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: 00517/2010
Datum: 05.04.2010

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

Kandidat: **UROŠ TOMŠIČ**

Naslov: **IZBOLJŠANJE NATANČNOSTI VODENJA ROBOTSKE ROKE Z ELEMENTI RAČUNALNIŠKEGA VIDA**

**IMPROVEMENT OF ROBOT ARM CONTROL USING COMPUTER VISION APPROACH**

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija

Tematika naloge:

Za vodenje industrijskih procesov se danes običajno uporabljajo dragi senzorski sistemi, prilagojeni za težke razmere v industrijskem okolju. Zaradi nezdružljivosti industrijskih sistemov z napravami široke potrošnje smo tako tudi pri vodenju procesov v prijaznih okoljih primorani uporabljati drage senzorske sisteme.

V delu raziščite možnosti uporabe višjih sistemov vodenja za povezovanje naprav v polju s cenovno ugodnimi izdelki široke potrošnje. Izbrani sistem uporabite za integracijo spletne kamere v sistem vodenja robotske roke. Pokažite, da lahko z uporabo osnovnih elementov računalniškega vida in predlaganega višjega sistema učinkovito izboljšamo natančnost vodenja obstoječe robotske roke.

Mentor:
prof. dr. Uroš Lotrič

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:
prof. dr. Franc Solina

UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Uroš Tomšič

# Izboljšanje vodenja robotske roke z uporabo elementov računalniškega vida

DIPLOMSKO DELO
NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: izr. prof. dr. Uroš Lotrič

Ljubljana, 2010

# IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani Uroš Tomšič,
z vpisno številko 63040300,

sem avtor diplomskega dela z naslovom:
Izboljšanje natančnosti vodenja robotske roke z elementi računalniškega vida

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Uroša Lotriča
- 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 6.7.2010 Podpis avtorja/-ice:

# Zahvala

Za vsestransko podporo, strokovno pomoč in nasvete pri izdelavi diplomske naloge bi se rad zahvalil mentorju na Fakulteti za računalništvo in informatiko v Ljubljani, izr. prof. dr. Urošu Lotriču. Zahvaljujem se tudi staršem, ki so mi študij omogočili.

# Kazalo vsebine

# Povzetek

V diplomski nalogi je predstavljena izdelava krmilnega in nadzornega sistema z elementi računalniškega vida za upravljanje z robotsko roko. V avtomatizaciji industrijskih procesov velja, da je strojna oprema prilagojena za delo v industrijskem okolju. Zaradi take prilagodljivosti je oprema zelo draga, poleg tega so ti sistemi nezdružljivi z napravami široke potrošnje, zato smo tudi pri vodenju procesov v opremi bolj prijaznih okoljih primorani uporabljati senzorske sisteme, ki so prirejeni delovanju v industrijskih razmerah. Cilj diplomske naloge je izboljšati vodenje robotske roke z uporabo cenovno ugodnih potrošniških elementov in integrirati sistem računalniškega vida v lasten nadzorni sistem. Pri tem pa uporabiti prosto dostopne in brezplačnih sisteme za zagotavljanje stabilne komunikacije in prenosa podatkov med krmilnim računalnikom in nadzornim sistemom.

Glavni del diplomske naloge, ki opisuje razvoj celotnega sistema, je razdeljen na dva dela. V prvem delu je predstavljen sistem krmiljenja robotske roke, struktura programa, avtomatski cikel upravljanja z in brez uporabe analize slike in postavitev v osnovno stanje. V drugem delu, je poudarjen predvsem razvoj nadzornega sistema, ki smo ga razvijali v programskem jeziku C# v okolju Visual Studio 2008. Sistem vsebuje osnovno vizualizacijo, upravljanje robotske roke v dveh režimih delovanja (avtomatski, ročni režim), vmesnik za upravljanje z alarmi in obvestili, nastavljanje podatkov v povezavi z vodenjem robotske roke, shranjevanjem podatkov v podatkovno bazo, obdelavo določenih podatkov in grafični ter tabelarični prikaz le-teh. Na sistem je priključena tudi spletna kamera, ki uporabniku aplikacije nudi pregled delovanja robotske roke in služi analiziranju ter obdelovanje podatkov v zvezi s finim pozicioniranjem roke. Sledi opis vgradnje žive slike v sistem, analiza slike in določanje kontrolnih točk ter opis sistema arhiviranja in obdelave podatkov.

Ključne besede: Avtomatizacija, krmilni računalnik (PLC), nadzorni sistem (SCADA), računalniški vid, analiza slike, komunikacijski protokoli, ogrodje .NET

# Abstract

The diploma work presents a control system with added computer vision system for more accurate robot hand positioning. In process automation computer equipment adapted for industrial environment is usually used. Such adaptability brings about very high costs. Since these systems are usually not compatible with consumer goods systems, it is necessary to use sensor systems adapted for operation in industry also in friendlier environments. The diploma work demonstrates that moderate efforts enable significant improvement of control systems with elements of consumer goods and the use of freely accessible software libraries.

In the aspect of the approach mentioned, we tried to improve the control of the robot hand. We built our own control system into which the elements of computer vision are included. The main part of the diploma work describing the development of the entire system is divided into two parts. In the first part the robot hand control system, the programme structure, the automatic control cycle with or without the use of an image analysis and the setting of the system into the initial state are presented. The second part is focused on the development of the SCADA system in the C# programme language in the Visual Studio 2008 environment. The system includes the basic visualisation, the robot hand control in the two automatic and manual operation regimes, the alarm/notification interface, the setting of relevant data, saving of data in the data base, processing of certain data and their graphic/table review. The system also includes a web camera interface that enables a user to supervise the robot hand operation. It is used in an analysis and processing during the fine positioning of the robot hand. It also includes the incorporation of a live image into the system including the image analysis, setting of the control points and description of data archiving and processing.



Key words: automation, programmable logic controller (PLK), control system (SCADA), computer vision, industrial communication protocols

# Uvod

Proizvodna podjetja se danes pri poslovanju srečujejo s konkurenco na globalnem trgu. Za nižanje cene izdelkov morajo vedno bolj optimizirati proizvodni proces tako glede porabe surovin in energentov kot tudi glede kvalitete izdelkov. Velik potencial podjetij se ne skriva le v zmožnosti čim večje proizvodnje, temveč predvsem v obvladovanju njihovih procesov. Potrebni pogoj za doseganje tega je nadzorovanje in takojšen vpogled v stanje procesa v realnem času ter morebitno ukrepanje v primeru odstopanj.

Zaradi želje po spremljanju in nadziranju krmilniško vodenih avtomatskih sistemov, se krmilne računalnike povezuje z nadzornimi sistemi. Ti sistemi, ki jih uporabljamo za upravljanje in nadzor avtomatiziranih procesov, predstavljajo vmesnik med človekom in strojem. Poleg vizualizacije (grafična predstavitev procesa) nudijo uporabnikom še druge funkcionalnosti, kot so spremljanje in nastavljanje tehnoloških parametrov (recepture), prikazovanje alarmov in napak povezanih z avtomatiziranim procesom, hranjenje podatkov v podatkovnih bazah in kasnejša grafična ter tabelarična predstavitev podatkov v uporabniku prijaznem vmesniku.

Na drugi strani se v industriji pri implementaciji različnih proizvodnih procesov zelo povečala uporaba računalniškega vida. Računalniški vid je področje računalništva, ki se ukvarja z računalniškimi sistemi, zmožnimi interpretacije in analize slikovnih podatkov. Naloge, ki so jih v preteklosti opravljali ljudje, predvsem za kontrolo kvalitete sestavnih delov ali končnih izdelkov, so prevzeli računalniki, saj so za enostavne in ponavljajoče se naloge veliko bolj natančni, hitri in neutrudljivi. Najbolj pogoste naloge računalniškega vida so razpoznavanje objektov, sledenje objektom, interpretacija scen in izvajanje raznih meritev. Uporablja se predvsem v računalniški, avtomobilski, farmacevtski in prehrambeni industriji [1].

Tako za nadzorne sisteme kot tudi za druge elemente sistemov vodenja v avtomatizaciji industrijskih procesov velja, da morajo biti prilagojeni za delo v industrijskem okolju. Oprema od krmilnih računalnikov, merilnih in izvršnih sistemov, sistemov za komunikacijo do nadzornih sistemov mora biti prilagojena industrijskemu okolju. To pomeni, da mora biti robustna, prirejena za ekstremne temperaturne razmere, vibracije, onesnaženja, elektromagnetne motnje in druge zunanje dejavnike [2]. Poleg omenjene strojne opreme je potrebno zagotoviti tudi ustrezno programsko opremo za izdelavo krmilnih programov, ter nadzornega in ostalih sistemov vodenja. Vse to pa za sabo potegne precejšnje stroške razvoja in posledično tudi nakupa te opreme.

Zaradi nezdružljivosti industrijskih sistemov z napravami široke potrošnje smo tako tudi pri vodenju procesov v opremi bolj prijaznih okoljih primorani uporabljati senzorske sisteme, ki so prirejeni delovanju v industrijskih razmerah. V diplomski nalogi smo želeli pokazati, da je z nekaj iznajdljivosti mogoče razviti lasten nadzorni sistem na eni strani in napreden merilni sistem na drugi strani.

Izdelali smo sistem, ki s pomočjo računalniškega vida izboljša natančnost vodenja robotske roke. Pri tem smo uporabili cenovno ugodne potrošniške senzorske komponente in integrirali sistem računalniškega vida v lasten nadzorni sistem. Pri tem smo se omejili na uporabo brezplačnih in prosto dostopnih sistemov za zagotavljanje prenosa podatkov med našimi programi in krmilnim računalnikom.

Za celovitost pregleda na področje industrijske avtomatizacije so v naslednjem poglavju predstavljeni krmilni računalniki, ki so jedro avtomatiziranih sistemov. Uporablja se jih za upravljanje in vodenje različnih procesov v industriji, pametnih hišah, in drugih avtomatsko vodenih sistemih. Predstavljena je strojna ter programska oprema proizvajalca Siemens, ki smo jo uporabili pri svojem delu. Sledi opis programskih jezikov in opis strukture programov, ki sem jih uporabil pri realizaciji sistema robotske roke.

V tretjem poglavju sem se posvetil splošnemu opisu nadzornih sistemov (SCADA), arhitekturi, principu komunikacije in opisu funkcionalnosti, ki jih taki sistemi vsebujejo. Nadzorne sisteme lahko ustvarimo z namenskimi inženirskimi orodji, ali pa jih razvijemo sami z uporabo običajnih razvojnih orodij.

Pri povezovanju krmilnih računalnikov z nadzornimi sistemi je zelo pomembna stabilna in robustna komunikacija, zato smo v četrtem poglavju podrobneje opisali industrijsko komunikacijo, protokole in področna vodila, s posebnim poudarkom na metodah povezovanja z višjimi sistemi vodenja.

V petem poglavju je nato na laboratorijskem problemu natančnega vodenja robotske roke predstavljen sistem komunikacije s krmilnim računalnikom z uporabo prosto dostopne knjižnice LibNoDave. Opravil in predstavil sem tudi teste in meritve ponovljivosti zajema in analize slike ter meritve natančnosti pozicioniranja robotske roke z in brez uporabe analize slike.

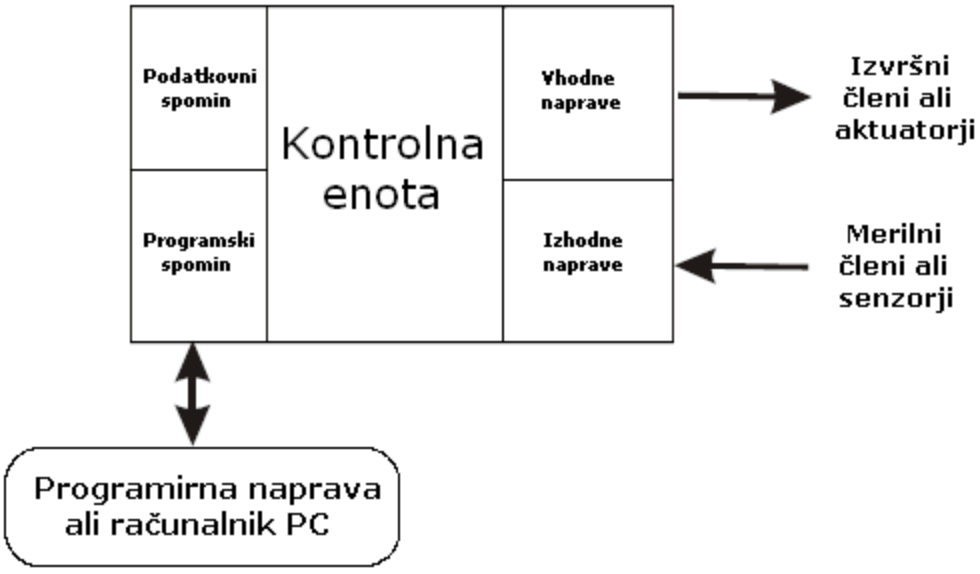
# 1. Programirljivi logični krmilniki

## 2.1 Strojna oprema

Krmilni računalnik oziroma PLK predstavlja jedro avtomatiziranega sistema, s katerim lahko realiziramo različne funkcije vodenja. Programirljive logične krmilnike ali krmilne računalnike na primer uporabljamo za upravljanje industrijskih strojev v kovinarski industriji, upravljanje linij za skladanje izdelkov in analiziranje ter vodenje procesov v kemičnih industrijah [3].

Zgodovina krmilnih računalnikov sega v pozna šestdeseta leta. Glavni razlog za razvoj takšne naprave je bil predvsem drago vzdrževanje in odpravljanje težav na velikih sistemih implementiranih z relejsko tehniko. Prvi krmilni računalnik MODICON 084 (ang. Modular Digital Controller) so razvili v podjetju Belford Associates in je bil kasneje prvi, ki se je proizvajal serijsko [4].

Krmilni računalnik ima klasično strukturo, kot je prikazana na sliki 2.1. Poleg centralno procesne enote ga sestavljajo še štiri glavne enote: programski spomin, podatkovni spomin, vhodne in izhodne enote.




*Slika 2.1: Struktura krmilnega računalnika.*

Iz slike 2.1 je razvidno, da sistem sestavlja tudi programirna naprava ali račundalnik PC, ki ga preko izbranih vodil priklopimo na krmilni računalnik. Uporabljamo ga za pisanje, testiranje, popravljanje, nalaganje programa in spremljanje napak na krmilnem računalniku [3].

Programski jeziki naj bi bili praviloma napisani po standardu IEC 61131-3, vendar se vsi proizvajalci tega standarda ne držijo. Najbolj razširjen in znan programski jezik za krmilne računalnike je kontaktna shema ali lestvični diagram (ang. Ladder Logic, LAD), ki izhaja iz stikalne tehnike. Glavna elementa modula sta mirovno in delovno stikalo [3].

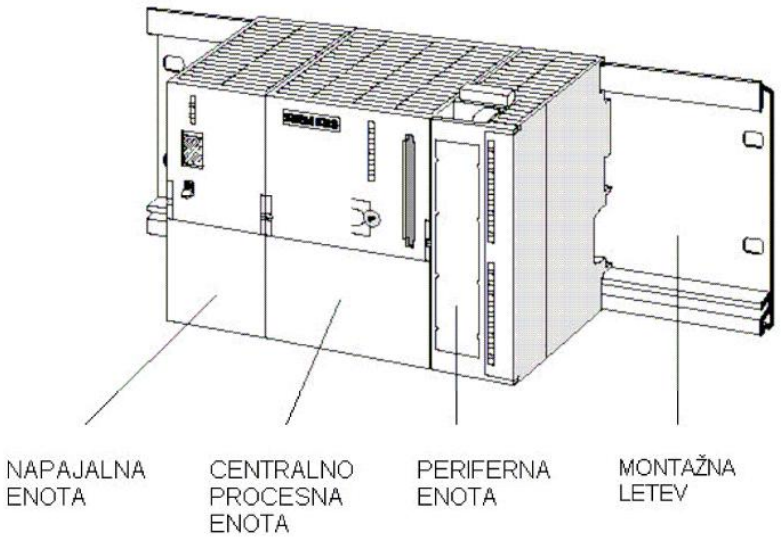
Ko je program naložen na krmilni računalnik, se ta prenese v delovni spomin in začne se posebna ciklična procedura izvajanja programa. Ta kontrolna procedura omogoča ponavljajoče izvajanje cikla, v katerem se preberejo vhodi, izvede se krmilni program in določijo se izhodne spremenljivke. Osnovna tipična kontrolna procedura je prikazana na sliki 2.2. Po vklopu krmilnika se najprej izvede test strojne opreme (ang. sanity check). Po opravljenem testu sistem prebere (preslika) vse vhodne spremenljivke in jih shrani v spomin. Šele zatem se izvede krmilni program, ki določi izhodne spremenljivke. Ta postopek preprečuje, da bi se katerakoli vhodna spremenljivka med izvajanjem programa spremenila, saj bi sprememba le-te lahko povzročila napačno določevanje vrednosti, ali pa bi prišlo do hazardov. V zadnji stopnji cikla se programsko določene izhodne spremenljivke preslikajo na fizične izhode [5].


*Slika 2.2: Cikel izvajanja programa.*

### 2.1.1 Krmilni računalnik Siemens S7-300

Krmilniki S7-300 imajo modularno konstrukcijsko izvedbo (slika 2.3). S kombinacijo komponent iz izbora S7-300 modulov lahko sestavimo sistem, ki ga zahteva projekt. Glede na namen posameznih modulov ločimo:

- napajalno enoto,
- centralno procesno enoto,
- periferne enote,
- funkcijske module,
- komunikacijske vmesnike.




*Slika 2.3: Krmilni računalnik s pripadajočimi moduli.*

Jedro celotnega sistema je centralna procesorska enota. Siemens v seriji S7-300 ponuja trinajst različnih krmilnikov, od tega sedem z osnovno (standardno) in šest z dodatno (31xC) opremo. Krmilnik izberemo glede na zahtevnost procesa, ki ga nameravamo voditi. Čim močnejši procesor je v sistemu, tem krajši so obdelovalni cikli in tem večji je njegov pomnilnik. Tudi število naprav in razširitvenih modulov, ki jih lahko priključimo na krmilnik, se ustrezno veča [6].

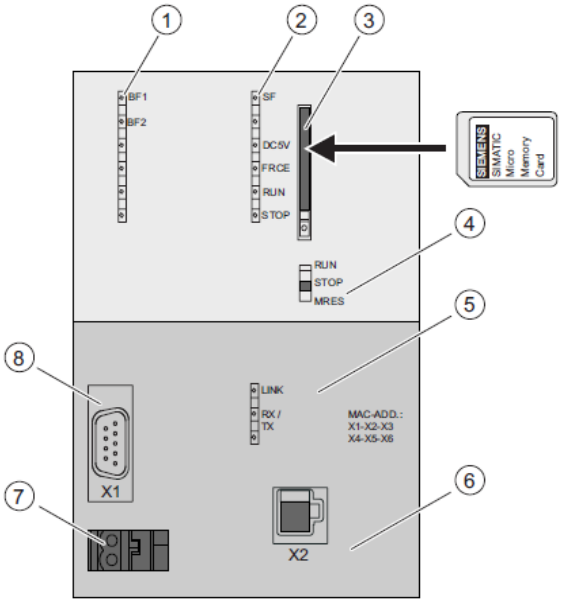
Lastnosti in nekatere karakteristike krmilnega računalnika S7-315-2 PN/DP, ki sem ga uporabil pri izvedbi krmiljenja diplomske naloge (slika 2.4):

- mikroprocesor za hitro obdelavo ukazov,
- interni pomnilnik 384 kilobajtov (RAM) delovnega spomina,
- krmilniki serije S7-300 potrebujejo tudi pomnilniško kartico (ang. Micro Memory Card), ki je na voljo s spominom od 64 KB do 8 MB, v mojem primeru sem uporabil kartico 64 KB spomina,
- izvedbeni časi:
    - za izvedbo bitne operacije najmanj 0,05 µs,
    - za izvedbo besedne operacije najmanj 0,09 µs,
    - za izvedbo aritmetičnih operacij najmanj 0,12 µs,
    - za izvedbo aritmetičnih operacij v plavajoči vejici njamanj 0,45 µs.
- 256 časovnikov z obsegom od 10 ms do 9990 s,
- 256 števcev z možnostjo štetja od 0 do 999,
- podatkovni spomin:
    - bitni spomin (M – ang. memory area) – 2048 bajtov – delovni spomin krmilnika,
    - podatkovni, funkcijski, organizacijski bloki – vseh skupaj je lahko 1024 z največjo velikostjo posameznega bloka 64 kB – podatkovni spomin krmilnika (MMC).
- naslovni spomin:
    - vhodni – 2024 B,

- izhodni – 2024 B,
- porazdeljeni:
  - vhodni – 2024 B,
  - izhodni – 2024 B,
- analogni:
  - vhodni – 1024 kanalov,
  - izhodni – 1024 kanalov,

- komunikacijska vmesnika:
  - integrirani vmesnik RS485, ki je lahko parametriran kot vmesnik MPI (ang. Multi Point Interface) ali PROFIBUS DP z možnostjo priklopa do 32 naprav,
  - PROFINET – industrijski Ethernet z omejitvijo priklopov do 32 naprav.

1. Statusni prikazovalnik prvega vodila.
2. Indikatorji stanja in napak.
3. Reža za SIMATIC Micro Memory CARD.
4. Stikalo za izbiranje stanja krmilnika.
5. Statusni prikazovalnik drugega vodila.
6. Drugo vodilo – PRFINET.
7. Priklop napajanja.
8. Prvo vodilo – MPI/DP [6].




*Slika 2.4: Krmilni računalnik 315-2 PN/DP.*

## 2.2 Programska oprema

Programska oprema STEP 7 predstavlja razvojno orodje za krmilnike Siemens SIMATIC S7-300, S7-400 in po novem tudi za S7-1200. Sestavlja ga več posameznih aplikacij, kjer vsaka opravlja določeno funkcijo. Tako imamo na voljo funkcije, ki nas spremljajo od začetka ustvarjanja projekta, do njegovega zaključka [7]. Funkcije lahko strnemo v naslednje skupine:

- funkcije za konfiguracijo strojne opreme,
- funkcije za konfiguracijo omrežij,
- funkcije za programiranje,
- funkcije za testiranje in servisiranje in
- funkcije za dokumentiranje in arhiviranje.

Glavni grafični vmesnik v programskem paketu STEP 7 je SIMATIC Manager. Ta nam iz različnih aplikacij zbere vse potrebne podatke za oblikovanje projekta kot celote. Znotraj samega projekta so podatki razdeljeni glede na funkcijo in predstavljeni kot objekti. Kadar želimo delati s posameznim objektom, se nam aktivira tudi ustrezno orodje za delo z njim [7].

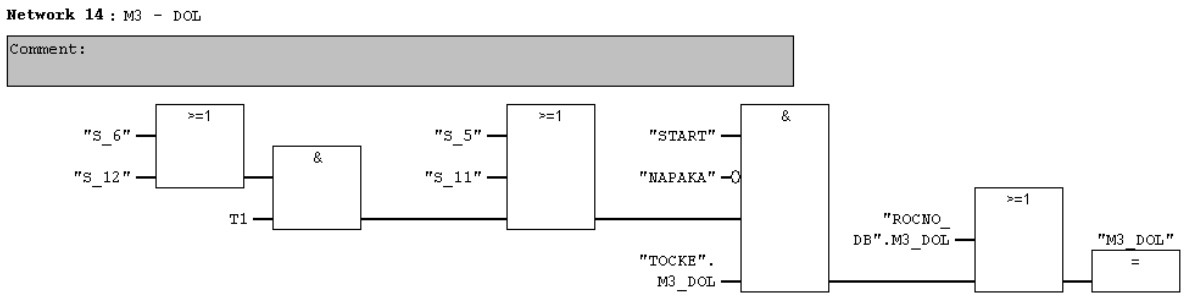
Programski paket STEP 7 nam dovoljuje programiranje v več programskih jezikih, ki so značilni za programirljive logične krmilnike. V aplikaciji, predstavljeni v nadaljevanju, sem uporabil naslednje jezike [7]:

- funkcijski načrt (ang. Function Block Diagram, FBD),
- kontaktni načrt (ang. Ladder Logic, LAD),
- nabor ukazov (ang. Statement List Programming Language, STL),
- strukturiran visokonivojski jezik (ang. Structured Control Language, SCL) in
- programiranje z grafom stanj (ang. State Graph, HiGraph).

Med programskimi metodami je možno tudi prevajanje iz višjih nivojskih jezikov v nižje nivojske jezike, na primer iz funkcijskega načrta v nabor ukazov, obratno pa le v primeru enostavne kode.

### 2.2.1 Funkcijski načrt – FDB

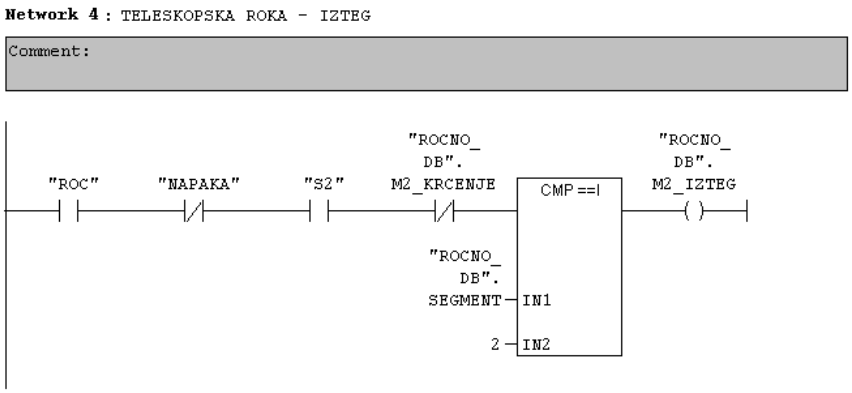
Funkcijski načrt je programski jezik, pri katerem ukaze opisujemo s standardnimi grafičnimi simboli. Programiranje je zato enostavno in hitro razumljivo. Primer funkcijskega načrta je prikazan na sliki 2.5. Iz osnovnih gradnikov lahko zgradimo tudi kompleksne funkcije. Programi, napisani kot funkcijski načrti, so zelo pregledni, kar nam olajša razhroščevanje programov. Ta programski jezik ni najbolj primeren za programiranje kompleksnih računskih operacij in za zahtevnejše obdelave podatkovnih struktur [8].




*Slika 2.5: Primer programske kode v funkcijskem načrtu.*

### 2.2.2 Kontaktni načrt – LAD

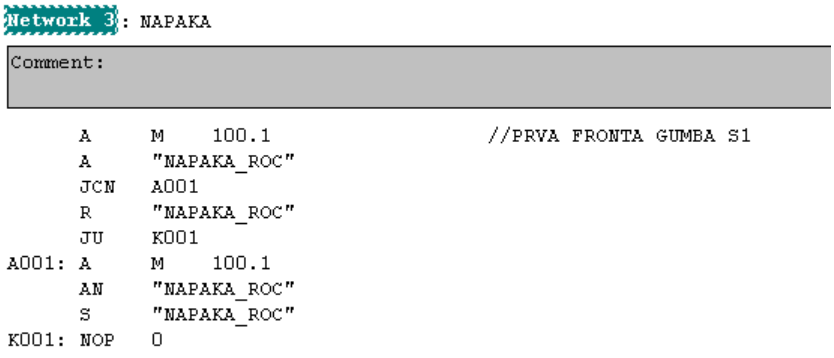
Kontaktni načrt je grafični programski jezik, kjer so ukazi predstavljeni s simboli, ki izhajajo iz stikalne (relejske) tehnike. Primer programske kode je predstavljen na sliki 2.6. Glavna elementa kontaktnega načrta sta delovno in mirovno stikalo, zato se programiranja v njem zelo hitro priučijo tudi električarji. Slabost tega programskega jezikaje nepreglednost, zato ni primeren za implementacijo kompleksnih nalog [8].




*Slika 2.6: Primer programske kode v kontaktnem načrtu.*

### 2.2.3 Nabor ukazov – STL

Naboru ukazov je programski jezik pri katerem ukaze vnašamo v tekstovni obliki. Kot vidimo na sliki 2.7, je struktura jezika je zelo podobna strojnim ukazom. Prednost nabora ukazov je v hitrem izvajanju programa, saj sta optimizirana tako čas obdelave kot tudi lokacija pomnilnika. V samem urejevalniku nam je delo že dodatno olajšano, saj lahko uporabljamo simbolično opisovanje, možne pa so tudi funkcije za iskanje ter sprotno preverjanje pravilnosti vnosa. Urejevalnik omogoča shranjevanje oziroma skladiščenje večkrat uporabljenih programskih delov v standardno programsko knjižnico, s čimer je omogočena kasnejša ponovna uporaba. Vsak ukaz v jeziku STL je sestavljen iz operacijske kode in operanda. Pri vsakem ukazu je mogoče vpisati dodatno označbo, ki jo lahko v programu uporabimo za sklicevanja, na primer pri pogojnih skokih. Tekstovni urejevalnik nam omogoča tudi vpis komentarjev. Slabost tega jezika je slaba preglednost nad večjimi programi in zato tudi težje razhroščevanje [8].




*Slika 2.7: Primer programske kode z uporabo nabora ukazov.*

### 2.2.4 Strukturiran visokonivojski programski jezik – SCL

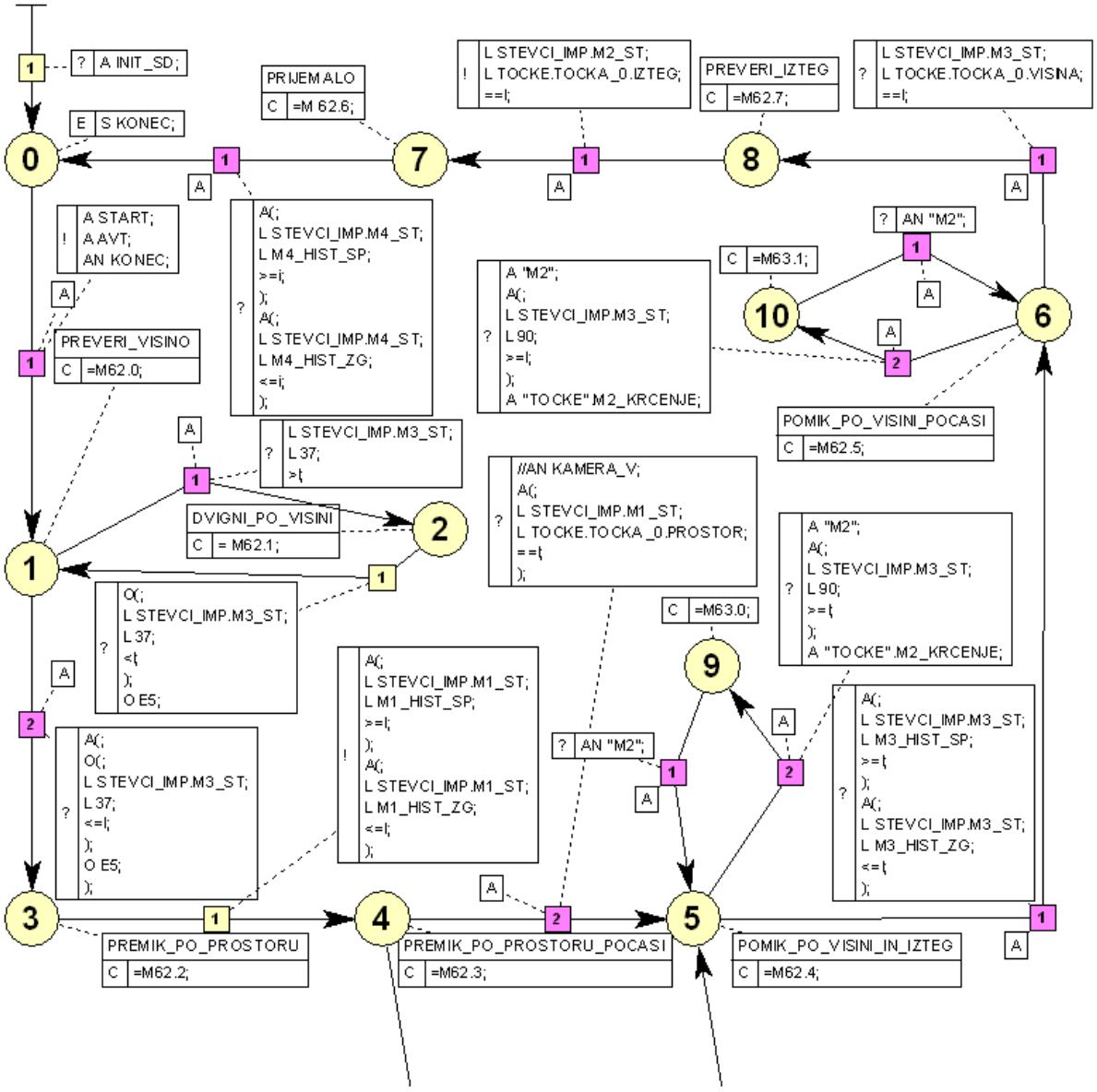
Jezik SCL je visokonivojski tekstovni programski jezik, ki ustreza standardu IEC 61131-3. Kot vidimo na sliki 2.8, je sintaksa zelo podobna programskemu jeziku Pascal. Prednost programiranja v jeziku SCL je v hitrejšem, preglednejšem programiranju, enostavnejšem, hitrejšem ustvarjanju zank in pogojnih skokov. Primeren je za programiranje zahtevnejših računskih operacij, algoritmov in za urejanje podatkovnih struktur. Omogoča sistemsko integracijo v druge programske jezike in podpira podatkovni ter blokovski sistem S7. Programski jezik SCL ni osnovni jezik programskega paketa Simatic STEP 7, saj ga je potrebno dodatno namestiti na sistem [9].

```
//IZRACUN HISTEREZ
    //HISTEREZA M1
    HISTEREZE.M1_HIST_ZG:=CIL_TOCKA.PROSTOR+HISTEREZE.M1_HIST;
    HISTEREZE.M1_HIST_SP:=CIL_TOCKA.PROSTOR-HISTEREZE.M1_HIST;
    //HISTEREZA M2
    HISTEREZE.M2_HIST_ZG:=CIL_TOCKA.IZTEG+HISTEREZE.M2_HIST;
    HISTEREZE.M2_HIST_SP:=CIL_TOCKA.IZTEG-HISTEREZE.M2_HIST;
    //HISTEREZA M3
    HISTEREZE.M3_HIST_ZG:=CIL_TOCKA.VISINA+HISTEREZE.M3_HIST;
    HISTEREZE.M3_HIST_SP:=CIL_TOCKA.VISINA-HISTEREZE.M3_HIST;
    //HISTEREZA M4
    HISTEREZE.M4_HIST_ZG:=CIL_TOCKA.PRIJEM+HISTEREZE.M4_HIST;
    HISTEREZE.M4_HIST_SP:=CIL_TOCKA.PRIJEM-HISTEREZE.M4_HIST;
    //HISTEREZA M3 KAMERA
    HISTEREZE.M3_KAMERA_ZG:=HISTEREZE.M3_KAMERA+HISTEREZE.M3_HIST;
    HISTEREZE.M3_KAMERA_SP:=HISTEREZE.M3_KAMERA-HISTEREZE.M3_HIST;

//DOLOCEVANJE SMERI PREMIKOV
    IF RACUNANJE=TRUE THEN

        PREPOVEDANA_STANJA:= S_14 OR S_15 OR S_16 OR S_19;

        //PREMIK PO PROSTORU
        IF (STEVCI_IMP.M1_ST > CIL_TOCKA.PROSTOR AND
            NOT PREPOVEDANA_STANJA) OR (S_16 AND KAMERA.PREMIK_P=0) THEN
            M1_DESNO:=1;
            M1_LEVO:=0;
        ELSIF (STEVCI_IMP.M1_ST < CIL_TOCKA.PROSTOR AND
               NOT PREPOVEDANA_STANJA) OR (S_16 AND KAMERA.PREMIK_P=1) THEN
            M1_LEVO:=1;
            M1_DESNO:=0;
        ELSE
            M1_LEVO:=0;
            M1_DESNO:=0;
        END_IF;
```

*Slika 2.8:Primer programske kode v strukturiranem visokonivojskim programskim jeziku.*

### 2.2.5 Programiranje s pomočjo grafa stanj – HiGraph

Grafični programski jezik S7-HiGraph je na voljo kot opcijski paket, ki ga je potrebno dodatno namestiti na sistem. Dovoljuje programiranje blokov, ki predstavljajo grafe stanj. Ti grafi dejansko predstavljajo končne avtomate. Realiziramo lahko Mealyev avtomat, Moorov avtomat ali njuno kombinacijo (slika 2.9). Programira se tako, da določujemo stanja, izhode in pogoje prehodov med stanji. Primeren je za programiranje koračnih verig. Dobro organiziran graf stanj nam omogoča sistematično programiranje in poenostavi razhroščevanje. Slabost jezika HiGraph v primerjavi z ostalimi je relativno velika poraba spomina [8].




*Slika 2.9: Primer programske kode narejene s pomočjo grafa stanj.*

### 2.2.6 Struktura programa

#### 2.2.6.1 Linearni program

Celoten uporabniški program v programirljivem logičnem krmilniku je lahko sklenjen v enem programskem bloku. Procesor izvaja program linearno po stavkih, kot so vpisani. V tem primeru je program nepregleden in njegovo razhroščevanje je zelo težavno.

#### 2.2.6.2 Strukturni program

Bolj prikladno je, da celoten uporabniški program razdelimo na bloke. Struktura takega programa je prikazana na sliki 2.10. Blok je samostojen del programa, ki se razlikuje od ostalih po svoji funkciji, nalogi in prioriteti. Ločimo več tipov različnih blokov [7].

- Organizacijski bloki (OB), regulirajo ciklično, časovno in alarmno obdelavo programov. Preko njih uporabnik dostopa do elementov operacijskega sistema. Njihovo izvajanje je določeno glede na prioritete.

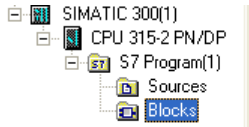
- Funkcije (FC), ki prav tako vsebujejo tehnološke funkcije, vendar so brez zmožnosti pomnjenja lokalnih spremenljivk čez več programskih ciklov. Lastnosti:
  - vrnejo rezultat (brez statičnih podatkov),
  - povečini so brez pomnjenja,
  - primerne so za programiranje pogosto uporabljenih funkcij.

- Funkcijski bloki (FB), so razlikujejo od funkcij (FC) po tem, da lahko pomnijo vse spremenljivke, tudi statične, v vnaprej definiranim točno določenim podatkovnem bloku (ang. instance data block). Do teh podatkov lahko vedno dostopamo iz kateregakoli mesta ali bloka v programu. V programu lahko funkcijski blok kličemo večkrat, če je potrebno, vsakič z drugimi nabori podatkov tako, da funkcijskemu bloku vsakič določimo drugi podatkovni blok. Ta blok na začetku izvajanja kode prenese vhodne podatke v določen podatkovni blok, na koncu izvajanja pa prenese podatke iz določenega podatkovnega bloka nazaj na izhode. Ta sistem zasede več podatkovnega prostora in procesorskega časa, vendar notranje izvajanje algoritma poteka zgolj z uporabo spremenljivk iz določenega podatkovnega bloka, zato je lahko to izvajanje nekoliko hitrejše kot pri funkcijah.

- Podatkovni bloki (DB), služijo za shranjevanje različnih uporabniških podatkov, ki so na voljo vsem delom programa. Omogočajo shramba lokalnih in globalnih podatkov v strukturirani obliki.

- Sistemski funkcijski bloki SFB, sistemske funkcije SFC in sistemski podatkovni bloki SDB so sestavni deli sistema in ne uporabljajo uporabniškega pomnilnika. Uporabnik jih lahko uporablja v svojem aplikativnem programu. Vsebujejo podatke za konfiguracijo posameznih modulov in komunikacijskih kanalov.

- Tabele UDT (ang. User Defined Types) so posebne podatkovne strukture, ki jih lahko deklariramo po svojih potrebah in jih uporabljamo v vseh modulih okolja Step 7. Uporablja se jih lahko kot elementarne podatkovne strukture v funkcijskih in podatkovnih blokih. Tabele UDT so shranjene le v izvorni kodi programa in se ne prenašajo na krmilni računalnik.




| | | | | |
|---|---|---|---|---|
| System data | --- | --- | --- | SDB |
| OB1 | | FBD | 412 | Organization Block |
| FC20 | CASI | STL | 498 | Function |
| FC30 | ROCNO | FBD | 336 | Function |
| FC31 | STEVCI_IM | FBD | 366 | Function |
| FC40 | TIM_S5TI | STL | 158 | Function |
| FC50 | IZHODI | FBD | 726 | Function |
| FC60 | SMERI | SCL | 892 | Function |
| FC61 | AVTOMATSKO | FBD | 1190 | Function |
| FC62 | CIKEL | HIGRAPH | 2086 | Function |
| FC63 | RESET_VERIGE | HIGRAPH | 576 | Function |
| FC65 | KONT_PREMIKOV | SCL | 252 | Function |
| FC66 | NAPAKE | FBD | 288 | Function |
| DB20 | DB_TIME | DB | 56 | Data Block |
| DB21 | DB_S5T | DB | 56 | Data Block |
| DB30 | ROCNO_DB | DB | 40 | Data Block |
| DB31 | STEVCI_IMP | DB | 44 | Data Block |
| DB40 | TOCKE | DB | 130 | Data Block |
| DB50 | NAPAKE_DB | DB | 48 | Data Block |
| DB62 | CIKEL_DB | DB | 348 | Data Block |
| DB63 | HISTEREZE | DB | 66 | Data Block |
| DB64 | RESET_DB | DB | 348 | Data Block |
| DB65 | KONT_PREMIKOV_DB | DB | 46 | Data Block |
| DB66 | KAMERA | DB | 54 | Data Block |
| UDT1 | TOCKA | STL | --- | Data Type |

*Slika 2.10: Struktura celotnega krmilnega programa.*

# 2. Nadzorni sistem – SCADA

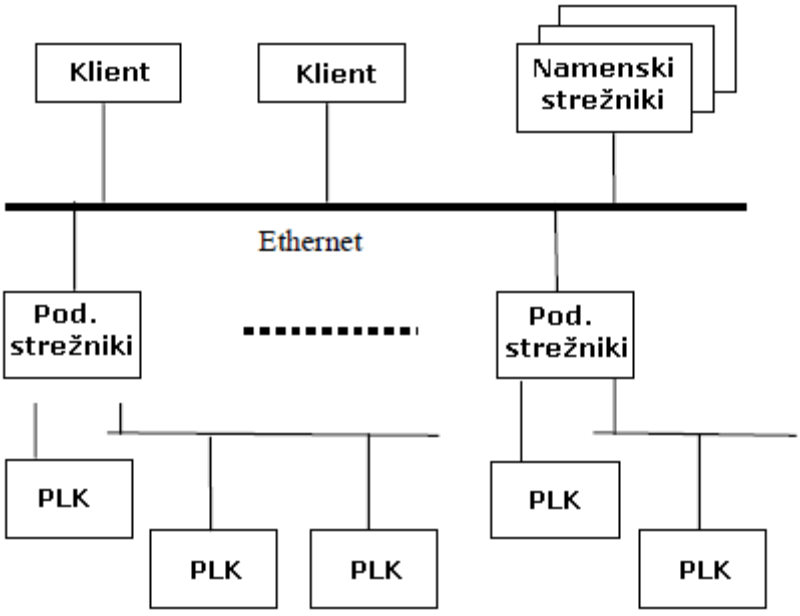
Programska oprema SCADA (ang. Supervisory Control and Data Aquisition) v današnjem času predstavlja nenadomestljiv del sistemov avtomatizacije. Nadzor nad procesom, grafični vmesnik človek-stroj (HMI), arhiviranje procesnih podatkov in alarmiranje so samo nekatere osnovne funkcije sistemov SCADA. Ti sistemi so se na trgu pojavili v sedemdesetih letih, kot namenska programska oprema, ki je v večini primerov dopolnjevala krmilnike ali porazdeljene krmilne sisteme. Čeprav je razmeroma dobro pokrivala osnovno funkcionalnost za specifičen primer (z izjemo grafike), je bila ta generacija programske opreme SCADA zelo neprilagodljiva v vseh pogledih - tako glede sprememb same aplikacije, kot tudi glede povezovanja z drugimi sistemi. V prvi polovici osemdesetih let so se na trgu pojavili splošno namenski sistemi SCADA, ki so temeljili na tehnologiji PC in delovali v okolju operacijskega sistema DOS. Te sisteme so v mnogih primerih izdelovale programerske hiše, ki niso dajale prednosti nobenemu od proizvajalcev krmilniške opreme. Eno od vodil razvoja je bilo ponuditi čim več funkcionalnosti za čim nižjo ceno. S časom se je razširila tako funkcionalnost, kot tudi tržni delež splošno-namenskih sistemov SCADA, vzporedno pa je potekal tudi neizbežen prehod na naslednike operacijskega sistema DOS [10]. Kasneje je

razvoj zelo hitro napredoval in za izdelavo teh sistemov so se pojavila razvojna orodja različnih proizvajalcev (Siemens, Ge Fanuc, Indusoft, … ). Pojavljati so se začele tudi nove tehnologije in standardi, kot najbolj znanega naj omenim standard OPC (ang. Object linking and embedding for Process Control), ki je v začetku deloval na osnovi COM/DCOM objektov, v zadnjem času pa se počasi uveljavljajo tudi drugi. Na primer OPC UA (ang. Unified Arhitecture), ki temelji na XML spletnih procesih [11]. Za implementacijo lastnih sistemov SCADA v višje nivojskih programskih jezikih kot so C, VB, C# in drugi so na voljo tudi razne knjižnice (Prodave, LibNoDave … ) in ActiveX kontrolniki.

## 3.1 Arhitektura

### 3.1.1 Arhitektura stojne opreme

Sistem SCADA lahko razdelimo na dva osnovna nivoja, in sicer na uporabniški nivo, ki streže odnosu človek-stroj ter podatkovni nivo, ki skrbi za podatkovne aktivnosti procesov. Podatkovni strežniki komunicirajo z napravami prek krmilnih računalnikov, ki so, tako kot je prikazano na sliki 3.1, priključeni na podatkovni strežnik prek področnih vodil [12].




*Slika 3.1: Tipična arhitektura strojne opreme.*

### 3.1.2 Arhitektura programske opreme

Produkti so večopravilni in so zasnovani tako, da v realnem času komunicirajo s podatkovno bazo, ki se nahaja na enem ali več strežnikih. Ti so odgovorni za zbiranje podatkov (kontrola alarmov, izračune, prijavljanje, arhiviranje) in upravljanje s parametri. Za nekatera od naštetih opravil je mogoče uporabiti tudi namenske strežnike [12].

## 3.2 Princip komunikacije

Notranja komunikacija strežnik-uporabnik in strežnik-strežnik poteka na osnovi vzorca objavi/naroči (ang. publish/subscribe) in uporablja protokol TCP/IP. Povedano z drugimi besedami, uporabnikova aplikacija se naroči na določen parameter, ki si ga lasti določen strežnik in le spremembe tega parametra so posredovane aplikaciji [12].

Dostop do podatkov na krmilnih računalnikih se izvaja s periodičnim povpraševanjem po podatkih (ang. polling). Uporabnik lahko čas vzorčenja podatkov iz krmilnega računalnika (ang. aquisition cycle) nastavi poljubno za različne parametre. Ponudniki imajo na voljo komunikacijske gonilnike za najbolj razširjene krmilne računalnike, ki uporabljajo različna področna vodila, kot na primer Profibus, Modbus in Profinet [12].

## 3.3 Funkcionalnosti

Glavne funkcionalnosti, ki jih sistemi SCADA vsebujejo so: uporabniški vmesniki, uporabniku prijazen grafični vmesnik, vmesnik za spremljanje in potrjevanje alarmov, izdelovanje poročil, vmesnik za arhiviranje podatkov in upravljanje z recepturami [12].

- Uporabniški vmesnik: uporabniki so locirani v uporabniške skupine, ki imajo definirane bralno/pisalne pravice za spreminjanje parametrov in pravice za upravljanje določenih funkcionalnosti.

- Grafični vmesnik (HMI): izdelki omogočajo preprost in uporabniku prijazen sistem za ustvarjanje vizualizacije sistema. Sestavljeni so iz več zaslonov na katerih prikazujemo grafične in tekstovne objekte. Podpirajo koncept generičnih grafičnih objektov, ki jim lahko pripnemo procesne spremenljivke. Te grafične objekte vstavljamo v sistem iz standardnih knjižnic z metodo povleci in spusti. Procesne in druge spremenljivke definiramo kot točke ali zaznamke (ang. tags), ki jim lahko nastavljamo različne lastnosti kot so na primer simbol, omejitve, čas vzorčenja.

- Alarmiranje: sistemi omogočajo na pregleden in uporabniku prijazen način spremljanje ter potrjevanje napak in obvestil, povezanih s tehnološkim procesom ali sistemom samim. Najnaprednejši sistemi omogočajo vpogled v zgodovino napak in shranjevanje v podatkovno bazo.

- Izdelovanje poročil: nekateri sistemi imajo možnost izdelovanja poljubnih poročil in neposrednega tiskanja.

- Arhiviranje: modul, ki omogoča shranjevanje želenih podatkov v podatkovno bazo ali kako drugo datoteko.

- Recepture: množico parametrov, ki neposredno vpliva na tehnološki proces, je mogoče zbrati v posebnem obrazcu recepture. Parametre za različne izdelke je mogoče

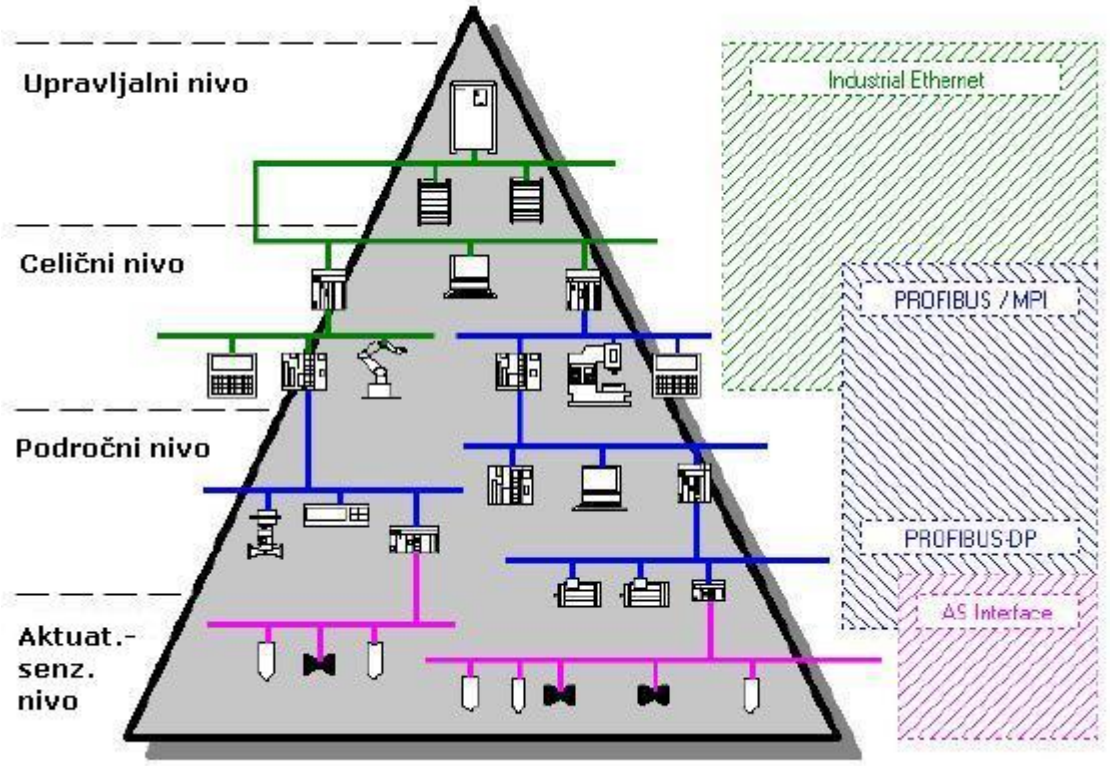
definirati in jih nato shranjevati kot podatkovne zapise določene recepture. Podatkovne zapise lahko urejamo, tiskamo, brišemo in nalagamo v/iz krmilnega računalnika.

# 4. Protokoli in načini povezovanje z višjimi sistemi vodenja

## 4.1 Industrijska komunikacija - protokoli

Protokol je formalen opis pravil za izmenjavo sporočil, ki jih je potrebno spoštovati, da se računalniški sistemi v omrežju lahko med seboj sporazumevajo. Industrijski protokoli določajo električne in fizikalne standarde, vrstni red bitov in bajtov, način vzpostavitve povezave, hitrosti prenašanja podatkov ter načine zaznavanja in odpravljanja napak.

Za vzpostavitev industrijske komunikacije so nam na voljo različni protokoli in področna vodila. Slika 4.1 prikazuje uporabo najprimernejših industrijskih protokolov po različnih plasteh avtomatiziranih sistemov [13].




*Slika 4.1: Uporaba področnih vodil po različnih plasteh avtomatiziranih sistemov.*

Kot prikazuje slika, se v višjih sistemih vodenja, kot so nadzorni sistemi (SCADA), sistemi upravljanja proizvodnje (ang. Manufacturing Execution System, MES) ter višji sistemi

planiranja proizvodnje (ang. Enterprise Resource Planning, ERP), najbolj pogosto uporablja industrijska komunikacija Ethernet. Ta je tudi najbolj zaželena in primerna.

Zaradi implementacije nadzornih sistemov na že obstoječe omrežje in zaradi dragih Ethernet komunikacijskih procesorjev, ni vedno mogoče vgraditi Ethernet omrežja, zato se pri povezovanju uporabljajo tudi druga industrijska področna vodila kot so na primer MPI, Profibus, Profinet.

### 4.1.1 Vodilo MPI

Vodilo MPI (ang. Multi Point Interface) je primernejše za uporabo v manjših omrežjih. Uporablja se lahko le s Simatic S7 izdelki, ki vsebujejo vmesnik MPI. Zasnovan je bil kot vmesnik za programiranje krmilnih računalnikov in s povečevanjem komunikacijskih zahtev hitro doseže svoje omejitve. Na vodilo MPI je lahko priklopljenih do 32 naprav. Najpogosteje uporabljena hitrost prenosa podatkov po vodilu je 187,5 kb/s, doseže pa lahko maksimalno 12 Mb/s. Iz PC računalnika lahko do vodila MPI dostopamo preko ustreznega vmesnika ali preko pretvornika PG/PC (RS232 – RS485). Pri vodilu MPI komunikacija med postajami poteka po principu žetona (ang. token). Pravico do dostopa do vodila se prenaša od postaje do postaje. Ko postaja prejme žeton ima pravico poslati sporočilo. V primeru, da sporočila nima, pošlje žeton naslednji postaji [13].

### 4.1.2 Vodilo PROFIBUS

Vodilo PROFIBUS (ang. Proces Field Bus) se največkrat uporablja na področnem nivoju in nivoju povezovanja med celicami. Vodilo je podprto z odprtim in neodvisnim industrijskim standard, ki je zasnovan v skladnosti z evropskim standardom EN 50170. Ta standard zahteva odprtost protokola in zmožnost povezovanja z napravami drugih proizvajalcev (ang. third-party components). Vgrajuje se ga v manjše in srednje velikih omrežjih, sestavljenih iz množice postaj. Na vodilo PROFIBUS lahko priključimo do 127 naprav.
Vodilo PROFIBUS se ločuje po aktivnih in pasivnih omrežnih postajah. Aktivne postaje za komunikacijo uporabljajo metodo podajanja žetona (ang. token ring), pasivne pa komunicirajo po principu gospodar-suženj. Aktivne postaje si po logičnem vrstnem redu (zaporedni naslovi) podajajo žeton znotraj obroča. Ko aktivna postaja dobi žeton, dobi vlogo gospodarja na omrežju. V časovnem intervalu, ki ga ima na voljo, lahko komunicira z aktivnimi postajami, ali pa zahteva podatke od pasivnih postaj, sužnjev. Sužnji nikoli ne dobijo žetona in nimajo pravice komuniciranja, lahko le odgovarjajo na zahteve gospodarjev.
Standard ločuje tri protokole:

- PROFIBUS DP (ang. Decentralized Periphery) je najbolj pogosto uporabljen protokol. Uporablja se predvsem za decentraliziranje periferije.

- PROFIBUS FMS (ang. Fieldbus Message Specification) je univerzalni komunikacijski protokol za bolj zahtevne komunikacije v višjih plasteh. Nudi mnogo več zahtevnejših funkcij za komunikacijo med inteligentnimi napravami v celičnem

nivoju. Na fizični plasti med protokoloma PROFIBUS FDS in PROFIBUS DP ni razlik, zato lahko delujeta na istem omrežju.

- PROFIBUS PA (ang. Process Automation) je razširjeni protokol DP. Omogoča zanesljiv in hiter prenos podatkov v procesni avtomatizaciji in priklop senzorjev ter aktuatorjev na skupno omrežno vodilo tudi v eksplozijsko nevarnih območjih.

Na fizični plasti lahko po standardu PROFIBUS uporabljamo električno ali optično omrežje.

- Električno omrežje – za prenos podatkov na fizični plasti uporablja RS-485 protokol in poseben oplaščen kabel z enojno parico. V mrežo lahko priklopimo do 127 naprav, pri čemer moramo na najmanj vsakih 32 naprav vgraditi še ojačevalec signala (ang. repeater). Hitrosti prenašanja podatkov po vodilu so lahko od 9,6kb/s do 12Mb/s.

- Optično omrežje – za prenos podatkov uporablja optični kabel. Prednosti uporabe optike so predvsem neobčutljivost na električne motnje in doseganje večjih razdalj, tudi do petnajst kilo metrov med posameznimi postajami. Hitrosti prenosa podatkov so od 9,6kb/s do in so neodvisne od dolžine optičnih vodnikov [13].

### 4.1.3 Industrijski Ethernet

Omrežni protokol Ethernet je vedno bolj razširjen v industrijskem okolju, uporablja se v avtomatizaciji in za nadzor procesov. Uporaba Ethernet protokola je najbolj primerna za nadzorni in celični nivo. Prednosti uporabe so predvsem hiter prenos velikih količin podatkov po dolgih razdaljah med velikim številom postaj, preprosta razširljivost in odprtost ter visoka razpoložljivost in globalna distribucija. Je komunikacijski omrežni protokol, ki ga uvrščamo v skupino omrežji po standardu IEEE 802.3.

Protokol Ethernet je sprva deloval s hitrostjo 10Mb/s in uporabljal povezovalno metodo z zaznavanjem prenosa in odkrivanjem trkov CSMA/CD (ang. Carrier Sense Multiple Access/Collision Detection). Trenutni protokoli podpirajo hitrosti do 100Mb/s (ang. Fast Ethernet), gigabitno (Gigabit Ethernet) in 10 Gb/s povezavo ter uporabljajo povezovalne metode usmerjanja (ang. routing) in preklapljanja (ang. switching).

Omrežje je lahko realizirano s tremi različnimi tehnologijami kot so: električna (parica), optična (optični vodniki) in brezžična tehnologija. Za prenašanje podatkov po omrežju uporablja protokole družine TCP/IP ali UDP/IP.

Z uporabo tehnologije preklapljanja, je širina omrežja postala skoraj neomejena. Protokol Ethernet omogoča tudi možnost vzpostavitve brezžične komunikacije, ki se lahko nevidno vgrajuje v mrežno strukturo. To pomeni, da so informacije na voljo povsod in ob vsakem času ter da je mogoč tudi mobilni dostop prek brezžičnega omrežja LAN, ki je povezan v internet ali intranet. Varnostni moduli zagotavljajo varnost omrežja pred nepooblaščenim dostopom in pred sabotažami ter vohunstvom.

Zaradi posebnih zahtev industrijskega okolja je bilo potrebno protokol Ethernet ustrezno prilagoditi. Za industrijsko rabo je nujna uporaba opreme, prilagojena industrijskemu okolju, kar pomeni, da je robustna, prirejena za ekstremne temperaturne razmere, vibracije, onesnaženja, elektromagnetne motnje in druge zunanje dejavnike [2]. Danes se za prenašanje podatkov med avtomatiziranimi sistemi ali med avtomatiziranimi sistemi in inteligentnimi partnerji (PC računalniki) uporabljajo ustrezno prilagojeni protokoli, ki jih označujemo s skupnim terminom industrijski Ethernet. Eden najbolj znanih med njimi je protokol Profinet.

### 4.1.4 Protokol Profinet

- Komunikacijski protokol Profinet je inovativen in odprt industrijski Ethernet standard (IEC 61918, za Profinet tudi IEC 61784-5-3) za industrijsko avtomatizacijo. Uporablja obstoječe standarde informacijske tehnologije in omogoča komunikacijo točka-točka med področnim in nadzornim nivojem. Standard Profinet temelji na industrijskem ethernetu in uporablja TCP/IP standard za določevanje parametrov, konfiguracijo in diagnostiko. Realno časovna komunikacija za uporabniške in procesne podatke poteka po istem podatkovnem vodilu. Bistveni nadgradnji protokola Profinet sta potrebni zaradi zahtev po komunikaciji v realnem času.

- Možnost nastavljanja prioritete posamičnim vozliščem. Po specifikaciji RT (ang. Real Time) se za komunikacijo uporablja poenostavljen ISO model, ki vključuje samo aplikacijsko, povezovalno in fizično plast, s čimer zagotavlja višje hitrosti pošiljanja podatkov z uporabo standardnih komponent strojne opreme v avtomatizaciji.

- Po specifikacija IRT (ang. Isochronous Real Time) gre za strojno podporo komunikaciji v realnem času. Med drugim omogoča tudi podatkovne prenose z metodo, ki vnaprej nastavi periodične intervale pošiljanja podatkov (ang. isochronous) z zelo kratkimi časi posodobitev za visoko dinamične aplikacije, ki upravljajo s pogoni (ang. motion control).

Profinet omogoča enostavno integracijo distribuiranih področnih naprav, na primer vhodno izhodnih modulov. V okolju Step 7 so te področne naprave dodeljene centralni kontrolni enoti, znani kot vhodno izhodni Profinet krmilnik (ang. IO controler).

Z uporabo ustreznega vmesnika (ang. proxy server) enostavno integriramo obstoječa področna vodila, na primer Profibus, na vodilo Profinet.

Omogoča nam tudi implementacijo modularne rešitve avtomatiziranega sistema (Profinet CBA, ang Component Based Automation). S komponentami na osnovi funkcionalnosti Profinet CBA, lahko razdelimo avtomatizirane sisteme v neodvisne module. Za izvedbo takega sistema nam je na voljo posebno Siemensovo inženirsko orodje v katerem grafično nastavimo povezave med moduli Simatic iMap [2].

## 4.2 Načini povezave z višjimi sistemi vodenja

Krmilne računalnike lahko povežemo z višjimi sistemi vodenja na več načinov. Na tržišču je na voljo bogata ponudba namenskih inženirskih razvojnih orodij, s katerimi lahko ustvarimo poljubne nadzorne sisteme, ne da bi imeli veliko izkušenj s programiranjem z višje-nivojskimi programskimi jeziki. Taka razvojna orodja imajo že vnaprej pripravljene določene gradnike, sklope (vizualizacija, alarmiranje, … ) in različne gonilnike za več vrst krmilnih računalnikov. Lahko pa izdelamo lasten nadzorni sistem z uporabo programerskih razvojnih orodij, kot je na primer Visual Studio (programski jezik C#). V primeru, da se odločimo za razvijanje lastne aplikacije, potrebujemo tudi sistem za komunikacijo in izmenjavo raznih podatkov s krmilnim računalnikom. Za komunikacijo lahko uporabimo različne plačljive ali prosto dostopne knjižnice, na primer LibNoDave, vtičnike za razvojna orodja, standard OPC, ki je industrijsko podprt in standardiziran (organizacija OPC Foundation), ter v primeru vodila Ethernet komunikacijo prek mrežnih vtičnikov (ang. internet socket communication).

### 4.2.1 Namenska razvojna okolja

Pri implementaciji nadzornega sistema je zelo pomembna izbira ustreznega razvojnega okolja. Uporaba takih sistemov je pogosta, na to kaže podatek, da ima skoraj vsako podjetje, ki se ukvarja s krmilno tehniko in izdelavo krmilnih računalnikov, svoje razvojno okolje za izdelavo nadzornih sistemov. Primer takega orodja proizvajalca Siemens je WinCC, ki je profesionalno inženirsko okolje in ponuja veliko možnosti za realizacijo poljubnega nadzornega sistema (SCADA). Ta sistem nudi poleg priklopa na Siemens krmilne računalnike tudi različne gonilnike za priklop na druge krmilne računalnike kot so Allen Bradley, Mitsubishi, Omron in drugi. Pri takih razvojnih okoljih je prednost predvsem v vnaprej pripravljenih posebnih objektih za komunikacijo, sistemu za upravljanje z alarmi, aktivnih gradnikih za izdelavo vizualizacije, gradnikih za enostavno izdelavo receptur in poljubnih poročil.

Za vzpostavitev komunikacije je na voljo uporabniku prijazno okno, ki omogoča nastavitev vrste krmilnika in ustrezne podatke za komunikacijo. To pomeni, da je mogoče določiti vrsto krmilnega računalnika, izbrati vrsto vodila (MPI, PROFIBUS, Ethernet, … ) in določiti komunikacijski naslov ter hitrost prenosa podatkov. Sistem v ozadju skrbi za stabilno komunikacijo, branje, pisanje in sinhronizacijo podatkov. Spremenljivke, s katerimi želimo manipulirati, lahko definiramo kot točke (ang. tags) v uporabniku prijaznem vmesniku. Lahko pa jih le zelo enostavno povežemo s spremenljivkami iz podatkovnih blokov ali iz simbolne tabele krmilnega programa.

Velika prednost takih in podobnih sistemov je v zelo enostavni izdelavi vizualizacije (grafična predstavitev tehnološkega procesa uporabniku). Pri tem nam sistem ponuja veliko izbiro tehnoloških in splošnih aktivnih gradnikov in simbolov, ki jih po metodi povleci in spusti (ang. drag and drop) vstavljamo v zaslone naše nadzorne aplikacije. Objektom lahko na

enostaven način dodelimo spremenljivke, ki nato določajo njihove lastnosti (razni grafični prikazi vrednosti, spreminjanje barve gradnikov…).

Sistem alarmiranja je zelo enostaven, saj od nas zahteva le besedilo napak ali opozoril, deklaracijo spremenljivke za proženje in potrjevanje alarmov ter napak.

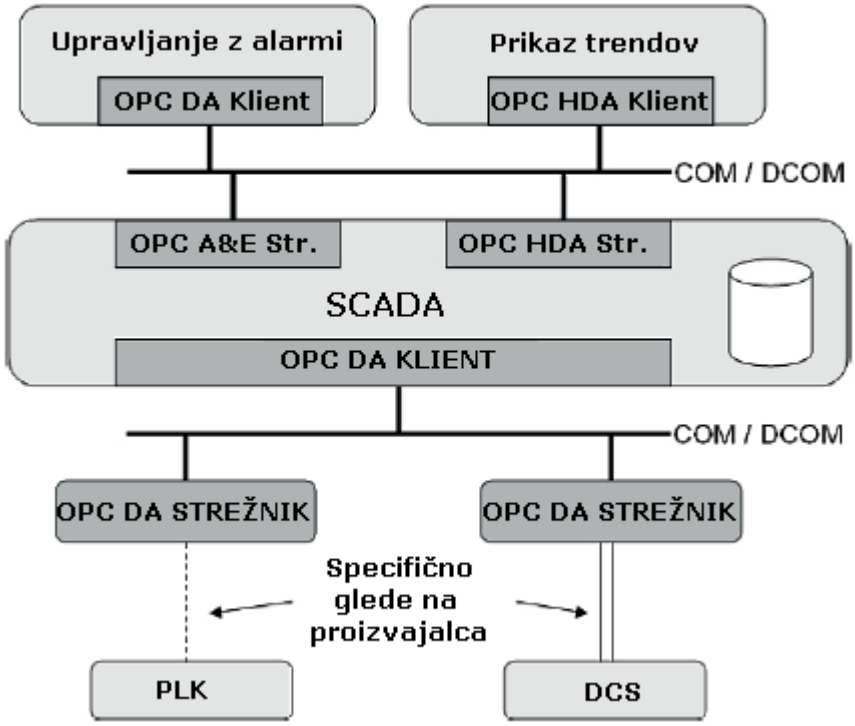
Za implementacijo bolj naprednih funkcij, kot je manipulacija s podatkovno bazo, prikazovanje raznih poizvedb, naprednih grafov in drugih funkcionalnosti, sta nam na voljo dva visoko nivojska jezika za pisanje skript in drugih podprogramov. V sistem sta vgrajena skriptni jezik Visual Basic in C.

Največja slabost namenskih sistemov za implementacijo nadzornih sistemov je velika cena razvojnih orodij in nujno potreben dokup licenc za izvajanje sistema pri uporabniku (razvojne in izvajalne licence).

### 4.2.2 Standard OPC

Standard OPC (ang. Object Linking and Embedding for Process Control) je široko sprejet robustni industrijski standard, ki omogoča izmenjavo podatkov med napravami različnih proizvajalcev in nadzornimi programi, brez lastniških omejitev. Standard OPC predlaga sistem komunikacije strežnik/odjemalec, ki skrbi za pravilno prenašanje podatkov med krmilniki in visokonivojskimi aplikacijami. Strežnik OPC lahko nenehno prenaša podatke na nižjem nivoju s krmilnimi računalniki, postajami HMI in na višjih nivojih s programskimi aplikacijami na računalnikih PC. Tudi v primeru, ko sta proizvajalca programske in strojne opreme različna, standard OPC poskrbi za neprekinjeno realno časovno komunikacijo. Standard OPC predstavlja odprto povezljivost v industrijski avtomatizaciji. Ta je zagotovljena s pomočjo oblikovanja in vzdrževanja nelastniških specifikacij standardov. Prvi standard OPC je nastal s sodelovanjem različnih vodilnih proizvajalcev sistemov za avtomatizacijo v sodelovanju z Microsoftom. Na začetku je bil sistem zastavljen na Microsoftovi tehnologiji OLE COM (ang. Component Object Model) in DCOM (ang. Distributed COM) tehnologiji. Da bi olajšali povezljivost, specifikacije definirajo standard, sestavljen iz objektov, vmesnikov, metod za uporabo in nadzor procesov ter aplikacij za avtomatizacijo proizvodnje. Objekti COM/DCOM zagotavljajo okolje, s katerim lahko ustvarjamo različne programske aplikacije. Ustanovljena je bila tudi organizacija OPC Foundation, ki skrbi za razvoj in razpečavo informacij ter novih specifikacij. Danes je na trgu mnogo ponudnikov strežnikov in odjemalcev OPC [11].

Tipična komunikacija po standardu OPC je implementirana v posebni plasti in deluje po principu odjemalec-strežnik (slika 4.2).




*Slika 4.2: Tipična uporaba OPC sistema odjemalec-strežnik*

Strežnik je prek področnega komunikacijskega vodila priključen na krmilni računalnik in skrbi za dostopnost procesnih podatkov. Potem, ko se odjemalec OPC poveže s strežnikom OPC lahko uporablja podatke, ki mu jih slednji nudi.

Za realizacijo tovrstnega sistema je potrebno namestiti ustrezen strežnik OPC, ki vsebuje gonilnike za komunikacijo s krmilnim računalnikom. Za delovanje strežnika OPC potrebujemo izvajalno licenco. Komunikacijo s strežnikom lahko v nadzorni aplikaciji, običajno gre za odjemalca OPC, s pomočjo dokumentacije o delovanju sistema OPC programiramo sami. Na tržišču so na voljo tudi rešitve, ki ponujajo vtičnike za razvojna okolja. Vtičniki nam nudijo vgrajene funkcije za komunikacijo s strežniki in omogočajo asinhrono pisanje in branje spremenljivk ter druge uporabne rešitve. Za izdelavo aplikacije s pomočjo vtičnikov ne potrebujemo izvajalnih ampak samo razvojno licenco, ki jo kupimo samo enkrat [11].

### 4.2.3 Knjižnica LibNoDave

Knjižnica LibNoDave podpira le priklop na krmilne računalnike proizvajalca Siemens in Vipa. Vgrajene ima gonilnike za povezavo s krmilnimi računalniki prek različnih področnih vodil kot so MPI, Profibus in Ethernet. Paket, ki je prosto dostopen na spletnem mestu http://libnodave.sourceforge.net, nam ponuja nabor knjižnic, ki jih lahko uporabimo v različnih programskih jezikih (C, Pascal, Pealr, C#, VB, Java). Vsebuje tudi primere uporabe

po vrstah jezikov in dokumentacijo, ki je zelo skopa. Knjižnica ni industrijsko podprta in uporablja se jo na lastno odgovornost.

Za vzpostavitev povezave je najprej potrebno definirati vmesnik (libnodave.daveInterface in libnodave.daveConnection) v katerem določimo povezavo in tip vmesnika, na primer TCP/IP, definiramo hitrost povezave, vrata prek katerih se povežemo ter naslov IP krmilnega računalnika. V primeru uporabe drugih vodil je postopek podoben. Koda, s katero vzpostavimo komunikacijo je predstavljena na sliki 4.3.

Knjižnica poleg branja in pisanja podatkov v in iz podatkovnih blokov, vhodnih in izhodnih spremenljivk ter pomnilniških bitov na krmilni računalnik, ponuja še dodatne napredne funkcije kot so:

- mehanizem za branje podatkov iz večjih podatkovnih blokov s samo eno zahtevo,
- postavitev krmilnega računalnika v zagon ali stop (ang. RUN/STOP mode),
- funkcijo s katero lahko shranimo celoten krmilni program iz krmilnika,
- funkcijo, ki primerja krmilni program naložen na krmilniku z krmilnim programom shranjenim na PC računalniku in
- izpis diagnostičnega spomina (ang. diafnostic buffer) v šestnajst bitnem zapisu.

```
public bool Povezava_TCP_IP_povezi(int rack, int slot, string naslov)
{
        libnodave.daveOSserialType fds;
        fds.rfd = libnodave.openSocket(102, naslov);
        fds.wfd = fds.rfd;
        _fds=fds;

        if (_fds.rfd > 0)
        {
              _di = new libnodave.daveInterface(_fds, "IF1", 0,
              libnodave.daveProtoISOTCP,  ibnodave.daveSpeed1500k);
              _di.setTimeout(100000);
              _dc = new libnodave.daveConnection(_di, 0, rack, slot);
              if (0 == _dc.connectPLC())
              {
                    return true;

              }
              else
              {
                    return false;
              }
        }
        else
              return false;
}
```

*Slika 4.3: Vzpostavitev povezave z krmilnim računalnikom prek TCP/IP protokola.*

Branje (slika 4.4) izvedemo z ukazom readBytes ali readBits, pri katerem navedemo izvor komunikacije, definiramo pomnilniški prostor iz katerega želimo brati zaporedno številko bloka, začetni odmik in velikost podatkov v bajtih ali bitih. Po uspešno izvedenem ukazu, so nam na voljo podatki v posebnem spominu (ang. buffer). Podatke moramo iz spomina pobirati z ustreznimi vgrajenimi ukazi, ki nam pretvorijo podatke v primeren podatkovni tip. Ti ukazi znajo pretvarjati predznačena in nepredznačena števila, ki so lahko velikosti bitov, bajtov, besed, dvojnih besed in znajo obračati kratek in debel konec besed (krmilni računalniki imajo v primerjavi z PC računalniki obrnjen kratek z debelim koncem besede) ter pretvarjati v realna števila po standardu IEEE 754, ki ga uporabljajo računalniki PC. Pisanje na krmilni računalnik je izvedeno na podoben način. Poleg omenjenih podatkovnih tipov lahko beremo in pišemo še ostale tipe podatkov, in sicer časovnike, števce in analogni vhodni ter izhodni pomnilniški prostor.

```
res = Global_DC.DC.readBytes(libnodave.daveDB, 31, 0, 8, null);
if (res == 0)
{
      Global_Tocka.prostor = Global_DC.DC.getS16();
      Global_Tocka.izteg = Global_DC.DC.getS16();
      Global_Tocka.visina = Global_DC.DC.getS16();
      Global_Tocka.prijem = Global_DC.DC.getS16();
}
```

*Slika 4.4: Branje koordinat iz podatkovnega bloka na krmilnem računalniku.*

Knjižnica LibNoDave se je pri realizaciji nadzornega sistema izkazala kot primerno orodje za vzpostavitev kontinuirane komunikacije s krmilnim računalnikom. Sistem komunikacije je vseskozi deloval neprekinjeno in stabilno.

### 4.2.4 Komunikacija prek internetnih vtičnikov

Internetni vtičnik ali mrežni vtičnik je končna točka dvosmernega poteka komunikacije med procesi v omrežju (internet). Predstavljajo mehanizem za zagotavljanje prenosa prihajajočega paketa na ustrezno aplikacijo, oziroma aplikacijsko nit. Mehanizem temelji na kombinaciji lokalnega in oddaljenega naslova IP ter številke vrat (ang. port noumber). Operacijski sistem vsako vtičnico posreduje komunikacijskemu aplikacijskemu procesu ali niti. Vtičnikov naslov je sestavljen iz naslova IP in številke vrat.

Tak sistem komunikacije lahko vzpostavimo le na področnem vodilu Ethernet. Za vzpostavitev komunikacije med krmilnim in računalnikom PC mora krmilni računalnik podpirati komunikacijo prek internetnih vtičnikov. Tako vrsto komunikacije podpirajo Siemensovi krmilni računalniki z že vgrajenim Ethetnet vodilom in Siemensovi komunikacijski procesorji Ethernet. Potrebno je nastaviti in sprogramirati pošiljanje in sprejemanje podatkov prek vtičnikov na obeh straneh, tako na računalniku PC, kot tudi na krmilnem računalniku. Na računalniku PC lahko z uporabo razvojnega okolja Visual Studio vključimo vtičnike z že vgrajeno sistemsko knjižnico System.Net.Sockets. Na krmilnem

računalniku pa je najprej potrebno nastaviti strojno konfiguracijo in na ta način omogočiti pošiljanje ter sprejemanje podatkov prek vtičnikov. To storimo s pomočjo vgrajenega Siemensovega orodja NetPro. Za dokončno realizacijo komunikacije je potrebna uporaba in nastavitev vgrajenih funkcijskih blokov.

Z metodo vtičnikov ni mogoče direktno pisati na naslovni prostor krmilnih računalnikov. Ko enota dobi paket podatkov, mora paket ustrezno obdelati in jih razvrstiti v primerne podatkovne tipe in strukture [21].

## 4.3 Za in proti

Pri implementaciji nadzornih sistemov v industriji, je zelo pomembna stabilnost aplikacije in zagotovljena nenehna povezava s krmilnimi računalniki. Uporaba namenskih razvojnih orodij je zelo primerna za implementacijo vizualizacije večjih sistemov, ki so lahko hkrati povezani z večjim številom krmilnih računalnikov. Prednost uporabe teh orodij je predvsem v naprej pripravljenih sklopih in gradnikih, ki jih lahko enostavno uporabimo v aplikaciji. Ti moduli poenostavijo in pospešijo izvedbo nadzornih sistemov. Pri izvedbi dodatnih funkcij obdelave podatkov, analizi arhivskih podatkov in vključitvi posebnih gradnikov, pa se moramo omejiti na zmožnosti uporabe vgrajenih skriptnih programskih jezikov. Za izvedbo nadzornih sistemov, z uporabo namenskih razvojnih orodij, je potreben nakup razvijalnih in izvajalnih licenc, pri čemer cena glede uporabe števila polj v aplikaciji ustrezno astronomsko raste.

V primeru implementacije nadzornega sistema z naprednejšimi ali posebnimi funkcijami obdelave podatkov, analize podatkov ali vgradnje dodatnih modulov (npr. živa slika) pa uporabimo programerska višje nivojska razvojna orodja. Za povezovalni sistem s krmilnimi računalniki lahko uporabimo industrijsko podprt in standardiziran sistem OPC. Prednost uporabe sistema OPC je predvsem to, da ga lahko uporabimo z ustreznimi gonilniki na vseh krmilnih računalnikih. Vgrajene ima tudi napredne funkcije za sinhrono in asinhrono branje ter pošiljanje podatkov na/iz krmilnih računalnikov. Slabosti takih sistemov so predvsem nestabilno delovanje cenenih strežnikov in morebitne težave z Microsoftovimi objekti COM/DCOM. Marsikatera nadgradnja operacijskega sistema (ang. Service Pack) zahteva dodatne nastavitve za pravilno delovanje omenjenih DCOM objektov. Potrebne so izvajalne licence strežnikov ter razvojne licence odjemalcev.

Pojavitev industrijskega Ethernet omrežja in naprav, ki podpirajo Ethernet komunikacijo je močno zmanjšala prevlado uporabe sistema OPC, saj je z uporabo internetnih vtičnikov prinesla prednost v komunikaciji med napravami. Za izvedbo take komunikacije ne potrebujemo licenc, potrebno je le, da krmilni računalnik podpira tako vrsto komunikacije. Pošiljanje podatkov se ne izvaja direktno na naslovne prostore krmilnih računalnikov ampak na vtičnike, iz katerih naprave preberejo in obdelajo prejete podatke. Za to je pa potrebno implementirati programsko rutino na obeh končnih postajah (PLK – PLK ali PLK - PC).
Alternativna rešitev je uporaba brezplačne komunikacijske knjižnice, ki jo lahko uporabimo v različnih programskih okoljih in različnih operacijskih sistemih (Windows, Linux). S

pomočjo knjižnice lahko komuniciramo le s krmilnimi računalniki proizvajalca Siemens in Vipa z uporabo vgrajenih lastnih, ali z uporabo Siemensovih komunikacijskih gonilnikov.

Pri preizkusu vgrajenih gonilnikov je sistem vseskozi deloval stabilno, kar vsekakor lahko naveden kot veliko prednost. Hkrati pa bi poudaril pomanjkanje informacij, navodil, dokumentacije in slabo tehnično podporo.

*Tabela 4.1: Primerjava povezovalnih sistemov.*

<table>
<tr><th colspan="3">Razvoj lastne aplikacije v . Net okolju</th><th>Namenska aplikacija za razvoj SCADA sistemov WinCC</th></tr>
<tr><td>Razvoj z uporabo internetnih vtičnikov</td><td>Razvoj s knjižnico LibNoDave</td><td>Razvoj z uporabo OPC strežnika</td><td>Enoten razvoj</td></tr>
<tr><td colspan="3">- potrebna razvojna licenca za programsko okolje (Visual Studio 2008 - 1000€)</td><td rowspan="2">- Potrebne razvojne in izvajalne licence.<br>Razvojna licenca 3170€, izvajalna licenca za 128 polj (ang. tags) - 2115€<br>- na voljo razni dodatni plačljivi moduli</td></tr>
<tr><td colspan="2">+ izvajalne licence niso potrebne</td><td>- potrebne izvajalne in razvojne licence (OPC foundation, SIEMENS OPC server…) nekje od 300 € dalje za server in od 150 € dalje za odjemalca</td></tr>
<tr><td>-- programiranje potrebno na strani PLK in na strani PC<br>- ne moremo pisati direktno na krmilniški naslovni prostor<br>+/- komunikacija mogoča s kmilnimi računalniki, ki podpirajo tak sistem komunikacije</td><td>- programiranje na strani PC<br>+/- vgrajene nekatere komunikacijske funcije - komunikacija mogoča le z krmilnimi računalniki proizvajalca Siemens in Vipa</td><td>- programiranje na strani PC<br>- potrebna konfiguracija OPC stržnika<br>+ ponuja vgrajene funkcije za komunikacijo<br>+ ponuja več funkcij<br>+ omogoča povezavo tudi z krmilnimi računalniki drugih proizvajalcev</td><td>+ najmanj programiranja, povezava s krmilniki je samoumevno zagotovljena<br>+ omogoča povezavo tudi z krmilnimi računalniki drugih proizvajalcev</td></tr>
<tr><td>+ podpora na strani proizvajalcev krmilnih računalnikov</td><td>- ni več podpore in ni industrijski standard</td><td>+ zelo močna podpora (OPC foundation), primerno za industrijo</td><td>+ odlična SIEMENS podpora, industrijski standard</td></tr>
</table>

| | |
|---|---|
| - objekti za delo z industrijskimi aplikacijami niso pripravljeni (potrebno jih je programirati, ali kupiti ustrezne gradnike) | ++ pripravljeni objekti za delo z industrijskimi aplikacijami (grafi, aktivni grafični objekti, administracija uporabnikov, alarmi, recepture, dobra podpora za večjezičnost...) |
| + neskončne možnosti razširitve (skalabilnost), (omogoča izdelava hevrističnih in statističnih funkcij) | - omejena razširljivost na programiranje skript v VBScript in C |
| + boljše možnosti za prikaz in obdelavo arhivskih podatkov | - slabše možnosti za prikaz in obdelavo arhivskih podatkov |

# 5. Natančno vodenje robotske roke

## 5.1 Namen aplikacije

Namen aplikacije je povezati krmilni računalnik z nadzorno aplikacijo, ki teče na osebnem računalniku, z uporabo prosto dostopnih, cenenih orodij in komponent. Poleg tega je namen vzpostaviti kontinuirano povezavo med krmilnim in osebnim računalnikom, implementirati sistem analize slike in obdelava podatkov.

Izdelal sem dva sistema, ki komunicirata en z drugim in upravljata z robotsko roko. Na višjem nivoju gre za vmesnik človek-stroj (sistem SCADA), ki nam nudi nekatere osnovne funkcionalnosti, kot so upravljanje v dveh režimih delovanja (servisni in avtomatski režim), vmesnik za upravljanje z napakami, nastavljanje podatkov povezanih s pozicioniranjem, arhiviranje podatkov v podatkovni bazi ter spremljanje, obdelava in grafična predstavitev določenih podatkov. Poleg osnovnih funkcij nam vmesnik omogoča spremljanje žive slike prek spletne kamere, ki je neposredno nameščena na robotsko roko. Spletna kamera ima na sistemu poleg funkcije prikazovanja žive slike še funkcijo zajema slik. Zajete slike program analizira in nato sprejema odločitve pri vodenju robotske roke. Na nižjem nivoju krmilni program prek vhodov in izhodov fizično vodi robotsko roko. Povezan je z višjim sistemom vodenja, ki prikazuje, posreduje in obdeluje podatke ter sprejema določene odločitve pri upravljanju robotske roke.

Uporabil sem robotsko roko proizvajalca Fishertechnik, ki je prek vhodno izhodnih enot priključena na krmilni računalnik proizvajalca Siemens S7-315 2PN/DP. Slednji je prek vodila Ethernet priključen na osebni računalnik, na katerem se izvaja aplikacija nadzornega sistema.

Robotsko roko sestavljajo štirje elektromotorji, napajani z enosmernim tokom, ki omogočajo krožno premikanje roke po prostoru (levo, desno), njen izteg naprej ali nazaj, premikanje po višini navzgor ali navzdol in odpiranje ter zapiranje prijemala. Vsak izmed naštetih gibov vsebuje poenostavljen dajalnik impulzov ali enkoder[1] za štetje obratov motorja, prek katerih lahko spremljamo premikanje roke. Slabost teh enkoderjev oziroma roke na splošno je, da je zelo težko izvesti natančno pozicioniranje, kar je posledica zelo velike razdalje med impulzi enkoderja. Pri premiku po prostoru je razdalja med dvema pulzoma pri srednje iztegnjeni roki okoli 5mm, pri daljšem iztegu roke od centra je razdalja še večja. To je tudi eden glavnih razlogov, da sem pozicioniranje nadgradil s pomočjo cenene spletne kamere in uporabe računalniškega vida. Poleg omenjenih signalov iz enkoderjev, je vsaka os sistema opremljena še s končnim stikalom. Njihova glavna funkcija je določevanje referenčnega položaja, hkrati pa jih uporabljamo tudi kot varnostna končna stikala, ki ob morebitnih mehanskih, električnih in programskih napakah preprečujejo, da bi prišlo do strojeloma.

[1] Dajalnik impulzov ali enkoder je elektro-mehanska naprava namenjena merjenju premikov elementov v gibanju. Uporabljata se dve vrsti enkoderjev: linearni in rotacijski. Enkoder je direktno povezan na pogon. Ker z njim merimo pot premika, je neodvisen od morebitnih variacij v hitrosti [14].

## 5.2 Opis krmilnega programa

Krmilni program za vodenje robotske roke sem pisal v različnih programskih jezikih v razvojnem okolju Step 7. Program sem strukturiral tako, da sem razdelil sistem na več funkcij in rutin (slika 2.10). Tako je program bolj pregleden in lažje berljiv. Strukturiranost programa omogoča preglednost in lažje odkrivanje ter odpravljanje morebitnih napak ali hroščev.

Osnova za pisanje krmilnega programa je poleg natančnega in podrobnega opisa delovanja sistema še vhodno izhodna tabela senzorjev in aktuatorjev.

Merilni členi ali senzorji (tudi tipala, odjemniki) so naprave, ki veličine iz okolja pretvorijo v obliko, primerno za nadaljnjo obdelavo z elektronskimi vezji. Najpogostejše veličine iz okolja so temperatura, tlak, pretok, sila, navor, hitrost vrtenja, pot pomika, kot zasuka in druge [15].

Izvršni členi ali aktuatorji so pretvorniki, ki sprejmejo električne signale in jih pretvorijo v fizično akcijo, dejanje. To je mehanizem, prek katerega je mogoče vplivati oziroma učinkovati na okolico. Glede na uporabljeno tehnologijo, so aktuatorji lahko električni, pnevmatični in hidravlični [15].

Vhodno izhodne spremenljivke deklariramo v simbolni tabeli tako, da jim določimo imena in ustrezne naslove, prek katerih krmilni računalnik komunicira z njimi. Dodamo jim lahko tudi uporabne komentarje. Osnovno simbolno tabelo z vhodno izhodnimi spremenljivkami predstavlja spodnja tabela. S črko I (ang. Input) so označene vhodne, s črko Q (ang. output) pa izhodne spremenljivke.

*Tabela 5.1: Simbolna tabela vhodno izhodnih spremenljivk.*

| **Ime** | **Naslov** | **Tip** | **Komentar** |
|---|---|---|---|
| E1 | I 0.0 | BOOL | VRTLJIVA MIZA – REFERENCA |
| E2 | I 0.1 | BOOL | VRTLJIVA MIZA – IMPULZ |
| E3 | I 0.2 | BOOL | TELESKOPSKA ROKA – REFERENCA |
| E4 | I 0.3 | BOOL | TELESKOPSKA ROKA – IMPULZ |
| E5 | I 0.4 | BOOL | DVIG – REFERENCA |
| E6 | I 0.5 | BOOL | DVIG – IMPULZ |
| E7 | I 0.6 | BOOL | PRIJEMALO – REFERENCA |
| E8 | I 0.7 | BOOL | PRIJEMALO – IMPULZ |
| S1 | I 1.3 | BOOL | NAPAKA VARNOSTNA TIPKA |
| S2 | I 1.4 | BOOL | GUMB START |
| S3 | I 1.5 | BOOL | GUMB ROČNO |
| S4 | I 1.6 | BOOL | GUMB STOP |
| S5 | I 1.7 | BOOL | GUMB AVTOMATSKO |
| | | | |
| M1 _SMER | Q 0.0 | BOOL | VRTLJIVA MIZA SMER: 0 - LEVO, 1 – DESNO |
| M1 | Q 0.1 | BOOL | VRTLJIVA MIZA – PREMIKANJE |

| | | | | |
|---|---|---|---|---|
| M2_SMER | Q | 0.2 | BOOL | TELESKOPSKA ROKA: 0 - IZTEG, 1 – KRČENJE |
| M2 | Q | 0.3 | BOOL | TELESKOPSKA ROKA – PREMIKANJE |
| M3_SMER | Q | 0.4 | BOOL | DVIG: 0 - GOR, 1 – DOL |
| M3 | Q | 0.5 | BOOL | DVIG – PREMIKANJE |
| M4_SMER | Q | 0.6 | BOOL | PRIJEMALO: 0 - SPUSTI, 1 – PRIMI |
| M4 | Q | 0.7 | BOOL | PRIJEMALO – PREMIKANJE |

Program, napisan strukturirano po blokih (slika 2.10), poleg glavnega (OB1) vsebuje še ostale funkcijske in podatkovne bloke.

- OB 1 je glavni programski blok v katerem so napisane osnovne funkcije za določevanje napak in pogojev za delovanje. Sprogramirano je tudi delovanje signalnih lučk, bele in rdeče. Kadar je sistem v napaki, gori rdeča luč. Bela pa predstavlja različne faze delovanja, in sicer bela je prižgana, ko deluje avtomatski cikel, utripa ko je cikel prekinjen in hitro utripa, ko se izvaja postavitev v osnovno stanje. V tem organizacijskem bloku so tudi klici ostalih funkcij in rutin sistema.

- FC20 (CASI) je funkcija, v kateri se preračunavajo časovniki, ki so uporabljeni v programu. Funkcija preračunava iz podatkovnega bloka DB20 (DB_TIME) v podatkovni blok DB21 (DB_S5T). Pomeni konverzijo podatkovnega tipa število (ang. integer) v podatkovni tip S5Time. Ta podatkovni tip je ostal v krmilnih računalnikih S7 še iz časov S5 sistemov. Gre za elementarni podatkovni tip sestavljen iz šestnajstih bitov. Prva dva bita sta nepomembna, druga dva določata časovno bazo (00 – milisekunde, 01 – desetinke, 10 – sekunde in 11 – 10s), naslednjih dvanajst bitov pa predstavlja vrednost časovnika v BCD (ang. binary coded decimal) formatu.

- FC30 (ROCNO) je funkcija za upravljanje robotske roke v servisnem režimu in je namenjeno testiranju opreme ter vzpostavitvi določenega stanja po morebitnem prekinjenem avtomatskem ciklu in servisiranju opreme. Ta režim izberemo s pritiskom na gumb S3 (ROČNO) na komandni plošči ali v nadzornem sistemu (SCADA). V tem režimu upravljamo z robotsko roko na način, da prek gumbov S2 (START), S4 (STOP) upravljamo z izbranim segmentom. Segmente izbiramo v nadzornem sistemu, za testiranje delovanja pa si lahko pomagamo s tabelo spremenljivk (ang. variable table), kjer lahko vrednosti spremenljivk neposredno vpisujemo v krmilni računalnik (slika 5.1).

- FC31 (STEVCI_IMP) – V sistemu so uporabljeni enkoderji za določevanje položaja posameznih segmentov robotske roke. Ti enkoderji niso pravi, ampak so poenostavljeni z uporabo mehanskega zobnika in stikala. Delovanje enkoderja sem moral uprizoriti z uporabo programske kode. Uporabil sem lastne števce, pri katerih se vrednost v primeru premikanja v določeno smer povečuje, pri premikanju v nasprotno smer pa zmanjšuje. Ti števci služijo določevanju točk in kontroliranju premikov robotske roke.

- FC 50 (IZHODI) – V tej funkciji se programsko določene in izračunane vrednosti v ročnem ali v avtomatskem režimu preslikajo na fizične izhode. Tukaj so dodani tudi varnostni pogoji.

- FC 60 (SMERI) je funkcija napisana v programskem jeziku SCL, kjer se preračunavajo meje za preklop robotske roke v različne hitrost (normalno in počasno gibanje). Izvaja se tudi preračunavanje koordinat točk in določevanje smeri robotske roke po posameznih gibih (npr. gib motorja M1 – levo/desno). Ker gre za navadne motorje napajane z enosmernim tokom, je počasnejše gibanje realizirano s pulzno širinsko modulacijo (ang. Pulse Wave Modulation, PWM). V mojem primeru je to signal s pravokotnimi impulzi, pri katerih pomeni visoka (ang. high) vrednost vklop motorja in nizka (ang. low) vrednost izklopa motorja. Modulacijo sem uprizoril z dvema časovnikoma, ki predstavljata visoko (deset milisekund) in nizko (tristo milisekund) stanje.

ROCNO_TEST -- Robotska_roka_2\SIMATIC 300(1)\CPU 315-2 PN/DP\S7 Program(1)

| | Address | Symbol | Symbol comment | Display format | Status value | Modify value |
|---|---|---|---|---|---|---|
| 1 | DB30.DBW 2 | "ROCNO_DB".SEGMENT | ŠTEVILKA SEGMENTA | DEC | | 2 |
| 2 | M 1.1 | "ROC" | ROC | BOOL | | |
| 3 | M 1.0 | "AVT" | AVT | BOOL | | |
| 4 | | | | | | |
| 5 | Q 0.1 | "M1" | VRTLJIVA MIZA - PREMIKANJE | BOOL | | |
| 6 | Q 0.1 | "M1" | VRTLJIVA MIZA - PREMIKANJE | BOOL | | |
| 7 | Q 0.2 | "M2_SMER" | TELESKOPSKA ROKA: 0 - IZTEG, 1 - KRČENJE | BOOL | | |
| 8 | Q 0.3 | "M2" | TELESKOPSKA ROKA - PREMIKANJE | BOOL | | |
| 9 | | | | | | |
| 10 | I 1.4 | "S2" | START | BOOL | | |
| 11 | I 1.6 | "S3" | STOP | BOOL | | |
| 12 | | | | | | |
| 13 | | | | | | |
| 14 | M 1.0 | "AVT" | AVT | BOOL | | |
| 15 | M 1.1 | "ROC" | ROC | BOOL | | |
| 16 | M 1.2 | "NAPAKA" | | BOOL | | |

*Slika 5.1: Tabela spremenljivk*

- FC61 (AVTOMATSKO) je funkcija upravljanja s števci, ki štejejo premikanje robotske roke, po vnaprej določenih točkah. Točke poljubno določimo na nadzornem sistemu. V odvisnosti od števcev se vršijo prenosi koordinat izbranih točk na aktualno točko, ki je trenutno v obdelavi. Izvajajo se tudi klici in obdelava funkcij cikla za avtomatsko upravljanje in cikla za postavitev v osnovno stanje. Več o tem sem napisal v nadaljevanju. V tej funkciji je realizirana tudi komunikacija s sistemom SCADA. Nadzorni sistem bere in pošilja tako zahteve, kot tudi podatke povezane z reguliranjem robotske roke. To so predvsem zahteve za začetek postopka analize slike s strani krmilnega računalnika. S strani nadzornega sistema se pošiljajo podatki o izračunanih

premikih in dovoljenja za nadaljevanje premikov ter druga sporočila povezana s pravilnim delovanjem sistema.

- FC62 (CIKEL) in FC 63 (RESET_VERIGE) sta funkciji napisani v programskem jeziku HiGraph. Vsebujeta koračni verigi vodenja robotske roke po avtomatskem ciklu in ciklu za postavitev v osnovno stanje. O tem več v posebnem poglavju.

- FC65 (KONT-PREMIKOV) je funkcija, s pomočjo katere kontroliramo in določamo premike robotske roke po posameznih segmentih delovanja, na način da funkcija prek prebranih (pulzi enkoderja) ter določenih (koordinate točk) spremenljivk vrača zastavice (bite). Zastavice označujejo prehode v preklop premikanja roke v nizko hitrost, ali označujejo položaj robotske roke na točno določeni poziciji.

- FC66 (NAPAKE) S pripadajočim podatkovnim blokom DB66 (DB_NAPAKE) upravljamo z zaznavanjem in potrjevanjem napak povezanih s sistemom. To so predvsem napake pogonov segmentov robotske roke. Napake motorjev preverjamo tako, da preverjamo spremembo omenjenih pulzov enkoderjev pri premikanju posameznih segmentov. V primeru neaktivnosti z neko minimalno časovno zakasnitvijo sprožimo napako, ki preprečuje nadaljnjo nepravilno delovanje robotske roke. Napake potrjujemo na nadzornem sistemu (SCADA), kjer se tudi evidentirajo v podatkovno bazo.

Tabela 5.2: Seznam morebitnih napak na robotski roki.

| Št. | Opis napake |
|---|---|
| 1 | Napaka motorja M1 - premik po prostoru! |
| 2 | Napaka motorja M2 - premik po iztegu! |
| 3 | Napaka motorja M3 - premik po višini |
| 4 | Napaka motorja M4 - premik prijemala! |

- Podatkovna struktura UDT 1 (TOČKA) predstavlja nov podatkovni tip, ki je sestavljen iz koordinat točke. Te koordinate so: prostor, izteg, višina in prijemalo. Tabela predstavlja preglednejše upravljanje s točkami in prenašanje na ustrezne segmente po sistemu.

### 5.2.1 Avtomatski režim obratovanja

Avtomatski režim je namenjen popolnemu avtomatskem upravljanju robotske roke. Premikanje roke se izvaja po vnaprej določenih točkah (koordinatah), ki so shranjene na krmilnem računalniku v podatkovnem bloku DB40. Koordinate točk določamo, spreminjamo, dodajamo in brišemo preko nadzornega sistema SCADA. V sistemu lahko določimo, koliko točk je aktivnih in možnost premikanja roke, ki je lahko nadzorovano in upravljano s pomočjo spletne kamere. Določimo lahko tudi vključenost funkcije ponavljanje koraka. Avtomatski režim izberemo z gumbom S5 na komandni plošči, lahko pa ga izberemo v nadzornem sistemu SCADA. Ko je režim izbran, s pritiskom na gumb START (S2) lahko zaženemo avtomatski cikel. Pri tem bela lučka signalizira delovanje cikla. Ustavimo ga s pritiskom na gumb STOP (S4). Daljši stisk gumba povzroči ničenje (ang. reset) koračne verige avtomatskega cikla in aktivacijo verige za postavitev v osnovno stanje.

Koračno verigo za upravljanje avtomatskega režima sem napisal v programskem jeziku HiGraph po principu končnega avtomata tipa Moore, kjer so naslednja stanja odvisna le od trenutnega stanja in vhoda. Slika 2.9 predstavlja osnovno verigo za upravljanje roke brez uporabe sistema spletne kamere.

Vsako stanje koračne verige pomeni najmanj en korak, premik, oziroma dogodek avtomatskega cikla. Koračna veriga je narejena univerzalno, ne glede na to kakšen cilj se bo izvajal. Lahko gre za prijem ali spust nekega objekta ali pa samo za premik robotske roke v določeno pozicijo. Na primer, če želimo z robotsko roko premakniti objekt iz ene točke v drugo in postaviti roko na poljubno tretjo pozicijo, moramo na nadzornem sistemu izbrati in določiti premik po treh točkah in za vsako točko, ne glede na funkcijo (prijem, spust … ), se bo odvil celoten cikel avtomatskega obratovanja. Ta je razdeljen na več stanj koračne verige.

- Stanje S_0: Inicializacija koračne verige. Izbira ustrezne točke za regulacijo roke. Prenos koordinat na registre za določevanje premikov.

- Stanje S_1: Ker se lahko robotska roka nahaja v katerikoli poziciji v prostoru, preverimo višino in jo dvignemo na ustrezen nivo. V primeru, da je dosežena minimalna višina za varno potovanje po prostoru, veriga nadaljuje pot v stanje S_3.

- Stanje S_2: V kolikor minimalna višina ni dosežena, se predvsem iz varnostnega razloga, roka dvigne na potovalno višino. V primeru skrajno nizke višine pri premiku po iztegu ali prostoru, bi lahko robotska roka zadela v drugi objekt in ga celo poškodovala, lahko pa bi poškodovala sama sebe.

- Stanje S_3: Roko premikamo po prostoru in iztegu. V primeru, da izteg doseže ciljno točko (koordinato) prej kot premik po prostoru, se ta ustavi, preden veriga zapusti to stanje. V nasprotnem primeru se izvaja v vseh naslednjih stanjih do stanja S_8 oziroma dokler ne doseže ciljne točke.

- Stanje S_4: Ko premik po prostoru preide v območje histereze ciljne točke, sistem preklopi v počasnejše gibanje z uporabo pulzne modulacije s čimer omogoči ustavitev roke bolj točno na poziciji.

- Stanje S_5: Ko roka doseže pozicijo po prostoru, se začne izvajati premik po višini. Območje histereze pa jo preklopi v nizko hitrost (stanje S_6).

- Stanji S_9 in S_10 sta vmesni stanji iz stanj S_5 ali S_6. Pri prehodu v eno izmed omenjenih stanj robotska roka predvsem zaradi varnostnih razlogov prekine izvajanje gibanja po višini. Izvajanje gibanja po višini se prekine, ko je trenutna koordinata višine manjša od določene varnostne meje in se roka po iztegu premika nazaj (se krči). V primeru, da bi roka imela nalogo prijeti objekt, ki bi bil že pod njo, hkrati pa bi se vršila oba premika (po višini in iztegu), bi lahko prišlo do nesreče, saj bi roka lahko zadela v objekt pod seboj.

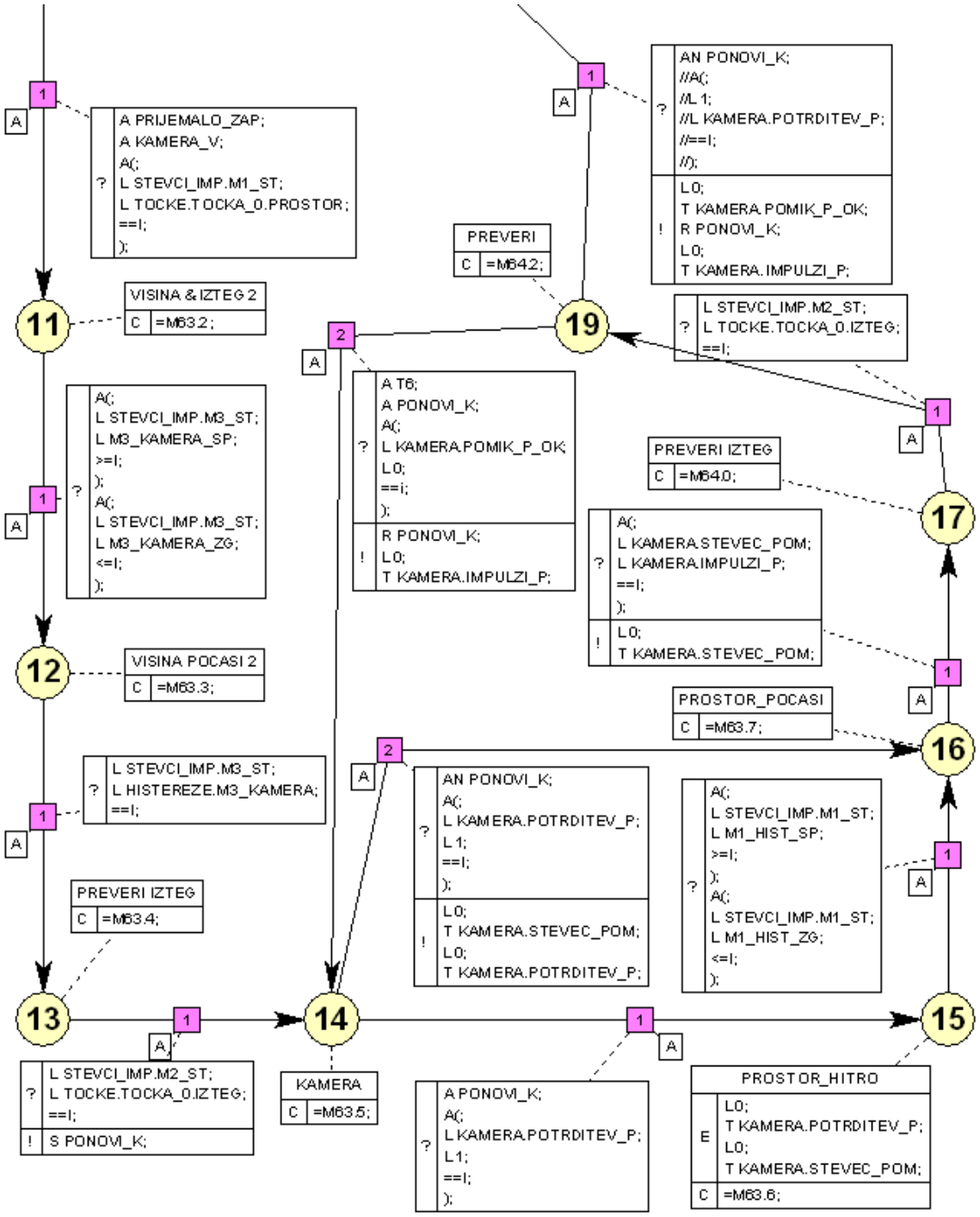
- Stanje S_8: To stanje verige se izvede zatem, ko roka doseže točko po višini. Preveri se premik po iztegu. V primeru, da se je premik po iztegu izvršil že v prejšnjih stanjih, veriga nadaljuje v stanje S_7. V nasprotnem primeru se ta premik izvede do konca.

- Stanje S_7: Končno je robotska roka na poziciji in izvede se še premik prijemala. Izvede se prijem ali spust kosa, odvisno od koordinate točke. Koračna veriga zatem nadaljuje v začetno stanje S_0. V primeru, da je to zadnja točka, se avtomatski cikel ustavi. V nasprotnem primeru se veriga izvede od začetka z novimi podatki o ciljni točki. Ta postopek se ponavlja do izvedbe zadnje točke premika, ali do ustavitve cikla s pritiskom na tipko »STOP« (S4).

*Tabela 5.3: Stanja osnovne koračne verige.*

| Stanja | Opis |
|---|---|
| S_0 | Inicializacija |
| S_1 | Preveri višino - varnost |
| S_2 | Dvigni po višini |
| S_3 | Premik po prostoru |
| S_4 | Premik po prostoru počasi |
| S_5 | Premik po višini |
| S_6 | Premik po višini počasi |
| S_7 | Premik prijemala |
| S_8 | Preveri pomik iztega |
| S_9 | Premik po višini - stop 1 |
| S_10 | Premik po višini - stop 2 |

Po zaključenem testiranju avtomatskega cikla osnovne koračne verige in po opravljenih meritvah točnosti regulacije roke, sem neposredno na sistem namestil spletno kamero, s katero sem želel izboljšati regulacijo (pozicioniranje) roke. Potrebno je bilo razširiti osnovno koračno verigo. Med stanji S_4 in S_5 sem vrinil drugi sklop verige, ki je namenjen upravljanju roke prek nadzornega sistema in spletne kamere (slika 5.2).




*Slika 5.2: Razširjena koračna veriga*

*Tabela 5.4: Stanja razširjene koračne verige.*

| Stanja | Opis |
|---|---|
| S_11 | Premik po višini in iztegu |
| S_12 | Premik po višini počasi |
| S_13 | Preveri pomik iztega |
| S_14 | Kamera |
| S_15 | Premik po prostoru |
| S_16 | Premik po prostoru počasi |
| S_17 | Preveri pomik iztega |
| S_19 | Kontrolno stanje |

Opis stanj razširjene koračne verige:

- Stanje S_11: Pogoj za prehod v stanje je dokončan premik po prostoru. V koordinati točke mora biti določeno, da je to prijem kosa in da je na nadzornem sistemu izbrana opcija pozicioniranje s pomočjo spletne kamere. Stanje določa paralelno premikanje robotske roke po dveh oseh, po iztegu in po višini.

- Stanje S_12: Ko roka doseže območje histereze koordinate posebne točke višine, ki je namenjena analiziranju slike in izračunavanju odstopanj od idealnega pomika, sistem preklopi na nizko hitrost gibanja. Posebna točka višine je na sistemu shranjena točka, ki določa optimalno višino za zajem slike.

- Stanje S_13: Sistem najprej preveri pomik po iztegu. V primeru, da se je že izvedel v prejšnjih stanjih, sistem nadaljuje v naslednje stanje. V nasprotnem primeru se ta izvede do konca in zatem lahko preidemo v stanje S_14.

- Sstanje S_14 je stanje, do katerega se je roka pozicionirala po koordinatah točk, razen po višini. Roka je po višini v posebni točki, ki je namenjena analiziranju slike in preračunavanju odstopanj od idealne tolerance. Ta točka je nekoliko višje kar omogoča zajem slike s perspektive in zato lažjo in kvalitetnejšo analizo slike na nadzornem sistemu. Postavijo se posebni biti, ki jih nadzorni sistem prebere in nato prevzame kontrolo nad robotsko roko. Ko nadzorni sistem konča z obdelavo slike in računanjem korekcijskih podatkov, pošlje krmilnemu računalniku signal za nadaljevanje in podatke za popravljanje pozicije dveh osi (premik po iztegu in prostoru). Koordinata (impulz enkoderja) pomeni v fizičnem merilu premik roke za približno pet milimetrov. Ko je potrebno izvesti premik deset ali več milimetrov, sistem ustrezno določi (izračuna glede na analizo zajete slike) novo koordinato in koračna veriga nadaljuje v stanje S_15. V primeru, da je premik manjši od pet milimetrov, se začne izvajati stanje S_16.

- Stanje S_15: v tem stanju se izvaja paralelni premik po iztegu in po prostoru z novo koordinato, ki jo je nadzorni sistem izračunal na podlagi analize slike. Premik se izvaja z normalno potovalno hitrostjo.

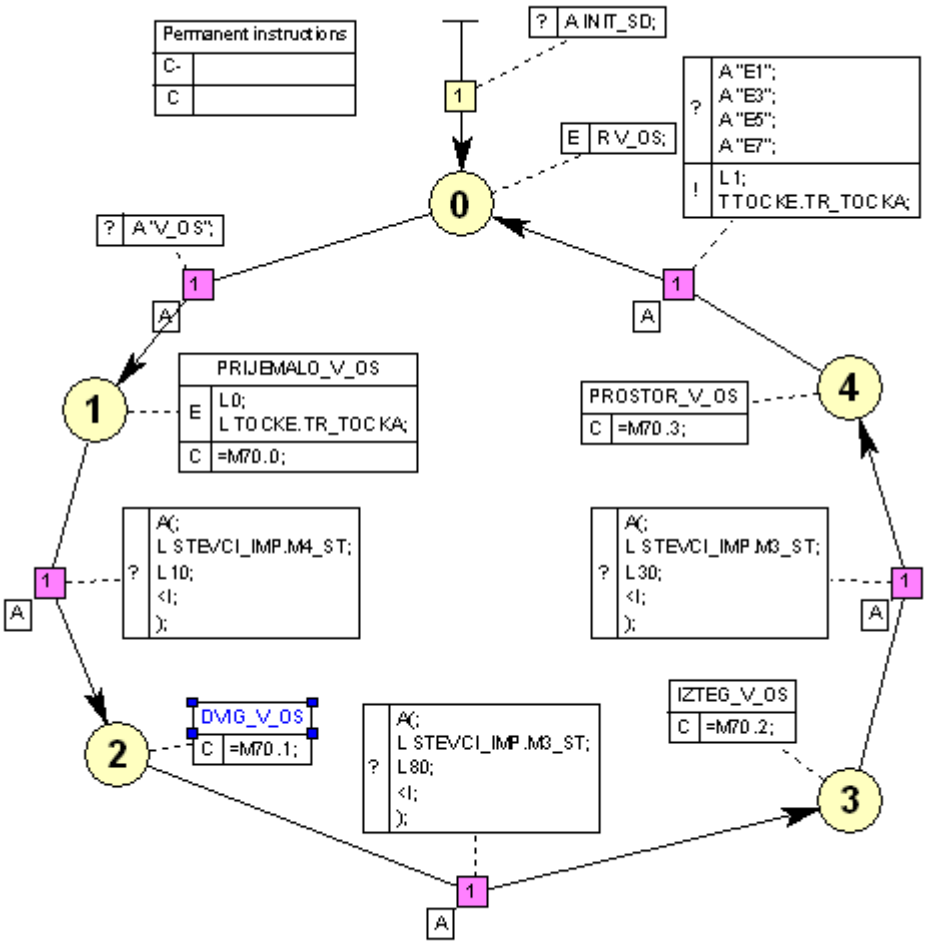
- Stanje S_16: V primeru manjše napake sistem preskoči izvajanje stanja S_15. To stanje se izvede tudi v primeru, da v prejšnjem stanju roka preide v območje histereze novo določene točke. Tukaj sistem premika roko po prostoru z nizko hitrostjo gibanja. Počasno gibanje je realizirano s pulzno modulacijo. Podatek o številu potrebnih pulzov je krmilnemu računalniku posredoval nadzorni sistem v stanju S_14. Krmilni računalnik v funkcijskem bloku avtomatsko izvede tudi štetje teh posebnih impulzov. Ko je premik končan veriga nadaljuje v stanje S_17.

- Stanje S_17: Premik po iztegu, je nekoliko bolj natančen in počasnejši od premika po prostoru, zato nadzorni sistem za korekcijo točke vedno izračuna novo koordinato. Premik po iztegu se vrši istočasno (ni izgube časa) kot premik po prostoru v prejšnjih stanjih.

- Stanje S_19: Sistem zagotavlja natančnost premika in preprečuje morebitne nepredvidljive napake analize slike na način, da še enkrat izvede postopek zajema, analizo slike ter izračun korekcijskih podatkov. V tem stanju se zopet postavijo ustrezni biti za komunikacijo z nadzornim sistemom, izvede se analiza in izračun pozicije. V primeru, da so podatki v mejah tolerance, krmilni računalnik dobi pogoj za nadaljevanje verige v stanje S_5. V nasprotnem primeru sistem ponovi korake od vključno stanja S_14 naprej.

### 5.2.2 Koračna veriga postavitev v osnovno stanje

Koračna veriga prikazana na sliki 5.4 je namenjena postavitvi robotske roke v osnovno ali referenčno stanje. Postopek postavitve sprožimo tako, da za dve sekundi držimo gumb STOP (S4). Postavitev v osnovno stanje oziroma pozicijo, ki jo sistem izvede prek štirih stanj, signalizira hitro utripanje bele lučke na komandni enoti. Stanje S_1 vodi postopek odpiranja prijemala, stanje S_2 sproži dvigovanje robotske roke. Ko dosežemo varnostno višino, sprožimo premik po iztegu (stanje S_3), in ko roka doseže drugo varnostno višino sprožimo še premik po prostoru (stanje S_4). Končna stikala premikov signalizirajo konec verige za postavitev v osnovno stanje (stanje S_0). Izvajanje te koračne verige zahteva ničenje (ang. reset) verige upravljanja avtomatskega cikla. Kazalec trenutne točke koordinat nastavimo na privzeto, prvo točko.




*Slika 5.3: Koračna veriga postavitve v osnovno stanje.*

*Tabela 5.5: Stanja verige za postavitev v osnovno stanje:*

| Stanja | Opis |
|---|---|
| S_1 | Odpiranje prijemala |
| S_2 | Dvigovanje roke |
| S_3 | Premik iztega |
| S_4 | Premik po prostoru |

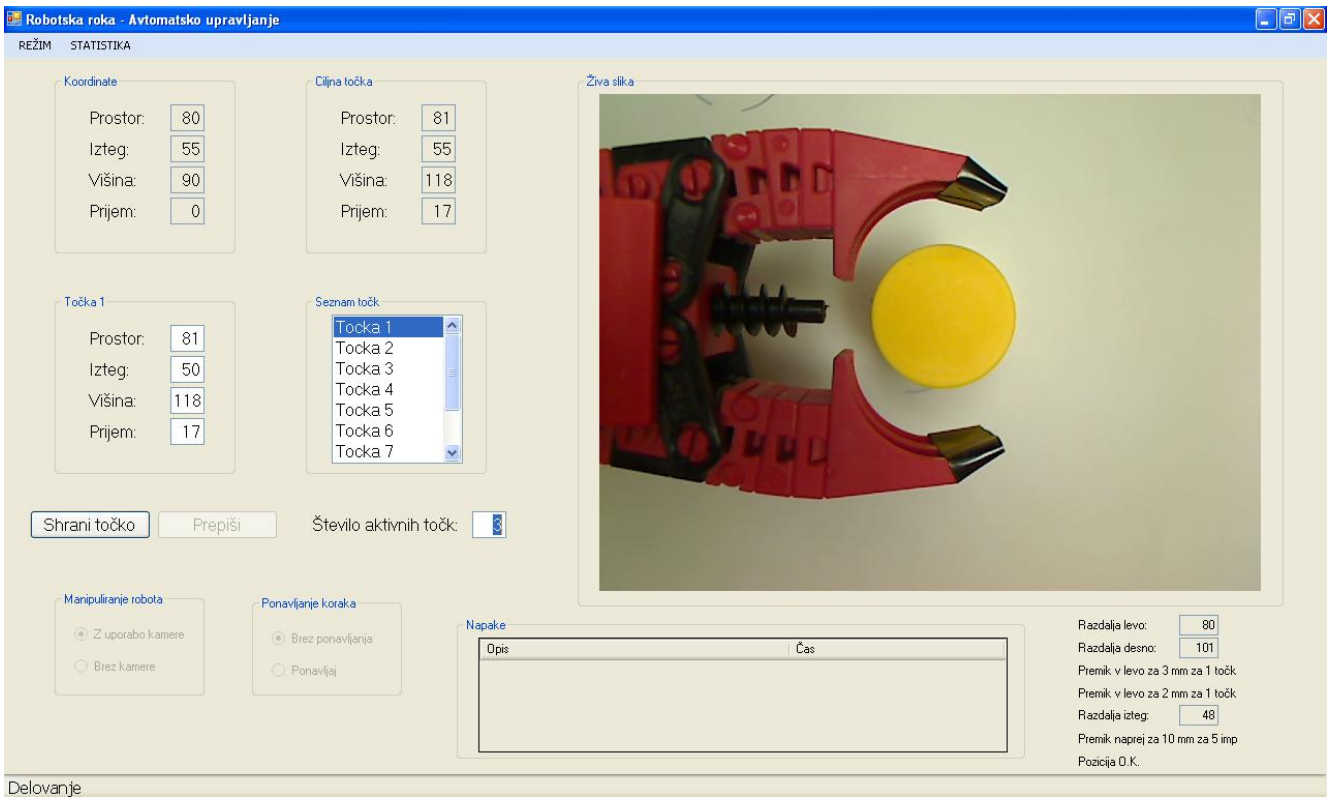
## 5.3 Nadzorni sistem

Nadzorni sistem sem razvijal v programskem jeziku C# in razvojnem okolju Microsoft Visual Studio 2008. Visual Studio je integrirano razvojno okolje (ang. Integrated Development Enviroment) za razvoj aplikacij, ki omogoča programskim razvijalcem lažje delo [16]. Programski jezik C# je moderen visoko nivojski objektno usmerjen programski jezik, ki je bil zgrajen na temeljih programskih jezikov Java, C, C++ in se uporablja za razvijanje aplikacij v ogrodju .NET [17].

Za komunikacijo s krmilnim računalnikom sem uporabil prosto dostopno knjižnico LibNoDave, ki jo je mogoče vključiti v .NET okolje. Nadzorni sistem sem poskušal narediti po videzu in funkcijah čimbolj podoben sistemom, ki se jih razvije z namenskimi razvojnimi okolji za nadzorne sisteme. Moj sistem vsebuje osnovno vizualizacijo, upravljanje v dveh režimih delovanja (ročno in avtomatsko), vmesnik za upravljanje z alarmi in obvestili, nastavljanje podatkov v povezavi s pozicioniranjem robotske roke, hranjenje podatkov v podatkovni bazi in obdelava nekaterih podatkov ter grafični prikaz. Na sistem je priključena tudi spletna kamera, ki uporabniku aplikacije nudi pregled dogajanja robotske roke in služi za analiziranje ter obdelovanje podatkov v povezavi s finim pozicioniranje robotske roke.

### 5.3.1 Opis aplikacije

Aplikacija je sestavljena iz treh oken. Glavno okno predstavlja neko osnovno vizualizacijo in avtomatsko upravljanje robotske roke. Poleg glavnega okna sta še okno za upravljanje z ročnim režimom in okno za prikaz arhivskih podatkov, izračun deležev časa, ko je robotska roka v določenem režimu delovanja (statusu) in prikaz teh statusov v obliki tortnega diagrama.




*Slika 5.4: Glavno okno nadzornega sistema*

Glavno okno nadzornega sistema je namenjeno upravljanju robotske roke, ki jo lahko vodimo v avtomatskem in v ročnem režimu. Okno je sestavljeno iz različnih gradnikov. Na vrhu je menijska vrstica, v kateri lahko izberemo ustrezen režim delovanja in aktiviramo novo okno za pregled statističnih podatkov. Okno vsebuje več okvirčkov za upravljanje s koordinatami, točkami, prikazom žive slike in sistemom upravljanja.

Zgornja dva predstavljata po štiri koordinate trenutne točke, v kateri se nahaja robotska roka, in ciljno točko, v katero je robotska roka namenjena. Ko teče avtomatski cikel, se v primeru izbire potovanja prek več točk, ciljna točka ustrezno spreminja.

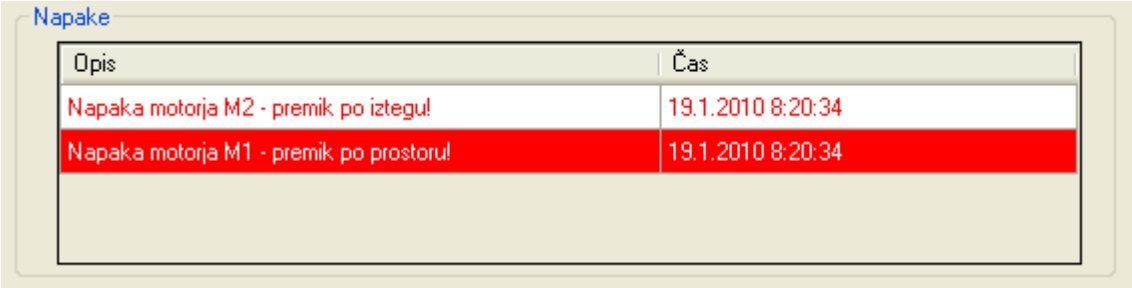
Srednja dva okvirja predstavljata zbirko točk in eno točko izbrano izmed te množice točk, ki jo lahko v mirovanju urejamo. Urejamo lahko poljubno točko iz desnega okvirja na način, da se s klikom izbrana točka prepiše v lev okvir. Urejamo jo lahko tako, da vpisujemo različne koordinate, lahko pa jih s klikom na gumb Prepiši vpišemo v trenutno točko urejanja in kasneje shranimo s klikom na gumb Shrani točko.

Spodnja dva okvirja predstavljata opcije pri upravljanju roke. Skrajno lev okvir predstavlja izbiro opcije upravljanja roke s pomočjo kamere in analize slike, ali premikanje le po nastavljenih koordinatah. V okvirju poleg njega lahko izberemo opcijo ponavljanja koraka. Po vključitvi avtomatskega cikla, sistem neprestano ponavlja izvajanje premikov po točkah, ki jih vnaprej definiramo. V enem ciklu izvajanja izvede toliko premikov po točkah, kolikor jih je definiranih v okvirju Število aktivnih točk. V spodnjem sredinskem delu zaslona se nahaja

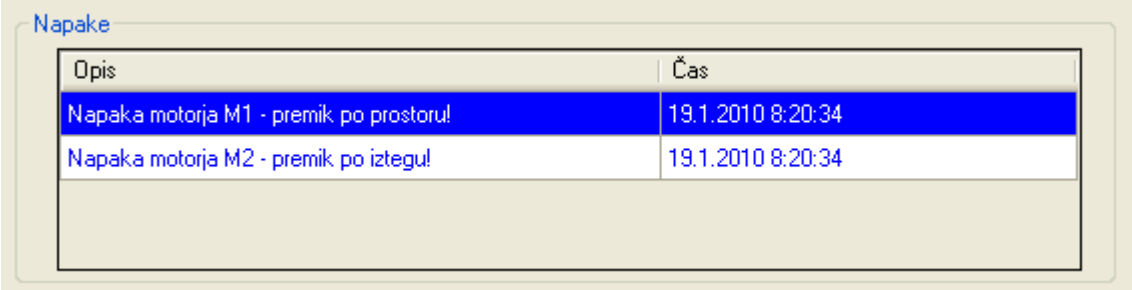
okvir za upravljanje z napakami, ki vsebuje opis in čas porajanja napake. V skrajno desnem robu se nam v primeru, da izberemo upravljanje roke s pomočjo kamere, v trenutku analize slike izpišejo podatki meritev in izračunov. V skrajno spodnjem delu aplikacij je še statusna vrstica, v kateri se izpisujejo razni statusi sistema (mirovanje, delovanje, napaka, … ).

Osveževanje določenih podatkov v aplikaciji se izvaja periodično in je realizirano s časovnikom, ki vsako sekundo izvede postopek osvežitve podatkov iz krmilnega računalnika. To so podatki trenutne točke, ciljne točke, status robotske roke, podatki o aktiviranem režimu in podatki, ki prikazujejo prisotnost napake na sistemu. Slika 4.4 prikazuje proces branja trenutnih koordinat pozicije robotske roke iz krmilnega računalnika.

Sistem alarmiranja je implementiran na krmilnem računalniku in ob pojavu napake se izvajanje avtomatskega ali ročnega cikla takoj ustavi. Nadzorni sistem v procesu osveževanja prebere, da je na sistemu napaka in s pomočjo prebranih podatkov lahko določi, katere vrste napaka se je pojavila. Sporočilo aktivne ali aktivnih napak, s časom pojava napake, se vpiše v podatkovno bazo in v okvir za prikazovanje napak. Napake se obarvajo v rdeče in preden ponovno zaženemo sistem bodisi z avtomatskim bodisi z ročnim režimom, moramo preveriti in odpraviti vzrok napake. Aktivirane napake je potrebno potrditi. Napako potrdimo z dvojnim klikom na vrstico v oknu za prikaz napak. Ko je napaka potrjena, je krmilnemu računalniku poslan potrditveni bit in sporočilo potrjene napake se obarva v modro barvo in se premakne na konec seznama aktiviranih napak.




*Slika 5.5: Aktivni napaki.*




*Slika 5.6: Potrjeni napaki.*

```
Res = Global_DC.DC.readBytes(libnodave.daveDB, 50, 0, 2, null);
if (res == 0)
{
      int preberiNapake = Global_DC.DC.getS16();
      bool M1_N = dolociBit(preberi_NAPAKE, 8);
      bool M2_N = dolociBit(preberi_NAPAKE, 9);
      bool M3_N = dolociBit(preberi_NAPAKE, 10);
      bool M4_N = dolociBit(preberi_NAPAKE, 11);
      ...
}
```

*Slika 5.7: Branje in določevanje podatkov o aktivnih napakah na sistemu.*

Okvir Živa slika služi za prikaz slike iz spletne kamere. To je slikovno okno (ang. picture box), v katerem se prikazuje video slika iz spletne kamere. Zajem slike se izvaja v novi niti, procesu s pomočjo knjižnic Avicap32 in Users32. Knjižnic DLL (ang. Dynamic Link Library), ki niso napisane direktno za .NET okolje ne moremo neposredno vključiti v aplikacijo, ampak jih lahko vključimo posredno, z uporabo procesa P/Invoke [18]. Ta proces je sestavljen iz treh primarnih korakov: deklaracije, vgradnja funkcije iz knjižnice DLL in upravljanja z napakami. V deklaraciji funkcije moramo navesti ime knjižnice DLL in ime želene funkcije. Slika 5.8 predstavlja deklaracijo funkcije za pregled priklopljenih spletnih video naprav na sistem.

Knjižnica Avicap32 vsebuje funkcije za zajem posnetkov AVI (ang. Audio Video Interleaved) in videa iz spletnih kamer ter drugih podobnih video naprav preko programskega vmesnika Windows API (ang. Application Programming Interface). Windows API je Microsoftovo jedro aplikacijskih programskih vmesnikov in je vgrajeno v Microsoftove operacijske sisteme Windows. Je vmesnik, ustvarjen s programom, ki omogoča interakcijo z drugo programsko opremo [19]. Prek procesa P/Invoke in knjižnice Avicap32 sem vgradil in uporabil funkcijo za pregled in iskanje priklopljenih naprav na računalnik PC ter funkcijo za prikaz žive slike iz spletne kamere.

```
[DllImport("avicap32.dll")]
protected static extern bool capGetDriverDescriptionA(short wDriverIndex,
      [MarshalAs(UnmanagedType.VBByRefStr)]ref String lpszName,
      int cbName, [MarshalAs(UnmanagedType.VBByRefStr)] ref String lpszVer,
int cbVer);
```

*Slika 5.8: Deklaracija knjižnice in funkcije za pregled priklopljenih spletnih video naprav na sistem.*

Knjižnica User32 je del zbirke knjižnic Windows okolja in se jo uporablja za ustvarjanje ter manipuliranje s standardnimi objekti uporabniškega vmesnika Windows. Ti elementi so

namizje, okna in meniji. V moji aplikaciji sem deklariral in uporabil tri funkcije: funkcijo za pošiljanje sporočil drugim oknom (SendMessage), funkcijo za spreminjanje pozicije in velikosti podrejenih oken (SetWindowPos) ter funkcijo za zaprtje podrejenega okna (DestroyWindow).

Postopek vključitve žive slike v aplikacijo (del kode prikazuje slika 5.9):

- z uporabo funkcije `capGetDriverDescriptionA` poiščemo ustrezno spletno napravo, priključeno na PC računalnik,
- nato naredimo podrejeno okno (ang. child window), ki upravlja z živo sliko iz spletne kamere in jo vgradi v slikovni okvir v glavnem oknu aplikacije,
- priključimo poiskano spletno napravo,
- podrejenemu oknu pošljemo sporočila za nastavitev skale prikazovalne slike, čas med dvema slikama in ukaz za začetek zajemanja slik iz spletne kamere in
- na koncu postopka zajeto sliko še prilagodimo oknu za prikaz žive sike v glavni aplikaciji.

Razred za zajem slike iz spletne kamere sem napisal s pomočjo objavljenega članka na svetovnem spletu [20].

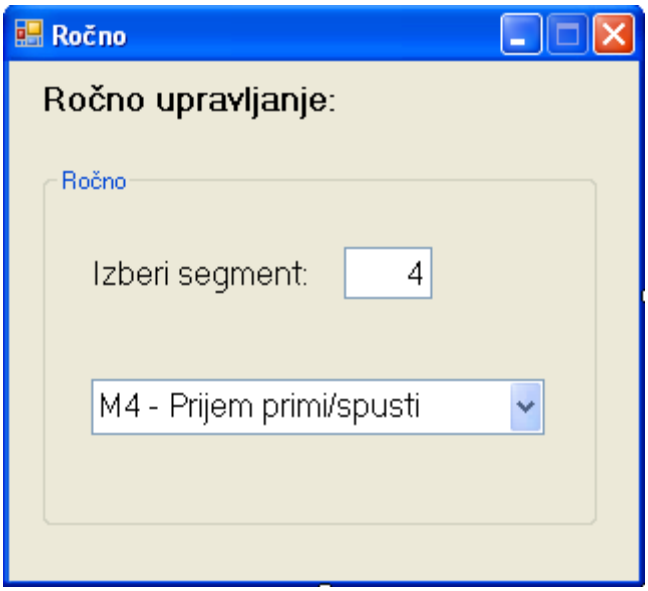
```
hHwnd = capCreateCaptureWindowA(ref DeviceIndex, WS_VISIBLE | WS_CHILD, 0,
0, 640, 480, oHandle.ToInt32(), 0);

// priključimo napravo
if (SendMessage(hHwnd, WM_CAP_DRIVER_CONNECT, DeviceID, 0) != 0)
{
      // nastavimo prikazovalno skalo slike
     SendMessage(hHwnd, WM_CAP_SET_SCALE, -1, 0);
     // nastavimo čas med dvema slikama
     SendMessage(hHwnd, WM_CAP_SET_PREVIEWRATE, 66, 0);
     // predvajanje slike iz spletne kamere
     SendMessage(hHwnd, WM_CAP_SET_PREVIEW, -1, 0);
     // razširimo okno, da se prilagodi slikovnemu okvirju
     SetWindowPos(hHwnd, HWND_BOTTOM, 0, 0, 640, 480, SWP_NOMOVE |
SWP_NOZORDER);
}
```

*Slika 5.9: Zajemanje žive slike iz spletne kamere.*

### 5.3.1 Ročno upravljanje

Ročno upravljanje robotske roke je namenjeno upravljanju sistema prek vnosa ročnih funkcij (segmentov). Omogoča nam vnos točk (koordinat) v sistem prek ročnega upravljanja robotske roke. Točke nato lahko uporabimo za upravljanje v avtomatskem režimu.

Za upravljanje v ročnem režimu v glavnem oknu aplikacije v menijski vrstici izberemo režim Ročno. Odpre se novo pomožno okno za upravljanje in izbor segmentov ročnega režima (slika 5.10).




*Slika 5.10: Pomožno okno za upravljanje z ročnim režimom.*

Z ročnim upravljanjem lahko testiramo pravilnost delovanja segmentov robotske roke, premaknemo roko v poljubno točko, ali premikamo roko na želene pozicije in jih shranimo v sistem kot točke, ki jih kasneje lahko uporabimo za upravljanje v avtomatskem delovanju.
Točko definiramo v več korakih.

- V oknu ročno izberemo ustrezen segment za upravljanje robotske roke po prostoru. Izberemo ga lahko tako, da kliknemo na puščico v kombiniranem oknu in izberemo ustrezen opis segmenta, ali pa enostavno vpišemo številko segmenta v tekstovno okno in potrdimo s tipko Enter. Na krmilni računalnik se vpiše izbran segment.
- Z izbranim segmentom upravljamo z gumboma START/STOP (S2/S4), ki sta na komandni plošči.
- Prvi in drugi korak ponavljamo za vse segmente, dokler ne pozicioniramo robotske roke na želeno točko.
- V glavnem oknu kliknemo na gumb Prepiši in vrednosti trenutnih koordinat se prepišejo v trenutno točko urejanja.
- Vrednosti lahko nato še z ročnim vnosom popravimo in popravke shranimo s klikom na gumb Shrani točko. Koordinate točke se shranijo na krmilni računalnik.

### 5.3.1 Avtomatsko upravljanje

Avtomatski režim je namenjen popolnemu avtomatskemu upravljanju robotske roke z možnostjo izbire opcij ponavljanja koraka izbranih točk in upravljanje roke s pomočjo analize slike, ki jo zajamemo s spletno kamero. Avtomatsko upravljanje je potrebno aktivirati bodisi v menijski vrstici pod zavihkom režim bodisi s pritiskom na gumb Avtomatsko (S5) na komandni plošči. Status izbranega režima nam prikazuje napis na vrhu v naslovni vrstici aplikacije (brez izbora, avtomatsko, ročno upravljanje).

Potrebno je izbrati število aktivnih točk, ki smo jih lahko definirali s pomočjo ročnega režima ali jih ročno vnesti v sistem. Avtomatski cikel sprožimo na komandni plošči s pritiskom na gumb Start (S2).

Z vključeno opcijo ponavljanja koraka, krmilni računalnik neprestano izvaja premike po izbranih točkah. To možnost uporabimo, ko gre za sistem oziroma nalogo robotske roke, ki iz približne točke, prenaša objekte v drugo točko. Na primer, prenašanje objektov, ki se nahajajo v približno vedno enaki točki (sistem tekočega traka, drče … ) na neko drugo lokacijo. Pri tem pa lahko izvedemo še kontrolo prisotnosti nekega vgradnega elementa na objektu in ga s pomočjo analize slike ustrezno razvrstimo v različne zabojnike (dober/slab kos).

Izbira opcije upravljanja robota s pomočjo spletne kamere pripomore k zagotavljanju točnosti pozicioniranja roke pri zajemanju objekta na določeni točki. Krmilni program pripelje robotsko roko na nastavljeno točko po vseh koordinatah, razen po višini. Po višini se pozicionira nekoliko višje, da omogoči nadzornemu sistemu boljšo perspektivo za zajem kvalitetnejše slike. Po zaključenem pozicioniranju na omenjeno točko, krmilni sistem postavi ustrezen komunikacijski bit. Nadzorni sistem vsakih sto mili sekund ta bit preverja in ob postavitvi bita izvede ustrezen postopek zajema in analize slike. Zatem izvede še preračun ustreznega premika v obliki nove koordinate ali pa v obliki impulzov (odvisno od obsega napake) in v spodnji desni rob aplikacije izpiše razdalje meritev v pikah in milimetrih ter popravek v obliki popravka koordinate ali v številu pulzov.

### 5.3.2 Pregled in analiza arhivskih podatkov

V industriji se pojavljajo vedno večje zahteve po shranjevanju in obdelavi podatkov, z namenom povečati optimizacijo proizvodnih procesov in aktivnosti. Jedro sistemov MES (ang. Manufacturing Execution System) je shranjevanje podatkov v podatkovni bazi in kasnejša obdelava v uporabniku prijaznem grafičnem vmesniku.

Prikaz in obdelava arhivskih podatkov se nahaja v novem oknu, do katerega lahko dostopamo, s klikom na Arhivski podatki v meniju glavne aplikacije. Okno nam omogoča tabelarični pregled porajanja napak na sistemu in izpis statusov delovanja robotske roke. Prek izračuna časovnih intervalov sta omogočena tudi tekstovni in grafični (v obliki tortnega diagrama) prikaz v odvisnosti od izbranega časovnega obdobja (slika 5.11).

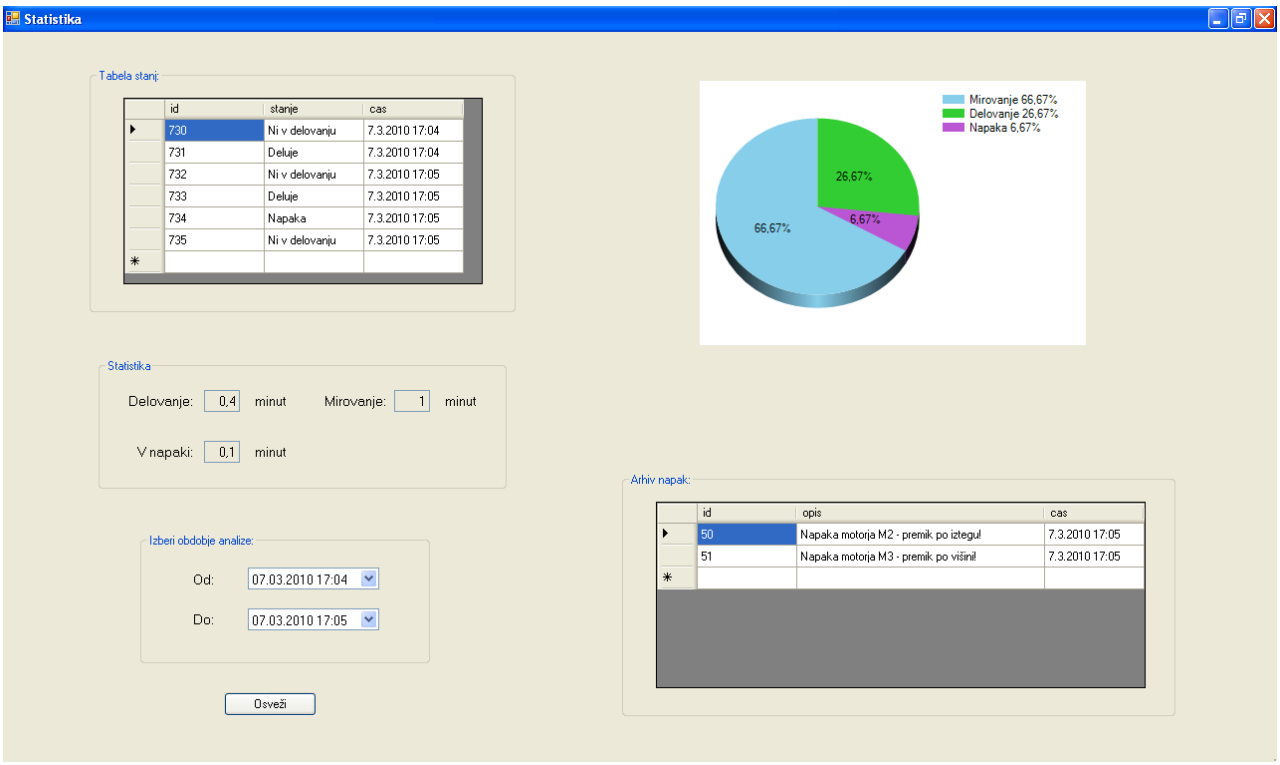
Podatki so arhivirani v podatkovni bazi Microsoft Access, na katero se priključimo z uporabo gonilnikov OLE DB (ang. Object Linking and Embedding, Database). To je Microsftov vmesnik API, namenjen za dostopanje do podatkov iz različnih virov, ki delujejo na enoten način.

Vsako napako, ki jo sistem prebere iz krmilnega računalnika, poleg izpisa v okvir za prikaz napak v glavnem oknu, vpiše še v podatkovno bazo. V podatkovno bazo se vpiše tudi čas nastanka napake.

Sistem vpisovanja statusov robotske roke in izračun časovnih intervalov po statusih:

- Ob vklopu aplikacije se v bazo zapiše trenutni status in čas. Čas se shrani tudi v javno spremenljivko `casZac`, ki jo kasneje uporabimo pri začetni poizvedbi iz podatkovne baze.
- Ob vsaki spremembi statusa robotske roke se omenjeni status in čas spremembe statusa vpišeta v podatkovno bazo.
- Ko odpremo okno Statistika, se avtomatsko izvede poizvedba iz podatkovne baze. Sistem prebere podatke, ki so se pojavili v časovnem intervalu, ki ga vnesemo v okvirček Izberi obdobje analize.
- Podatke statusov in napak, ki so se zgodili v omenjenem časovnem obdobju, zapišemo v okvir za prikaz podatkovnih tabel.
- Zatem se izvede izračun časovnih intervalov statusov (delovanje, mirovanje in napaka).
- Na koncu se v okvir Statistika izpiše izračunan čas statusov v minutah in nariše graf v obliki tortnega diagrama, ki prikazuje delež časa, ko je naprava v določenem statusu (slika 5.11).

V primeru, da nas zanimajo podatki za poljubno določeno časovno obdobje, to obdobje vpišemo v okvir Izberi obdobje analize in pritisnemo na gumb Osveži. Ponovno se izvede zgoraj opisan postopek.




*Slika 5.11: Prikaz in obdelava arhivskih podatkov.*

```
OleDbCommand cmdGetRecords = new OleDbCommand();
cmdGetRecords.Parameters.Clear();
try
{
      cmdGetRecords.CommandText = query;
      cmdGetRecords.CommandType = CommandType.Text;
      cmdGetRecords.Parameters.AddWithValue("@zac", casZac);
      cmdGetRecords.Parameters.AddWithValue("@kon", casKon);
      cmdGetRecords.Connection = Global_baza.Baza;
      OleDbDataAdapter daGet = new OleDbDataAdapter();
      daGet.SelectCommand = cmdGetRecords;
      DataSet dtGet = new DataSet();
      cmdGetRecords.CommandTimeout = 240;
      dtGet.Locale = System.Globalization.CultureInfo.InvariantCulture;
      daGet.Fill(dtGet);
      _dt = dtGet.Tables[0];
      dgvStatusi.DataSource = _dt;
      analiziraj();
      narisiGraf();
}
```

*Slika 5.12: Poizvedba statusov robotske roke za izbran časovni interval.*

### 5.3.3 Implementacija sistema za zajemanje in analiziranje slike

Za zajemanje video posnetkov in slik sem uporabil spletno kamero proizvajalca Logitech model Webcam Pro 9000, z ločljivostjo do dveh milijonov točk. Kamero sem namestil neposredno na robotsko roko. Za zajem in obdelavo slike sem ustvaril novo okensko aplikacijo. Na začetku sem za zajemanje slike uporabil Microsoftov model WIA (ang. Windows Image Acquisition). Model WIA je aplikacijski programski vmesnik (API) in je vgrajen v operacijske sisteme Windows. Knjižnica grafični programski opremi omogoča komuniciranje s slikovno strojno opremo (digitalne kamere, skenerji in druga digitalna video oprema) [23].

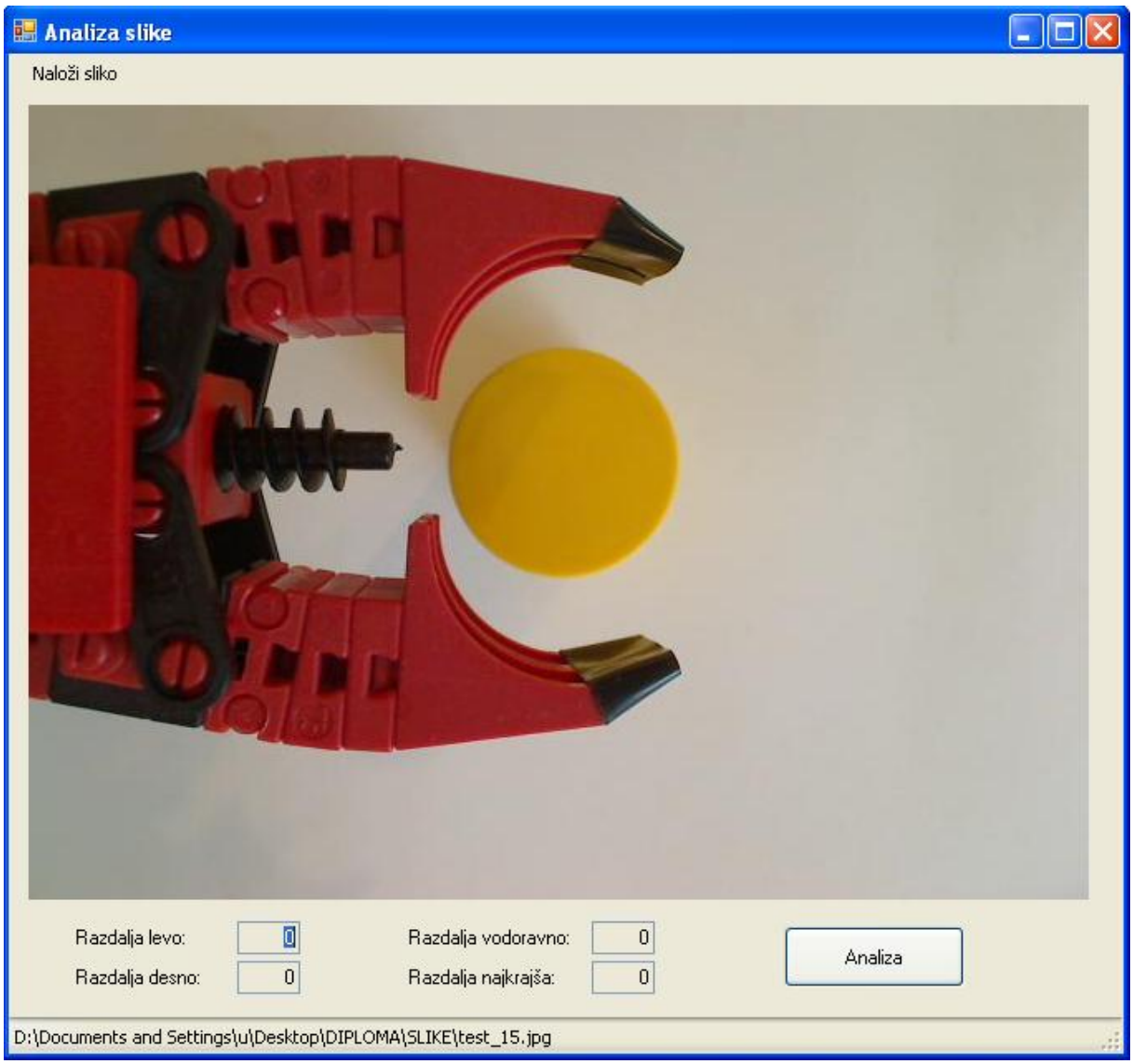
```
try
{
    oItem=oDevice.ExecuteCommand(CommandID.wiaCommandTakePicture);
    ImageFile imgFile = oItem.Transfer(jpegGUID) as WIA.ImageFile;
    SaveImageAs();
    imgFile.SaveFile(path);
    oItem = null;
}
```

*Slika 5.13: Zajem in shranjevanje slike.*

Kamera omogoča sistem avtomatskega nastavljanja parametrov (barvni odtenki, osvetlitev in ostrina slike). Po večkratnih poizkusih zajema slike, se je izkazalo, da je slika kvalitetnejša, če parametre za višino, pri kateri posnamemo sliko za analizo, nastavimo ročno. Prav tako se je s testiranjem pokazalo, da je ločljivost slike 640x480 dovolj dobra in natančna, za izvedbo mojega sistema analiziranja slike.

Ko sem določil optimalne parametre in višino za zajemanje slik ter naredil sistem zajemanja in hranjenja postnetkov, je bilo potrebno slike pretvoriti v primeren format za obdelavo ter določiti merilne točke in barvne filtre za prepoznavanje točk. Sliko sem pretvoril v bitno polje (ang. bitmap). Vsaka točka polja je sestavljena iz pike (ang. pixel) in je predstavljena v RGB barvnem formatu.

Robotska roka ima prijemala rdeče barve, objekt, ki ga prenaša, je rumene barve. Ker po formatu RGB (ang. Red, Green, Blue) med rdečo in rumeno barvo ni velike razlike, sem skrajne robove roke prelepil s črnim trakom ter za podlago uporabil bel trši papir. Na ta način sem lažje določil kontrolne točke in barvne filtre.




*Slika 5.14: Zajem slike preko WIA vmesnika.*

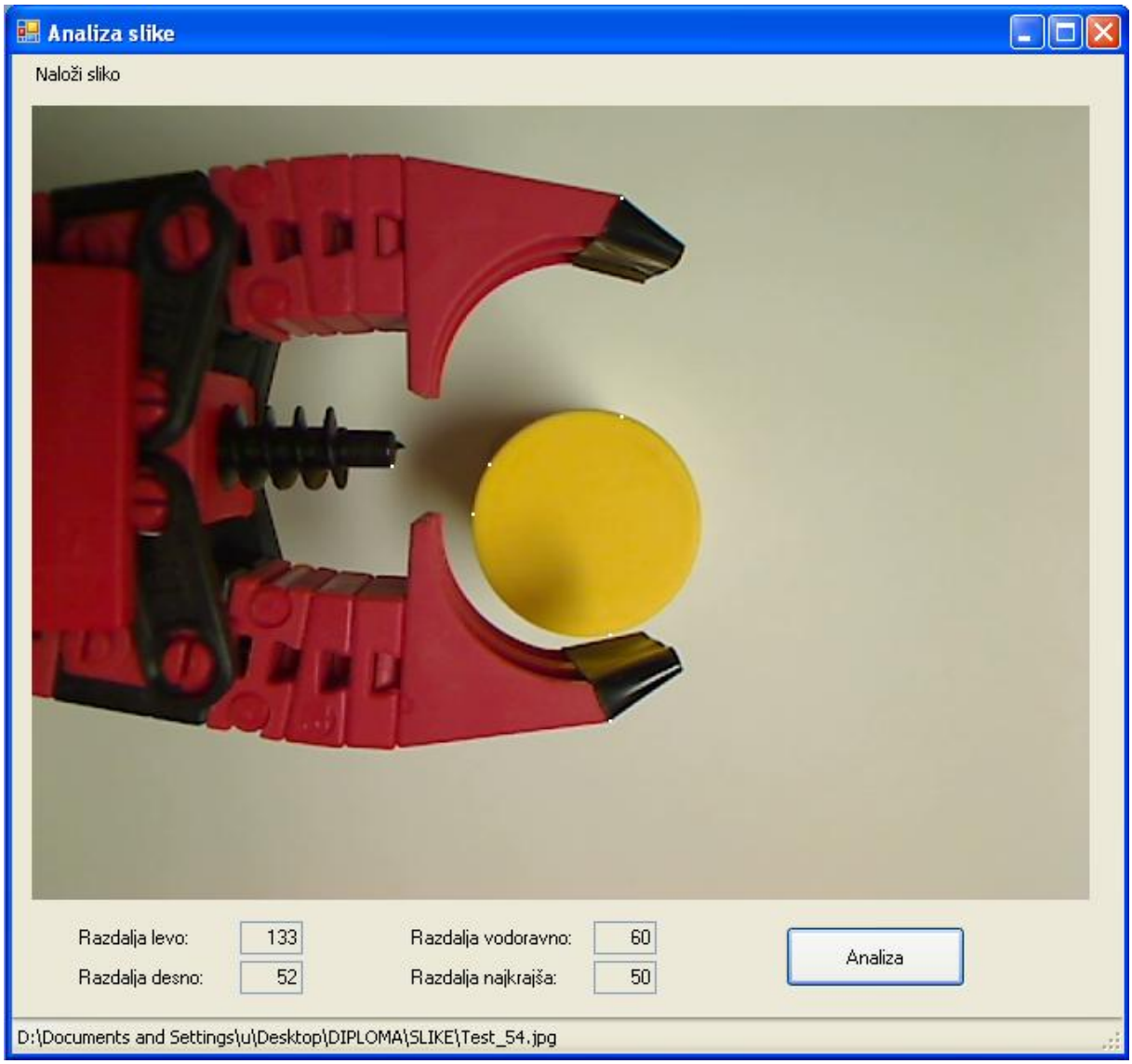
Ko so bile testne slike posnete, sem s pomočjo programa Slikar (ang. Paint) določil barvne filtre na način, da sem s pomočjo orodja za prepoznavanje barve (ang. pick color) zajel več vzorcev nekega objekta in tako določil barvno območje v RGB formatu. Tabela 5.6 prikazuje določitev območja prepoznavanja rumene točke na objektu. Iz objekta sem vzel deset vzorcev barv in določil ustrezno številčno območje v formatu RGB. Na podlagi območja (min. in maks. vrednosti) sem lahko kasneje napisal filter za prepoznavanje rumenega odtenka. Za prepoznavanje črne barve sem izvedel enak postopek.

*Tabela 5.6: Zajete točke rumenega objekta v RGB formatu:*

| | RDEČA | ZELENA | MODRA |
|---|---|---|---|
| 1. | 222 | 180 | 62 |
| 2. | 209 | 156 | 18 |
| 3. | 162 | 117 | 24 |
| 4. | 191 | 144 | 36 |
| 5. | 205 | 147 | 21 |
| 6. | 157 | 115 | 29 |
| 7. | 215 | 177 | 60 |
| 8. | 193 | 144 | 25 |
| 9. | 195 | 143 | 33 |
| 10. | 190 | 139 | 24 |
| | | | |
| min. | 157 | 115 | 18 |
| maks. | 222 | 180 | 62 |

Za določanje razdalje objekta od prijemal robotske roke, sem določil šest kontrolnih točk, prek katerih sem izračunal ustrezne podatke za popravljanje pomika (po iztegu in po prostoru). Določil sem tri razdalje, od tega dve za določevanje razdalje po prostoru in eno za določevanje razdalje po iztegu. Izračun odmika objekta od leve strani prijemala poteka tako, da določimo kontrolno točko na skrajno višji točki, ki je na črni podlagi na robu roke. Druga kontrolna točka za izračun razdalje objekta od leve strani prijemala je prva rumena točka na koordinati *x* črne točke. Razlika koordinat *y* obeh točk nam pove odmik objekta od prijemala (na levi strani). Računanje odmika objekta od desne strani prijemala poteka podobno. Najprej poiščemo skrajno spodnjo točko na robu prijemala, nato je potrebno poiskati še zadnjo točko rumenega odtenka na osi *x* črne točke. Ker najprej določimo črno točko, ki je na sliki nižje od rumenega objekta, je potrebno analizo slike še enkrat ponoviti, da dobimo še rumeno točko na osi *x* črne točke. Ko to točko določimo, lahko izračunamo odmik na desni strani objekta. Na podlagi obeh razdalj lahko izračunamo koliko in v katero smer je potrebno premakniti roko.

Za izračun razdalje za kontroliranje iztega določimo dve točki: prva je na objektu med prijemali (na vretenu) v spodnjem skrajnem desnem robu, druga pa je prvi rumen odtenek, ki ni odvisen od koordinat črne točke na vretenu. Razdalja med koordinatama točk na osi *x* določa razdaljo, ki jo primerjamo z optimalno razdaljo, shranjeno v sistemu. Na podlagi primerjave, lahko izračunamo potreben premik. Upoštevati moramo minimalne odmike (histereze). Analizo slike prikazuje slika 5.15, kjer so kontrolne točke označene z belimi pikami.




*Slika 5.15: Analiza slike in določevanje kontrolnih točk.*

Po uspešno dokončanem sistemu analize slike in določevanju kontrolnih točk, je bilo potrebno sistem vgraditi v glavno aplikacijo nadzornega sistema. Ker sem hotel v nadzornem sistemu prikazovati tudi živo sliko, ki prikazuje stanje robotske roke, sem moral spremeniti sistem zajema slike iz WIA (podpira le zajem slik) na sistem, ki uporablja razred WebCam. Po spremembi sistema, so se začele pojavljati težave pri analiziranju in določanju točk. Večkrat sem dobil nepravilne ali celo nemogoče rezultate. Ker je bil sistem že implementiran v samo aplikacijo in povezan s krmilnim računalnikom, je bilo zelo težko ugotoviti vzroke za napake. Implementiral sem poseben sistem, ki mi je vsako zajeto sliko iz spletne kamere shranil na disk z imenom Test in določenim indeksom. Ta indeks sem uporabil v podatkovni bazi (povezava slike z analiziranimi podatki), v katero sem shranjeval tudi odtenke RGB, ki so se ujemali po določenem barvnem filtru za zajem črne in rumene točke. Odtenke RGB je sistem prepoznal kot pravilne kontrolne točke. Tako sem lahko kasneje analiziral sliko in jo primerjal s pridobljenimi vrednostmi, shranjenimi v podatkovni bazi. Glavni vzrok za napake je bila sprememba sistema zajemanja slik iz WIA na sistem, ki je zajemal slike direktno iz

žive slike. Zajete slike so se razlikovale v kvaliteti zajema. Slike, zajete z uporabno razreda WebCam, so bile kvalitetnejše in so posledično vsebovale tudi bolj žive barve v primerjavi s slikami posnetimi z uporabo sistema WIA (primerjava slik 21 in 22). Drugi, tudi zelo pomemben vzrok, je bil spreminjanje svetlobe in pojav senc na objektu. Ker sem delovanje sistema testiral v različnih delih dneva (različna osvetljenost), so bili tudi rezultati analize slike in pozicioniranja slabši. Rešitev problema je bil popravek filtrov za iskanje točk in vgradnja luči na sistem, ki je zagotavljala konstantno osvetljenost objektov, ne glede na svetlobo iz okolice.

Sistem zajema in analize slike bi lahko izvedli s primernimi industrijskimi komponentami. Za primer tovrstne uporabe lahko vzamemo podjetje National Instrument, ki ponuja bogato zbirko kamer in ustrezne programske opreme. Pametna kamera NI 1762 z resolucijo 640x480 z že vključeno osnovno programsko opremo stane 1849 € [22].

#### 5.3.3.1 Meritve in predstavitev rezultatov

Opravil sem nekaj meritev, s katerimi sem preveril ponovljivost analize slike iz iste točke in naredil primerjavo reguliranja roke s pomočjo spletne kamere in brez nje.

Tabela 5.7 prikazuje meritve ponovljivosti rezultatov analize slike. Zajem in analizo slike sem opravil desetkrat, vedno iz iste pozicije. Rezultati v tabeli so zaradi boljše predstave predstavljeni v pikah (ang. pixel). Na tej višini (višina za zajem in analizo slike) dvanajst pik predstavlja približno en milimeter. Iz tabele je razvidno, da je ponovljivost zajema in analiziranja slike dobra, saj se po posamezni meri meritev razlikuje za največ eno piko.

*Tabela 5.7: Meritve ponovljivosti:*

| ŠT. MERITVE | RAZDALJA LEVO | RAZDALJA DESNO | RAZDALJA OD ROKE |
|---|---|---|---|
| 1 | 92 | 90 | 52 |
| 2 | 92 | 89 | 52 |
| 3 | 92 | 89 | 53 |
| 4 | 92 | 89 | 52 |
| 5 | 92 | 89 | 52 |
| 6 | 92 | 89 | 53 |
| 7 | 92 | 89 | 52 |
| 8 | 92 | 90 | 52 |
| 9 | 92 | 90 | 52 |
| 10 | 93 | 89 | 53 |

Opravil sem deset meritev pozicioniranja roke s pomočjo spletne kamere ter deset meritev brez uporabe kamere in analize slike. Meritve sem opravil tako, da sem s pomočjo ročnega režima določil ciljno točko, na katero sem vodil roko iz različnih pozicij. Prvih pet premikov se je roka premikala v ciljno točko iz desne strani, v ostalih petih poizkusih pa iz leve strani. Za vsak premik sta opravljeni dve meritvi: najprej je bila opravljena meritev brez analize slike, zatem sem roko s pomočjo ročnega režima vrnil nazaj na začetno referenčno točko in izvedel še poizkus s pomočjo kamere. Rezultati meritev so predstavljeni v tabeli 5.8. Tabela je sestavljena iz številke meritve, začetne točke, napake brez in z uporabo kamere, popravljene točke in ciljne točke. Začetna točka je definirana s koordinatami prostora, iztega, višine in prijemala. Napaki brez in z uporabo analize slike sta podani v milimetrih in sta zapisani kot napaka pomika ter napaka iztega. Predznak pomika določa zamaknjenost iz smeri: levi zamik je predstavljen z negativnim, desni pa s pozitivnim številom. Zamik po iztegu je predstavljen na podoben način, le da negativni predznak predstavlja zamik nazaj, pozitivni predznak pa zamik naprej. Popravljena točka predstavlja točko, v kateri se sistem nahaja po procesu analize in popravljanja pozicije. Zapisani sta le koordinati premika po prostoru in po iztegu.

*Tabela 5.8: Meritve premikov z in brez uporabe analize slike.*

| ŠT. MERITVE | ZAČETNA TOČKA | NAPAKA BREZ [mm] | NAPAKA Z [mm] | POPRAVLJENA TOČKA | CILJNA TOČKA |
|---|---|---|---|---|---|
| 1 | 00; 00; 00; 00 | -5 ; 5 | 1 ; 0 | 80; 47 | 81; 50 |
| 2 | 35; 10; 00; 00 | 6 ; -5 | 1 ; 0 | 82; 53 | 81; 50 |
| 3 | 40; 15; 50; 00 | 3 ; -2 | 0 ; 0 | 81; 51 | 81; 50 |
| 4 | 71; 30; 74; 00 | 5 ; -9 | 1 ; 1 | 82; 55 | 81; 50 |
| 5 | 74; 74; 55; 00 | 3 ; -7 | 0 ; 1 | 81; 54 | 81; 50 |
| 6 | 86; 40; 40; 00 | 5 ; -1 | 0 ; -1 | 82; 50 | 81; 50 |
| 7 | 91; 55; 52; 00 | 4 ; -4 | 0 ; -1 | 81; 48 | 81; 50 |
| 8 | 88; 76; 13; 00 | 3 ; -3 | 0 ; -1 | 81; 48 | 81; 50 |
| 9 | 106; 04; 60; 00 | 5 ; 0 | 1 ; 0 | 82; 50 | 81; 50 |
| 10 | 99; 80; 08; 00 | 6 ; 4 | -1 ; 0 | 81; 48 | 81; 50 |

Rezultate meritev je bilo zaradi pomanjkanja prostora in nelinearnosti prijemal zelo težko natančno izmeriti, zato sem zaokrožil posamično meritev na milimeter natančno. Vrednost napake, popravljene prek spletne kamere, so največ en milimeter. V primeru, da bi imeli drugače upravljan sistem in bi namesto motorjev z enosmernim tokom uporabili servo pogone z absolutnimi enkoderji, bi lahko s tem sistemom analiziranja slike zagotovili popravljanje na desetinke milimetra natančno. Pozicioniranje brez uporabe sistema popravljanja zgreši idealno točko za več milimetrov. Iz tabele je razvidno, da se je roka lahko po določenem številu premikov tudi boljše pozicionirala, saj nisem vedno peljal roke v referenčno točko, ampak sem jo premikal naprej na ciljno točko v avtomatskem režimu (z/brez sistema analize

slike), nazaj na poljubno začetno točko, pa v ročnem režimu. Pri tem so se izvajali premiki levo/desno, naprej/nazaj in pojavila so se odštevanja interferenc napak premikov. Zato se je lahko roka pozicionirala bolj natančno.

### 5.3.1 Komunikacija s PLK

Krmilni in PC računalnik, na katerem se izvaja nadzorna aplikacija, sta povezana z vodilom Ethernet. Povezava se vzpostavi prek vgrajenih gonilnikov za Ethernet komunikacijo iz knjižnice LibNoDave. V funkciji za vzpostavitev komunikacije nastavimo naslov IP krmilnega računalnika in komunikacijska vrata (Siemens in Vipa krmilniki poslušajo na vratih 102), številko letve (ang. rack) ter oznako krmilnika na letvi. Ko poženemo nadzorno aplikacijo, se najprej vzpostavi povezava s krmilnim računalnikom. Ta komunikacija poteka kontinuirano skozi celotno izvajanje aplikacije. Ko izključimo aplikacijo, izklopimo povezavo s krmilnim računalnikom in sprostimo internetni vtičnik (slika 5.16). Funkcija za vzpostavitev komunikacije, ki jo predstavlja slika 4.3 je napisana v razredu G_Povezava in je javna funkcija, kar pomeni, da jo lahko kličemo v različnih razredih in oknih. Spremenljivka DC, ki je tipa `libnodave.daveConnection` kaže na vzpostavljeno povezavo. Prek te spremenljivke lahko upravljamo s povezavo (branje/pisanje spremenljiv na/iz krmilnega računalnika … ).

Za upravljanje stabilne povezave skrbi nova nit (proces), ki periodično preverja povezavo prek pomožnega bita (Povezava) in jo v primeru prekinitve poskuša vzpostaviti nazaj. Ob neuspehu, sistem poskuša povezavo v periodičnih ciklih vzpostaviti nazaj. V primeru, da ena od funkcij, ki je namenjena bodisi za pisanje ali branje iz/v krmilni računalnik, naleti na težavo in se ne izvede pravilno, sistem pomožni bit resetira. Proces, ki skrbi za povezavo, deluje v ozadju, neodvisno od delovanja aplikacije in ne obremenjuje glavne aplikacije ter v primeru, da se je povezava prekinila, blokira kakršnokoli nadaljno komuniciranje s krmilnim računalnikom (slika 5.17).

Komunikacija med krmilnim in osebnim računalnikom, ki sem jo izvedel prek knjižnice LibNoDave, je vseskozi delovala stabilno.

```
public void Povezava_TCP_IP_close()
{
        _dc.disconnectPLC();
        libnodave.closeSocket(_fds.rfd);
}
```

*Slika 5.16: Sprostitev komunikacije z PLK.*

```
while (true)
{
      if (G_Povezava.Povezva == false)
      {
            G_Spremenljivke sp = new G_Spremenljivke();
            G_Spremenljivke.izhodi.branje = false;
            G_Spremenljivke.vhodi.branje = false;
            if(povezava.Povezava_TCP_IP_povezi(_rack, _slot,
            _naslovP))
                  G_Povezava.Povezva = true;

      }
      ...
}
```

*Slika 5.17: Del kode v procesu za upravljanje s povezavo.*

# 6. Sklepne ugotovitve

Cilj diplomske naloge je bil izdelati krmilni in nadzorni sistem za upravljanje robotske roke. Nalogo smo rešili z namestitvijo prosto cenovno ugodne spletne kamere namenjene široki potrošnji in uporabo prosto dostopnih programskih knjižnic. Zgradili smo svoj nadzorni sistem, v katerega smo poleg osnovnih funkcionalnosti integrirali tudi zajem slike iz spletne kamere in njeno analizo z elementi računalniškega vida.

Uporaba programskega okolja Visual Studio in programskega jezika C# pri razvijanju lastnega nadzornega sistema se je pokazala kot prava izbira, saj lahko razvijemo poljuben sistem skorajda brez omejitev. Največ težav sem imel s sistemom vgradnje spletne kamere in pri odkrivanju napak napačne analize slike. Največ težav so predstavljale sence in neenakomerna osvetljenost objekta v prostoru. Odpravil sem jih z uporabo barvnih filtrov ter z vgradnjo luči, ki je z dokaj enakomerno osvetljenostjo objekta občutno zmanjšala vpliv zunanje svetlobe.

Na podlagi opravljenih meritev in obdelave rezultatov lahko povzamem, da celoten sistem deluje dobro in stabilno. S povezavo med krmilnim računalnikom in našim nadzornim sistemom ni bilo posebnih težav. S spletno kamero in sistemom zajema analize slike, ki sem ju uporabil, bi lahko roko pozicionirali še natančneje, vendar oprema robotske roke, ki deluje s pomočjo direktno vodenih enosmernih elektromotorjev, tega ne dovoljuje.

Rešitev za boljše delovanje takega sistema bi bila zamenjava obstoječih motorjev s servo pogoni in absolutnimi optičnimi dajalniki impulzov. Možna nadgradnja sistema bi bila tudi predelava sistema zajema in obdelave slike, ki bi nam omogočila, da bi objektu sledili kontinuirano ne glede na položaj robotske roke. S tem bi se izognili zamudnemu pozicioniranju robotske roke na izbrano višino in ustavljanje sistema za zajem slike.

# 7. Viri


[1] (2010) Franc Solina, Računalniški vid nekdaj in danes
Dostopno na:
http://eprints.fri.uni-lj.si/199/1/Solina_ROSUS2006.pdf.

[2] (2010) Siemens, Simatic Net Industrial Ethernet Networking Manual, str. 19 – 24
Dostopno na:
http://support.automation.siemens.com/WW/llisapi.dll/csfetch/27069465/SYH_Industria_Ethernet_Networking_Manual_76.pdf?func=cslib.csFetch&nodeid=28940210&forcedownload=true.

[3] (2010) James Vernon, Programmable logic control, str. 1-2
dostopno na: http://www.control-systems-principles.co.uk/whitepapers/programmable-logic-control.pdf.

[4] (2010) Stancho Djiev, Programmable logic controllers (PLC), str. 1
Dostopno na: http://anp.tu-sofia.bg/djiev/PDF%20files/PLC-1.pdf.

[5] (2010) Programmable logic controllers, str. 9 - 10
Dostopno na: http://www.eod.gvsu.edu/~jackh/books/plcs/chapters/plc_intro.pdf.

[6] (2010) Siemens, S7-300CPU 31xC and CPU 31x:Specifications, str. 161 - 243
Dostopno na: http://www.ad.siemens.com.cn/products/as/simaticplc/s7-300/download/s7300_cpu_31xc_and_cpu_31x_manual_en-US_en-US.pdf.

[7] (2010) Siemens, Working with STEP 7
Dostopno na: http://support.automation.siemens.com/WW/llisapi.dll?query=6ES7810-4CA08-8BW1&func=cslib.cssearch&content=adsearch/adsearch.aspx&lang=en&siteid=cseus&objaction=cssearch&searchinprim=&nodeid99=.

[8] (2010) Siemens, Programming with STEP 7, str. 197 - 206
Dostopno na:
http://support.automation.siemens.com/WW/llisapi.dll?query=programming+with+step7&func=cslib.cssearch&content=adsearch/adsearch.aspx&lang=en&siteid=cseus&objaction=cssearch&searchinprim=&nodeid99=.

[9] Siemens (2000), Information and Training, Programming with S7-SCL, Course ST-7SCL, str. 3 – 5.

[10] S. Sokolič (1999), Trendi v razvoju Scada sistemov, Avtomatika 1, str. 26, 27.

[11] (2010) T. Hannelius, M. Shoroff, Embedding OPC Unified Architecture, str. 1 – 2
Dostopno na:
http://www.wapice.com/wapice_cms/files/HanneliusSchroffTuominen_%20Embedding_OPC UA.pdf.

[12] (2010) A. Daneels, W. Salter, What is SCADA?, str. 1 - 3
Dostopno na:
http://www.elettra.trieste.it/ICALEPCS99/proceedings/papers/mc1i01.pdf.

[13] (2010) Siemens, WinCC Communication Manual, str. 19 – 25
Dostopno na: http://faziliao.jonweb.net/siemens/SW/COMB1V5_E.PDF.

[14] (2010) Rok Slokar, Optični sistemi v industriji, str. 6
Dostopno na:
http://vicos.fri.uni-lj.si/data/alesl/semSlokar.pdf.

[15] (2010) Senzorji in aktuatorji,
Dostopno na:
http://colos.fri.uni-lj.si/eri/RAC_SISTEMI_OMREZJA/html/RSO-OKOLJE/Senzorji_aktuatorji.html.

[16] (2010) Microsoft Visual Studio 2008.
Dostopno na: http://msdn.microsoft.com/en-us/vstudio/default.aspx.

[17] (2010) Microsoft Visual C#
Dostopno na: http://msdn.microsoft.com/en-us/vcsharp/default.aspx.

[18] (2010) An Introduction to P/Invoke and Marshaling on the Microsoft .NET Compact Framework
Dostopno na: http://msdn.microsoft.com/en-us/library/aa446536.aspx.

[19] (2010) Windows API
Dostopno na: http://msdn.microsoft.com/en-us/library/aa383750.aspx.

[20] (2010) Integrate the web Webcam functionality using C#
Dostopno na:
http://www.c-sharpcorner.com/UploadFile/yougerthen/810262008070218AM/8.aspx.

[21] (2010) How do you program the communication blocks FC5 and FC6?
Dostopno na:
http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&objId=17853532&objAction=csOpen&nodeid0=27103175&lang=en&siteid=cseus&aktprim=0&extranet=standard&viewreg=WW.

[22] (2010) National Instruments, NI 1722 Smart Camera
Dostopno na:
http://sine.ni.com/nips/cds/view/p/lang/en/nid/204078.

[23] (2010) Windows Image Acquisition (WIA)
Dostopno na:
http://msdn.microsoft.com/en-us/library/ms630368(VS.85).aspx.