UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Andrej Šmuc

Programski modul za krmiljenje proizvodnih procesov z uporabo digitalne kamere

DIPLOMSKO DELO
NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: doc. dr. Peter Peer

Ljubljana, 2011

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00546/2011
Datum: 04.04.2011

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **ANDREJ ŠMUC**

Naslov: **PROGRAMSKI MODUL ZA KRMILJENJE PROIZVODNIH PROCESOV Z UPORABO DIGITALNE KAMERE**

PROGRAM MODULE FOR CONTROL OF PRODUCTION PROCESSES WITH USE OF DIGITAL CAMERA

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija

Tematika naloge:

V uvodu opišite osnove avtomatizacije proizvodnih procesov z vidika programerja ter opišite problem, ki ga rešujete v svojem delu. Nato predstavite krovno aplikacijo, ki bo uporabljala razviti modul, razvojno okolje in potrebno strojno opremo v fazi avtomatizacije proizvodnje. Sledi naj opis razvoja samega programskega modula in njegovo ovrednotenje na primeru uporabe.

Mentor:
doc. dr. Peter Peer

Dekan:
prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani Andrej Šmuc,

z vpisno številko 63010148,

sem avtor diplomskega dela z naslovom:

Programski modul za krmiljenje proizvodnih procesov z uporabo digitalne kamere

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom doc. dr. Petra Peera
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 17. 5. 2011 Podpis avtorja:

Zahvala

Na tem mestu bi se zahvalil dr. Borutu Lenardiču, ki mi je z idejo in dostopom do strojne opreme omogočil, da je diplomska naloga nastala.

Zahvaljujem se mentorju doc. dr. Petru Peeru za pomoč in spodbudo pri pisanju diplomske naloge.

Zahvala gre tudi Zvonku Boštjančiču za mnenja in nasvete pri implementaciji programske opreme in Petru Jakopiču, ki mi je predstavil proces izdelave optičnih vlaken in svetoval pri vključevanju razvite programske opreme v proces.

Posebna zahvala gre staršem za potrpežljivost in podporo pri študiju in Katji, ki mi je pri pisanju diplomske naloge stala ob strani.

Kazalo

Seznam uporabljenih kratic in simbolov

SCADA – Supervisory Control And Data Acquisition (nadzorna kontrola in pridobivanje podatkov)

HMI – Human Machine Interface (vmesnik človek–stroj)

PLC – Programmable logic controller (programabilni kontrolnik)

ROI – Region Of Interest (interesno območje)

XML – Extensible Markup Language (razširljiv označevalni jezik)

Povzetek

Diploma opisuje programsko rešitev za kontrolo proizvoda industrijskega procesa s pomočjo digitalne kamere. Aplikacija, katere del je kamerin programski modul za krmiljenje, je bila posebej razvita za kontrolo procesa proizvodnje surovcev za optična vlakna. Zaradi svoje generične zasnove je razvita rešitev primerna tudi za druga industrijska področja. Rešitev deluje kot samostojen modul znotraj aplikacije. Pri procesu izdelave kamera zajema slike obdelovanca in jih pošilja svojemu nadzornemu sistemu, ki je zaradi svoje generične zasnove sposoben zajeto sliko procesirati na najrazličnejše načine. Pridobljeni podatki se pošljejo v obdelavo ostalim napravam, ki so del večjega sistema SCADA. Programski modul za delo s kamero je tako po meri narejen kontrolni sistem, uporaben v realnem industrijskem okolju, kjer je prilagodljivost na spremenljive zahteve proizvodnega procesa ključnega pomena.

Ker rešitev s kamero rešuje zaokrožen problem v industriji, diploma opisuje osnove nadzora industrijskih procesov in komunikacije med napravami v proizvodnji. Opisani so ključni elementi, potrebni za računalniški nadzor industrijskega procesa. Programski modul je razvit v okolju .NET in poizkuša izkoristiti prednosti, ki mu jih to okolje ponuja v primerjavi z v industriji bolj uveljavljenimi tehnologijami. Diploma poizkuša bralcu predstaviti svet avtomatike z vidika .NET razvijalca in predstaviti prednosti takšne izbire tehnologije pred bolj tradicionalno usmerjenimi tehnologijami v svetu avtomatike.

Ključne besede:

SCADA, kamera, avtomatika, kontrola, proizvodnja, okolje .NET

Abstract

The diploma thesis describes software solution for controlling industrial processes with the help of digital camera. The developed software module is a part of application developed originally for optic fibre preform manufacturing, but can also be applied to other fields of industry. The digital camera solution functions as an independent module inside control application. During manufacturing process it captures images of manufactured part and sends them into its subsystem for further processing. Module is developed in such a manner that multiple different algorithms can easily be applied to images, according to process needs. Acquired data is then sent to various devices, which are part of bigger SCADA system. Camera control module is custom made control system useful in real industrial environment, where adaptation to process changes plays a vital role in production of high quality products.

Since developed solution solves specific industrial problem as a whole, the diploma also describes basics of automation and process control. Basic elements needed for computer based process control are also described. In contrast to more widely accepted solutions in industry, software was developed using Microsoft .NET framework. The diploma tries to introduce the aspect of developing automation solutions from .NET developer's point of view. It also discusses advantages of using such technology in contrast to more traditional technologies used in automation.

Key words:

SCADA, camera automation, control, manufacturing, .NET framework

Poglavje 1

Uvod

1.1 Industrijska avtomatika

Človeka v razvitem svetu vsepovsod obkrožajo raznovrstne dobrine. Kadar se odločamo o nakupu nekega izdelka, lahko pogosto izbiramo med celo paleto podobnih proizvodov in prodajalci nas prepričujejo o njihovi kakovosti. Če nam je bilo včasih dovolj imeti na izbiro nekaj osnovnih izdelkov, katere je relativno preprosto proizvesti, je velika ponudba kompleksnih in visokotehnoloških izdelkov dandanes nekaj samoumevnega. Proizvajalci v svoje izdelke tako vključujejo najrazličnejše materiale in polizdelke, razvite s kompleksnimi postopki. Obstoja teh izdelkov in materialov se niti ne zavedamo, vendar pa bi se ob njihovem pomanjkanju hitro zavedeli njihove nepogrešljivosti pri sodobnem načinu življenja.

Malokdo se vpraša, kako taki izdelki pridejo na prodajne police, kako so sestavljeni in kako jih, zaradi pogosto izjemno visokega povpraševanja, uspejo proizvesti v dovolj velikih količinah. Seveda si pri omenjenih vprašanjih proizvajalci pomagajo z najrazličnejšimi stroji in napravami. Nihče si ne more predstavljati proizvodnje avtomobilov, pri kateri bi vse delo opravljal človek ročno. V sodobnih obratih to delo opravljajo stroji in to z veliko večjo natančnostjo in hitrostjo kot bi bil to sposoben narediti človek. Obstaja veliko zapletenih tehnoloških procesov, pri katerih človek igra manjšo vlogo in naprave, ki opravljajo določeno delo, le nadzira.

Ker so tehnološki procesi vedno bolj zapleteni in zahtevajo hitro in natančno odzivnost glede na parametre sistema (seveda ob visoki učinkovitosti), je računalnik postal nepogrešljiv pri krmiljenju proizvodnih procesov. V sodobni industriji je računalnik postal že tako nepogrešljiv kakor sam stroj, ki delovni proces opravlja. Kadar slišimo, da v določeni industriji ne uporabljajo ra-

čunalnikov pri krmiljenju proizvodnega procesa, nam pogosto pride na misel, da najbrž proizvajajo izjemno enostaven izdelek in to ne v posebno velikih količinah. Učinkovitost v njihovem proizvodnjem procesu pa je verjetno veliko manjša kot bi bila, če bi imeli računalniško podprto krmiljenje proizvodnje.

Področje avtomatizacije proizvodnje je izjemno široko. Ker se v različnih vrstah industrije in tudi znotraj posameznih panog izdelke proizvaja na različne načine z uporabo različnih, velikokrat tudi specifično prilagojenih strojev, se inženirji soočajo s problemom njihovega krmiljenja. Manjši problemi takega tipa se pojavljajo v industriji, kjer so tehnološki postopki že dobro preverjeni in utečeni, rešitve na področju njihovega krmiljenja pa dobro znane. Takšni procesi ne zahtevajo posebno velikega truda za njihovo računalniško podporo.

Večje težave pa imajo inženirji pri avtomatizaciji bolj kompleksnih in specifičnih tehnoloških postopkov, ki so ponekod še v fazi raziskav in razvoja. V takih primerih se pogosto zgodi, da so naprave v tehnološkem postopku popolnoma heterogene, postopki pa zahtevajo izjemno veliko prilagodljivost sistema na različne oblike prilagoditev. Pri načrtovanju kontrolnih sistemov za kompleksne tehnološke postopke se tako inženirji srečujejo z izjemno velikim številom parametrov. Računalniški sistem, ki krmili tak tehnološki proces mora biti sposoben hitro preračunavati različne vrednosti, katere mu posredujejo senzorji. Ker je takih izračunov pogosto zelo veliko, morajo inženirji poskrbeti za primerno zmogljivost sistema in optimizacijo algoritmov. Krmilni sistem nadzoruje stroje in naprave v realnem času, torej si ne more privoščiti počasne odzivnosti. Podatke o stanju procesa mora dobiti hitro, jih v najkrajšem možnem času obdelati, se odločiti o nadaljnem poteku proizvodnega procesa ter sporočiti ustreznim napravam ustrezna navodila.

Tudi zanesljivost delovanja je pri kontrolnih sistemih v industriji bistvenega pomena. Pri uporabi večine aplikacij s katerimi se srečujemo pri vsakodnevnem delu in zabavi, si načrtovalci programske opreme lahko privoščijo občasne izpade delovanja. Kadar pride do napake sistema, uporabnik aplikacijo zažene znova in nato nadaljuje delo brez težav. Izpada kontrolnega sistema pa si načrtovalci v industriji ne morejo privoščiti, saj je to pogosto povezano z velikimi stroški. Računalniški sistemi velikokrat krmilijo tehnološke procese, v katerih nastopajo nevarne in zdravju škodljive snovi, zato je nemoteno delovanje tu izjemnega pomena. Industrijski kontrolni sistemi morajo tako vsebovati varnostne mehanizme, ki ob morebitnem izpadu delovanja sistema preprečijo poškodbe ljudi in zmanjšajo ali preprečijo gmotno škodo.

1.2 Strojna oprema za nadzor naprav v industriji

Ker v industriji računalniki delujejo v posebnih razmerah, zahteve glede stabilnosti delovanja in zmogljivosti pa so visoke, se je na tem področju uveljavila strojna oprema, posebej prilagojena zahtevam industrije. Pomembna naloga industrijskih nadzornih sistemov je povezati okolico (naprave, senzorje ipd.) z delom kontrolnikov, ki izvajajo nadzorno logiko. Čeprav obstaja nešteto raznovrstnih naprav, s katerimi lahko nadzorni sistem komunicira pa komunikacija med njimi ponavadi poteka s pomočjo digitalnih in analognih signalov. Na nadzorni sistem bi torej lahko gledali kot na črno škatlo, ki pridobiva informacije od določenega števila vhodnih signalov, informacije obdela in jih nato posreduje okolju v obliki množice izhodnih signalov. Ponavadi mora delovati zelo hitro. Če naprava pridobiva podatke o hitrosti vrtenja motorja, ki kontrolira precizne procese in se mora odzivati v najkrajšem možnem času, mora biti sposobna podatke ustreznih vhodnih signalov obdelati pravočasno ter nastaviti vrednost ustreznih izhodnih kontrolnih signalov. Časa za obdelavo podatkov v tem primeru ni na pretek. Odzivnost naprave je izjemnega pomena in ponavadi bistveno večja, kot je to zahtevano od domačih računalnikov.

V industriji se za krmiljenje pogosto uporablja drugačne naprave, kot so na primer kontrolniki PLC (angl. Programmable Logic Controller) [1]. Za razliko od običajnih računalnikov je kontrolnik PLC načrtovan za priključitev večjega števila vhodov in izhodov. Posebej je prilagojen delu v industrijskem okolju, kjer lahko vladajo višje temperature, pojavljajo se lahko električne motnje iz okolja ter vibracije. Takšni kontrolniki so ponavadi razširljivi, po potrebi lahko na njih priključimo dodatno število vhodov in izhodov. Stanje sistema, ki ga nadzorujejo, lahko posredujejo drugim napravam, ki takšne podatke obdelujejo na različne načine. Kontrolniki PLC so primerni za vzdrževanje določenega stanja sistema neodvisno od ostalih delov sistema. Če se komunikacija z ostalim delom sistema prekine, lahko vsak kontrolnik PLC še naprej opravlja svoje funkcije in tako ohranja naprave, ki jih upravlja, v željenem načinu delovanja.

1.3 SCADA

Kadar govorimo o sistemu SCADA (angl. Supervisory Control And Data Acquisition)[2], govorimo o sistemu, ki pokriva celoten spekter nadzora in kontrole nad procesom in napravami, ki v takem procesu sodelujejo. Večinoma se to nanaša na kontrolo strojev in procesov v industriji ali na nadzor nad večjimi

in kompleksnejšimi napravami, kot so elektrarne, čistilne naprave, letališča ipd.

Sistemi, ki sodelujejo v industrijskih procesih in bi jih želeli nadzorovati, so različnih dimenzij in kompleksnosti. Lahko govorimo o kontroli enega stroja, ki obdeluje kovino, morda o tekstilni tovarni, kjer množica avtomatiziranih strojev proizvaja tkanino. Lahko pa govorimo o večji geografski lokaciji, kjer centralni nadzorni sistem kontrolira in zbira podatke o desetinah vetrnih elektrarn ter nadzornikom omogoča popolno kontrolo nad njihovim delovanjem ne glede na to, kje se nadzorna oseba v nekem trenutku nahaja. Sistem SCADA ponavadi sestavljajo vsaj naslednje osnovne komponente:

- vmesnik za povezavo človek–stroj (angl. Human Machine Interface – HMI)
- kontrolniki naprav
- komunikacijska infrastruktura.

Komponente skupaj sestavljajo sistem, zmožen nadzorovati še tako zapleten proces, ki bi bil brez takšnega nadzora neobvladljiv

1.3.1 Vmesnik za povezavo človek-stroj

Vmesnik HMI [3] v sistemu SCADA predstavlja vez med napravami in človekom. Z njegovo pomočjo so operaterjem na razumljiv način predstavljeni podatki o delovanju sistema. Preko njega lahko operater prebere podatke o tlaku v določeni črpalki, električni napetosti v določenem daljnovodu ali temperaturi peči za taljenje železa. Glede na predstavljene podatke lahko operaterji sprejmejo odločitve o poteku procesa. Vmesniki HMI operaterjem poleg predstavitve sistemskih podatkov ponujajo interaktiven nadzor nad sistemom. Po potrebi lahko z njim regulirajo pretoke različnih snovi, odpirajo ali zapirajo različne ventile, omogočeno jim je nastavljanje temperature različnih peči ipd.

Dober vmesnik HMI operaterja pravočasno opozarja na morebitna kritična stanja in napake v sistemu. Če določena sistemska spremenljivka (na primer tlak v kotlu) preseže svoje okvire, vmesnik HMI to jasno prikaže, najpogosteje v obliki alarma, kar omogoči operaterju, da se na dogodek ustrezno odzove (na primer preko vmesnika HMI odpre ventil za zmanjšanje tlaka v kotlu). Prav tako ščiti sistem pred morebitnimi napakami uporabnika. Prepreči lahko posredovanje napačnih vrednosti različnih parametrov, ki bi sistem morebiti lahko poškodovali (na primer morebitno nastavitev previsoke temperature, ki bi povzročila eksplozijo kotla).

Izziv načrtovalcev vmesnikov HMI je predstaviti realno stanje kompleksnih naprav na uporabniku razumljiv način. Mnogokrat je zaželjeno, da vmesnik HMI prikazuje čimbolj realno stanje resničnega sistema. Tako so v vmesniku poleg elementov, ki neposredno kontrolirajo obnašanje določenih komponent sistema (na primer ventilov), pogosto prikazane še cevi, ki določene ventile povezujejo. Prikazane so tudi komponente sistema, ki jih s pomočjo vmesnika HMI ne moremo kontrolirati, vendar je koristno vedeti, da na določenem delu naprave obstajajo (na primer zasilni ventili za ročno sproščanje pritiska v sistemu). Vse to služi namenu, da si operaterji, ki uporabljajo sistem, čim lažje predstavljajo fizično napravo. Pri načrtovanju vmesnika HMI, je treba upoštevati tudi morebitne različne vrste izobrazbe uporabnikov sistema. Inženirja, vajenega dela s kompleksnimi izračuni, morebiti zanimajo podrobni parametri v procesu in želi imeti čimvečji nadzor nad vsako najmanjšo podrobnostjo sistema. Tehnologa, ki pregleduje fizično kondicijo sistema pa pri uporabi vmesnika HMI morda zanima le, kako na čimbolj preprost način hitro odpreti ali zapreti določen ventil med servisom določenega dela sistema.

Vmesnik HMI mora biti posebej prilagojen delu v industrijskem okolju. V nasprotju z delovnim okoljem običajnega uporabnika računalnika, ki ima čas raziskovati številne funkcije sistema, je v industrijskem okolju zaželjeno, da je nepotrebnega klikanja čim manj. Odzivnost na uporabnikove spremembe mora biti hitra. Način, s katerim so funkcije prikazane na vmesniku, naj bo preprost in razumljiv. Tako je v okolju, kjer operaterji pri delu uporabljajo zaščitne rokavice, zaželjeno, da uporabnik nastavlja določene parametre sistema z uporabo velikih kontrol na zaslonu občutljivem na dotik. Precizno klikanje s pomočjo računalniške miške ali uporaba svetlobnega peresa je v industrijskem okolju nezaželjena.

1.3.2 Kontrolniki naprav

Zapletene sisteme sestavlja veliko število naprav, ki v njem zasedajo različne vloge. V industrijskih procesih uporabljamo od najpreprostejših ventilov in kontrolnikov pretoka, do bolj zapletenih naprav kot so specialne peči, stružnice za obdelovanje kovin, različnih senzorjev ipd. Kadar želimo naprave nadzorovati s pomočjo računalniških sistemov, moramo poskrbeti za vmesnik med napravo in sistemom za nadzor naprave.

Kontrolnik, ki povezuje računalniški sistem in napravo, mora poskrbeti za komunikacijo med njima. Računalniku mora omogočiti dostop do podatkov naprave v digitalni obliki, podatke posredovane v digitalni obliki iz računalnika pa posredovati napravi v obliki električnih signalov. Tako bi lahko pre-

prost ventil s pomočjo kontrolnika nadzorovali tako, da bi računalniški sistem kontrolniku posredoval digitalno vrednost 1 za odprtje ventila, kontrolnik pa bi ventilu poslal signal v obliki električne napetosti 24V, kar bi sporočilu ventilu naj se odpre. V primeru, da bi računalniški sistem kontrolniku sporočil digitalno vrednost 0, bi ventilu poslal signal v obliki električne napetosti 10V, kar bi pomenilo, da želimo ventil zapreti. Na podoben način in z različnimi kontrolniki lahko nadzorujemo stotine ali tisoče naprav v še tako zapletenem sistemu.

Nekateri deli sistema, še posebej bolj zapleteni, morajo znotraj sistema delovati z določeno stopnjo avtonomije. Njihovo delovanje bi želeli urejati z bolj ali manj zapleteno logiko, ki bi lahko delovala neodvisno od sistema, ta pa bi pridobival podatke o njenem delovanju in z določenimi nastavitvami urejal, na kakšen način naj se logika obnaša. V takšnih primerih je smiselna uporaba kontrolnikov PLC [1]. Takšni kontrolniki izvajajo svoj program in poskrbijo za delovanje delov sistema, ki ga nadzorujejo. Kontrolnikov PLC je lahko v sistemu zelo veliko in vsak ureja točno določeno področje poteka procesa. Podatke o delovanju lahko pošiljajo svojim nadzornim sistemom, ki jih združujejo in posredujejo v ostale dele sistema SCADA, dokler niso podatki na voljo operaterjem, ki uporabljajo vmesnik HMI.

Z zagotavljanjem avtonomnega delovanja delov sistema lahko poskrbimo za dodaten nivo varnosti v sistemu. Ob napaki v delu sistema, ki ga nadzoruje kontrolnik PLC s svojo logiko, bi kontrolnik ugotovil, da je prišlo do nepredvidenega dogodka in začel izvajati varnostne postopke. Tako bi sistem spravili v varno stanje. Tudi če bi se prekinila povezava do glavnega nadzornega sistema, bi lokalni kontrolnik varno upravljal napravo. Ob ponovni vzpostavitvi povezave z glavnim nadzornim sistemom pa bi se mu posredovali podatki o načinu delovanja kontrolnika. Operater bi s pomočjo vmesnika HMI ugotovil, da del sistema deluje v posebnem načinu in bi se na dogodek ustrezno odzval. Z avtonomnim krmiljenjem delov sistema bi na tak način vzpostavili višjo stopnjo redundance in varnosti.

1.3.3 Komunikacijska infrastruktura

Ključni lastnosti infrastrukture, ki skrbi za prenos podatkov med različnimi deli sistema SCADA, sta vsekakor hitrost in zanesljivost. Skozi zgodovino razvoja sistemov SCADA so njihovi proizvajalci uvedli različne komunikacijske protokole, prilagojene za industrijsko okolje [4]. Veliko teh protokolov je splošno sprejetih, nekateri pa so standardizirani in jih podpira večina večjih razvijalcev sistemov SCADA. Takšni protokoli podpirajo izjemno hiter prenos

podatkov in so prilagojeni okolju, kjer je pri izračunih in ravnanju s podatki pomembna vsaka milisekunda.

Za povezave med različnimi napravami in sistemi nekateri uporabljajo serijske povezave (RS-232, RS-485), uveljavile pa so se (in se uveljavljajo čedalje bolj pogosto) tudi Ethernet povezave [5].

Nekateri proizvajalci sistemov SCADA so razvili aplikacije, ki omogočajo dostop do vmesnika HMI tudi preko interneta. Tako je operater nenehno obveščen o delovanju procesa, ki ga nadzoruje. Če je zaradi narave upravljanega procesa mogoče, lahko s pomočjo mrežnih vmesnikov operater določene dele sistema upravlja neodvisno od svoje lokacije.

1.4 Pregled vsebine

Namen prvega poglavja je bralcu predstaviti pogled na svet industrijske avtomatike z vidika razvijalcev programske opreme. Predstavljeni so osnovni problemi, s katerimi se srečujemo pri krmiljenju strojev in naprav. Predstavljene so osnove strojne opreme, prilagojene za industrijski svet in razloženi principi sistemov SCADA. Prvo poglavje bralca seznani z osnovami nadzora in kontrole ter komunikacije med nadzornim sistemom in napravo, ki jo sistem nadzira.

Drugo poglavje predstavi specifičen pristop k reševanju problemov kontrolnih sistemov. Predstavljeno je okolje .NET za razvoj programske opreme. Z njim je razvita programska oprema, ki skupaj s strojno opremo podjetja Beckhoff oblikuje zaključen sistem za kontrolo strojev in proizvodnih procesov, katerega del je tudi rešitev opisana v tem diplomskem delu. Opisane so osnove programske rešitve, katere del je obravnavani programski modul.

Tretje poglavje podrobno opisuje razvito rešitev za nadzor proizvodnega procesa s pomočjo digitalne kamere. Opisana je uporabljena strojna oprema in njene zmogljivosti. Opisane so programske knjižnice za upravljanje digitalne kamere ter principi postopka zajema slike za obdelavo. Podrobno so opisani deli programske rešitve, razvite za delo s kamero, njena arhitektura in komunikacija s preostalim delom aplikacije. Opisani so tudi postopki za programiranje po meri narejenih algoritmov za obdelavo slike in njihovo integracijo v sistem.

Četrto poglavje predstavi uporabniku rezultate uporabe razvite programske rešitve v realnem proizvodnem okolju.

Peto poglavje obnovi bistveno vsebino diplomskega dela in obravnava možnosti za izboljšanje razvite programske rešitve.

Poglavje 2

Opis obstoječega sistema

2.1 Obstoječa programska rešitev

Razvit programski modul za nadzor procesa s kamero je integriran v aplikacijo za nadzor in kontrolo industrijskih procesov. Ta je primerna predvsem za kontrolo bolj zapletenih postopkov, ki se hitro spreminjajo in od kontrolnega sistema zahtevajo številne prilagoditve. Aplikacija se uporablja za krmiljenje procesa izdelave surovca za optična vlakna [6], primerna pa je tudi za uporabo v drugih industrijskih panogah. Postopek izdelave optičnih vlaken je zapleten proces, pri katerem je treba v realnem času usklajevati komunikacijo med stotinami signalov in naprav. Naprave vključujejo od preprostih ventilov in regulatorjev pretoka, stikal in pogonov motorjev, do naprav izdelanih po meri samo za potrebe procesa, ki za krmiljenje zahtevajo zapleteno logiko. Aplikacija je razvita v okolju .NET, podpora komunikaciji s kontrolnimi moduli različnih proizvajalcev pa omogoča enostavno razširljivost in prilagodljivost za delovanje v različnih okoljih. Skupaj s široko paleto krmilnikov in komunikacijskih modulov podjetja Beckhoff je odličen izbor za krmiljenje najrazličnejših procesov, ki so dovolj zapleteni, da je njihova računalniška kontrola smiselna.

Aplikacija ponuja uporabniku široko paleto funkcij za krmiljenje procesov:

- vsestranski vmesnik HMI za nadzor in kontrolo sistema v realnem času
- izvajanje zapletenih procesov v obliki receptov [7]
- povezljivost z krmilniki PLC in komunikacijskimi moduli različnih proizvajalcev
- močna orodja za analizo stanja sistema, ki ga uporabnik nadzoruje

- varnostne mehanizme, ki med kontrolo procesa varujejo naprave in uporabnika

Sistem omogoča modularno vključevanje kontrolnih sistemov narejenih po meri za krmiljenje specialnih naprav, ki sodelujejo v procesu. Kontrola za nadzor procesa s pomočjo digitalne kamere je eden takšnih modulov.

2.2 Opis problema

Proces izdelave surovca za optična vlakna je zapleten in navadno traja tudi več ur. Obdelovanec je podvržen zaporedju obdelav, ki na različne načine spreminjajo njegove lastnosti. Parametre naprav, ki sodelujejo pri obdelavi materiala, je treba med procesom nenehno spremljati in prilagajati. Nekatere parametre lahko prilagodimo na podlagi vizualnega izgleda obdelovanca ali njegovih delov med postopkom obdelave. V ta namen je smiselno razviti poseben modul, ki s pomočjo digitalne kamere in različnih algoritmov za obdelavo zajete slike spremlja stanje obdelovanca med procesom. Tako lahko bodisi avtomatsko nadzira izbrane procesne parametre bodisi služi operaterju kot pomoč pri sprejemanju odločitev in mu omogoča ročno prilagajanje procesnih spremenljivk. Ker se proces zaradi prilagoditev spreminja, je pomembno, da je razviti modul prilagodljiv. Podpirati mora uporabo raznovrstnih algoritmov za obdelavo slike in omogočati vključevanje spremenljivega števila procesnih spremenljivk v postopek delovanja modula. Rešitev mora biti prilagodljiva do te mere, da jo je mogoče uporabiti tudi za vizualni nadzor drugih proizvodnih procesov (ob uporabi aplikacije, katerega del je razviti modul).

2.3 Okolje .NET

Pri načrtovanju novega programskega sistema se vedno pojavi vprašanje izbire orodja za njegovo izdelavo. Včasih je izbira enostavna, saj smo (predvsem v zelo specifičnih primerih) pri izbiri vezani na funkcionalnosti, ki jih določeno orodje ponuja. V veliki večini primerov pa se nam ponuja pestra paleta orodij in platform, izbira med njimi pa je vse prej kot lahka.

Programska okolja so se skozi zgodovino razvijala predvsem zaradi vprašanja, kako razvijalcem programske opreme omogočiti lažje (ter hitrejše in posledično cenejše) načrtovanje in izdelavo programske kode. Izbira okolja je odvisna od želja in potreb razvijalca. Nobene ovire ni, da se ne bi lotili pisanja programskega sistema v zbirnem jeziku ali celo strojni kodi. Takšna odločitev

bi nam vsekakor pustila proste roke pri upravljanju z viri računalnikov, na katerem bi tekla naša programska oprema. V tem primeru nihče ne bi mogel oporekati hitrosti in učinkovitosti naše programske opreme – seveda pod pogojem da bi bili naši algoritmi optimalni. Vendar pa bi nas pri razvoju najverjetneje ustavil pogled na ceno takšnega razvoja, tako z vidika časa kot denarja.

V takšnem primeru je vsekakor bolje izbrati bolj napredno programsko okolje, ki namesto nas poskrbi za določene vidike razvoja programske opreme. Gre predvsem za vidike, ki nas pri delu upočasnjujejo in povečujejo možnosti napak. Pri izbiri se navadno soočamo z vprašanjem, koliko programerske svobode smo pripravljeni žrtvovati na račun hitrosti in udobja pri razvoju. Zanimiva možnost izbire je Microsoftovo okolje .NET [8], ki je trenutno na voljo v svoji 4. različici.

Okolje .NET poenostavlja razvoj programske opreme z abstrakcijo operacijskega sistema. Razvijalcem prihrani čas, ki bi ga drugače porabili s prilagajanjem aplikacije operacijskemu sistemu in jim omogoča, da svoj čas namenijo svoji programski opremi. Okolje poskrbi za določena nižjenivojska opravila, s katerimi bi se moral ukvarjati razvijalec, kar pripomore k hitrosti in učinkovitosti razvoja programske opreme. Pri velikem številu razvojnih okolij se mora razvijalec ukvarjati z alociranjem pomnilnika, kar je zamudno opravilo in lahko vodi do hudih programskih napak, ki so mnogokrat težko izsledljive. V okolju .NET to ni problem, saj za ta opravila poskrbi namesto razvijalca.

Pri razvoju v različnih okoljih se mnogokrat pojavlja pojem "ponovnega izumljanja nečesa". Kadar razvijalec pride do programske naloge (ponavadi zelo nizkonivojske), lahko ugotovi, da bi mu pomagala določena avtomatizirana funkcija, ki pa je okolje, v katerem razvija, ne podpira. V ta namen se odloči razviti razne funkcije in metode, ki bi jih okolje moralo priskrbeti že samo po sebi. Pri tem izgublja dragoceni čas. Ker razvijalec te funkcije le malokdaj razvije v obliki ločene knjižnice, ki bi se dala vključiti v druge aplikacije (ali pa zaradi različnih razlogov teh knjižnic ni dovoljeno vključiti v ostale aplikacije), so omenjene funkcije ponavadi izgubljene, ko se razvijalec loti dela na naslednji aplikaciji. V primeru, da uporablja isto orodje in naleti na podoben programski izziv, se lahko loti ponovnega razvoja že razvitih funkcij. Pogosto se zgodi, da takšne funkcije probleme rešujejo na neoptimalen način. Okolje .NET s svojim širokim naborom dobro preizkušenih knjižnic poskrbi za rešitev takšnih situacij. Nizkonivojsko programiranje naj bi bilo razvijalcu prihranjeno, saj je zanj, bolj kot okolje v katerem razvija, pomembna aplikacija, ki jo razvija.

Okolje .NET za učinkovitejši razvoj ponuja tudi široko paleto že izdelanih

kontrol in rešitev, s katerimi razvijalcu poenostavi razvoj aplikacij. Z uporabo teh rešitev (in če je mogoče s prilagoditvijo svoje aplikacije tem rešitvam), lahko razvijalec v nekaj dneh ponudi stranki svojo rešitev, katere razvoj bi drugače lahko trajal tedne. Seveda ima tako razvijalsko udobje svojo ceno.

Kritiki .NET okolja (in podobnih okolij) bi rekli, da je njihova slabost ravno v poenostavitvi opravil uporabniku, saj razvijalec nima popolne kontrole nad nizkonivojskimi sistemskimi opravili. To neizogibno nekoliko podaljša izvajalni čas aplikacije. Vendar pa je časovna potratnost sistema ponavadi kritična le na določenih delih aplikacije. Ponavadi so takšni deli aplikacij specialni algoritmi za izvajanje specifične naloge, ki je le del večjega programskega paketa. Postavlja se vprašanje, ali je vredno celotno aplikacijo zasnovati na platformi, ki omogoča večji nadzor nad sistemskimi viri uporabniku, vendar pa je neprimerno bolj okorna za razvoj, zaradi nekaj manjših (a vendar časovno kritičnih) delov sistema.

Mnogokrat razvijalec po podrobnem pregledu ugotovi, da rešitve za njegove programerske izzive ne zahtevajo direktne kontrole nad sistemskimi viri (torej kontrole nad viri mimo nadzorovanega okolja). Kadar pa vseeno tak nadzor potrebuje, okolje .NET zanj ponuja rešitev. Posamezne dele kode lahko na poseben način označi in dovoljeno mu je uporabiti način dostopa do pomnilnika, kot smo jih vajeni pri nekaterih drugih programskih jezikih (in okoljih). Takšen primer so kazalci, s katerimi lahko neposredno manipulira s pomnilnikom. Vendar pa se mora v tem primeru razvijalec na določenih delih kode soočati s problemi, ki so mu v nadzorovanem okolju prihranjeni (kot so problemi s pisanjem v zaščiten del pomnilnika ipd.).

Zaradi tega lahko rečemo, da je okolje .NET prilagodljivo okolje za razvijalca, ki ga ščiti pred njegovimi napakami in mu olajša razvoj programske opreme. Prihrani mu ukvarjanje s podrobnostmi, ki so sicer za delovanje njegove aplikacije nujne, vendar z vidika obnašanja aplikacije nepomembne. Navsezadnje bi se moral razvijalec pri načrtovanju dobre aplikacije bolj ukvarjati z vprašanjem, kaj naj njegova aplikacija dela in ne z vprašanjem, kako naj njegova aplikacija (na določenem sistemu in v določenem okolju) deluje.

V industrijskem svetu se za razvoj kontrolnih sistemov navadno uporabljajo rešitve, razvite v različnih programskih jezikih. Veliko kontrolnih aplikacij pred dobro uporabniško izkušnjo postavlja zanesljivost pri upravljanju procesa in zmogljivost delovanja. Veliko aplikacij pa uporabniškemu vmesniku ne namenjajo takorekoč nikakršne pozornosti, pomembno je samo, da je z njimi mogoče opraviti zahtevano delo. Tako so v industriji pogosto prisotne aplikacije, ki leta opravljajo isti postopek, so neprilagodljive in imajo razvit le osnoven uporabniški vmesnik. Okolje .NET je, zaradi širokega nabora raznovrstnih kontrol

in visoke prilagodljivosti, idealno orodje za razvoj aplikacij, od katerih se na eni strani pričakuje dobro uporabniško izkušnjo ter zmogljivost in zanesljivost delovanja na drugi strani.

2.4 Krmilniki Beckhoff

Obstoječa rešitev za povezavo z napravami, ki jih krmili, podpira uporabo strojne opreme različnih proizvajalcev. Od vsakega proizvajalca posebej je odvisno na kakšen način njihove rešitve podpirajo uporabo razvojnih okolij. Nekateri med njimi za povezavo z njihovo strojno opremo priskrbijo samo osnovne knjižnice (ponavadi napisane v programskem jeziku C / C++). Drugi poleg tega ponujajo še raznovrstne programske knjižnice za krmiljenje njihove opreme z uporabo razvojnih okolij kot so .NET, Java ipd. Kadar želimo pri razvoju uporabiti okolje .NET, se je za dobro rešitev izkazala strojna oprema proizvajalca Beckhoff.

Podjetje Beckhoff se na področju avtomatizacije usmerja večinoma v rešitve, ki delujejo s pomočjo osebnih računalnikov (PC-based control). V svetu osebnih računalnikov je bila dolgo časa kontrola procesov, ki pri avtomatizaciji zahtevajo hitro odzivnost in visoko stopnjo zanesljivosti nadzornega sistema, nepredstavljiva. Osebni računalniki so veljali za relativno nestabilen sistem, ki ni sposoben daljše časovno obdobje neprekinjeno krmiliti proizvodnega procesa. Do takšnega slovesa so jim nekoliko pomagale splošno znane težave operacijskih sistemov. Tudi dejstvo, da so bile strojne komponente osebnega računalnika med seboj dobro združljive bolj v teoriji kot v praksi (in to je posledično vplivalo na stabilnost sistema), osebnim računalnikom ni pomagalo pri slovesu v svetu industrije. Z razvojem osebnih računalnikov in mrežne tehnologije pa se je počasi spremenilo tudi to prepričanje in danes osebni računalniki čedalje bolj pridobivajo na slovesu tudi v svetu industrijske avtomatike. Podjetje Beckhoff je, z raznovrstno strojno opremo za različne namene krmiljenja in avtomatizacije procesov, eno izmed vodilnih podjetij na tem področju [9]. Osebne računalnike z industrijskimi krmilniki povezuje njihova programska oprema TwinCat [10], ki z implementacijo komunikacijskih protokolov, primernih za industrijo, poskrbi za dovolj hitro in robustno komunikacijo med osebnim računalnikom in industrijskim okoljem.

Beckhoff ponuja razvijalcem dobro vzdrževane programske knjižnice za povezavo z njihovim sistemom tudi za okolje .NET. Razvijalci lahko z njimi na preprost način dostopajo do funkcij sistema TwinCat, ki vzdržuje komunikacijo med različnimi možnimi arhitekturami kontrolnikov sistemov v industrijskem

okolju. Integracija z različnimi programskimi moduli, narejenimi po meri, je zato preprosta. Knjižnice poskrbijo za komunikacijo med različnimi moduli in sistemom TwinCat, ta pa podatke posreduje do kontrolnih modulov, ki jih nadzoruje.

Poglavje 3

Programski modul za nadzor procesa z digitalno kamero

V skladu s koncepti obstoječe aplikacije sem izdelal programski modul, ki omogoča nadzor procesov z digitalno kamero. Ta modulu med delovanjem pošilja sliko produkta v izdelavi, modul pa s temi podatki in uporabo različnih algoritmov v realnem času računa različne parametre, pomembne za krmiljenje procesa. Nekateri izračunani parametri so na voljo sistemu za krmiljenje in nadzor procesa, nekateri pa tudi operaterju sistema v pomoč pri morebitnem ročnem posredovanju med izdelavo produkta.

3.1 Kamera za nadzor procesa

Pri načrtovanju rešitve sem uporabil digitalno kamero podjetja ImagingSource, model DMK 31AG03.I (slika 3.1). Osnovna specifikacija kamere je podana v tabeli 3.1 [11].

Slika 3.1: Kamera DMx 31AG03.I

Ločljivost:	1024×768
Barvni formati:	Y800, RGB8
Hitrost zajemanja slike:	do 30 slik na sekundo
Dimenzije:	višina: 50,6mm, širina: 50,6mm, dolžina: 50mm

Tabela 3.1: Specifikacije kamere

Kamera podpira zajem izključno črno-belih slik, vendar barve pri ugotavljanju parametrov, ki jih potrebujemo za proces izdelave surovcev optičnih vlaken, niso relevantne. Pomembna lastnost naprave je sposobnost avtomatskega kontroliranja zaslonke in to je tudi eden od razlogov, zakaj je bila za potrebe procesa izbrana prav ta kamera.

Kamero priključimo na računalnik preko gigabitnega ethernet omrežja. Za programsko kontrolo naprave so v več programskih jezikih na voljo napravi priložene knjižnice IC Imaging Control. Aplikacija, katere del je kontrola kamere, je napisana v jeziku C# in ker so kontrolne knjižnice za kamero na voljo tudi v okolju .NET, so logična izbira za to programsko rešitev.

3.1.1 Barvni formati kamere

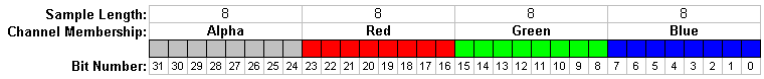
V računalniškem svetu predstavljamo podatke digitalno. Podatki, ki jih kamera zajema in pošilja sistemu, na katerega je priključena, morajo biti v digitalni obliki. Tudi podatki o barvah niso izjema. Kakor so pomembne dimenzije, v katerih se slika pošilja sistemu, tako je treba za obdelavo slike vedeti, v kakšnem formatu se sistemu posreduje informacija o barvah. Kamera, ki sem jo uporabil pri izdelavi rešitve ne podpira zajema barvnih slik. Podatke o barvah (odtenkih sivine) pošilja v enem izmed dveh formatov: RGB8 ali Y800.

3.1.1.1 Format RGB

V formatu RGB [12] je vsaka barva podana z določitvijo številskih vrendosti njenih komponent. Vsako barvo, predstavljeno v tem formatu, se da določiti kot seštevek vrednosti, ki predstavljajo njeno rdečo (R), zeleno (G) in modro (B) komponento. Če imajo vse komponente vrednost 0, je predstavljena barva črna, v primeru, da imajo vse komponente maksimalno vrednost, pa je barva bela. Število barv, ki jih lahko predstavimo je odvisno od števila bitov, ki jih pri določeni predstavitvi uporabimo za vsako barvno komponento.

Nekateri sistemi (tudi sistem, v katerega je integrirana rešitev s kamero), predstavljajo barve v 32-bitnem formatu RGBA (slika 3.2 [13]). Format uporablja po 8 bitov za predstavitev vsake barvne komponente in ostalih 8

bitov za predstavitev transparentnosti. Transparentnost (alfa) je ponazorjena z 8 biti tako, da vrednost 0 pomeni popolno transparentnost (torej je slikovni element neviden), maksimalna vrednost pa pomeni popolno netransparentnost slikovnega elementa.



Slika 3.2: 32-bitna RGBA predstavitev barve

3.1.1.2 Monokromatski RGB8 format

V sistemih, kjer zajemanje barvnih slik ni podprto, se včasih odločimo za predstavitev sivin v monokromatskem RGB8 formatu. Namesto komponent za rdečo, zeleno in modro, v tem primeru 8 bitov opisuje 256 možnih intenzitet sivin. Sliko tako sestavljajo samo slikovni elementi z različnimi odtenki sive barve.

3.1.1.3 Format YUV

Ta format za predstavitev barv uporablja nekoliko drugačen sistem kot format RGB. Tudi YUV format uporablja 3 vrednosti za določitev barve. Vrednost Y določa svetlost barve, dobimo pa jo s seštevkom obteženih vrednosti R, G in B. Uporabljene uteži so lahko različne, vendar se v modernejših formatih uporablja formula:

$Y = 0,2125R + 0,7154G + 0,0721B$

Vrednosti U in V v formatu določata barvo, dobi pa se ju z uporabo formul:

$U = B - Y$

$V = R - Y$

Zgodovinsko je format tesno povezan z razvojem televizije. S pojavom barvnih televizij so se razvijalci soočili s problemom, kako zakodirati podatke o barvah, s tem da bi obstoječi črno-beli televizorji še vedno normalno predvajali sliko. Komponenti Y so dodali še barvni komponenti U in V. Barvni televizorji so sposobni barvni komponenti, skupaj s komponento Y, dekodirati in pretvoriti v RGB signal.

Kadar konvertiramo digitalni video, pri računanju ponavadi skaliramo vrednosti YUV komponent tako, da ima komponenta Y vrednosti v razponu 16-235, komponenti U in V pa vrednosti v razponu 16-240. Vrednost 128 v primeru U in V predstavlja izhodišče. Črna barva bi bila tako predstavljena z YUV vrednostmi (16, 128, 128), bela barva pa z vrednostmi (235, 128, 128).

3.1.1.4 Format Y800

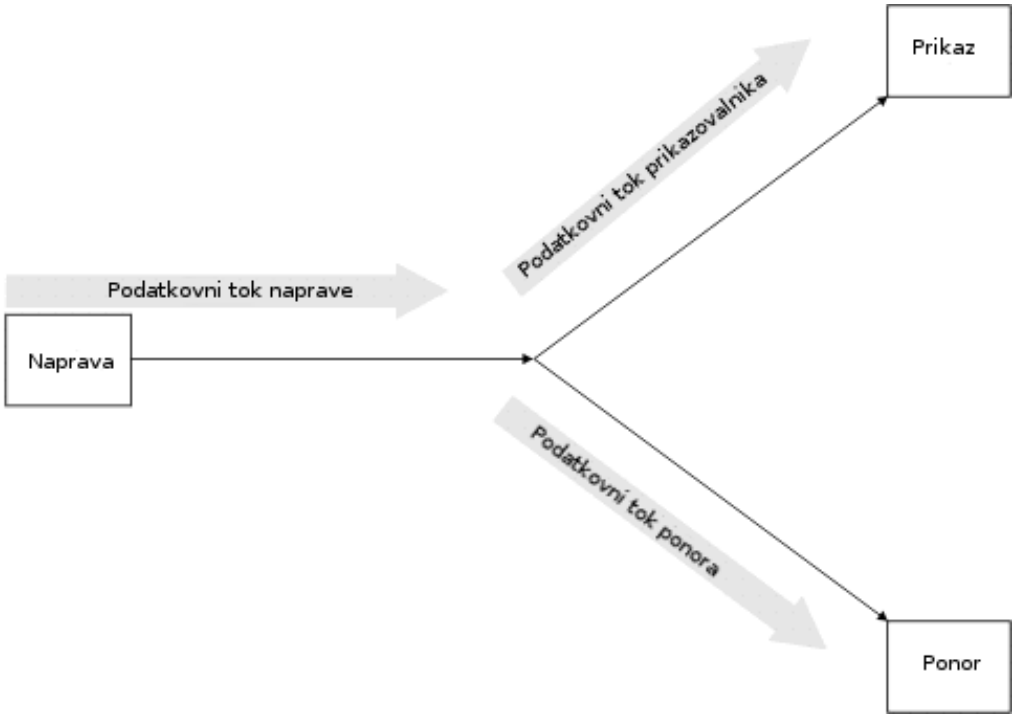
Format Y800, uporabljen pri konvertiranju kamerine slike, je preprosta oblika YUV formata, ki pri prezentaciji ne uporablja barvnih komponent (komponent U in V). Podobno kot pri monokromatskem RGB8 formatu, sliko sestavljajo slikovni elementi različnih intenzitet sivin, le da so v primeru kodiranja s formatom Y800 predstavljene le s komponento Y. Ker kamera prednastavljeno predstavlja sliko v tem formatu, sem ga pri implementaciji modula uporabil tudi jaz.

3.2 Osnovni koncepti delovanja naprave

Knjižnica za delo z omenjeno kamero je zasnovana tako, da na čimbolj preprost način razvijalcu omogoči zajem kamerine slike in njeno manipulacijo. Ob nakupu naprave dobimo priloženo tudi knjižnico za delo z njo. Tako lahko uporabnik zlahka izkoristi zmogljivosti naprave, ki jo uporablja. Knjižnica tako omogoča operacije nad slikovnimi elementi zajete slike za potrebe različnih algoritmov, kot tudi nekatere priročne možnosti kot so približevanje in pomik slike.

Slika 3.3 prikazuje postopek zajema slike z uporabljeno kamero. Slike, ki jih zajema naprava, lahko ponazorimo kot tok podatkov. Ta teče od izvora – naprave, ki podatke zajema, do ponora – sistema, ki podatke potrebuje. Ker je podatke priročno obdelovati ločeno za potrebe prikaza in potrebe algoritmov za obdelavo slike, je podatkovni tok mogoče razdeliti na dva dela. To sta del za prikazovanje podatkov in del za obdelavo podatkov. Oba dela podatkovnega toka sta identična, saj gre v bistvu za isti podatkovni tok, ki potuje v različne smeri. Če torej uporabimo nespremenjen podatkovni tok za prikaz podatkov in za obdelavo podatkov, potem sistem obdeluje točno tako sliko, kot je prikazana na uporabnikovem zaslonu.

Priročnost takšne razdelitve se pokaže v primeru, da želimo uporabniku prikazati nekoliko spremenjene podatke, kot jih obdeluje algoritem. Zaradi deljenega podatkovnega toka lahko dodajamo različne filtre [14] za obdelavo slik na del podatkovnega toka namenjenega prikazu slike in del podatkovnega toka, namenjenega obdelavi podatkov z algoritmi (ponor). Takšen način obdelave podatkov nam omogoča preprosto dodajanje grafičnih elementov na prikazovalni del, ki pa zaradi deljenosti podatkovnega toka ne moti algoritmov pri izračunavanju parametrov. Kadar želimo filtrirati podatke za oba dela podatkovnega toka, lahko dodajamo filtre na osnovni podatkovni tok, ki poteka iz naprave.



Slika 3.3: Arhitektura modula za zajemanje in obdelavo slike

Vsako udobje pri razvijanju sistemov zahteva svoje sistemske vire. Vendar pa nam prilagodljivost obdelave podatkov omogoča dober nadzor nad dodatno porabo le–teh. Arhitektura sistema, ki uporablja deljen podatkovni tok, nam ponuja različne možnosti tudi pri varčevanju z razpoložljivimi zmogljivostmi sistema. Če želimo optimizirati performanse, lahko na preprost način ugasnemo prikazovalnik ali različne grafične elemente, ki jih morebiti dodajamo na podatkovni tok, speljan v prikazovalnik podatkov. Optimizacijo se da doseči tudi z racionalnim postavljanjem filtrov podatkov na podatkovnemu toku. Seveda pa je vprašanje, ali nam aplikacija, ki jo razvijamo to dopušča.

Pri načrtovanju sistema je treba previdno razmisliti o vseh pomembnejših funkcionalnostih (še posebej zahtevnih s stališča porabe sistemskih virov). Če ugotovimo, da se nam obdelava podatkov za namen določene funkcionalnosti bolj izplača na nivoju kamere, jo bomo, raje kot na nivoju aplikacije, vklopili na nivoju same naprave. S previdnim načrtovanjem lahko tako obdržimo eleganco in preprostost pri razvoju, ki nam jo omogoča knjižnica za delo s kamero, ne da bi se zato odpovedali dobrim performansem sistema, ki ga razvijamo.

3.3 Programska rešitev kontrole s kamero

Z vidika uporabnika je modul kamere razdeljen na dva dela:

- kontrolnik SCADA
- glavno okno kamere.

3.3.1 Kontrolnik SCADA

Razvit modul kamere je integriran v sistem SCADA in tako del večje sistemske rešitve (glej 3.5). Vmesnik HMI omogoča vizualen prikaz kamere na shemi celotnega sistema, z njegovo pomočjo pa lahko uporabnik dostopa do prikaza glavnega okna kamere. Kontrolnik je navidez sestavljen samo iz grafičnega dela, ki prikazuje kamero in gumba, s katerim dostopamo do glavnega okna kamere. Vendar lahko uporabnik s pomočjo kontrolnikovih nastavitev poleg izgleda nastavlja še nekatere druge parametre, ki so nujni za delovanje kamere, niso pa spremenljivi med samim delovanjem sistema. Kontrolnik kamere, vključen v celotno shemo SCADA, predstavlja naslednji opis XML:

```
<ScadaObject xsi:type="IcImagingDevice" height=""
Identifier=""zindex="" x="" y="" width="">
<Description></Description>
<BackgroundColor></BackgroundColor>
<BorderSize></BorderSize>
<BorderStyle></BorderStyle>
<BorderColor></BorderColor>
<ExecutionOrder></ExecutionOrder>
<ManualOverride></ManualOverride>
<Name></Name>
<Visible></Visible>
<AutoIrisOffset></AutoIrisOffset>
<MaxExposure></MaxExposure>
<BackgroundImage></BackgroundImage>
<DeviceName></DeviceName>
<FilterName></FilterName>
<FilterParameters></FilterParameters>
<FlipHorizontal></FlipHorizontal>
<FlipHorizontalDisplay></FlipHorizontalDisplay>
<AutoMaxExposure></AutoMaxExposure>
```

```
<FlipVertical></FlipVertical>
<FlipVerticalDisplay></FlipVerticalDisplay>
<FrameRate></FrameRate>
<ImageFormat></ImageFormat>
<InputNames></InputNames>
<LicenceKey></LicenceKey>
<ROIHeight></ROIHeight>
<ROIWidth></ROIWidth>
<Rotation></Rotation>
<RotationDisplay></RotationDisplay>
<ShowSerialNumber></ShowSerialNumber>
</ScadaObject>
```

z opisa lahko razberemo nekatere nastavljive lastnosti, ki so posredovane sistemu preko kontrolnika SCADA. Njegove lastnosti lahko razdelimo na dva dela. Nekatere določajo splošen izgled in obnašanje kontrolnika, z nekaterimi pa uporabnik določa bolj precizno obnašanje kamere ali algoritmov, ki obdelujejo njeno sliko.

Lastnost Description

Uporabniku prijazen opis naprave, ki ponavadi vsebuje bolj podrobne informacije o napravi. To je še posebej uporabno, kadar imamo več naprav istega tipa in želimo uporabniku ponuditi dodatne podatke o specifični instanci naprave.

Lastnost BackgroundColor

Lastnost določa barvo ozadja naprave, prikazane v vmesniku HMI. Podana je v formatu RGBA, pri čemer so vrednosti ločene z vejicami.

Lastnost BorderSize

Lastnost določa širino okvirja kontrolnika SCADA, kot je prikazan uporabniku v vmesniku HMI. Vrednost je podana v slikovnih elementih.

Lastnost BorderStyle

Uporabnik s to lastnostjo določa stil okvirja, ki omejuje kontrolnik SCADA.

Lastnost BorderColor

Lastnost določa barvo okvirja kontrolnika SCADA.

Lastnost ExecutionOrder

Lastnost določa vrstni red, po katerem glavna aplikacija preračunava vrednosti algoritmov naprav. Lastnost je uporabna v sklopu integracije modula v glavno aplikacijo.

Lastnost ManualOverride

Določa, ali je po zagonu sistema uporabniku dovoljena interakcija s kontrolnikom SCADA. S tem parametrom lahko uporabniku med delovanjem sistema preprečimo uporabo naprave.

Lastnost Name

Uporabniku prijazno ime naprave, ki se uporablja za identifikacijo na vmesniku HMI.

Lastnost Visible

Določa vidnost kontrolnika SCADA uporabniku.

Lastnost BackgroundImage

Lastnost določa pot do morebitne slike, ki naj se prikaže v ozadju kontrolnika SCADA na vmesniku HMI.

Lastnost AutoIrisOffset

Odpiranje in zapiranje zaslonke kamere se kontrolira s pošiljanjem signala določene frekvence napravi. Ta frekvenca se od naprave do naprave razlikuje. S pomočjo te lastnosti se kalibrira fiksno točko, pri kateri se zaslonka ne odpira niti ne zapira. V našem primeru je vrednost tega parametra 0.

Lastnost MaxExposure

Parameter nastavlja maksimalno vrednost hitrosti zaslonke kamere. Služi kot varovalka, ki preprečuje, da bi se v izračun posredovali dve enaki sliki.

Lastnost AutoMaxExposure

Lastnost ureja vklop avtomatskega prilagajanja parametra izpostavljenosti svetlobi pri zajemu slike na določeno hitrost zajema slike sistema tako, da se pri tem hitrost pošiljanja slike ne zmanjša. Če uporabnik nastavi višjo hitrost pošiljanja slike, bo funkcija AutoMaxExposure parameter MaxExposure nastavila na določeno vrednost, če bo uporabnik hitrost zmanjšal, bo ta funkcija parameter MaxExposure povečala.

Lastnost DeviceName

Lastnost določa model kamere, s pomočjo katerega se sistem poveže na napravo. Veljavne vrednosti te lastnosti določa proizvajalec kamere.

Lastnost FilterName

S to lastnostjo podajamo ime algoritma, ki ga želimo uporabljati pri obdelavi slike, pridobljene iz naprave. To nam omogoča uporabo različnih algoritmov za obdelavo slike. Veljavna vrednost lastnosti je odvisna od implementacije algoritma v programski kodi in mora ustrezati specifikacijam posameznega algoritma.

Lastnost FilterParameters

Nastavitev se uporablja za posredovanje posebnih parametrov algoritmom za obdelavo slike. Podobno kot pri imenu algoritma, mora nabor nastavitev ustrezati specifikacijam posameznega algoritma.

Lastnost FlipHorizontal

Lastnost sporoča algoritmom kamere, ali naj pred obdelavo sliko obrnejo po horizontali. Zrcaljenje poteka samo za potrebe algoritma, na prikazno sliko ta nastavitev ne vpliva.

Lastnost FlipVertical

Omogoča zrcaljenje slike po vertikali. Zrcaljenje poteka samo za potrebe algoritma, na prikazno sliko ta nastavitev ne vpliva.

Lastnost FlipHorizontalDisplay

Lastnost se uporablja za morebitno obrnitev slike na prikazovalnem delu kamere. To je uporabno še posebno v primeru, ko bi radi le rotirali prikaz slike, ne pa tudi slike same (zaradi obnašanja algoritmov, ki sliko obdelujejo). Zrcaljenje poteka samo za potrebe prikazne slike, na izvajanje algoritma to zrcaljenje ne vpliva.

Lastnost FlipVerticalDisplay

Omogoča zrcaljenje slike po vertikali. Zrcaljenje poteka samo za potrebe prikazne slike, na izvajanje algoritma to zrcaljenje ne vpliva.

Lastnost FrameRate

S pomočjo te nastavitve lahko uporabnik nastavlja število sličic, ki jih v sekundi zajame kamera in pošlje v sistem. Večja kot je vrednost parametra, bolj tekoča je slika, vendar to bolj obremeni sistem. Pri uporabi kamere na sistemih z manj zmogljivimi procesorji je smiselno zmanjšati vrednost te lastnosti. Pri opisani rešitvi je uporabljena maksimalna hitrost zajema slike, ki ga podpira kamera, to je 30 sličic na sekundo.

Lastnost ImageFormat

Lastnost določa format, v katerem so zajete slike posredovane sistemu. Trenutno je podprta samo možnost uporabe formata Y800.

Lastnost InputNames

Lastnost je namenjena določanju posebnih generičnih vhodnih parametrov v napravo. Uporabnik lahko določi maksimalno 10 posebnih vhodnih parametrov. Za vsak parameter, kateremu uporabnik poda ime (imena so med seboj ločena z znakom ';'), sistem generira uporabniški vmesnik na glavnem oknu kamere. Vrednost lastnosti lahko vsebuje tudi prazne vnose, s katerimi lahko uporabnik preskoči vhodne parametre, ki jih ne želi prikazati na uporabniškem vmesniku. Primer: Če želi uporabnik od desetih možnih vhodnih parametrov v glavnem oknu kamere prikazati 1., 4. in 5., potem lahko to stori tako, da nastavi vrednost lastnosti InputNames na naslednji način: Input0;;;;Input4;Input5. Tip gradnika se določi glede na tip signala, ki je priključen na določen vhod modula (glej poglavje 3.5 in 3.4). Če je na vhod priključen digitalni signal, bo modul generiral gradnik za določanje digitalnih vrednosti (polje za izbiro

s kljukico), če pa je priključen signal analogen, bo modul generiral drsnik in vnosno polje, s katerim bo uporabnik lahko nastavljal analogne vrednosti. Razpon vrednosti na drsniku so določene glede na minimalno in maksimalno vrednost priključenega signala.

Lastnost LicenceKey

S to lastnostjo uporabnik poda licenčni ključ za uporabo kontrolnih knjižnic kamere. Licenčni ključ je priložen vsaki napravi posebej. Kontrolne knjižnice so navadno vključene v ceno kamere. Uporabljena kamera skupaj s kontrolnimi knjižnicami stane 670 EUR.

Lastnost ROIHeight

Lastnost določa višino interesnega območja kamere. Vrednost je podana v številu slikovnih elementov.

Lastnost ROIWidth

Lastnost določa širino interesnega območja kamere. Vrednost je podana v številu slikovnih elementov.

Lastnost Rotation

S to lastnostjo lahko rotiramo sliko, poslano iz kamere, za potrebe algoritma. Lastnost ne vpliva na prikazno sliko, marveč samo na sliko poslano algoritmu. Možna je rotacija za 90, 180 in 270 stopinj. Veljavne vrednosti lastnosti so: None, Deg90, Deg180, Deg270.

Lastnost RotationDisplay

Lastnost deluje podobno kot lastnost Rotation, le da v tem primeru rotira sliko za potrebe prikazovalnika. Lastnost ne vpliva na sliko poslano algoritmu.

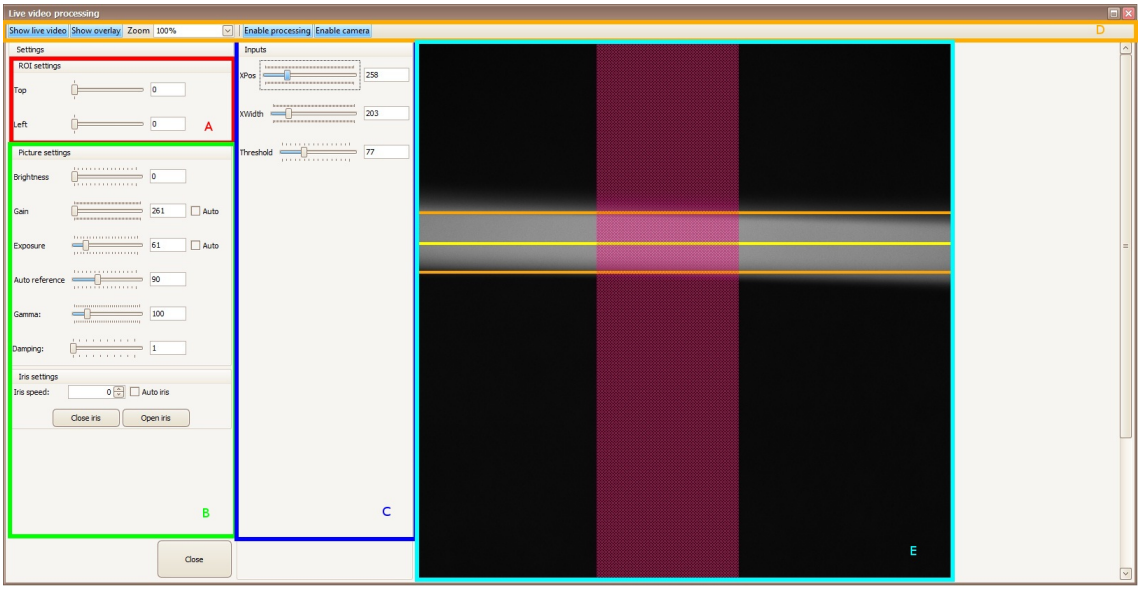
Lastnost ShowSerialNumber

Uporabnik lahko s to lastnostjo določi, ali naj se na kontrolniku SCADA prikaže serijska številka kamere. To je uporabno za namen hitre identifikacije naprav na vmesniku HMI.

3.3.2 Glavno okno kamere

Glavno okno kamere sem razdelil na pet delov:

- glavni meni (segment D na sliki 3.4)
- nastavljanje interesnega območja (segment A na sliki 3.4)
- nastavljanje parametrov slike (segment B na sliki 3.4)
- nastavljanje generičnih vhodnih parametrov (segment C na sliki 3.4)
- prikaz slike (segment E na sliki 3.4).



Slika 3.4: Glavno okno kamere

3.3.2.1 Glavni meni

Glavni meni sestavljajo naslednji gradniki:

Gumb "Show live video"

S to opcijo lahko uporabnik prikaže ali skrije prikazovanje slike, poslane iz kamere. V primeru, da je slika skrita, algoritem v ozadju še vedno preračunava njene parametre. Opcija je uporabna predvsem v primeru, ko želimo začasno

izboljšati performance sistema na račun izklopa prikaza slike, hkrati pa imeti odprto okno za nastavljanje njenih parametrov.

Gumb "Show overlay"

Ta opcija uporabniku omogoča vklop in izklop prikaza grafičnih elementov slike. Na sliki, ki jo kamera pošilja, grafični elementi označujejo pomembne dele, kot so robovi in interesna območja.

Izbirni menu "Zoom"

S pomočjo te opcije lahko uporabnik povečuje, ali pomanjšuje prikazno sliko. Uporabnik lahko izbira več območij do maksimalnega povečanja slike na 200% originalne vrednosti, ali do minimalnega območja 25% originalne velikosti slike.

Gumb "Enable processing"

Gumb deluje kot stikalo in v primeru, da je vklopljen, omogoči aplikaciji izvajanje algoritma za obdelavo slike. V primeru, da je opcija izklopljena, aplikacija preskoči izračunavanje parametrov, ki jih aplikacija pridobiva s pomočjo zajete slike.

Gumb "Enable camera"

S pomočjo tega gumba lahko uporabnik popolnoma izklopi delovanje kamere. V tem primeru se izklopi tako zajem in pošiljanje slike aplikaciji, kot tudi izračunavanje parametrov.

3.3.2.2 Nastavljanje interesnega območja

Uporabniški vmesnik za nastavljanje interesnega območja sestavljata dve kontroli, sestavljeni iz drsnika s pripadajočim vnosnim poljem (slika 3.4). Z njuno pomočjo lahko uporabnik nastavlja pozicijo interesnega območja. Kontroli se imenujeta "Top" in "Left". Prva premika pozicijo zgornjega levega kota interesnega območja v koordinatnem sistemu po osi Y, druga po osi X.

Širino in višino definicijskega območja lahko uporabnik nastavlja v nastavitvah kontrolnika SCADA kot lastnosti naprave (glej poglavje 3.3.1). Parametra sta nastavljiva samo med ustavljenim stanjem sistema. To je nujno, ker nastavljanje parametrov slike, ki določata višino in širino interesnega območja, vplivata na format videa, ki ga zajema kamera. Format videa se med pošiljanjem slike aplikaciji ne sme spreminjati. Omenjena parametra vplivata tudi

na definicijsko območje vrednosti, nastavljivima s kontrolama za premikanje pozicije. Kontrola "Top" ima definicijsko območje od 0 do 767 slikovnih elementov – višina interesnega območja, kontrola "Left" pa definicijsko območje od 0 do 1023 slikovnih elementov – širina interesnega območja, pri čemer pa so na voljo samo vrednosti v tem območju, ki so deljive z 8.

3.3.2.3 Nastavljanje lastnosti slike

S pomočjo tega dela uporabniškega vmesnika lahko uporabnik nastavlja lastnosti slike kamere in tako sliko prilagodi, da lahko algoritem kar najbolj učinkovito opravlja svoje delo. Nastavljive so naslednje lastnosti kamere:

Lastnost Brightness

Lastnost se uporablja za nastavljanje svetlosti slike, ki jo zajemamo s kamero. Sistem vsakemu slikovnemu elementu prišteje konstanto, kar pomeni, da vsako sliko posvetli ali potemni. Višja vrednost lastnosti pomeni svetlejšo sliko.

Lastnost Gamma

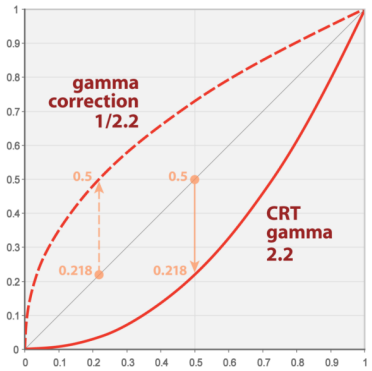
Lastnost se uporablja za nastavljanje vrednosti popravka game slike, ki jo zajemamo. S tem kompenziramo nelinearno obnašanje parametrov slike.

Popravek gama je parameter, ki nam omogoča pravilno prikazati zajeto sliko na ekranu. Pri definiranju svetlosti slikovnega elementa se na monitorjih pojavljajo odstopanja od nastavljene vrednosti svetlosti. Če določenemu monitorju pošljemo informacijo, da je svetlost nekega slikovnega elementa x, se v praksi izkaže, da naprava prikaže svetlost tistega slikovnega elementa z vrednostjo x^{γ}.

Parameter γ se lahko od monitorja do monitorja razlikuje. Tipično se na monitor pošilja vrednost svetlosti slikovnega elementa med 0 in 1. To pomeni, da če je vrednost parametra $\gamma = 2,2$ in nastavljena vrednost slikovnega elementa 0,5, bo prikazana vrednost na zaslonu imela svetlost $0,5^{2,2}$, kar znese 0,2176. Torej bo prikazana slika na monitorju videti temnejša, kot bi glede na nastavitve želeli. Situacijo opisuje formula:

$L = V^{\gamma}$,

kjer je V nastavljena vrednost svetlosti in L prikazana vrednost svetlosti. Rešitev za omenjeni problem je popravek gama. Vhodni signal, ki določa svetlost slikovnega elementa, popravimo tako, da dobljeno vrednost potenciramo z inverzom parametra γ (slika 3.5 [15]).



Slika 3.5: Graf vrednosti signala in njegovega gamma popravka

Pri vpeljevanju popravka gama torej uporabimo formulo:
$L' = L^{\frac{1}{\gamma}}$

V primeru vrednosti parametra 2,2, nastavljene vrednosti svetlosti slikovnega elementa 0,5 in dobljene vrednosti 0,2176, dobimo vrednost $L' = 0,2176^{\frac{1}{2,2}} = 0,2176^{0,45455} = 0,5$, kar pa ustreza originalni vrednosti svetlosti slikovnega elementa, ki jo želimo nastaviti.

Lastnost Gain

Lastnost se uporablja za ojačanje slike, ki jo zajemamo. Vsak slikovni element se pomnoži z nastavljeno vrednostjo. S povečanjem te funkcije tako izboljšamo kontrast slike. Če sliko ojačamo preveč, lahko na njej opazimo šum.

Lastnost AutoGain

Lastnost omogoča vklop, ali izklop avtomatske funkcije nastavljanja ojačanja slike. Pri nastavljanju vrednosti si modul pomaga z vrednostjo lastnosti AutoReference.

Lastnost Exposure

Lastnost nastavlja čas osvetlitve slike, ki jo zajemamo. Večja kot je njena vrednost, več svetlobe bo prišlo do senzorja. Pri zajemanju slike zelo svetlega predmeta mora biti čas osvetlitve majhen.

Lastnost AutoExposure

Lastnost omogoča avtomatsko regulacijo časa osvetlitve slike. Pri nastavljanju vrednosti si modul pomaga z vrednostjo lastnosti AutoReference.

Lastnost AutoReference

Ta lastnost se uporablja kot referenčna vrednost za avtomatske algoritme, ki regulirajo avtomatsko osvetlitev, avtomatsko ojačanje slike in avtomatsko nastavljanje zaslonke. Glede na vrednost tega parametra se bodo algoritmi odločali, kako nastavljati druge parametre glede na trenutno stanje slike, ki jim jo kamera posreduje. V primeru, da je glede na vrednost lastnosti slika pretemna, potem sistem najprej odpira zaslonko, dokler je popolnoma ne odpre. Če to ne zadostuje povečuje parameter izpostavljenosti svetlobi do vrednosti lastnosti MaxExposure. Kadar tudi to ni dovolj začne povečevati vrednost lastnosti Gain. V primeru, da je slika presvetla, sistem izvede postopek, obraten postopku za primer pretemne slike: najprej zmanjša vrednost lastnosti Gain, nato krajša čas izpostavljenosti svetlobi, nazadnje pa začne z zapiranjem zaslonke.

Lastnost AutoIris

Lastnost omogoča avtomatsko regulacijo odpiranja ali zapiranja zaslonke kamere. Pri nastavljanju vrednosti si modul pomaga z vrednostjo lastnosti AutoReference.

Lastnost IrisSpeed

Lastnost, s katero lahko kontroliramo hitrost odpiranja ali zapiranja zaslonke kamere. Knjižnica za delo s kamero ponuja štiri možne hitrosti: 1, 2, 3 in 4, kjer manjša številka pomeni počasnejšo hitrost premikanja zaslonke.

Lastnost OpenIris

S pomočjo tega gumba uporabnik ročno odpira zaslonko kamere. Zaslonka se odpira s hitrostjo, določeno v lastnosti IrisSpeed. Zaslonka se odpira toliko časa, dokler uporabnik zadržuje miškin gumb v pritisnjenem stanju nad gumbom “Open iris”.

Lastnost CloseIris

S pomočjo tega gumba uporabnik ročno zapira zaslonko kamere. Hitrost zapiranja določa lastnost IrisSpeed. Zaslonka se zapira toliko časa, dokler uporabnik zadržuje miškin gumb v pritisnjenem stanju nad gumbom "Close iris".

Lastnost Damping

S to lastnostjo blažimo spremembe pri avtomatskem nastavljanju parametrov kamere. Večja kot je vrednost lastnosti, manj radikalne bodo avtomatske spremembe slike, ki jih nastavljajo avtomatski algoritmi.

3.3.2.4 Nastavljanje generičnih vhodnih parametrov

Programskemu modulu kamere lahko uporabnik določi do 10 generičnih vhodnih parametrov. Z njimi sistemu posreduje vrednosti, katere lahko algoritem za obdelavo slike uporablja med svojim delovanjem. Definira se jih tako, da kontrolniku SCADA nastavimo vrednost lastnosti InputNames, v kateri definiramo imena generičnih vhodnih parametrov (glej poglavje 3.3.1). Za vsako ime, ki se nahaja v lastnosti InputNames, sistem zgenerira kontrolo na uporabniškem vmesniku. Vrednost generičnih parametrov je posredovana v sistem s pomočjo signala, ki je lahko digitalen, ali analogen (glej 3.4). Glede na tip signala, ki je priključen na mesto poimenovanega generičnega parametra, se zgenerira ustrezen uporabniški vmesnik: drsnik in vnosno polje za analogne signale in kontrolnik za izbiro stanja tipa boolean za digitalne signale.

3.3.2.5 Prikaz slike

Okno za prikaz slike prikazuje sliko, ki jo pošilja kamera preko prikazovalnikovega podatkovnega toka. Nanjo vplivajo nastavitve, ki so specifične za prikazovalnikov tok podatkov. Prednastavljena je velikost slike 1024×768 slikovnih elementov. Sliko lahko na prikazovalniku zmanjšamo in prikažemo samo njen del tako, da v nastavitvah kontrolnika SCADA definiramo območje ROI (z uporabo lastnosti ROIWidth in ROIHeight), ki je manjše od prednastavljene velikosti slike. V tem primeru lahko določamo kateri del slike je prikazan na zaslonu. To storimo z uporabo drsnikov dela uporabniškega vmesnika za nastavljanje interesnega območja slike na glavnem oknu kamere.

Slika je uporabniku prikazana samo v primeru, da je v glavnem menuju izbrana opcija "Enable camera". Kadar je slika uporabniku vidna, lahko po želji vključimo opcijo grafičnega prikaza izračunanih, ali nastavljenih parametrov z

opcijo “Show overlay”. Grafični parametri, prikazani na zaslonu, se razlikujejo od filtra do filtra, njihove specifikacije morajo biti podane v specifikaciji filtra.

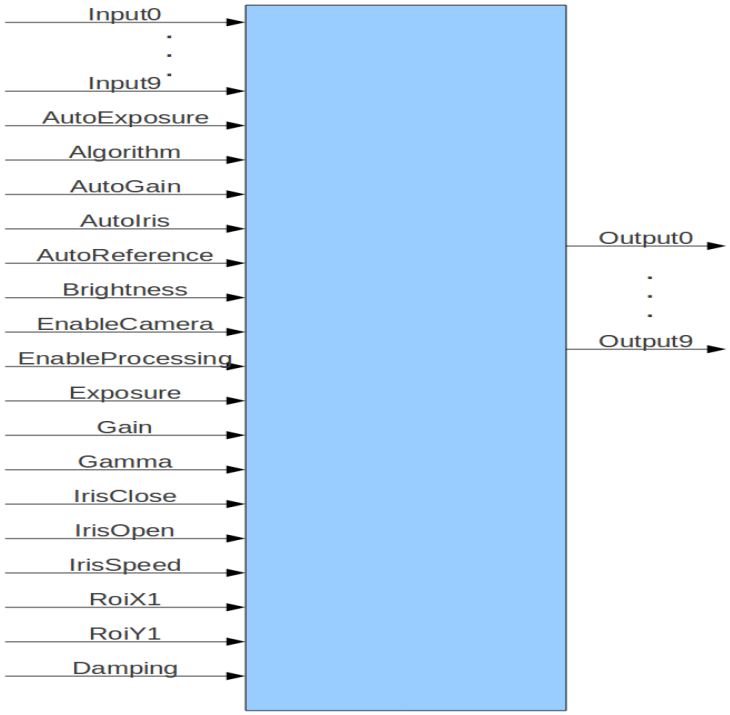
3.4 Povezava modula z glavno aplikacijo

Programski modul za nadzor kamere podatke za delovanje pridobiva iz dveh virov. Podatki, ki se med delovanjem sistema ne spreminjajo, so shranjeni v podatkovni bazi, modul pa jih prebere ob inicializaciji povezave s fizično napravo.

Vrednosti podatkov se nastavlja s pomočjo posebnih urejevalnikov lastnosti naprav, njihove spremembe pa glavna aplikacija upošteva šele ob vsakem novem zagonu in se med delovanjem sistema ne spreminjajo. Za parametre, katerih vrednost se med delovanjem sistema spreminja, poskrbi poseben komunikacijski mehanizem, vgrajen v sistemski del glavne aplikacije. Vsak takšen podatek v modulu kamere je povezan z aplikacijo kot bodisi vhodni bodisi izhodni signal. Programski modul za nadzor kamere si lahko predstavljamo kot črno škatlo (slika 3.6): vhodni signali posredujejo informacije iz zunanjega okolja v modul kamere, ta s pomočjo pridobljenih podatkov opravi izračune in jih nato posreduje v zunanje okolje v obliki izhodnih signalov.

Signali za prenos podatkov med modulom kamere in glavno aplikacijo so lahko dveh tipov: digitalni ali analogni. Digitalni signali lahko zavzamejo vrednost 0 ali 1, analogni pa katerokoli vrednost, predstavljivo s 64 bitno predstavitvijo števila s plavajočo vejico.

Podatke, ki jih v okolje s pomočjo izhodnih signalov pošilja modul kamere, lahko deli sistema SCADA uporabljajo za različne namene, bodisi za dodatno obdelavo podatkov bodisi za komunikacijo z različnimi fizičnimi napravami s pomočjo kontrolnikov PLC. Podobno se podatki, ki se spreminjajo med delovanjem sistema SCADA iz zunanjega okolja, prenašajo v modul kamere s pomočjo vhodnih signalov. Takšni podatki lahko pridejo iz različnih virov. Lahko so rezultat različnih nastavitev v grafičnem uporabniškem vmesniku, lahko so rezultat različnih algoritmov, ki delujejo v sistemu SCADA, lahko pa so tudi vrednosti, posredovane iz fizičnih naprav, kot so raznovrstni senzorji povezani preko kontrolnikov PLC. Takšen način komunikacije modula kamere z glavno aplikacijo omogoča preprosto integracijo z obstoječim sistemom SCADA, se odziva na spremembe okolja in vanj pošilja ustrezne podatke.

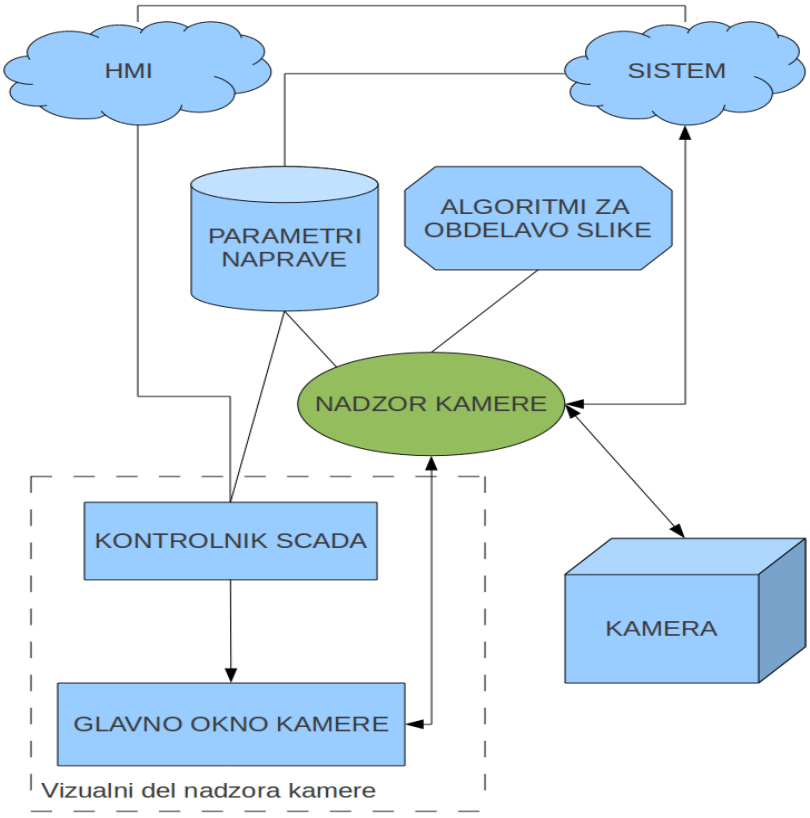


Slika 3.6: Shema programskega modula za nadzor kamere

3.5 Arhitektura sistema

Rešitev za nadzor proizvodnega sistema s pomočjo kamere je zasnovana tako, da znotraj obstoječega sistema deluje kot zaključena celota, ki združuje kontrolne elemente za delo s kamero. Vsebuje potrebne mehanizme za komunikacijo z glavno aplikacijo. To modulu kamere omogoča, da se z le nekaj manjšimi nastavitvami v konfiguraciji vključi v celoten sistem SCADA. modul kamere vsebuje vizualne dele, ki uporabniku ponujajo preprost grafični vmesnik za delo s kamero. Programski deli nadzorujejo napravo, prenašajo podatke med deli kontrole kamere in jih usklajujejo. Posebni komunikacijski deli pa poskrbijo za komunikacijo med modulom in aplikacijo.

Na sliki 3.7 je prikazana osrednja programska enota, poimenovana nadzor kamere, ki sem jo implementiral kot osrednjo stično točko za povezavo delov modula. Služi kot povezava med fizično napravo, vizualnimi deli modula kamere in glavno aplikacijo. Ob zagonu sistema preveri ali je kamera prisotna v omrežju. Če je njeno iskanje uspešno, poskrbi za osnovno vzpostavitev ko-



Slika 3.7: Arhitektura sistema nadzora kamere

munikacijskih vmesnikov med modulom in glavno aplikacijo. Ko je postopek končan, lahko deli modula s pomočjo nadzora kamere nastavljajo parametre na fizični napravi. Takšni parametri so odpiranje in zapiranje zaslonke, nastavljanje različnih parametrov zajema slike (svetlost, gama popravek ipd.). Vsi podatki, ki jih obdeluje modul kamere, se prenašajo med deli celotnega sistema s pomočjo nadzora kamere. To vključuje podatke, ki jih modul pošilja glavni aplikaciji, glavna aplikacija modulu ali katerikoli del sistema SCADA kameri (fizični napravi). Nadzor kamere pri inicializaciji povezave z napravo poskrbi tudi za nastavljanje vseh morebitnih filtrov za obdelavo slike na podatkovni tok podatkov, ki teče od izvora do ponora slike.

Z vidika celotnega proizvodnega sistema je glavna naloga podsistema za delo s kamero posredovanje podatkov, ki jih modul kamere priskrbi iz pri-

dobljene slike. Sistem je zasnovan na način, da je za pridobivanje različnih podatkov preprosto zamenjati algoritem, ki iz slike izlušči podatek, pomemben za potrebe procesa. Nadzor kamere ob inicializaciji preveri ali ima na voljo implementacijo algoritma za obdelavo slike. Ta poleg postopka za pridobivanje podatkov, pomembnih za industrijski proces, vsebuje tudi kodo, ki (če je ta možnost na voljo) informacije o podatkih tudi grafično prikaže na prikazovalnem delu podatkovnega toka. Če je algoritem prisoten, modul kamere na zahtevo glavne aplikacije njenemu sistemskemu delu posreduje parametre, ki jih izračuna pri obdelavi slike.

Sistemski del glavne aplikacije izračunava vrednosti s frekvenco 50 milisekund. Glavna aplikacija torej lahko s tako frekvenco zahteva ali pošilja podatke od modula kamere. Ker je modul vezan na sistemske parametre, od katerih nekateri nadzirajo funkcije fizične naprave, bi se lahko zgodilo, da bi napravo "poplavili" s prehitrimi zahtevki po spremembah parametrov. Pri posredovanju parametrov med modulom in fizično napravo zato posreduje posebna programska nit, ki skrbi izključno za prenose parametrov med tema dvema deloma rešitve [16]. Nit poskrbi, da se spremembe posredujejo izključno ob spremembah parametrov, tako iz smeri fizične naprave kot iz smeri programskega modula. Deluje tudi kot blažilnik hitrih sprememb, ki bi potencialno lahko povzročile napake in neodzivnost programskih knjižnic, s katerimi se kontrolira fizično napravo.

Vizualni del sistema za nadzor kamere je sestavljen iz kontrolnika SCADA in glavnega okna kamere. Kontrolnik SCADA služi za integracijo v obstoječ vmesnik HMI. Sistemski del glavne aplikacije in vmesnik HMI sta povezana s komunikacijskimi mehanizmi, ki si preko nadzora kamere sporočajo vrednosti parametrov. Sistem glavne aplikacije je tako obveščen o spremembah vrednosti parametrov, ki jih uporabnik nastavlja v vizualnih delih modula kamere, ti pa so obveščeni o spremembah, ki bi morebiti izvirale iz sistemskega dela glavne aplikacije. Na ta način je uporabniku zagotovljena ne samo možnost nadzora nad spreminjanjem podatkov, marveč tudi možnost videti realno sliko parametrov sistema.

Glavno okno kamere je poseben del za nadzor parametrov modula kamere, ki je dostopno preko kontrolnika SCADA. Čeprav lahko uporabnik do glavnega okna kamere dostopa le s pomočjo kontrolnika SCADA, pa ima lastne mehanizme za komunikacijo z nadzorom kamere. Ko je uporabniku glavno okno kamere na voljo, to za svoje delovanje in komunikacijo s sistemom ne potrebuje kontrolnika SCADA in se obnaša kot popolnoma neodvisen del aplikacije. To je še posebej primerno za primer, ko želi uporabnik imeti glavno okno kamere na voljo tudi, ko uporablja kakšen drug del aplikacije in želi imeti zaprt vmesnik

HMI.

Kontrolnik SCADA, sistemski del glavne aplikacije in nadzor kamere imajo dostop do podatkov shranjenih v podatkovni bazi (slika 3.7). Ob inicializaciji sistema, nadzor kamere iz nastavitev v podatkovni bazi prebere vrednosti nekaterih parametrov, kot so serijska številka kamere in tip kamere. To so parametri, ki jih nadzor kamere navadno potrebuje le ob zagonu sistema in se med delovanjem sistema ne spreminjajo.

Modul kamere je sposoben izvajati različne algoritme za obdelavo slike, vendar ne več algoritmov hkrati. Aktivni algoritem se izbere s pomočjo nastavitve vrednosti lastnosti kontrolnika SCADA, zamenja pa se ga lahko le, ko je izvajalni sistem glavne aplikacije ustavljen. Ob zagonu glavne aplikacije nadzor kamere pregleda programsko kodo in ugotovi ali vsebuje implementacijo algoritma z imenom, podanim v lastnosti FilterName kontrolnika SCADA.

3.6 Implementacija algoritmov za obdelavo slike

Zajete slike lahko obdelujemo z implementacijo filtrov za obdelavo slike, ki jih knjižnice kamere po zajemu slike znajo uporabljati. Za implementacijo in uporabo filtra je v programski kodi najprej treba kreirati razred filtra za obdelavo slike in dedovati [17] od razreda FrameFilterImpl. Nato je potrebno implementirati tri metode: GetSupportedInputTypes, GetTransformOutputTypes in Transform, ki skrbijo za obdelavo slike [18].

3.6.1 Metoda GetSupportedInputTypes

```
public void GetSupportedInputTypes(ArrayList frameTypes)
```

Metoda definira podprte načine kodiranja slikovnih elementov, ki jih filter sprejme v obdelavo. Tipe kodiranja dodajamo v podatkovno strukturo ArrayList. Primer za dodajanje podprtosti formatu Y800:

```
public override void GetSupportedInputTypes(
    ArrayList frameTypes)
{
    frameTypes.Add(new TIS.Imaging.FrameType(
        TIS.Imaging.MediaSubtypes.Y800));
}
```

3.6.2 Metoda GetTransformOutputTypes

```
public bool GetTransformOutputTypes(FrameType inType,
    ArrayList outTypes)
```

Metoda definira morebitne transformacije med formati v katerih so podani slikovni elementi zajete slike. Parameter inType definira vhodni format, parameter outTypes pa vsebuje formate, katere lahko filter transformira iz danega formata podanega v parametru inType. Metoda vrne vrednost true, če je napolnila parameter outTypes z vrednostmi in false, če ni na voljo nobena transformacija. V primeru, da transformacije formata, v katerem so podani slikovni elementi, ne želimo, lahko med izhodne formate podamo kar vhodni format in vrnemo vrednost true:

```
public override bool GetTransformOutputTypes(FrameType inType,
    ArrayList outTypes)
{
    outTypes.Add(inType);
    return true;
}
```

3.6.3 Metoda Transform

```
public bool Transform(IFrame src, IFrame dest)
```

Sistem nad vsako zajeto sliko izvede metodo Transform. To je metoda, ki izvaja dejansko obdelavo slike. V parametru src je podana vhodna slika, v parametru dest pa izhodna slika. V primeru, da izvedemo transformacijo nad parametrom src in ga zapišemo v parameter dest uspešno, metoda vrne vrednost true. Če vrnemo vrednost false, se bo slika podana v parametru src zavrgla.

Zaradi lažje obdelave slike se uporablja unsafe blok kode, ki ga podpira programski jezik C# in nam s tem dovoljuje uporabo kazalcev. Znotraj takšnega bloka mora razvijalec, podobno kot v nekaterih drugih programskih jezikih, sam poskrbeti za upravljanje s pomnilnikom, saj mu pri tem okolje .NET pusti proste roke. Tako se lahko s pomočjo kazalcev premikamo po besedah (angl. bytih) vhodne slike in nastavljamo vrednosti besed izhodne slike.

```
public override bool Transform(IFrame src, IFrame dest)
{
```

```
        unsafe
        {
        if (dest.Ptr == null) return false;
            byte* pIn = src.Ptr;
            byte* pOut = dest.Ptr;
            //zapišimo nespremenjen slikovni element
            //na lokaciji kazalca *pIn
            //na pozicijo kazalca *pOut
            *pOut = *pIn;
        }
    }
```

Ker metoda Transform teče v drugi programski niti [16], je treba v primeru, da želimo kodi metode posredovati kakšne posebne parametre, ki smo jih določili v algoritmu, te posredovati v bloku BeginParameterTransfer() in EndParameterTransfer(). Blok znotraj teh dveh metod poskrbi za sinhronizacijo in se tako izogne neželenim ustavitvam sistema zaradi dostopa do iste pomnilniške lokacije iz več programskih niti. Po končani implementaciji zahtevanih metod filtra lahko filter uporabimo tako, da kreiramo njegovo instanco, jo posredujemo metodi FrameFilterCreate in filter je pripravljen. Lahko ga dodamo na ustrezen podatkovni tok kot vsak drug filter, ki je del knjižnice IC Imaging Control.

Ker je modul kamere zasnovan tako, da bi bila implementacija algoritmov za obdelavo slike kar se da preprosta, za določene zgoraj opisane vidike implementacije posebnih algoritmov uporabniku ni treba skrbeti. Če uporabnik želi implementirati nov algoritem za obdelavo slike, mora kreirati razred algoritma, ki deduje od razreda FilterBase in ga opremiti z atributom ImagingFilter, s katerim označi ime algoritma. Primer deklaracije takšnega razreda:

```
[ImagingFilter(~BrightnessThresholdFilter~)]
internal class BrightnessThresholdFilter : FilterBase
{
}
```

V takšnem razredu lahko uporabnik implementira dve metodi za obdelavo slike: ProcessImageData, v kateri implementira algoritem za obdelavo slike in PaintOverlay, kjer lahko implementira morebiten dodaten prikaz podatkov na sliki.

3.6.4 Metoda ProcessImageData

```
protected override unsafe void ProcessImageData(
    TIS.Imaging.IFrame data)
```

Znotraj te metode uporabnik implementira algoritem za obdelavo slike. Do vhodnih parametrov kamerinega modula lahko dostopa s pomočjo objekta RuntimeParameters. Tako bi do prvega vhodnega parametra modula dostopal na naslednji način:

```
int xPos = (int)IcControl.RuntimeParameters.Input0;
```

Podatke lahko pošilja na izhode modula z metodo SetOutputUnsafe. Na prvi izhod modula bi tako poslal vrednost 100 na naslednji način:

```
IcControl.SetOutputUnsafe(0, 100);
```

Do posameznih slikovnih elementov slike lahko uporabnik znotraj te metode dostopa na enak način kot je opisan pri uporabi metode Transform (glej 3.6.3).

3.6.5 Metoda PaintOverlay

```
public override void PaintOverlay(OverlayBitmap overlay)
```

V primeru, da želi uporabnik na prikazovalni del glavnega okna kamere dodati grafične elemente, ki jih kamera ne zajema, lahko to stori z implementacijo te metode. S klicem metode:

```
Graphics g = overlay.GetGraphics();
```

dobi dostop do standardnega C# grafičnega objekta, s pomočjo katerega lahko riše poljubne grafične elemente, katere modul kamere prikaže na prikazovalnem delu glavnega okna kamere. Več o delu z grafičnim objektom bralec lahko najde v [19].

V primeru uporabljenega algoritma pri procesu izdelave surovca optičnega vlakna, je bilo potrebno za implementacijo novega algoritma implementirati metodo ProcessImageData in metodo PaintOverlay za prikaz dodatnih informacij na zajeti sliki (specifičen primer uporabe razvitega modula je opisan v poglavju 4). Ukvarjanje z bolj specifičnimi vidiki vključevanja algoritmov v modul je razvijalcu algoritmov prihranjeno. Z opisanimi načini dostopa do parametrov modula (glej 3.6.4) ima uporabnik pri implementaciji algoritmov na voljo vsa sredstva za komunikacijo s sistemom glavne aplikacije preko vhodov in izhodov modula kamere.

Poglavje 4

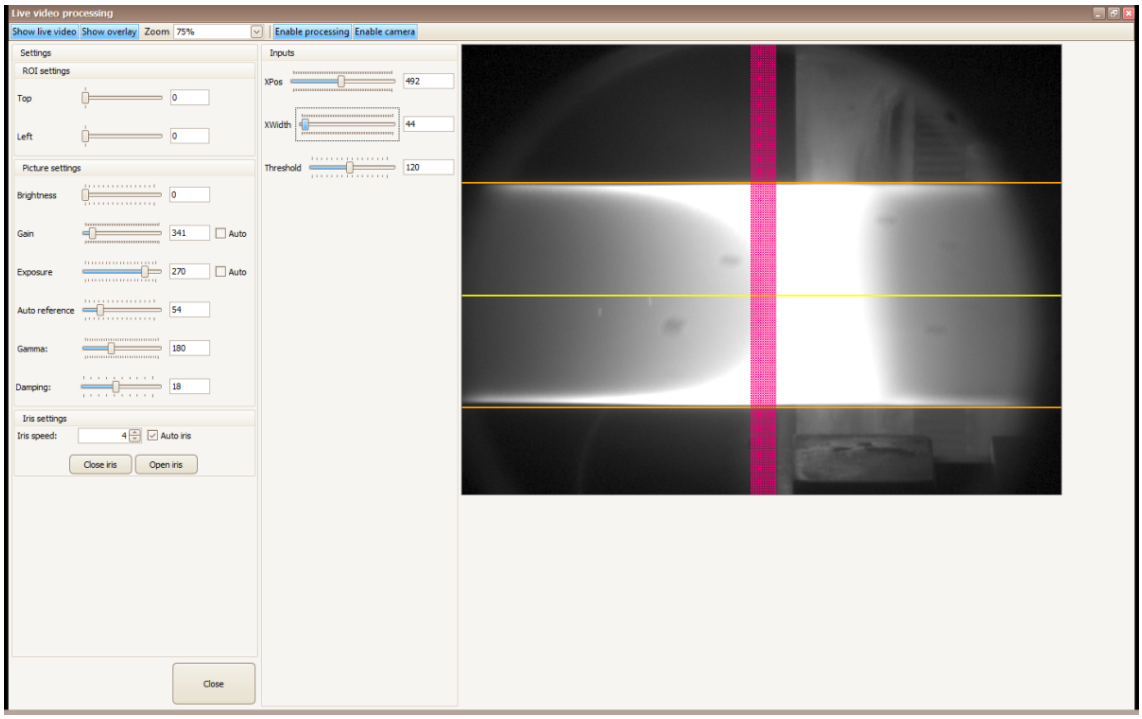
Primer uporabe

Modul, ki sem ga razvil in opisal v tej diplomski nalogi, je bil najprej preizkušen pri nadzoru procesa izdelave surovca za optična vlakna [6]. Kamera je v realnem času zajemala sliko obdelovanca in jo preko nadzora kamere (glej poglavje 3.5) pošiljala v sistem glavne aplikacije. Ta je rezultate obdelave slike nato uporabil za različne izračune, na njihovi podlagi pa je nadzoroval potek določenih delov proizvodnega procesa. Razviti modul je operaterju, ki je vodil proces, omogočil prilagoditev modulovega vmesnika potrebam konkretnega proizvodnega procesa. Glede na potrebe procesa je operater nastavil lastnostim kontrolnika SCADA naslednje vrednosti:

```
<MaxExposure>333</MaxExposure>
<FilterName>BrightnessThresholdFilter</FilterName>
<FilterParameters>0</FilterParameters>
<FlipHorizontal>true</FlipHorizontal>
<FlipHorizontalDisplay>false</FlipHorizontalDisplay>
<AutoMaxExposure>false</AutoMaxExposure>
<FlipVertical>false</FlipVertical>
<FlipVerticalDisplay>false</FlipVerticalDisplay>
<FrameRate>30</FrameRate>
<ImageFormat>Y800</ImageFormat>
<InputNames>XPos;XWidth;Threshold</InputNames>
<ROIHeight>768</ROIHeight>
<ROIWidth>1024</ROIWidth>
<Rotation>Deg90</Rotation>
<RotationDisplay>None</RotationDisplay>
```

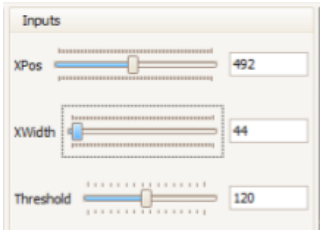
Interesno območje slike, določeno z lastnostima ROIWidth in ROIHeight, je operater nastavil kar na celotno območje zajete slike, to je 1024×768 slikovnih

elementov (za opis lastnosti kontrolnika SCADA glej poglavje 3.3.1). Za potrebe uporabljenega algoritma za obdelavo slike, je operater sliko rotiral za 90 stopinj in preslikal po horizontali. To sta mu na preprost način omogočili lastnosti kontrolnika SCADA "Rotation" in "FlipHorizontal". Rezultat nastavitev prikazuje slika 4.1 glavnega okna kamere narejena med izvajanjem procesa.



Slika 4.1: Glavno okno kamere med potekom procesa

Parametre za potrebe proizvodnega procesa je računal algoritem z imenom BrightnessThresholdFilter. Algoritem je za potrebe procesa želel razviti naročnik sam, kar modul, ki sem ga razvil, tudi omogoča. Na željo naročnika zaradi konkurenčne prednosti algoritem tukaj ni opisan. Naročnik je algoritem lahko vključil v sistem s postopkom, opisanim v poglavju 3.6. Ker so želeli nekatere rezultate in parametre algoritma prikazati na zaslonu, sem njihovi implementaciji algoritma dodal možnost prikaza določenih parametrov na prikazovalni del glavnega okna kamere (slika 4.1). Pomembnejši rezultati obdelave slike, ki jih je omenjeni algoritem računal in pošiljal sistemu glavne aplikacije v obdelavo, so središče in robovi obdelovanca. Središče sem v prikazovalnem delu glavnega okna slike prikazal z rumeno horizontalno črto po celi



Slika 4.2: Avtomatsko generiran uporabniški vmesnik za nastavljanje vhodnih parametrov

širini slike, robova pa z oranžnima horizontalnima črtama.

Za potrebe delovanja algoritma je operater kontrolniku SCADA podal tri vhodne parametre: XPos, XWidth in Threshold. Vsi trije parametri so bili posredovani z analognimi signali (glej 3.4). Modul kamere je s pomočjo vrednosti, podanimi v lastnosti InputNames kontrolnika SCADA, generiral tri vhodne kontrolnike, sestavljene iz drsnika in vnosnega polja. Z njimi je bilo mogoče nastavljati vrednosti omenjenih vhodnih parametrov (slika 4.2). Za opis postopka avtomatskega generiranja uporabniškega vmesnika glej poglavje 3.3.2.4.

Vrednosti parametrov XPos in XWidth sem na željo naročnika grafično prikazal na prikazovalnem delu glavnega okna kamere (glej poglavje 3.6.5). Pri uporabljeni implementaciji algoritma sta prikazana z območjem rdeče barve, ki poteka po celotni višini sike od njenega zgornjega do spodnjega roba. Parameter XPos določa pozicijo skrajno levega dela tega območja, parameter XWidth pa širino tega območja. Z uporabo avtomatsko generiranih drsnikov in vnosnih polj za omenjena parametra, lahko operater med obdelavo slike nastavlja vrednosti teh parametrov. Rdeče območje se premika glede na nastavitve in mu nazorno prikazuje rezultate njegovega dela.

Zaradi prilagodljive zasnove modula kamere si je operater procesa lahko popolnoma prilagodil razviti modul glede na potrebe specifičnega surovca za optična vlakna, ki ga je v tistem trenutku obdeloval. Prilagojen uporabniški vmesnik modula kamere je operaterju omogočil nadzor nad obdelovancem v procesu, preprosto kontrolo določenih (tudi posebej za ta primer definiranih) procesnih spremenljivk in spremljanje rezultatov njegovih nastavitev na zaslonu v realnem času.

Poglavje 5

Zaključek

Pri izdelavi diplomske naloge sem načrtoval in implementiral programsko rešitev za krmiljenje industrijskega procesa s pomočjo digitalne kamere. Posebej v ta namen sem razvil uporabniški vmesnik za upravljanje tako s fizično napravo kot s parametri procesa.

Uporabniški vmesnik se brez težav integrira v obstoječi sistem kot del vmesnika HMI. S pomočjo programskih knjižnic modul nadzira nastavitve kamere in od nje pridobiva informacije. Vmesnik za nadzor kamere je zasnovan tako, da se vanj lahko vključijo različni algoritmi za obdelavo slik, ki lahko služijo različnim namenom. Rezultati, ki jih algoritmi producirajo, se lahko posredujejo drugim algoritmom v obdelavo, če pa je zaradi narave procesa smiselno, lahko te rezultate aplikacija posreduje različnim komunikacijskim modulom ali kontrolnikom PLC za neposredno kontrolo naprav. Ker je programska rešitev integrirana v sistem SCADA obstoječe aplikacije (in bi v skrajnem primeru skupaj s fizičnimi krmilniki lahko bila sama svoj, sicer zelo okrnjen, sistem SCADA), bi lahko rekli, da je razvita generična rešitev, ki jo lahko uporabimo za kontrolo različnih industrijskih procesov.

Kot verjetno pri vsakem kontrolnem sistemu, bi se tudi pri razvitem modulu lahko našle možnosti za izboljšave. Poleg izboljšav uporabniškega vmesnika je v procesu še nekaj prostora za optimizacijo z dodajanjem različnih filtrov na podatkovni tok zajete slike. Možnost izboljšave vidim tudi v morebitni dodatni funkcionalnosti za dinamično dodajanje in odvzemanje algoritmov za obdelavo slike. Ker pa so v industriji funkcionalnosti kontrolnega sistema v veliki meri določene s procesom, ki ga upravljajo, je generična rešitev, ki dopušča raznovrstne dodelave, dobra osnova za reševanje zapletenih izzivov krmiljenja proizvodnih procesov.

Slike

Tabele

Literatura

[1] Bolton W.: Programmable logic controllers. Elsevier, Burlington, ZDA, 2009

[2] Bailey D., Wright E.: Practical SCADA for industry. Elsevier, Oxford, Velika Britanija, 2003, strani 1-4, 17

[3] Wiles J.: Techno Security's guide to securing SCADA. Elsevier, Burlington, ZDA, 2007, stran 66

[4] Mahalik N. P.: Fieldbus technology: Industrial network standards for real-time distributed control. Springer, Berlin, 2003, strani 10-12

[5] Kalapatapu R.: SCADA protocols and communication trends. Dostopno na:
http://www.isa.org/intech/April2005/NetCommDept
(datum zadnjega obiska: 10. 4. 2011)

[6] Freudenrich C.: How fiber optics work. Dostopno na:
http://communication.howstuffworks.com/fiber-optic-communications/fiber-optic5.htm (datum zadnjega obiska: 9. 4. 2011)

[7] Brandl D.: Design patterns for flexible manufacturing. ISA - Instrumentation, Systems and Automation society, United states, 2006, strani 20-22

[8] Microsoft, Overview of the .NET framework. Dostopno na:
http://msdn.microsoft.com/en-us/library/a4t23ktk.aspx
(datum zadnjega obiska: 15. 4. 2011)

[9] PC-based Control: The new performance class of Beckhoff Industrial PCs. PC-Control, November 2010, strani 8-11

[10] TwinCat 3 - Convergence of technologies. Dostopno na:
http://www.pc-control.net/pdf/042010/pcc_0410_twincat3_e.pdf
(datum zadnjega obiska: 20. 4. 2011)

[11] The Imaging Source Cameras. Dostopno na:
http://www.theimagingsource.com/en_US/products/cameras/gige-ccd-mono/dmk31ag03i/
(datum zadnjega obiska: 25. 4. 2011)

[12] Petrou M., Petrou C.: Image processing: The fundamentals, 2nd edition. John Wiley & Sons, Chichester, Velika Britanija, 2010

[13] RGBA color space. Dostopno na:
http://en.wikipedia.org/wiki/RGBA_color_space
(datum zadnjega obiska: 25. 4. 2011)

[14] The Imaging Source Europe GmbH: IC Imaging Control .NET: Frame filters. Dostopno na:
http://www.imagingcontrol.com/en_US/support/documentation/dotnet/tech_FrameFilter.htm (datum zadnjega obiska: 20. 4. 2011)

[15] Gamma correction. Dostopno na:
http://en.wikipedia.org/wiki/Gamma_correction
(datum zadnjega obiska: 25. 4. 2011)

[16] Microsoft: Threads and Threading. Dostopno na:
http://msdn.microsoft.com/en-us/library/aa720724%28v=vs.71%29.aspx
(datum zadnjega obiska: 15. 4. 2011)

[17] Microsoft: Inheritance (C# programming guide). Dostopno na:
http://msdn.microsoft.com/en-us/library/ms173149%28v=VS.100%29.aspx
(datum zadnjega obiska: 15. 4. 2011)

[18] The Imaging Source Europe GmbH: Writing a Frame Filter: Binarization. Dostopno na:
http://www.imagingcontrol.com/en_US/support/documentation/dotnet/Binarization.htm
(datum zadnjega obiska: 20. 4. 2011)

[19] Graphics class. Dostopno na:
http://msdn.microsoft.com/en-us/library/system.drawing.graphics.aspx
(datum zadnjega obiska: 25.4.2011)