý. Ovšem nevýhodou je, že nepracuje při focení přes sklo, nebo s použitím různých předsádek, kdy se poloha ostřícího mechanismu posouvá do jiné vzdálenosti a měření vzdálenosti objektu je zkreslené délkou použité předsádky. To ve spojení s malou hloubkou ostrosti může zapříčinit špatné zaostření (front focus, back focus).

Pasivní autofokus využívá obrazový snímač pro ostření obrazu tak, že analyzuje obraz na základě kontrastu hran obsažených ve snímku. Přístroj posouvá polohu roviny zaostření do té doby, dokud nenajde optimální ostrost hrany objektu. Tento způsob lze použít při fotografování přes sklo i s předsádkami, ale nefunguje při malém množství dopadajícího světla na snímač.

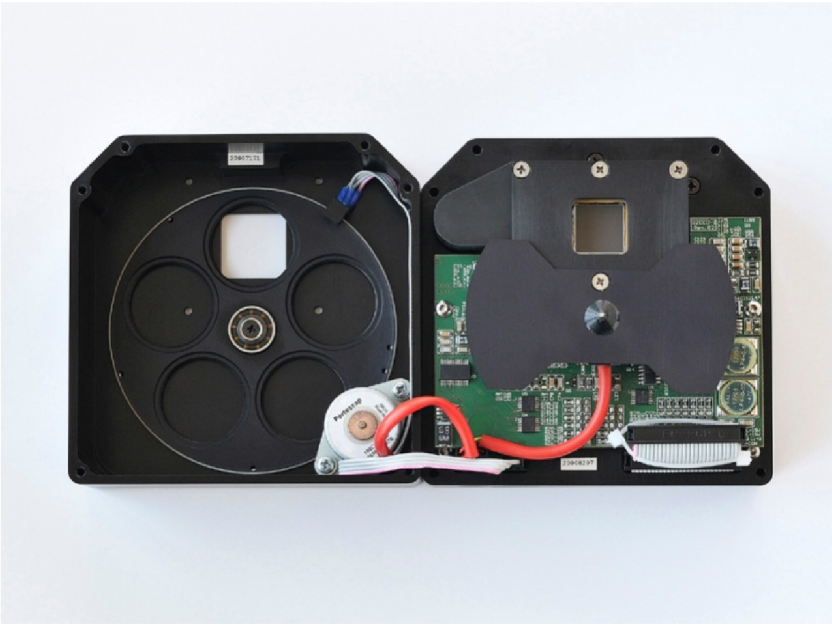
3.3 Life view

Režim živého náhledu (life view) se objevil se vznikem elektronických snímačů, slouží k okamžitému prohlížení obrazu, který se naexponuje na snímači. Obraz je tedy přes objektiv přiveden na světlocitlivý snímač, dále je získaná informace pomocí A/D převodníků převedena do digitální formy a poslána nejen do obrazového procesoru, který data dále zpracovává, ale také na display přístroje, nebo pomocí různých rozhraní do PC ovládajícího přístroj. To umožňuje vidět obraz průběžně a například kontrolovat ostrost obrazu a správnou kompozici.

4. Charakterizace kamery G2-3200

Obrázek 10: Kamera G2-3200 firmy Moravské přístroje [7]

Tato kamera od společnosti Moravské přístroje je přístroj, určený pro astronomické účely. Využívá nízkošumový full-frame (plnoformátový) snímač typu CCD od společnosti Kodak. Dále podporuje rychlé USB 2.0 komunikační rozhraní s maximální rychlostí přenosu 480 Mbps (Megabit per sekond - Megabitů za sekundu), dvoustupňové peltierovo chlazení optického snímače pro snížení tepelného šumu, které je možné kombinovat s vodním chlazením. Přístroj má také zaintegrované kolo s různými filtry v různém spektru. [7]



Obrázek 11: Pohled dovnitř kamery. Na levé části je vidět podavač filtrů, v pravé části závěrka snímače a ovládací elektronika. [7]

Snímač je typu CCD Kodak KAF-3200ME s rozlišením 3,2 MPx (2184x1472Px) a rozměry 14,9x10mm. Velikost jednoho pixelu je 6,8x6,8 μm. Bližší informace jsou v tabulce 1.

Tabulka 1: Parametry použitého CCD snímače Kodak KAF-3200ME

Rozlišení	2184 (H-horizontálně) × 1472 (V-vertikálně) pixelů
Velikost pixelu	6,8 μm (H) × 6,8 μm (V)
Obrazová plocha	14,9 mm (H) × 10 mm (V)
Plná kapacita pixelu	~55 000 e^-
Kapacita výstupního bodu	~110 000 e^-
Temný proud	0,8 e^-/s/pixel při 0 °C
Zdvojení tepelného šumu	6 °C

Kamera využívá 16bitového A/D převodníku s korelovaným dvojitým vzorkováním pro vysoký dynamický rozsah. Termoelektrické chlazení dvěma peltierovými články je schopno ochladit snímač až o 50°C. Jejich horká strana je potom chlazena 50mm ventilátorem. Napájení je 12V DC s příkonem až 40W, použit je originální napájecí zdroj. Kamera má vlastní stabilizátor napětí, tudíž není nutné, aby napájecí zdroj stabilizátor obsahoval. [7]

Tabulka 2: Informace o napájení a spotřebě kamery.

Napájení hlavy kamery	12 V DC
Spotřeba kamery	15 W bez chlazení
	40 W chlazení 100%
Vstupní napětí adaptéru	100-240 V AC/50-60 Hz
Výstupní napětí adaptéru	12 V DC/5 A
Maximální výkon adaptéru	60 W

Přístroj používá otočnou mechanickou závěrku, jejíž nejkratší expoziční čas je 85ms. Nejdelší čas expozice sice není limitován, ale je omezen saturací čipu. K dispozici jsou různé adaptéry pro připojení objektivů, jsou to: standardní 1.25" a 2" adaptéry, T-závit a adaptéry pro objektivy Canon EOS, Nikon apod. [7]

Při použití standardních objektivů fotoaparátů kamera neumí ovládat clonu. Moderní objektivy totiž komunikují přímo s přístrojem, který clonu řídí. Objektivy firmy Nikon například používají mechanicky řízenou clonu z těla fotoaparátu, firma Canon zase používá elektronicky řízenou clonu. Kamera by tedy musela ovládat více způsobů řízení a komunikace s objektivy, nehledě na to, že každý výrobce používá jiné protokoly pro komunikaci s objektivy a jiné mechanické upevnění objektivů. Možným řešením by bylo

použít starší objektivy, které mají manuální clonový kroužek (kde lze nastavit clonu ručně). V našem případě je použit objektiv Canon, který má (pokud není nastaveno řídicí elektronikou jinak) clonu otevřenou na nejmenší hodnotu clonového čísla (plně otevřenou clonu).

5. Diagnostické metody využívající detekci záření pomocí CCD kamery

5.1 Metoda elektroluminescence

Elektroluminescence je děj, při kterém elektrony přeskakují z vyšší, do nižší energetické hladiny. Při přeskoku mezi hladinami uvolní přebytečnou energii v podobě elektromagnetického záření (fotonu). Díky tomu lze snadno diagnostikovat různé nepravidelnosti (defekty) fotovoltaického článku.

Měření probíhá tak, že se k fotovoltaickému článku připojí stejnosměrné napětí a tím se vybudí proud protékající článkem v propustném směru. To vyvolá buzení fotonů, které zbůsobují luminescenci. Záření se snímá kamerou se světlocitlivým snímačem, díky tomu lze analyzovat nepřesnosti ve struktuře článku. Měření musí probíhat v absolutní tmě, kvůli eliminaci chyby způsobené dopadáním vnějšího světla. Z tohoto důvodu se používá temná komora.[1]

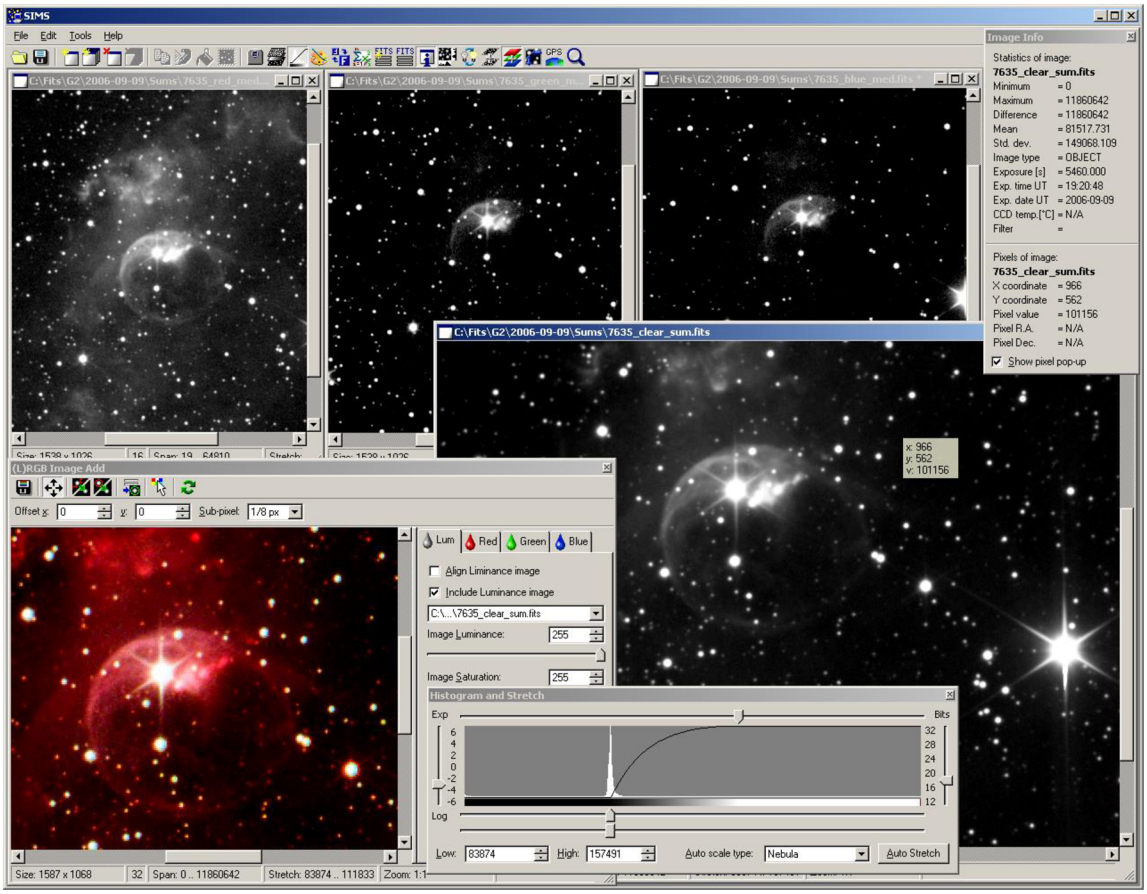
5.2 Temná komora

Pro účely měření metodou elektroluminiscence byla vyrobena speciální temná komora, vyrobená z hliníkového plechu a mající tři části. Horní část obsahuje CCD kameru a její účel je nepropouštět světlo ke kameře a zabránit dopadu světla přes systém vzduchového chlazení na zadní stranu CCD snímače kamery. V prostřední části je umístěno kontaktní pole pro uchycení měřeného fotovoltaického článku a průchodky pro připojení kabeláže k měřicím přístrojům. Spodní část komory obsahuje nerezovou vanu s polystyrenovou izolační nádobou, do které se lije kapalný dusík pro podchlazení měřeného článku. Dále je celý vnitřní povrch, kvůli eliminaci odrazu světla, nastříkán černou matnou barvou. [1]

Obrázek 12: Temná komora používaná pro diagnostiku fotovoltaických článků. [1]

6. Program SIPS

SIPS (Scientific Image Processing System) je originální software pro obsluhu kamery dodaný výrobcem. Jedná se o aplikaci určenou ke hvězdářským účelům. Náhled je patrný na obrázku 13.



Obrázek 13: Program SIPS [7]

Tento program umožňuje jednak řízení expozice, ovládání chlazení, nastavení požadovaného filtru, režim nízkého šumu, ale i pořizování automatických sekvencí snímků, nebo kalibraci. K dispozici jsou také pokročilé funkce pro úpravu nafocených snímků jako zobrazení a úprava histogramu, obrazové filtry, výřezy, nebo automatické skládání snímků a řada dalších nástrojů. Samotné uživatelské rozhraní umožňuje noční režim, který tolik neoslňuje uživatele při použití ve tmě. K dispozici jsou i ovladače třetích stran pro různé podpůrné systémy. Program SIPS také podporuje automatickou pointaci montáže dalekohledu s použitím samostatné pointační kamery.

7. Základní funkce pro komunikaci s kamerou

V této kapitole jsou rozebrány funkce, které jsou obsaženy v DLL (Dynamic-Link Library-dynamicky linkovaná knihovna) knihovnách výrobce, a zprostředkovávají základní ovládání přístroje, jako navázání komunikace, nastavení parametrů, ovládání závěrky, expozice a načtení fotografie do ovládacího PC. Tyto API (Application Programming Interface-rozhraní pro programování aplikací) funkce jsou naprogramovány v programovacím jazyce C.

Ovladače pro kamery G2 se skládají ze dvou základních částí. Systémový ovladač, který běží jako součást jádra operačního systému. Tento ovladač zajišťuje komunikaci nejnižší úrovně po USB. Jedná se o soubory *g2ccd.sys* a *g2ccd.inf*. Dále ještě existuje katalogový soubor *g1ccd.cat*, který obsahuje digitální podpis o kompatibilitě s operačními systémy Windows. [9]

Druhá část jsou API funkce v podobě DLL knihovny, na kterých je postavena samotná aplikace. Obsahuje funkce pro navázání spojení s kamerou, čtení informací z přístroje, nastavení parametrů, otevírání a zavírání závěrky, expozici, načtení obrazu, atd. Jedná se o soubor „*g2ccd2.dll*“. Tento soubor bude také nedílnou součástí nového softwaru.

7.1 Používané datové typy:

API výrobce kamery používá (pro přehlednost) vlastní datové typy, jejich definice je znázorněna zde:

```
typedef int            INTEGER;
typedef unsigned       CARDINAL;
typedef float          REAL;
typedef double         LONGREAL;
typedef unsigned char  CHAR;
typedef unsigned char  BOOLEAN;
typedef void *         ADDRESS;
```

7.2 Funkce API

Zde jsou probrány jednotlivé funkce API s jejich parametry a návratovými hodnotami. [9]

```
void __cdecl Enumerate(
     void ( __cdecl *CallbackProc ) ( CARDINAL )
);
```

Funkce *Enumerate()* umožňuje vyhledávání všech kamer, které jsou v daném okamžiku připojeny k ovládacímu PC.

Její parametr je ukazatel na funkci zpětného volání *CallbackProc()* s jedním celočíselným parametrem. Tato funkce je volána pro každou připojenou kameru a jejich identifikátory jsou předány jako parametr. Potom může aplikace nabídnout uživateli seznam připojených zařízení, ze kterých lze vybírat.

Pokud je aplikace vytvářena pro práci s pouze jednou kamerou, nebo je známý identifikátor zařízení (například je získán z INI souboru, kde je identifikátor definován), nemusí být tato funkce volána.

```
CCamera *__cdecl Initialize(
     CARDINAL Id
);
```

Ovladač je navržen tak, aby mohl manipulovat s více kamerami najednou. Jednotlivé kamery rozlišuje pomocí handlerů jednotlivých instancí. Handler je definován jako ukazatel na skrytou strukturu:

```
struct CCamera;
```

Tento ukazatel je navrácen funkcí *Initialize()*. Této funkci je nezbytné předat identifikátor kamery. Jestliže funkce vrátí hodnotu NULL, nemůže být konkrétní kamera použita. Buď není vůbec připojena, nebo jí používá jiný software.

```
void __cdecl Release(
     CCamera *PCamera
);
```

Pokud není potřeba kameru nadále používat, je její manipulátor uvolněn voláním funkce *Release()*. Žádné jiné funkce (s výjimkou *Enumerate()* a *Initialize()*) potom již nemohou být volány.

```
void __cdecl RegisterNotifyHWND(
     CCamera *PCamera,
     ADDRESS NotifyHWND
);
```

Ovladač může aplikaci oznámit, že byla připojena, nebo odpojena kamera. Oznámení je posláno zprávou Windows, jejíž HWND byl předán jako parametr funkce *RegisterNotifyHWND*. Zprávy jsou:

```
#define WM_CAMERA_CONNECT 1034
#define WM_CAMERA_DISCONNECT 1035
```

Pokud se aplikace dlouhou dobu nezajímá o toto oznámení, může zavolat funkce *RegisterNotifyHWND* s parametrem NULL.

Volání funkce *Release()* automaticky zavolá i funkci *RegisterNotifyHWND()* s parametrem HWND = NULL.

```
BOOLEAN __cdecl GetBooleanParameter(
      CCamera *PCamera,
      CARDINAL Index,
      BOOLEAN *Boolean
);
```

Funkce vrací hodnotu typu BOOLEAN, která závisí na parametru Index. Jestliže funkce nemůže rozpoznat hodnotu parametru *Index*, vrací hodnotu *false*.

Parametr *Index* může nabývat těchto hodnot:

- *gbpConnected*: Vrací *true* jestliže je kamera v daném okamžiku připojena
- *gbpSubFrame*: Vrací *true* jestliže kamera podporuje čtení výřezu obrazu
- *gbpReadModes*: Vrací *true* jestliže kamera podporuje více způsobů čtení
- *gbpShutter*: Vrací *true* jestliže je kamera vybavena mechanickou závěrkou
- *gbpCooler*: Vrací *true* jestliže je kamera vybavena aktivním CCD chladičem
- *gbpFan*: Vrací *true* jestliže může být ovládán chladicí ventilátor kamery (zapnutý/vypnutý)
- *gbpFilters*: Vrací *true* jestliže kamera obsahuje kolo s filtry
- *gbpGuide*: Vrací *true* jestliže je kamera schopna navádět teleskop
- *gbpElectronicShutter*: Vrací *true* jestliže je CCD snímač kamery vybaven elektronickou závěrkou
- *gbpRGB*: Vrací *true* jestliže má kamera Bayerovu RGBG masku na CCD snímači
- *gbpCMY*: Vrací *true* jestliže má kamera CMY filtry na CCD snímači
- *gbpCMYG*: Vrací *true* jestliže má kamera CMYG filtry na CCD snímači

- *gbpDebayerXOdd*: Vrací *true* jestliže Bayerova maska začíná na lichém, horizontálním pixelu
- *gbpDebayerYOdd*: Vrací *true* jestliže Bayerova maska začíná na vertikálním lichém pixelu
- *gbpInterlaced*: Vrací *true* jestliže CCD snímač pracuje v prokládaném režimu

```
BOOLEAN __cdecl GetIntegerParameter(
      CCamera *PCamera,
      CARDINAL Index,
      CARDINAL *Num
);
```

Funkce vrací celočíselný parametr (typu Integer), který závisí na parametru *Index*. Jestliže funkce nemůže rozpoznat hodnotu parametru *Index*, vrátí hodnotu *false*.

Parametr *Index* může nabývat těchto hodnot:

- *gipCameraId*: vrátí Identifikátor aktuální kamery
- *gipChipW*: Vrátí šířku CCD snímače v pixelech
- *gipChipD*: Vrátí výšku CCD snímače v pixelech
- *gipPixelW*: Vrátí šířku CCD snímače v nanometrech
- *gipPixelD*: Vrátí výšku CCD snímače v nanometrech
- *gipMaxBinningX*: Vrátí maximální binning v horizontálním směru
- *gipMaxBinningY*: Vrátí maximální binning ve vertikálním směru
- *gipReadModes*: Vrátí počet čtecích režimů kamery
- *gipFilters*: Vrací počet filtrů, které jsou k dispozici
- *gipMinimalExposition*: Vrací nejkratší možný čas závěrky dané kamery v milisekundách

```
BOOLEAN __cdecl GetStringParameter(
      CCamera *PCamera,
      CARDINAL Index,
      CARDINAL String_HIGH,
      CHAR *String
);
```

Funkce vrací řetězec, který závisí na použitém parametru. Pokud funkce nemůže rozpoznat hodnotu parametru *Index*, vrátí hodnotu *false*.

Získaný řetězec je nakopírován do bufferu, na který ukazuje hodnota ukazatele „*String". Funkce kontroluje maximální velikost bufferu, aby předešla jeho přetečení. Index posledního znaku v bufferu (indexování začíná od nuly) musí být předán jako hodnota parametru *String_HIGH*. Pokud je velikost bufferu větší, než předávaný řetězec, ukončovací nulový znak řetězce se zkopíruje také.

Parametr *Index* může nabývat těchto hodnot:

- *gspCameraDescription*: Vrací popis kamery
- *gspManufacturer*: Vrací název výrobce
- *gspCameraSerial*: Vrátí sériové číslo kamery
- *gspChipDescription*: Vrátí popis použitého CCD snímače

```
BOOLEAN __cdecl GetValue(
      CCamera *PCamera,
      CARDINAL Index,
      REAL *Value
);
```

Funkce *GetValue*() vrací hodnotu typu float závislou na parametru *Index*. Pokud funkce nemůže rozpoznat hodnotu parametru *Index*, vrátí hodnotu *false*.

GetValue() je podobná funkcím GetBoolean/Integer/StringParameter, ale předchozí tři funkce vracely statickou hodnotu (nezávislou na okamžitém stavu kamery). Návratovou hodnotou je okamžitý stav přístroje (například teplota snímače, atd.).

- gvChipTemperature: Vrací okamžitou teplotu CCD snímače ve stupních Celsia
- gvHotTemperature: Vrací okamžitou teplotu teplejší strany chladiče ve stupních Celsia
- gvCameraTemperature: Vrací okamžitou teplotu uvnitř těla kamery ve stupních Celsia
- gvEnvironmentTemperature: Vrací okamžitou teplotu vnějšího prostředí ve stupních Celsia
- gvSupplyVoltage: Vrací okamžitou hodnotu napětí napájecího zdroje
- gvPowerUtilization: Aktuální využití cladiče CCD snímače (hodnota od 0 do1)

- gvADCGain: Vrací okamžitý zisk A/D převodníku v elektronech/ADU

```
void __cdecl SetTemperature(
     CCamera *PCamera,
     REAL Temperature
);
```

Nastaví požadovanou teplotu snímače. Teplota se udává ve stupních Celsia. Pokud kamera nemá chlazení, nemá tato funkce žádný efekt.

```
BOOLEAN __cdecl SetBinning(
     CCamera *PCamera,
     CARDINAL x,
     CARDINAL y
);
```

Nastaví požadovaný binning při čtení. Pokud kamera binning nepodporuje, nemá tato funkce význam.

Binning je technika, která se používá v astronomických přístrojích, kde je potřeba, kvůli malému množství dopadajícího světla na snímač, výrazně zvýšit jeho citlivost. Binning spočívá ve slévání nábojů získaných v sousedních obrazových bodech tak, že dohromady dají jen jeden bod. Výsledný obraz má sice menší rozlišení, ale větší citlivost, která je důležitá při exponování při malé intenzitě světla. Binning může být vertikální i horizontální. V praxi se používá obou současně tak, že se sečtou nejčastěji náboje z 2x2, nebo 3x3 pixelů, to znamená, že výstupní obraz bude mít 4x, nebo 9x méně obrazových bodů. [8]

```
void __cdecl ClearCCD(
     CCamera *PCamera
);
```

Odvede veškerý nahromaděný náboj z CCD snímače (vymaže obraz na snímači).

```
void __cdecl Open(
     CCamera *PCamera
);
```

Otevře závěrku kamery. Jestliže má kamera mechanickou závěrku, nemá tato funkce využití.

```
void __cdecl Close(
      CCamera *PCamera
);
```

Zavře závěrku kamery. Jestliže má kamera mechanickou závěrku, nemá tato funkce využití.

```
BOOLEAN __cdecl GetImage(
      CCamera *PCamera,
      INTEGER x,
      INTEGER y,
      INTEGER w,
      INTEGER d,
      CARDINAL BufferLen,
      ADDRESS BufferAdr
);
```

Funkce *GetImage()* načte obraz z kamery. Obraz je matice *w*d* (šířka x výška) 16bitových pixelů. *BufferAdr* je ukazatel na buffer, do kterého je obraz zkopírován. *BufferLen* představuje velikost tohoto bufferu v Bytech. Pokud v bufferu není dostatek místa pro uložení obrazu, volání funkce selže.

Pokud kamera nepodporuje čtení výřezu obrazu, parametry x a y musí být nula. Parametry w a d musí mít hodnotu rozlišení snímače.

```
BOOLEAN __cdecl GetImageExposure(
      CCamera *PCamera,
      LONGREAL ExpTime,
      BOOLEAN UseShutter,
      INTEGER x,
      INTEGER y,
      INTEGER w,
      INTEGER d,
      CARDINAL BufferLen,
      ADDRESS BufferAdr
);
```

Gx CCD kamery umožňují dvě možnosti počítání expozičního času. Dlouhé expozice lze řídit přímo z obslužného PC, v takovém případě je postup následný:

1. Inicializace funkcí *ClearCCD()* – Odvedou se náboje ze snímače, a ten je připraven k expozici (neobsahuje žádnou obrazovou informaci).

2. Otevření závěrky funkcí *Open()*.

3. Odměření samotné expoziční doby pomocí časovače.

4. Zavření závěrky – funkce *Close()*.

5. Načtení obrazu – funkcí *GetImage(...)*.

Pro expozice délky desítek sekund až minut nejsou drobné nepřesnosti v časování tolik důležité (mohou být, v závislosti na rychlosti PC, 2 až 15 milisekund). Tyto nepřesnosti mohou být velmi zásadní pro krátké expoziční časy. Proto byla vytvořena funkce *GetImageExposure()*. Při použití této funkce si kamera sama odměří expoziční čas s přesností na desítky mikrosekund. Parametry x, y, w, d jsou totožné jako u předchozí funkce.

V tomto případě není nezbytné ovládat samostatně závěrku a mazat náboj CCD snímače před provedením expozice. Kamera při použití funkce udělá vše automaticky. Nyní stačí exponovat pouze jedním krokem:

1. Volání funkce *GetImageExposure(...)*

Parametr *ExpTime* je požadovaná doba expozice v sekundách. Parametr *UseShutter* udává ovladači, zda má vytvořit tmavý, nebo světlý snímek. Tato funkce má využití u snímacího režimu s nízkým šumem.

```
void __cdecl GetLastError(
     CCamera *PCamera,
     CARDINAL ErrorString_HIGH,
     CHAR *ErrorString
);
```

Pokud volání nějaké funkce selže (vrátí hodnotu *false*), funkce *GetLastError()* vrátí popis chyby. Řetězec *ErrorString* je předán podobně jako u volání *GetStringParameter()*.

```
BOOLEAN __cdecl EnumerateReadModes(
      CCamera *PCamera,
      CARDINAL Index,
      CARDINAL Description_HIGH,
      CHAR *Description
);
```

Spočítá všechny čtecí módy poskytované kamerou. Tento výpočet nepoužívá žádné zpětné volání. Funkce je volána (nejprve s indexem 0) opakovaně, pokaždé s inkrementovaným indexem tak dlouho, dokud nevrátí hodnotu *false*. Řetězec *Description* je předáván obdobně jako u volání funkce *GetStringParameter()*.

```
BOOLEAN __cdecl SetReadMode(
      CCamera *PCamera,
      CARDINAL mode
);
```

Nastaví požadovaný čtecí mód.

```
BOOLEAN __cdecl EnumerateFilters(
      CCamera *PCamera,
      CARDINAL Index,
      CARDINAL Description_HIGH,
      CHAR *Description,
      CARDINAL *Color
);
```

Spočte všechny filtry v kameře. Tento výpočet nepoužívá žádné zpětné volání. Funkce se volá od počátečního indexu 0 opakovaně až do doby, kdy funkce vrátí hodnotu *false*. Index se při každém volání inkrementuje. Tím se vypočte počet filtrů v přístroji. Řetězec s popisem se předává podobně, jako u volání *GetStringParameter()*. Parametr *Color* naznačuje název barvy, který může být použit v GUI aplikace.

```
BOOLEAN __cdecl SetFilter(
      CCamera *PCamera,
      CARDINAL index
);
```

Nastaví požadovaný filtr podle indexu.

8. Nový obslužný software

Nový obslužný software je navržen s ohledem na jednoduchost a rychlost práce. Jak bylo popsáno v předchozích kapitolách, originální software je sice velmi sofistikovaný, ale většina nabízených funkcí nemá pro naše účely žádné využití a naopak některé funkce vůbec nemá (např. živý náhled).

Program musí být schopen rozpoznat připojenou kameru (v případě, kdy je připojeno více kamer, nabídnout možnost vybrat tu, se kterou chceme pracovat), nastavit potřebné parametry, jako je teplota snímače, nastavení filtru, čas expozice a snímací režim. Stěžejní dovedností je režim živého náhledu, který umožní jednodušší zaostření obrazu. Nesmí chybět také možnost uložení exponovaného snímku do souboru v plné kvalitě.

Software byl vytvořen jako standalone aplikace pro 32bitové systémy Microsoft Windows© ve vývojovém prostředí Borland® C++ Builder® (verze 10.0.2288.42451 Update 2), za použití programovacího jazyka C++. Název programu je *G2CameraSoftware*.

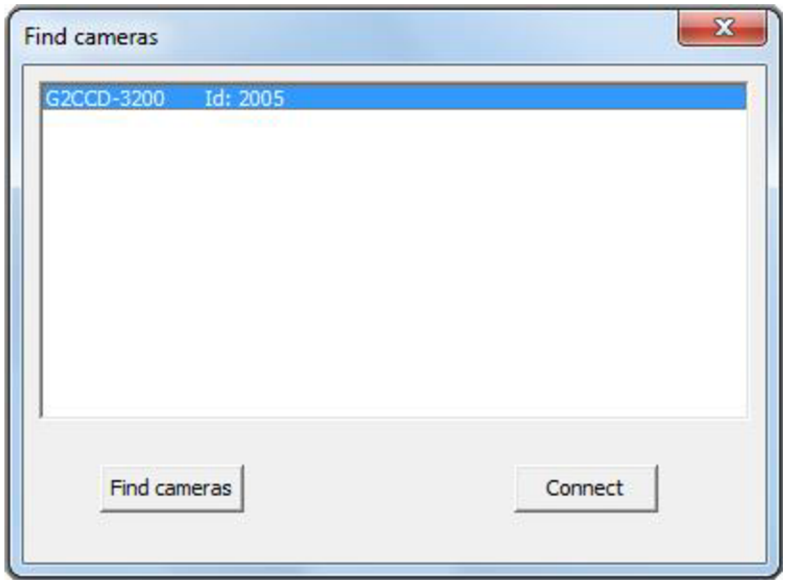
Aby bylo možné tento program používat, a navázat komunikaci s kamerou, je nutné nejprve nainstalovat hardwarové ovladače pro toto zařízení. Ovladače jsou k dispozici ke stažení na webových stránkách výrobce. Tento konkrétní typ kamery byl vyráběn v několika hardwarových revizích. Každá revize používá různé API knihovny, proto je program schopen obsluhovat kameru pouze jedné revize, a to revize č. 2. Vzhledem k zapouzdření používání knihoven by neměl být problém s přechodem na jiné DLL.

8.1 Uživatelské rozhraní

Náhled uživatelského rozhraní je patrný na obrázku 14. V levé části hlavního okna se nachází ovládací panel s nastavením kamery a expozice. Vpravo je zobrazovací část pořízených snímků a živého náhledu. Program také obsahuje hlavní menu s položkami *File*, *Camera* a *Help*, sloužící k uložení snímku do souboru, vypnutí programu, načtení a připojení kamer. Součástí nabídky je i jednoduché dialogové okno s informacemi „o aplikaci“.

Obrázek 14: Náhled uživatelského rozhraní nové aplikace.

Při spuštění programu není připojeno žádné zařízení, tudíž všechny ovládací prvky jsou zakázané. Nejprve je nutno spustit vyhledávání kamer pomocí nabídky *Camera-Connect*. Spustí se dialogové okno viz. obrázek 15. V tomto okně se nachází seznam načtených kamer a tlačítka *Find cameras* a *Connect*. Pokud ještě nebyl dříve vytvořen seznam kamer, je nutno spustit jejich vyhledávání tlačítkem *Find cameras*. Po doběhu vyhledávání se vypíše seznam připojených zařízení, kde si může uživatel vybrat požadovanou kameru kliknutím na příslušnou položku a stiskem tlačítka *Connect*. Pokud není zvolena žádná položka, tlačítko *Connect* je deaktivováno. V případě, že enumerace kamer proběhla dříve, není potřeba provádět vyhledávání a stačí pouze vybrat zařízení ze seznamu. Po potvrzení, se dialogové okno zavře a do ovládacího panelu se načtou příslušné hodnoty.



Obrázek 15: Dialogové okno pro vyhledávání a připojení kamer.

Ovládací panel slouží k přehlednému nastavení a zobrazení důležitých parametrů zařízení. Pokud připojená kamera nedisponuje určitou funkcí, je ovládací prvek dané funkce zakázán. Tento panel obsahuje název a id připojené kamery, zobrazuje teplotu snímače a umožňuje její nastavení. Rozsah hodnot lze nastavit v intervalu <-20°C;40°C>. Teplota se vykresluje každých 10 sekund. Dalšími prvky jsou nabídka použitých filtrů s tlačítkem pro nastavení a nabídka čtecího módu. Při připojení kamery je implicitně nastaven čtecí mód, který je první v seznamu. Dále v pořadí je pole pro nastavení expozičního času v jednotkách sekund (hodnoty musí být větší než nula, jinak nelze spustit expozici). Panel ještě obsahuje tlačítko *Expose* pro spuštění snímání obrazu. Expozici je možno spustit pouze pokud je vyplněn snímací čas. Samotnou expozici lze zastavit stejným tlačítkem, pokud ovšem expozice skončila a probíhá stahování obrázku do PC, není možno proces zastavit. Poslední tlačítko je pro spuštění živého náhledu. Funkce opět vyžaduje vyplnění expozičního času, jinak ji nelze spustit. Zastavení živého náhledu se provádí stejným tlačítkem, jako spuštění. Nad tlačítky *Expose* a *Life view* se nachází pole pro výpis aktuálního stavu kamery. Vypisuje se zde průběh expozice, načtení snímků, průběh živého náhledu (vykreslovaný segment) apod.

Živý náhled je realizován jako postupné vykreslování částí obrazu, které probíhá cyklicky, dokud není uživatelem zastaveno. Původní plán byl vykreslovat celý snímek najednou, ale pro nízkou rychlost načítání obrazu do PC, nebylo možné tuto techniku použít. Proto je snímek rozdělen na 6 stejných segmentů a během živého náhledu se postupně načítají a vykreslují tyto jednotlivé segmenty. Díky tomuto rozdělení trvá načtení snímku podstatně kratší dobu, protože se stahuje pouze jeho dílčí úsek. V praxi to umožňuje snadnější přeostřování obrazu. Další přidanou funkcí během živého náhledu, je zesílení načteného snímku. Díky tomu lze použít kratší expoziční časy, což zrychlí dobu překreslení. Zesílení signálu sice způsobí nežádoucí šum, ale ten v případě živého náhledu nezpůsobuje takový problém, protože slouží pouze k nastavení objektivu kamery před řádnou (dlouhou a bezšumovou) expozicí. Další popis živého náhledu je v kapitole testování.

Pro uložení vytvořeného snímku, slouží položka hlavní nabídky *File-Save*. Po stisku se objeví dialogové okno s nabídkou umístění souboru a možností zadat jeho název. Snímky se ukládají ve formátu *bmp*.

Rozměry panelu pro zobrazení snímku jsou v poměru 3:2, což odpovídá poměru stran použitých snímačů, takže nedochází k deformaci snímku. Snímky mají rozlišení v řádech megapixelů, což je mnohem více, než je rozlišení zobrazovacího panelu, proto zde dojde ke zmenšení snímku viz další kapitoly.

Pro zvýšení přehlednosti zdrojového kódu nové aplikace a také pro zvýšení bezpečnosti přístupu k některým proměnným, ukazatelům a funkcím, byly vytvořeny třídy, které zapouzdřují jednotlivé celky. Třídy jsou:

- *CameraDriver*,
- *CameraFinder*
- *Cameras*.

Díky použití těchto tříd, se zdrojový kód hlavního těla programu stane přehledným a snadno udržovatelným. Bez jejich použití, by byl velmi znesnadněn další vývoj aplikace, a také by z důvodu nepřehlednosti hrozil vznik závažných chyb programu, jako například neuvolňování alokované paměti.

8.2 Třída CameraDriver

Třída *CameraDriver* byla vytvořena pro zapouzdření práce s knihovnami a importovanými funkcemi pro ovládání kamery, které tvoří základ celé aplikace. Úkolem této třídy je načtení DLL knihovny, získání funkcí z této knihovny, a vytvoření metod, které budou odkazovat na její funkce takovým způsobem, aby jejich používání bylo jednodušší, než v případě přímého použití těchto funkcí přes ukazatele.

Zde jsou příslušné DLL knihovny načteny tzv. explicitně. Knihovny se neimportují do aplikace během překladu, ale jsou připojeny až během chodu programu. To umožňuje práci s nimi až v případě, kdy je třeba. V tomto případě dochází k importu knihoven po spuštění aplikace, těsně před vyhledáváním kamer připojených k ovládacímu PC.

Třída je obsažena v souborech *CameraDriver.h* a *CameraDriver.cpp*. Těsně před definicí samotné třídy, jsou ještě nadefinovány typy ukazatelů na jednotlivé funkce knihoven. Jejich ukázka je uvedena na obrázku 16. Kompletní výpis se nachází v příloze na CD. Vzhledem k tomu, že aplikace je vytvořena v jazyce C++, ale funkce DLL knihoven jsou kvůli kompatibilitě vytvořeny v jazyce C, bylo potřeba použít „C“ volací konvence *__cdecl*. V případě použití jiné konvence např. *__fastcall,* která je defaultní u použitého překladače, by jednotlivé návratové hodnoty funkcí byly špatně interpretovány a došlo by k chybě.

```
typedef void (__cdecl* LPFNDLLENUMERATE)(void ( *CallbackProc)(g2ccd::CARDINAL ));
typedef g2ccd::CCamera *(__cdecl* LPFNDLLINITIALIZE)(g2ccd::CARDINAL);
typedef void (__cdecl* LPFNDLLRELEASE) (g2ccd::CCamera*);
typedef void (__cdecl* LPFNDLLREGISTERNOTIFYHWND) (g2ccd::CCamera *PCamera,
                                                   g2ccd::ADDRESS NotifyHWND);
...
```

Obrázek 16: Ukázka definicí několika vybraných ukazatelů na funkce.

Kompletní definice třídy *CameraDriver*, je uvedena v příloze. *Private* (skryté) položky třídy jsou jednotlivé ukazatele na funkce knihoven. Pomocí těchto ukazatelů, se volají funkce pro ovládání kamery. Jako *public* (veřejné), jsou nadefinovány konstruktor, destruktor a jednotlivé metody, jež představují importované funkce.

V těle konstruktoru dochází nejprve k načtení knihovny *g2ccd2.dll* funkcí *LoadLibrary()*. Ta vrátí ukazatel na knihovnu (její handler). V případě selhání, je vrácená výjimka, která se odchytává při volání konstruktoru v těle GUI. Následují bloky pro načtení jednotlivých funkcí do příslušných *private* ukazatelů. Načítání funkcí, je opět ošetřeno zachytáváním výjimek, pro případ selhání běhu konstruktoru. Ukázka části konstruktoru je na obrázku 17.

```
CameraDriver::CameraDriver()
{
 this->hDLL = LoadLibrary("g2ccd2.dll");

 if (this->hDLL == NULL)
 {
  throw "Loading of \"g2ccd2.dll\" failed!";
 }

 this->lpfnDllEnumerate = (LPFNDLLENUMERATE)GetProcAddress(this->hDLL,"Enumerate");
 if (this->lpfnDllEnumerate == NULL)
  {
   FreeLibrary(this->hDLL);
   throw "Loading functions from \"g2ccd2.dll\" failed!";
  }

 ...
```

Obrázek 17: Ukázka části konstruktoru třídy CameraDriver.

Tělo destruktoru, obsahuje uvolnění instance připojené DLL knihovny. Pokud by se tento handler neuvolnil, knihovna by zůstala otevřená v paměti, což je nežádoucí.

Jednotlivé metody této třídy, jak již bylo poznamenáno, zapouzdřují funkce, které se načetly z knihoven. Definice některých metod je vidět na obrázku 18. Parametry každé metody odpovídají parametrům volaných importovaných funkcí. Výhodou použití zapouzdření knihovních funkcí v metodách třídy CameraDriver je, že již není nutné funkce volat přes jejich ukazatel, a také došlo k přetypování vstupních a výstupních parametrů a návratových hodnot. Tato drobnost Nebo vytváří další úroveň abstrakce, která odstiňuje klientské třídy od implementačních detailů. Některé funkce vracejí hodnotu datového typu g2ccd::BOOLEAN, který je nadefinován jako unsigned char. Bylo zjištěno, že návratové hodnoty jsou pouze {0,1}, proto mohlo dojít k přetypování na standardní bool.

```
bool CameraDriver::GetValue(g2ccd::CCamera *PCamera, unsigned Index, float *Value)
{
 return lpfnDllGetValue(PCamera, Index, Value);
}

void CameraDriver::SetTemperature(g2ccd::CCamera *PCamera, float Temperature)
{
 lpfnDllSetTemperature(PCamera, Temperature);
}

bool CameraDriver::SetBinning(g2ccd::CCamera *PCamera, unsigned x, unsigned y)
{
 return lpfnDllSetBinning(PCamera, x, y);
}
```

Obrázek 18: Ukázka několika metod třídy CameraDriver.

Na obrázku 18 je příklad několika metod třídy CameraDriver. V těle metod většinou dojde pouze k volání odpovídajících funkcí pomocí jejich skrytých ukazatelů. Dochází také k přetypování vstupních a výstupních parametrů a návratových hodnot. Ve většině případů lze datové typy přetypovat automaticky, u některých metod (například *GetBooleanParameter*()) muselo být v těle dané metody provedeno přetypování explicitně.

Použitá knihovna *g2ccd2.dll*, je knihovna odpovídající hardwarové revizi kamery č. 2, která na rozdíl od té předchozí používá rozhraní USB 2.0. Pro tuto revizi kamery je program vyvíjen. V nové aplikaci je použita poslední verze knihoven, které jsou obsaženy v originálním programu SIPS 2.1. Soubor *g2ccd2.dll* musí být uložen v adresáři s exe souborem aplikace, jinak nedojde k naimportování knihoven. Rovněž zde musí být umístěny soubory *vproc4.dll* a *vtools4.dll*, na nichž je soubor *g2ccd2.dll* závislý.

8.3 Třída CameraFinder

Třída *CameraFinder* byla vytvořena pro zapouzdření operací vyhledání připojených zařízení k ovládacímu PC. Jedná se o jednoduchou třídu, jejíž účel je vyhledat připojené kamery a vrátit jejich handlery (manipulátory).

Třída je definována v souborech *CameraFinder.h* a *CameraFinder.cpp*. Jelikož tato třída používá funkce importovaných knihoven, je závislá na existenci třídy *CameraDriver*. Ukazatel na instanci třídy *CameraDriver* je nutno vložit jako vstupní parametr konstruktoru této třídy. Proto musí být v těle programu vytvořena nejprve instance třídy *CameraDriver* a až poté může být vytvořena instance této třídy.

Skryté položky třídy jsou ukazatel na Instanci třídy *CameraDriver* (ta zprostředkovává funkce knihoven), a parametr *CameraNum*, který slouží pro uložení počtu připojených kamer během jejich vyhledávání.

Veřejné metody třídy jsou *konstruktor*, *destruktor*, *GetCameraNumber()* (vrací počet připojených kamer), a *FindCameras()*. Nejdůležitější metodou třídy je metoda

FindCameras(). Ta má za úkol provést vyhledání připojených kamer, získání jejich manipulátorů a navrácení pole těchto manipulátorů pro všechna připojená zařízení. Výpis definice třídy *CameraFinder* je na obrázku 19.

```
void EnumerateCallback (unsigned Id);


class CameraFinder
{
 private:

  CameraDriver *Driver;
  unsigned int CameraNum;

 public:

  CameraFinder(CameraDriver *Driver);
 ~CameraFinder();

  g2ccd::CCamera** FindCameras();
  unsigned GetCameraNumber();

};
```

Obrázek 19: Hlavička třídy CameraFinder. Na prvním řádku je patrná definice globální finkce zpětného volání EnumerateCallback().

Před definicí samotné třídy je ještě vytvořena globální funkce zpětného volání *EnumerateCallback()*. Funkce je volána cyklicky tolikrát, kolik zařízení je připojeno k PC. Její parametr Id představuje identifikační číslo zařízení, které je pro každou kameru unikátní a je uvedeno na štítku přístroje. Pomocí těchto čísel lze kamery snadno rozlišovat. Počet připojených kamer se určí podle počtu volání této funkce.

V konstruktoru této třídy se pouze předá ukazatel na instanci třídy *CameraDriver* a počítadlo připojených kamer se vynuluje. Destruktor je prázdný, protože alokovaná paměť s polem manipulátorů se předá do další třídy, která tyto manipulátory uvolní ve svém destruktoru (vysvětleno v dalších kapitolách). Tělo metody *GetCameraNumber()* pouze vrací počet enumerovaných kamer.

V těle metody *FindCameras()* dochází nejprve k vytvoření statického pole Id kamer. Toto pole má velikost 100 prvků, což je maximální počet kamer, které je program schopen obsluhovat. Dále je zavolána funkce *Enumerate()*, jejímž parametrem je funkce *EnumerateCallback()*. Funkce v argumentu je volána cyklicky pro každou kameru, která je připojena a v jejím těle se uloží Id dané kamery a zvýší se počet připojených kamer. Samotná enumerace běží ve vlastním vlákně, proto je nutné počkat (pomocí systémové funkce *Sleep()*), až vlákno doběhne, a poté teprve manipulovat se získanými hodnotami. Doba čekání byla stanovena na 2 sekundy (samotné hledání trvá v řádech milisekund), proto je zde poměrně velká rezerva. Po doběhu enumerace se vytvoří dynamické pole

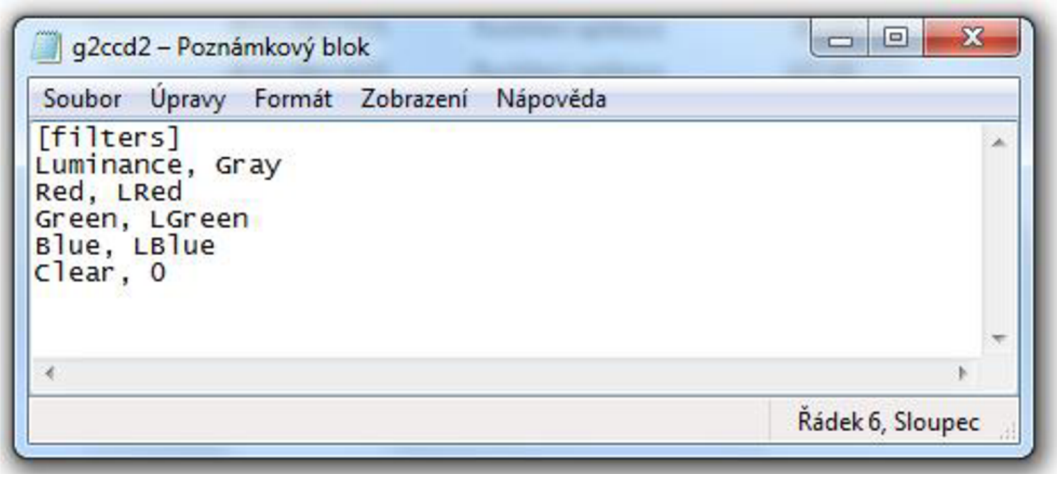
ukazatelů na strukturu *CCamera,* což jsou manipulátory každé kamery a uvolní se alokovaná paměť pomocných proměnných. V závěru metody se předá, jako návratová hodnota, pole ukazatelů na kamery (pole manipulátorů). Výpis této metody je uveden v příloze na CD.

8.4 Třída Camera

Třída kamera je fasádou nad nízkoúrovňovou strukturou *CCamera* a DLL funkcemi. Odstiňuje klientské třídy od implementačních detailů. Každá její instance představuje právě jednu připojenou kameru. Jejím hlavním úkolem je zapouzdřit všechny potřebné operace konkrétní kamery a uložit v sobě veškeré důležité údaje o připojeném zařízení (viz níže).

Třída je závislá na existenci instance třídy *CameraDriver*. *Camera* potřebuje pro konstrukci instance handler daného zařízení, který předá třída *CameraFinder* jako návratovou hodnotu při vyhledání a připojování kamer. Po tomto předání již není potřebná existence instance třídy *CameraFinder*. Ta totiž při své destrukci neuvolňuje handlery připojených zařízení. Obvykle sice bývá uvolnění v destruktoru dané třídy, zde ale přebírá (nad vytvořenými handlery) zodpovědnost právě třída *Camera*. V hlavním těle programu je již při vyhledávání a připojení kamer pro každou nalezenou kameru vytvořena instance třídy *Camera*. Díky tomu je znemožněno, aby zůstal handler některé kamery zapomenut a neuvolněn, což by způsobilo pád aplikace.

Konstruktor třídy potřebuje při svém volání jako vstupní parametry ukazatel na ovladač kamery, a handler připojené kamery. Ukazatel na ovladač je vlastně ukazatel na instanci třídy *CameraDriver*, a handler dané kamery je získán enumerací kamer třídou *CameraFinder*. Tyto parametry jsou uloženy jako skryté položky třídy. V konstruktoru dochází k inicializaci hodnot, které představují vlastnosti kamery, tj. načtení rozlišení snímače, popisu kamery, nebo její id. Dále se provede enumerace čtecích módů a filtrů (pokud jimi zařízení disponuje). Filtry a čtecí módy jsou uloženy ve skrytých položkách třídy. Jejich předání je realizováno pomocí samostatných metod. Proces enumerace filtrů je poněkud odlišný od enumerace čtecích módů. Zatímco kamera disponuje informacemi o všech čtecích módech, které může předat při volání funkce pro jejich vyhledání, nemá žádné informace o použitých filtrech na filtrovém kole. Kamera pouze ví, zdali jsou filtry použity, ale neví jaké filtry, a v jakém pořadí jsou nainstalovány. To se může lišit od potřeb uživatele. Proto je seznam filtrů načten z inicializačního souboru *g2ccd2.ini*. O načtení informací z tohoto souboru se stará sama funkce *EnumerateFilters()*. Struktura tohoto souboru je velmi jednoduchá a lze ji vidět na obrázku 20.



Obrázek 20: Struktura inicializačního souboru se seznamem filtrů.

Destruktor třídy má za úkol uvolnit alokovanou paměť (např. řetězec *Description* obsahující popis kamery). Také zde dochází k uvolnění handleru kamery, který byl získán během enumerace voláním funkce *Release()*. Třída tedy převzala zodpovědnost za uvolnění tohoto manipulátoru (viz. výše).

Metody třídy jsou postaveny tak, aby umožňovaly jednoduchou práci s funkcemi kamery. Funkce importovaných knihoven potřebují jako vstupní argumenty handler kamery a další individuální parametry, které se liší podle připojeného zařízení (např. rozlišení CCD snímače). Tyto parametry jsou zde načteny v konstruktoru a dále je není potřeba vždy pro danou funkci zadávat. Díky tomuto zapouzdření je zajištěno snadnější ovládání kamery. Také je zamezeno vzniku chyb, jako například špatné zadání rozlišení snímače, které by mělo za následek chybu běhu programu.

Detailní výpis třídy je uveden v příloze na CD. Následující tabulka obsahuje přehled metod s popisem jejich funkce.

Tabulka 3:Výpis metod třídy Camera s popisem jejich funkce.

`Camera(CameraDriver *driver, CCamera *pcamera)`	konstruktor
`~Camera()`	destruktor
`float GetTemperature()`	Vrací teplotu snímače.
`void SetTemperature(float temperature)`	Nastaví požadovanou teplotu snímače ve stupních celsia..
`unsigned GetChipW()`	Vrací horizontální počet pixelů snímače.
`unsigned GetChipD()`	Vrací vertikální počet pixelů snímače.
`unsigned GetId()`	Vrací Id zařízení.
`char* GetCameraDescription()`	Vrací popis zařízení.
`void Clear()`	Vymaže obraz ze snímače.
`void Open()`	Otevře závěrku.
`void Close()`	Zavře závěrku.
`bool GetImage(unsigned int Size, void *Buffer)`	Načte obraz z kamery. Size představuje velikost bufferu v Bytech, Buffer je adresa paměti pro uložení obrazu. Pokud selže načtení snímku vrátí hodnotu *false*.
`bool IsCooler()`	Vrací *true*, pokud zařízení disponuje chladičem.
`bool IsFilter()`	Vrací *true*, pokud má zařízení filtry.
`bool IsReadModes()`	Vrací *true*, pokud zařízení podporuje čtecí módy.
`char** GetReadModes()`	Vrátí čtecí módy zařízení.
`int GetNumberReadModes()`	Vrátí počet čtecích módů.
`bool SetReadMode(unsigned Mode)`	Nastaví požadovaný čtecí mód. Pokud nastavení selže, vrátí *false*.
`char* GetFilters()`	Vrátí seznam filtrů kamery.
`int GetNumberFilters()`	Vrátí počet filtrů.
`bool SetFilter(unsigned Index)`	Nastaví požadovaný filtr. Pokud se nasavení nepovede, vrátí *false*.
`bool IsSubFrame()`	Pokud má kamera možnost čtení výřezu snímku vrátí true.

8.5 Hlavní tělo programu

Tato kapitola popisuje strukturu hlavního těla programu, jednotlivé metody a řídicí události, jejich funkce a princip činnosti. Protože zdrojový kód aplikace je velmi rozsáhlý, bude jeho kompletní výpis uveden v příloze na CD. Celá aplikace se skládá ze tří formulářů. Hlavní okno programu, formulář pro vyhledání a připojení kamer, a jednoduchý formulář s nápovědou.

Formulář pro připojení kamer je vytvořen v souborech *Connect.h*, *Connect.cpp* a *Connect.dfm*. Jedná se o třídu *TForm3*. Třída obsahuje pouze tři objekty *ListBox* pro výpis kamer a dvě tlačítka *FindCameras* a *Connect*. Dále je zde nadefinováno pět metod. Nejdůležitější jsou *ButtonFindCamerasClick()* a *ButtonConnectClick()*.

V těle metody *ButtonFindCamerasClick()* nejprve dochází k nastavení ovládacích prvků, pokud je již připojeno nějaké zařízení, odpojí se, a dále proběhne vyhledání všech připojených zařízení. Při běhu metody se vytvoří instance třídy *CameraFinde*r, která se na konci opět uvolní poté, co se získané manipulátory předají do instancí třídy *Camera*. Načtené pole kamer se uloží do ukazatele *Cameras*, který je globální proměnnou hlavního formuláře. Totéž se provede s uložením počtu kamer do proměnné *CameraNumber*. Při vyhledávání se do příslušného pole vypíše seznam zařízení pro pozdější výběr.

Událost *ButtonConnectClick()* nejprve nastaví do globální proměnné *ActiveCamera* index aktivní kamery. Všechny připojené kamery jsou uloženy v poli a tato proměnná uvádí, se kterým zařízením chceme pracovat. Pokud bychom chtěli pracovat s jinou kamerou ze seznamu, stačí pouze změnit index aktivní kamery. V rámci této události dále proběhne načtení a zobrazení popisu kamery, enumerace filtrů, čtecích módů a načtení teploty. Tyto enumerace a načtení proběhnou, pokud zařízení má dané možnosti. Pokud zařízení disponuje chlazením, nastaví se automaticky teplota na 20°C. V případě, že kamera nemá některou funkci, jsou její ovládací prvky deaktivovány.

Další metody (události) této třídy jsou velmi jednoduché a mají za úkol pouze nastavit dané prostředí tak, aby nebylo možné špatnou obsluhou provést krok, který není žádoucí (např spustit událost *ButtonConnectClick()* v době, kdy ještě není vybrána žádná kamera). Hlavička třídy *TForm3* (Connect.h) je uvedena na obrázku 21.

```
void EnumerateCallback (unsigned Id);


class CameraFinder
{
 private:

  CameraDriver *Driver;
  unsigned int CameraNum;

 public:

  CameraFinder(CameraDriver *Driver);
 ~CameraFinder();

  g2ccd::CCamera** FindCameras();
  unsigned GetCameraNumber();

};
```

Obrázek 21: Hlavička třídy TForm3 (Connect.h).

Nejdůležitější částí je hlavní formulář vytvořený v souborech *Main.h*, *Main.cpp,* a *Main.dfm*. Jedná se o třídu *TForm1*. Formulář obsahuje hlavní menu, které bylo popsáno výše.

Při konstrukci tohoto formuláře (spuštění programu) dojde k zakázání ovládacích prvků a všech časovačů. Tak nemohou nastat události pro zařízení, které ještě nebylo připojeno. V dalším kroku se vytvoří instance třídy *CameraDriver* (načte se DLL knihovna). Při destrukci formuláře se uvolní veškerá alokovaná paměť, a pokud jsou připojené kamery, odpojí se.

Nastavení teploty snímače zajišťují prvky *EditSetTemperature* (textové pole) a tlačítko *ButtonSetTemperature*. Zadávání požadované teploty je ošetřeno při stisku tlačítka, pokud je hodnota mimo rozsah <-20°C;40°C>, objeví se okno s chybovou hláškou, obsah pole se vymaže a kurzor se nastaví zpět do tohoto pole. Pro vypisování teploty slouží objekt *LabelTemperatureValue*, do kterého se hodnota vypisuje každých 10 sekund pomocí časovače *TimerTemperature*.

Pro nastavení požadovaného filtru slouží rozbalovací nabídka *ListBoxFilters* a tlačítko *ButtonSetFilter*. Enumerace probíhá při připojování kamery. Další ovládací prvek je *ListBoxReadingModes*, do kterého se podobně jako v případě filtrů, uloží při vyhledávání podporované čtecí módy. Potřebný mód se spustí stiskem tlačítka *ButtonSetReadMode*.

Důležitým objektem je pole pro zadávání expozičního času (*EditExposeTime*). Zde probíhá kontrola správného vyplnění pomocí události *OnExit()*. Kontrola odhalí, jestli je číslo zadáno jako kladné, pokud ne, vrátí kurzor zpět do pole a vymaže jeho starý obsah. Další

kontrolou je, zda pole není prázdné při spuštění expozice, nebo živého náhledu. Popis bude uveden níže.

Ovládací panel ještě obsahuje tlačítka *ButtonExpose* a *ButtonLifeView*. Stiskem *ButtonExpose* je spouštěna (zastavována) expozice. Pro rozpoznání jestli probíhá expozice, či nikoli, slouží globální proměnná *expose*, má-li hodnotu *true*, expozice zrovna probíhá. Po stisku tlačítka se tedy ověří, neběží-li zrovna expozice. Pokud ano, je zastavena a ovládací prvky jsou opět aktivovány. Neběží-li expozice, jsou deaktivovány určité ovládací prvky, které by mohly ovlivnit probíhající snímání obrazu. Dále je vymazán obsah snímače, otevřena závěrka a spuštěn časovač *TimerExpose*, který odměřuje dobu expozice a nastaví se příslušné ovládací prvky. Tím tato událost končí a zodpovědnost za probíhající expozici přebírá časovač *TimerExpose*. Jakmile doběhne tento časovač, spustí se událost *TimerExposeTimer()*, kde se dokončí snímání obrazu. Opět se nastaví potřebné ovl. prvky, zavře se závěrka kamery, načte se snímek do globální proměnné *Buffer* a dojde k úpravě snímku pro zobrazení. Úprava spočívá v přepočtení rozsahu snímku na rozsah, který používá příslušný zobrazovací panel (bude popsáno v dalších odstavcích). Poté se spustí rutina na zobrazení snímku v panelu s náhledem, hodnota proměnné expose se překlopí na *false* a znovu se aktivují všechny potřebné ovládací prvky.

Na přepočtení rozsahu byla pro přehlednost vytvořena funkce *UniformRange()*, která projde celé pole s načtenými obrazovými body a přepočítá jejich hodnotu na rozsah 8 bitů.

O vykreslování obrazu do náhledu se stará funkce *DrawImage()*. Pořízený snímek má větší rozměry, než panel pro vykreslení náhledu, proto musí být nejprve zmenšen a poté zobrazen. V tomto případě probíhá vykreslení tak, že se vypočte poměr originálních rozměrů a rozměrů zobrazovacího pole, a cyklus zobrazí pouze pixely, které jsou od sebe vzdáleny právě o vypočtený poměr. Tím se přebývající pixely vynechají a tak je snímek zmenšen.

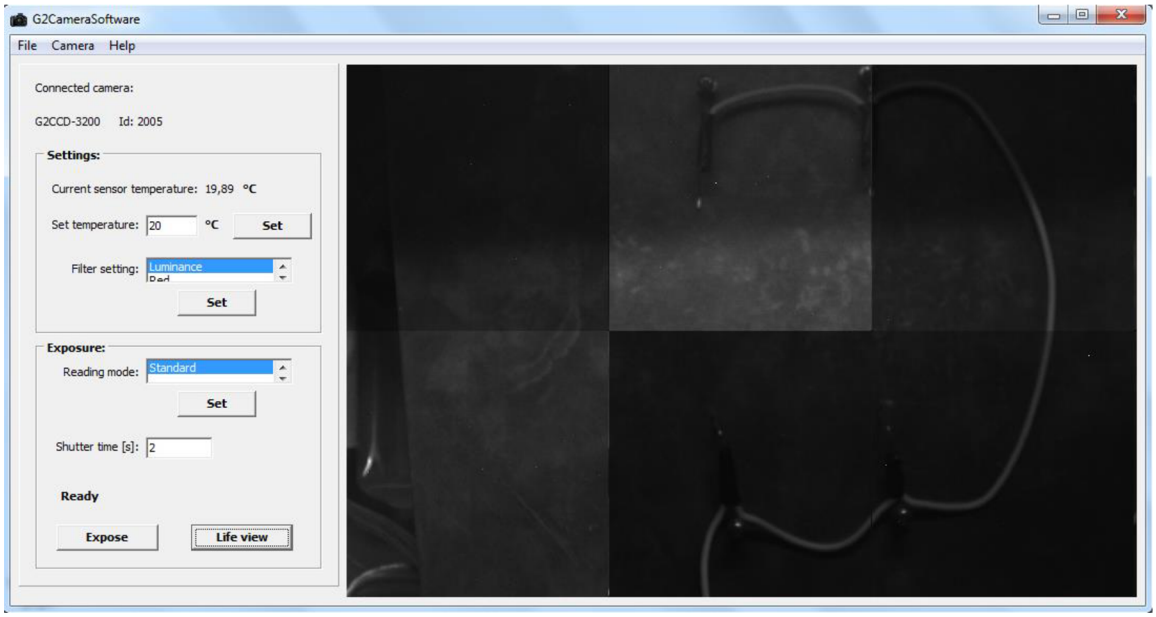
Pokud bude uživatel chtít uložit snímek do souboru, stiskem *File-Save* se aktivuje událost *Save1Click*. V těle této události dojde k vykreslení snímku v plném rozlišení do neviditelného pole *TImage* (podobně jako u zobrazení náhledu) a k volání dialogového okna sloužícího k zadání názvu souboru. Po potvrzení názvu se uloží bitmapa pomocí funkce *SaveToFile()*. Díky tomu, že pole *TImage* je nastaveno jako neviditelné, může se definovat jeho velikost přesně na velikost originálního snímku a přitom neovlivnit velikost hlavního okna programu.

Živý náhled pracuje velmi podobně jako funkce expozice, také využívá časovače pro odměření expoziční doby, ale kvůli požadavkům na rychlost muselo být snímání obrazu realizováno poněkud odlišným způsobem. Při spuštění události se zkontroluje, jestli kamera podporuje čtení výřezu snímku, pokud ne, není možné živý náhled použít. V opačném případě dojde opět k inicializaci zakázáním určitých ovládacích prvků, otevře se závěrka, spustí časovač a opět nastaví potřebné ovl. prvky (včetně proměnné *expose*). Po

doběhu časovače se spustí obslužná rutina *TimerLifeViewTimer()*. Zde se provede zavření závěrky, zastavení časovače a načtení obrázku funkcí *GetImageLV()*. Původně se snímek načítal celý. Načtení celého snímku trvá přibližně 12 sekund, což je pro živý náhled spolu s expozičním časem příliš dlouhá doba. Navíc bylo nutno exponovat minimálně 10 sekund. V součtu je čas vykreslení nového snímku příliš dlouhý. Proto je obraz rozdělen na 6 stejných segmentů a při běhu jedné expozice živého náhledu se provede vyfocení a vykreslení vždy jen jedné šestiny snímku. K tomu slouží příkaz switch, který pro každý segment nastaví přesné úseky snímané části fotografie a pozici pro vykreslení v panelu náhledu. Segment je určen globální proměnnou *Segment*, která se v každém cyklu inkrementuje (v případě nejvyšší hodnoty se nastaví na nulu a začne od prvního segmentu). Po načtení dané části snímku, se provede zesílení signálu a přepočet na potřebný rozsah. Díky zesílení je možno zkrátit dobu expozice, nevýhodou je však zvýšení šumu. Na konci této události se opět zavolá událost stisku tlačítka *ButtonLifeView* a celý cyklus proběhne znovu. Překreslování probíhá stále dokola, dokud jej uživatel nezastaví.

8.6 Testování

Testování probíhalo přímo v temné komoře s otevřenými dvířky, kdy dochází k přípravě expozice za použití živého náhledu.



Obrázek 22: Živý náhled. Zde jsou vidět jednotlivé segmenty obrazu.

Expoziční čas při standardní expozici v použité komoře s otevřenými dvířky bylo nutno nastavit na hodnotu minimálně 10 sekund. Ideální je 20sekundová expozice. Ovšem při 10sekundové expozici lze vidět obraz poměrně dobře, což je pro nastavení kamery ideální. Načtení celého snímku trvá přibližně 12 sekund, což je pro živý náhled spolu s expozičním časem nedostatečně krátká doba. Použitím funkce zesílení signálu byl dobře rozpoznatelný obraz získán již při 2sekundové expozici. Tato funkce je použita v živém náhledu. Zesílením dochází ke zvýšení míry šumu v obraze. Šum není natolik znatelný, aby mohl

zhoršit proces ostření. Bylo tedy zjištěno, že při použití zesílení se při živém náhledu zkrátí minimální nutná doba expozice z 10 sekund na 2 sekundy.

Jak již bylo uvedeno, načtení celého snímku trvá 12 sekund, proto byla pro režim živého náhledu vytvořena segmentace snímku, kdy se postupně exponuje a vykresluje šest částí obrazu. Díky tomu se do PC načítá pouze šestina snímku, což zrychluje dobu čtení. Testováním bylo zjištěno, že čas pro načtení jedné části je přibližně 5 sekund. To v součtu s dobou expozice (2 sekundy) znamená překreslení části snímku za 7 sekund. Doba vykreslení obrazu a obsluhy kamery (čištění snímače, otevření a zavření závěrky…) je téměř zanedbatelná, proto zde není uvedena.

Další funkcí, kterou bylo nutno otestovat v praxi, je přepočet dynamického rozsahu načteného snímku při vykreslování v GUI, nebo ukládání do bitmapového souboru. Ukázalo se, že při vytvoření stejné expozice (stejný expoziční čas, čtecí režim, míra osvětlení, ohnisková vzdálenost objektivu, clona a rovina zaostření) byly snímky z originálního softwaru a tímto novým programem stejné.

Snímek načtený z kamery má minimálně dvojnásobný rozměr, nežli je rozměr vykreslovacího panelu aplikace, proto dochází při vykreslování ke zmenšení snímku, což ale nemá vliv na kvalitu obrazu. Do souboru se pak snímek ukládá v plném rozlišení.

Pro účely testování aplikace nebylo k dispozici větší množství kamer, tudíž nemohl být otestován maximální počet připojených zařízení, a ani přepínání mezi kamerami, které jsou k PC připojené. Program je však vytvořen takovým způsobem, že pokud by rutina na vyhledávání a připojování kamer nebyla funkční, nebylo by možné připojit ani jednu kameru.

Protože byl program vyvíjen na PC s operačním systémem Windows 7 (64bit), a aplikace se bude používat převážně na PC s Windows XP, bylo nutno program otestovat na tomto systému. Nebyl nalezen žádný problém a aplikace fungovala standardně.

Poslední vlastností programu je změna velikosti náhledu během změny rozměrů hlavního okna. Překreslením okna se zavolá událost *OnResize()*. V těle této události dojde k přenastavení velikosti panelu pro náhled a novému vykreslení snímku. V případě, kdy je současně spuštěn živý náhled, se velikost náhledového panelu sice zvětší, ale dojde k vymazání celé plochy a k vykreslení obrazu dojde až s vytvořením snímku dalšího segmentu.

9. Závěr

Jak bylo uvedeno v úvodních kapitolách, nejjednodušší řešení, jak zrychlit proces diagnostiky (konkrétně ostření obrazu) v temné komoře, je použití manuálního ostření v kombinaci s živým náhledem snímaného obrazu. Proto byl vytvořen nový ovládací program, který umí základní nastavení přístroje, vytvoření, zobrazení a uložení expozice a živý náhled. Program se podařilo uskutečnit jako standalone aplikaci, která je snadno přenositelná a lze ji používat na PC s operačním systémem Windows. Aplikace je vytvořena jako 32bitová, ale lze ji (díky zpětné kompatibilitě) použít na nových (64bitových) verzí operačního systému Windows.

Podařilo se splnit hlavní část zadání, a to sice urychlit diagnostiku fotovoltaických článků v temné komoře urychlením přípravné fáze. Během testování bylo zjištěno, že při otevřené komoře v režimu živého náhledu stačí pro expozici přibližně 2 sekundy (díky funkci zesílení), tím se, spolu s použitím segmentace snímku, podařilo zkrátit čas vykreslení jednoho segmentu na přibližně 7 sekund. To umožňuje rychlejší zaostření a nastavení ohniskové vzdálenosti objektivu.

Ukládání snímku používá obrazový formát bmp, protože funkce pro vytváření těchto souborů je standardně obsažena ve VCL (Visual Component Library) knihovnách použitého vývojového prostředí. V případě potřeby by bylo možné doplnit možnost ukládání v jiných obrazových formátech.

Během vytváření expozice bylo kvůli vykreslování a ukládání bitmapového obrázku nutno zmenšit dynamický rozsah snímku. Kamera odesílá do obslužného PC obraz v dynamickém rozsahu 16 bitů (tj. 0-65535), avšak struktura *bmp* souboru (stejně tak jako objekt třídy TImage který slouží k vykreslování obrazu) umožňuje pouze rozsah 8bitový (0-255). Po přepočítání obrazu na menší rozsah nebyla ztráta dynamického rozsahu znatelná, což jen potvrzuje popis v kapitole testování, kde je uvedeno, že originální software musí tento přepočet udělat také.

Při testování bylo k dispozici pouze jedno zařízení (viz kapitola testování), proto nebylo možné otestovat v praxi práci s více zařízeními najednou.

Původně byla celá rutina expozice vložena do jedné události (tlačítko *Expose*). To ovšem znamenalo problém při dlouhých expozicích. Program totiž používal pro odměření času snímání systémovou funkci čekání *Sleep()*, která znemožňovala (po dobu expozice) doběh události tlačítka. To mělo za následek zamrznutí programu, takže nebylo možné obsluhovat např. nastavení teploty snímače. V praxi jsou použity expoziční časy v řádech minut, čili program přestal odpovídat a operační systém jej chtěl ukončit. Řešením tohoto problému bylo použití objektu *Timer*, který odměřoval expoziční čas a až teprve po doběhu tohoto času byla zavřena závěrka a načten obraz.

Během vývoje aplikace se objevilo několik problémů. Nejprve nebylo možné spustit komunikaci mezi ovládacím PC a kamerou (nefungovala enumerace zařízení). Ani při připojení kamery přes její (již známé) id nebylo možné načíst parametry tohoto zařízení, ani nastavit hodnoty, jako např. teplota chladicího systému. Po podrobném zkoumání byla objevena chyba v připojování knihoven. Zařízení G2-3200 bylo vyrobeno ve více hardwarových revizích. Původní revize používala právě používanou knihovnu g2ccd.dll. Ovšem námi používaná kamera je novějšího typu (revize 2) a používá knihovnu g2ccd2.dll. Změnou importované knihovny byl problém odstraněn. Další problém byl při změně používané kamery, anebo ukončování práce s ní. Důvodem bylo zapomenuté volání uvolnění kamery funkcí *Release()*. Proto vždy docházelo k vyvolání výjimky a pádu aplikace. Tyto problémy se díky konzultacím s výrobcem podařilo odstranit.

Vytvořenou aplikaci je možné dále rozvíjet. Jedním z návrhů pro další vývoj je například možnost ovládání více typů kamer než je G2-3200. To by znamenalo používat více importovaných knihoven, které obsluhují dané typy zařízení. Aplikace by také mohla umožňovat ukládání snímků do více obrazových formátů jako například *jpeg*, či *png*. Další novou vlastností by mohlo být zobrazení histogramu, možnost nastavitelného zesílení obrazu, jednoduchých výřezů, či zobrazování více snímků najednou (např. pomocí záložek). Jednou z užitečných vlastností by mohla být volba zobrazení snímku. Například ve dvou režimech jako stoprocentní zvětšení s možností posuvu, a zobrazení celého snímku (současný stav).

Seznam literatury

1. STEINER, Patrik. *Charakterizace elektrických vlastností solárních článků pomocí CCD kamery*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 54 s. Vedoucí diplomové práce Ing. Jiří Vaněk, Ph.D.

2. VÁVRA, Václav; ŠTELCL, Jindřich. *Základy digitálních dokumentačních technik a možnosti jejich využití* [online]. 2008 [cit. 2011-12-02]. Základní principy expozice, práce s digitální technikou. Dostupné z WWW: <http://is.muni.cz/do/1499/el/estud/prif/ps08/digitech/web/kapitola2/3.html>.

3. *Digimanie* [online]. 1998 [cit. 2011-12-05]. Dostupné z WWW: <www.digimanie.cz>.

4. Soubor:Blooming ccd.jpg. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 2.10.2007, last modified on 9.5.2011 [cit. 2011-12-05]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Soubor:Blooming_ccd.jpg>

5. SPRING, Kenneth R. ; FELLERS, Thomas J. ; DAVIDSON, Michael W. . *Nikon Microscopy* [online]. 2011 [cit. 2011-12-06]. Introduction to Charge-Coupled Devices (CCDs). Dostupné z WWW: <http://www.microscopyu.com/articles/digitalimaging/ccdintro.html>

6. Charge-coupled device. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2011-12-06]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Charge-coupled_device>

7. MORAVSKÉ PŘÍSTROJE. *G2 CCD Uživatelská příručka*. Zlín, 2010.

8. Úvod do techniky CCD čipů. *CCD kamery pro astronomii* [online].21-09-2011 [cit. 2012-04-17]. Dostupné z: http://ccd.mii.cz/art?id=303&lang=405

9. MORAVSKÉ PŘÍSTROJE. *Manuál k ovladačům kamery Gx*. Zlín, 2011.

10. MATOUŠEK, David. *C Builder: řešené příklady*. 1. vyd. Praha: BEN - technická literatura, 2010, [94] s. ISBN 978-80-7300-258-9.

11. Linking Explicitly. MICROSOFT. *Microsoft Developer Network* [online]. © 2013 [cit. 2013-05-11]. Dostupné z: http://msdn.microsoft.com/en-us/library/784bt7z7.aspx

12. Pointers. *Cplusplus.com* [online]. 2000-2013 [cit. 2013-05-11]. Dostupné z: http://www.cplusplus.com/doc/tutorial/pointers/

13. C++ výjimky. *Linuxsoft.cz* [online]. 2003-2013 [cit. 2013-05-16]. Dostupné z: http://www.linuxsoft.cz/article.php?id_article=1801

Seznam zkratek

A/D	Analogově/Digitální
AC	Alternating Current – střídavý proud
API	Application Programming Interface – rozhraní pro programování aplikací
CCD	Charge-Coupled Device – zařízení s vázanými náboji
CMOS	Complementary Metal–Oxide–Semiconductor
CMY	Cyan, Magenta, Yellow – modrozelená, purpurová, žlutá
CMYG	Cyan, Magenta, Yellow Green - modrozelená, purpurová, žlutá, zelená
DC	Direct Current – stejnosměrný proud
DLL	Dynamic Link Library – dynamicky připojená knihovna
EV	Exposure Value – expoziční jednotka
GUI	Graphic User Interface – grafické uživatelské rozhraní
Id	Identification – identifikační číslo
ISO	International Stantards Organization – mezinárodní organizace pro standardizaci
jpeg	join photographic experts group
LBIC	Light Beam Induced Current – proud indukovaný světelným paprskem
Mbps	Megabit per second – megabity za sekundu
MPx	Megapixel
PC	Personal Computer – osobní počítač
RGB	Red, Green, Blue – červená, zelená, modrá
SIPS	Scientific Image Processing System – vědecký systém pro zpracování obrazu
USB	Universal Seriál Bus – univerzální sériová sběrnice
VCL	Visual Component Library – knihovny vizuálních komponent

Seznam obrázků

Seznam tabulek

Seznam příloh

A. Zdrojové kódy (na přiloženém CD)

B. Program G2CameraSoftware (na přiloženém CD)