Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Rok Šolar

Avtomatizacija perifernih naprav ob brizgalnem stroju

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Uroš Lotrič

Ljubljana, 2017

COPYRIGHT. Rezultati diplomske naloge so intelektualna lastnina avtorja in Fakultete za računalništvo in informatiko Univerze v Ljubljani. Za objavo in koriščenje rezultatov diplomske naloge je potrebno pisno privoljenje avtorja, Fakultete za računalništvo in informatiko ter mentorja.

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Za proizvodnjo brizganja plastičnih izdelkov izdelajte programsko opremo za vodenje perifernih naprav ob brizgalnem stroju in poiščite rešitev za optimizacijo tiskanja različnih nalepk na škatle s končnimi izdelki. Pri tem poskrbite, da bo programska koda dovolj splošna, da jo bo mogoče uporabiti na vseh proizvodnih linijah v podjetju. Poseben poudarek namenite preglednosti kode in potrebni dokumentaciji, ki bo omogočala enostavne nadgradnje v prihodnosti ter tako doprinesla k zmanjševanju stroškov proizvodnje in povečevanju konkurenčne prednosti podjetja.

Iskreno se zahvaljujem mentorju izr. prof. dr. Urošu Lotriču za vse nasvete, strokovnost in podporo pri izdelavi te diplomske naloge.
Zahvalil bi se tudi sodelavcem za vso pomoč pri zasnovi dela.
Posebna zahvala pa gre družini, Brigiti in ostalim bližnjim, ki so mi stali ob strani tekom celotnega študija.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
PLC	programmable logic controller	programirljivi logični krmilnik
IEC	International Electrotechnical Commission	mednarodna komisija za elektrotehniko
POU	program organization unit	programska organizacijska enota
IL	instruction list	seznam ukazov
ST	structured text	strukturirani tekst
LD	ladder diagram	lestvični diagram
FBD	function block diagram	funkcijski blokovni diagram
SFC	sequential function chart	sekvenčni funkcijski diagram
RAM	random-access memory	bralno-pisalni spomin
HSC	high speed counter	hitri števec
RTC	real time clock	ura realnega časa
IDE	integrated development environment	integrirano razvojno okolje
HTTP	hypertext transfer protocol	protokol za prenos hiperteksta
TCP	transmission control protocol	protokol za nadzor prenosa
XML	extensible markup language	razširljiv označevalni jezik
MES	manufacturing execution system	proizvodni informacijski sistem
UML	unified modeling language	poenoteni jezik modeliranja
ASCII	American standard code for information interchange	ameriški standardni nabor za izmenjavo informacij
EPL	Eltron programming language	programski jezik Eltron
ZPL	Zebra programing language	programski jezik Zebra
HTML	hypertext markup language	jezik za označevanje hiperteksta
CSS	cascading style sheets	kaskadne stilske predloge

Povzetek

Naslov: Avtomatizacija perifernih naprav ob brizgalnem stroju

Avtor: Rok Šolar

V tem delu je predstavljeno načrtovanje in izdelava novega programa za programirljive logične krmilnike ter nadgradnja sistema za tiskanje nalepk. V procesu proizvodnje so krmilniki uporabljeni za avtomatizacijo perifernih naprav ob vsakem stroju za brizganje plastike ter za komunikacijo z nadrejenimi sistemi. Obstoječ program na krmilnikih je zastarel, nedokumentiran ter prvotno ni bil izdelan za proizvodni proces v taki obliki, kot se v podjetju izvaja danes. Nov program ima na razumljiv in urejen način implementirane vse zahteve trenutnega proizvodnega procesa, je ustrezno dokumentiran ter je načrtovan na način, da bo v prihodnosti omogočal učinkovite nadgradnje. Te lastnosti so v veliki meri dosežene z uporabo programerske paradigme programiranja na osnovi končnih avtomatov. Opisana je tudi nadgradnja sistema za tiskanje nalepk, ki sedaj omogoča tiskanje različnih nalepk. V diplomskem delu bodo najprej opisani elementi sistema ter kako so umeščeni v proces proizvodnje. Nato bodo predstavljeni glavni koncepti dela s krmilniki, uporabljen krmilnik ter standard IEC 61131-3. V glavnem delu bo opisano načrtovanje novega programa, nekaj pomembnejših algoritmov delovanja ter nadgradnja sistema za tiskanje nalepk. Na koncu bo opisano še dokumentiranje programov za krmilnike.

Ključne besede: programirljivi logični krmilniki, avtomatizacija, periferija, IEC 61131-3, CyBro-2, tiskanje nalepk, dokumentacija.

Abstract

Title: Automation of injection moulding machine peripheral equipment

Author: Rok Šolar

The thesis presents the process of planning and implementing of a new software solution for programmable logic controllers and the upgrade of the label printing system. In the production process, controllers are used to automate peripherals along side every injection moulding machine and to communicate with superior systems. The existing software code is obsolete, undocumented and was not originally designed for the production process in the form that is carried out in the company today. The new program implements all the requirements of the current production process in a comprehensible way, is properly documented and designed in a way that will enable effective upgrades in the future. These properties are largely achieved by using the automata-based programming paradigm. The upgrade of the printing system is also described, which now allows printing of various labels. The thesis will first describe the elements of the system and how they are placed in the production process. Next, the basic concepts of working with controllers, the controller that we will use and the standard IEC 61131-3 will be described. The main part will describe the design of the new program, some of the more important algorithms from the code, and the upgrade of the printing system. Finally, documentation of programs for controllers will be described.

Keywords: programmable logic controllers, automation, peripherals, IEC 61131-3, CyBro-2, printing labels, documentation.

Poglavje 1

Uvod

Programirljivi logični krmilniki (ang. programmable logic controllers – PLC, v nadaljevanju samo krmilniki) so postali ključen element v avtomatizaciji proizvodnje. S hitrim razvojem krmilnikov, ki ga lahko primerjamo z razvojem osebnih računalnikov [7], ter njihovo standardizacijo, krmilniki postajajo vedno bolj zmogljivi, omogočajo vedno več funkcij in prevzemajo vedno večjo vlogo v avtomatizaciji proizvodnje. Zaradi vseh odgovornosti, ki jih krmilniki prevzemajo, pa je potrebna kvalitetna, berljiva, nadgradljiva in dobro dokumentirana programska koda, ki omogoči učinkovito in dolgoročno delovanje procesa proizvodnje.

Najpomembnejši elementi proizvodnje različnih vrst zapiralne embalaže in drugih tehnično zahtevnih izdelkov za farmacevtsko, medicinsko in elektroindustrijo so stroji za brizganje plastike ter podporni sistemi, ki krmilijo vso pripadajočo periferijo ter komunicirajo z nadrejenimi sistemi.

Težavo trenutnega perifernega sistema v proizvodnji predstavlja programska koda, ki je uporabljena na krmilnikih. Ta je bila izdelana s strani zunanjih izvajalcev, je zastarela, nedokumentirana in vsebuje veliko nedokumentiranih nadgradenj, saj se je proces proizvodnje od izdelave obstoječe programske kode precej spremenil. Te lastnosti so jo naredili nerazumljivo, povzročajo napake v proizvodnji ter težave pri nadgradnjah sistemov, odvisnih od delovanja krmilnikov. Poleg tega predstavlja težave tudi sistem za tiskanje

nalepk. Star sistem omogoča tiskanje le ene, privzete nalepke, nekateri kupci pa zahtevajo na škatlah drugačne nalepke. Te nove nalepke so se do naše nadgradnje sistema tiskale ročno. Cilj dela je torej načrtovati, izdelati in ustrezno dokumentirati nov program za krmilnike v proizvodnji. Poleg tega je cilj tudi načrtovati in izdelati nov sistem tiskanja nalepk, ki bo omogočal tiskanje različnih nalepk glede na podatke delovnega naloga.

V delu bomo najprej opisali proizvodnjo podjetja ter kako so krmilniki in vse periferne naprave, ki jih krmilnik krmili, umeščeni v proces proizvodnje. Tretje poglavje bo namenjeno predstavitvi krmilnika, ki ga podjetje uporablja in katerega program bo načrtovan in izdelan. Ob tem bomo opisali tudi nekaj splošnih lastnosti krmilnikov ter standard IEC 61131-3, ki opisuje programske jezike za programiranje krmilnikov. V četrtem poglavju bomo predstavili načrtovanje novega programa – razloge za nov program, krmilno omarico ter glavno programersko paradigmo, po kateri bo program načrtovan – programiranje na osnovi končnih avtomatov. V petem poglavju bomo opisali izdelan program, torej strukturo programa ter implementacijo nekaterih pomembnejših funkcij. V šestem poglavju bomo podrobneje predstavili problem tiskanja nalepk, pri čimer bomo najprej opisali delovanje starega sistema, nato pa še izdelan nov sistem ter njegove prednosti. Ob tem bomo opisali tudi prototipno spletno aplikacijo, ki smo jo uporabili za testiranje sistema, dokler ta ni implementiran v proizvodni informacijski sistem podjetja. V sedmem poglavju bomo predstavili način dokumentiranja programa, v osmem pa še analizo, pomanjkljivosti ter možnosti nadgradnje sistema.

Poglavje 2

Proizvodnja in komponente

2.1 Opis

Glavna dejavnost podjetja je izdelava rešitev na področju zapiralne embalaže za tube, steklenice in druge vrste embalaže. Poleg tega ponujajo tudi rešitve na področju tehnično zahtevnih izdelkov za farmacijo, medicinsko in elektroindustrijo. Podjetje izdelavo teh izdelkov realizira večinoma s proizvodnim procesom brizganja plastike z zapiralnimi silami [17]. Brizganje plastike je eden izmed najbolj pogosto uporabljenih procesov za proizvodnjo plastičnih izdelkov. Izveden je v obliki cikličnega procesa, ki vključuje hitro polnjenje kalupa s staljeno plastiko (ali pa tudi s kakšnim neplastičnim materialom), čemur sledi ohlajevanje ter na koncu izmet izdelanih kosov [6]. Pomemben del procesa proizvodnje pa predstavljajo operacije, ki se morajo izvesti po odpiranju orodja oziroma po izmetu izdelanih kosov. Sam proces izdelave kosov z brizganjem plastike je običajno visoko avtomatiziran. Problem nastane, ko se orodje odpre in je potrebno izdelane kose in podatke o izdelanih kosih ustrezno obdelati. V mnogih primerih je ta del procesa izveden ročno, kar kosom poviša ceno. Na tem mestu nastane največji potencial za nižanje stroškov proizvodnje [13]. V našem proizvodnem procesu, po odpiranju orodja in izmetu, prevzamejo kose periferne naprave, ki se nahajajo ob vsakem stroju in so krmiljene s programirljivim logičnim krmilnikom. S tem

se potrebno ročno delo drastično zmanjša.

2.2 Periferija in proces proizvodnje

Postavitev periferije je prikazana na sliki 2.1. Sestavljena je iz šestih komponent:

- drča z izmetno loputo,
- tekoči trak,
- spustna loputa,
- tehtnica in terminal,
- vrtiljak,
- tiskalnik.

Slika 2.1: Stroj in periferija.

2.2.1 Drča z izmetno loputo

Izmetna loputa, prikazana na sliki 2.2, se nahaja na izmetni drči stroja. Njena naloga je preusmerjanje izdelanih kosov bodisi v izmet, če so kosi slabi, ali na tekoči trak za nadaljevanje procesa, če so kosi dobri. Krmiljena je preko digitalnega izhoda na krmilniku na sledeč način: če je izhod nastavljen na 1, potem je izmetna loputa nastavljena na dobre kose, v nasprotnem primeru pa je nastavljena na izmet.

Slika 2.2: Izmetna loputa na izmetni drči stroja.

2.2.2 Tekoči trak

Namen tekočega traku je, da pripelje kose od izmeta stroja do spustne lopute in tehtnice, torej, da kose dvigne na višino škatel. Krmiljen je preko digitalnega izhoda na krmilniku (0 – trak stoji, 1 – trak teče).

2.2.3 Spustna loputa

V spustni loputi, prikazani na sliki 2.3, se nabirajo dobri kosi, ki pridejo po tekočem traku. Ko je loputa polna, se kosi spustijo v škatlo. Če vsebuje tudi tehtnico, potem se kontrolira glede na težo kosov, v nasprotnem primeru pa se spustna loputa kontrolira glede na število brizgov, ki jih stroj opravi, ter ostalih vhodnih parametrov delovnega naloga, kot sta predpostavljena teža enega kosa ter število aktivnih gnezd orodja. Krmiljena je preko digitalnega izhoda na krmilniku (0 – loputa je zaprta, 1 – loputa je odprta).

Slika 2.3: Spustna loputa.

2.2.4 Tehtnica in terminal

Tehtnica je izdelek podjetja Libela Elsi iz Celja. Sestavljena je iz tehtalnega dela in terminala DPA 3T, prikazanega na sliki 2.4 [12], preko katerega nastavljamo parametre tehtanja ter preko katerega krmilnik bere podatke tehtanja. Krmilnik s tehtnico komunicira preko serijske povezave z uporabo

protokola RS-232. Pri tem je potrebno poudariti, da sta tehtnica in pripadajoč terminal opcijska. Če je vključena v proces proizvodnje, potem njene meritve služijo kot osnova za odpiranje spustne lopute ter določanje količine izdelanih kosov v škatli.

Slika 2.4: Terminal tehtnice.

2.2.5 Vrtiljak za škatle

Na vrtiljaku, prikazanem na sliki 2.5, so razvrščene škatle za pakiranje izdelanih kosov. Ko krmilnik izračuna, da je škatla polna, pošlje vrtiljaku signal za obrat. Ta se nato premakne in pod spustno loputo pristavi naslednjo prazno škatlo. Vrtiljak se prav tako krmili preko digitalnega izhoda na krmilniku. Ob prehodu digitalnega izhoda na krmilniku iz 0 na 1 se vrtiljak zavrti za en segment.

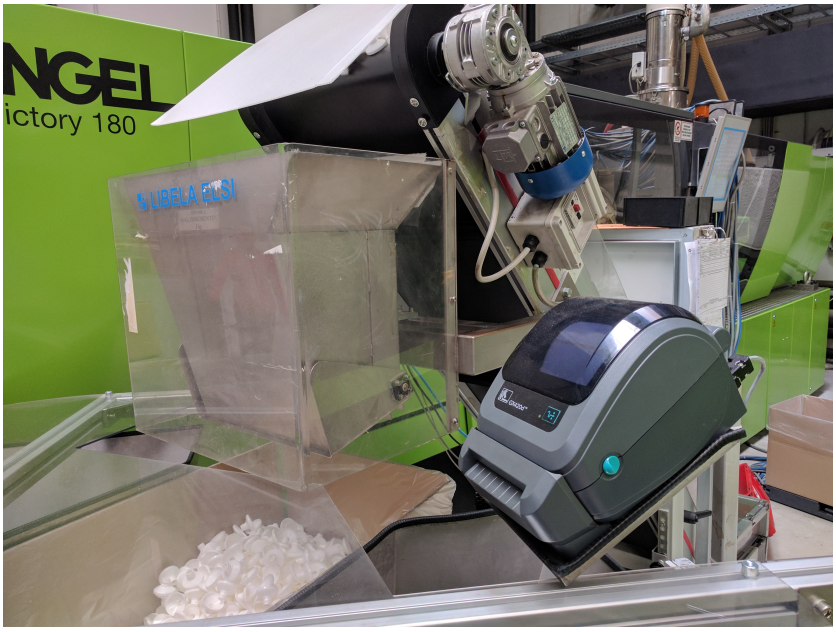
Slika 2.5: Vrtiljak s škatlami.

2.2.6 Tiskalnik

Tiskalnik je izdelek podjetja Zebra Technologies iz Združenih držav Amerike in je tipa GX420d (slika 2.6). Namenjen je tiskanju nalepk, ki jih delavec nalepi na škatle z izdelki. Tiskanje nalepke se proži pred ali po obratu vrtiljaka, odvisno od konfiguracije. Možne konfiguracije sistema bomo predstavili v poglavju 2.3. Krmilnik s tiskalnikom komunicira na isti način kot s terminalom tehtnice – z uporabo serijske povezave in protokolom RS-232. Več o delovanju ter konfiguraciji tiskalnika bomo predstavili v poglavju 6, kjer bomo opisali nadgradnjo sistema za tiskanje nalepk.

2.3 Konfiguracije sistema

Ker nekateri stroji nimajo tehtnice in terminala tehtnice ter ker so lahko ob nekaterih strojih periferne naprave drugače postavljene, imamo na volji štiri različne konfiguracije sistema:



Slika 2.6: Tiskalnik nalepk nad vrtiljakom škatel in ob spustni loputi.

1. Tehtanje kosov in zaporedje najprej tisk potem obrat. Če skupek perifernih naprav stroja vsebuje tudi tehtnico in terminal, potem se spustna loputa, vrtiljak ter tiskalnik prožijo na podlagi teže izdelanih kosov. Spustna loputa se odpre, ko je dosežena ciljna teža v tehtnici, vrtiljak in tiskalnik pa se prožita, ko je dosežena ciljna teža v škatli. Del prve konfiguracije je tudi lastnost, da se ob polni škatli najprej izvede tiskanje nalepke, šele nato obračanje vrtiljaka.

2. Tehtanje kosov in zaporedje najprej obrat potem tisk. Druga konfiguracija prav tako vsebuje tehtnico in terminal, le da se nalepke tiskajo pred obračanjem vrtiljaka.

3. Štetje kosov in zaporedje najprej tisk potem obrat. Tretja konfiguracija je namenjena primeru, ko nimamo tehtnice. Torej se spustna loputa, vrtiljak ter tiskalnik prožijo na podlagi števila dobrih brizgov stroja ter ostalih parametrov delovnega naloga. Vrstni red proženja tiskalnika in vrtiljaka je enak prvi konfiguraciji – najprej se proži tiskanje nalepke,

nato pa obrat vrtiljaka.

4. Štetje kosov in zaporedje najprej obrat potem tisk. Četrta konfiguracija je prav tako brez tehtnice, le da je ponovno zamenjan vrstni red tiskanja nalepke ter obračanja vrtiljaka.

Ne glede na to katero konfiguracijo imamo izbrano pa lahko spustno loputo, vrtiljak in tiskalnik vedno krmilimo tudi ročno. To je realizirano s stikalom, ki preklaplja med ročnim in avtomatskim načinom ter tipko, kot je to prikazano na sliki 2.7. Delovanje stikala in tipke je sledeče: za ročno upravljanje periferije moramo najprej prestaviti stikalo na ročni način. Nato s pritiskom na tipko odpremo spustno loputo. Če tipko pridržimo do zapiranja lopute, potem se proži tudi obrat vrtiljaka ter tisk nalepke. Vrstni red obrata vrtiljaka in tiskanja nalepke je odvisen od izbrane konfiguracije.

Slika 2.7: Stikalo in tipka za ročno krmiljenje spustne lopute, vrtiljaka in tiskalnika.

Poglavje 3

Krmilnik CyBro-2

Ob vsakem stroju v proizvodni hali stoji krmilnik, ki krmili vso periferijo in komunicira z nadrejenimi sistemi. Uporabljen je krmilnik CyBro-2, ki je izdelek podjetja Cybrotech iz Združenega kraljestva. V nadaljnjih podpoglavjih bomo na splošno opisali delo s programirljivimi logičnimi krmilniki ter podrobneje opisali krmilnik CyBro-2.

3.1 Programirljivi logični krmilniki

Programirljivi logični krmilniki so posebna vrsta mikroprocesorskih računalnikov, namenjeni predvsem uporabi v industrijskih okoljih [1]. Predhodniki krmilnikov so bili relejski kontrolni sistemi, torej trdo ožičena (ang. hard wired) vezja, zgrajena iz elektronskih komponent, kot so releji in stikala. Ker so bili taki sistemi trdo ožičeni, so bili prilagojeni samo nalogi, za katero so bili načrtovani [8]. Zato so morebitne napake ali kasnejše spremembe zahtevale novo ožičenje sistema. Relejski kontrolni sistemi so fizično zasedli veliko prostora, zahtevali veliko dela za izdelavo in še več za kakršnekoli kasnejše spremembe. Težavo je predstavljalo tudi testiranje, ki ga ni bilo mogoče izvesti dokler ni bil celoten sistem postavljen. Glavni cilj prvih krmilnikov je bil torej nadomestiti relejske kontrolne sisteme.

Poleg tega, da so krmilniki rešili opisane težave relejskih kontrolnih siste-

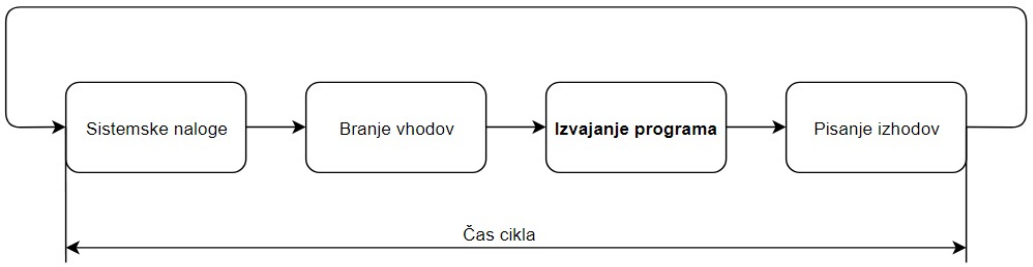
mov, lahko med razloge za uspeh prvih krmilnikov štejemo tudi enega izmed programskih jezikov, ki so ga uporabljali – lestvični diagram (ang. ladder diagram) [7]. Ta je bil osnovan na električnih shemah, ki opisujejo relejsko logiko in dovoljuje omejeno znanje programiranja za izdelavo programov, saj je programiranje zelo podobno risanju električnih shem.
Z razvojem raznih drugih vrst krmilnikov in računalnikov, se postavi vprašanje zakaj sploh uporabljati krmilnike namesto osebnih računalnikov. Krmilniki so po arhitekturi sicer podobni osebnim računalnikom, vendar obstaja nekaj ključnih razlik [14]:

- krmilniki so zmožni delati v okoljih z vibracijami, ekstremnimi temperaturami in vlažnostjo, kar za osebne računalnike ne drži,
- krmilniki so lahko bolj razumljivi za programiranje, saj uporabljajo tudi lažje razumljive programske jezike, kot so na primer lestvični diagrami,
- krmilniki navadno ne uporabljajo naprav kot so tipkovnice, trdi diski in zasloni ampak so zgrajeni kot samostojne neodvisne enote.

Programiranje krmilnikov je nekoliko drugačno, kot programiranje klasičnih programov za osebne računalnike. Znotraj krmilnika se neprestano izvaja ciklični proces branja vhodov, izvajanja programa ter pisanja izhodov, kot je to prikazano na sliki 3.1. Pri tem dobimo nov parameter, tipičen za krmilnike, ki nam pove koliko časa se izvaja cikel (ang. scan time). Ta je po navadi ranga nekaj milisekund in je odvisen od velikosti programa, velikosti pomnilnika ter tipa procesorja [7].

3.2 Standard IEC 61131-3

Hiter razvoj mikrotehnologij stalno odpira vrata novim in bolj naprednim programirljivim logičnim krmilnikom. S tem pa vedno bolj narašča tudi potreba po standardizaciji novih tehnologij. Od prvih krmilnikov je bilo predlaganih kar nekaj standardov, ki pa niso dobili dovolj velike mednarodne podpore. Mednarodna komisija za elektrotehniko (ang. International



Slika 3.1: Cikel procesa krmilnika.

Electrotechnical Commission – IEC) je uspela ustvariti prvi standard, ki je dosegel dovolj veliko mednarodno in industrijsko podporo – IEC 61131. Predstavljen je v obliki družine standardov, ki opisujejo celoten življenjski cikel krmilnikov, oziroma združuje razna določila in priporočila v zvezi z njimi. Za to delo je pomemben predvsem tretji del (IEC 61131-3), ki se nanaša na programske jezike krmilnikov [11].
Osnovni koncept oziroma osnovna enota programa krmilnika, predstavljena v tem standardu je programska organizacijska enota (ang. program organization unit – POU). Programske organizacijske enote so [11]:

- Funkcija (ang. function): običajna funkcija, klicana s parametri ali brez parametrov. Za iste parametre vedno vrne isti rezultat (nima spomina). Primer tega so lahko običajne aritmetične ali primerjalne funkcije.

- Funkcijski blok (ang. function block): funkcija, klicana s parametri ali brez parametrov. Ima spomin in si lahko zapomni status (instanca). Primer tega so časovniki in števci.

- Program: je vrh strukture in ima dostop do vhodov in izhodov krmilnika ter jih lahko naredi dostopne ostalim enotam.

Opisuje tudi osnovne elemente, lastnosti spremenljivk, semantična in sintaktična pravila ter pet standardnih programskih jezikov, ki se uporabijo za programiranje programskih organizacijskih enot [9, 11, 10]:

- Seznam ukazov (ang. instruction list – IL): je nizkonivojski tekstovni jezik, na pogled zelo podoben zbirniku, saj se koda piše v obliki strojnih ukazov in ne programskih stavkov. Po navadi je uporabljen kot vmesni jezik, v katerega se prevede koda ostalih tekstovnih in grafičnih jezikov.

- Strukturiran tekst (ang. structured text – ST): je visokonivojski tekstovni jezik, ki ga na pogled lahko primerjamo s programskim jezikom Pascal. Z njim se izognemo strojnim ukazom in nam omogoča uporabo programskih stavkov, s čimer koda postane veliko krajša in preglednejša kot pri seznamu ukazov. Zaradi nujno potrebnega prevajanja so programi običajno daljši in počasnejši.

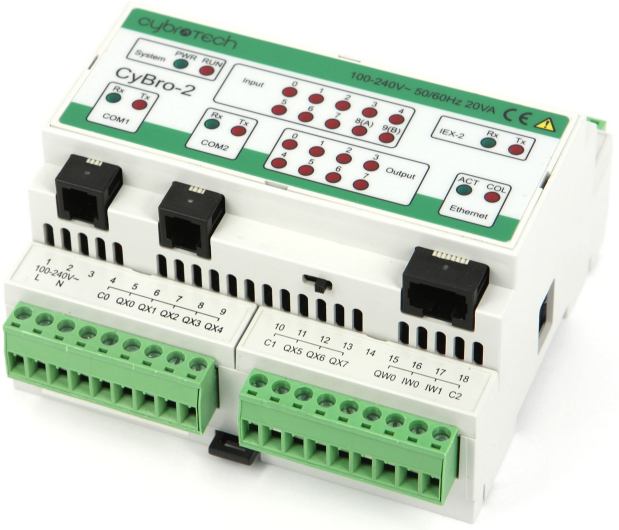
- Lestvični diagram (ang. ladder diagram – LD): je zelo razširjen grafični programski jezik. Osnovan je na podlagi električnih shem, ki opisujejo relejsko logiko, kar pomeni, da je lahko razumljiv in zahteva le osnovno znanje programiranja. Uporabljen je bil na prvih krmilnikih in je inženirjem omogočil lažji prehod iz relejskih kontrolnih sistemov na krmilnike.

- Funkcijski blokovni diagram (ang. function block diagram – FBD): je drugi grafični programski jezik. Opisuje obnašanje funkcij, funkcijskih blokov in programov v obliki med seboj povezanih standardnih ali uporabniško določenih grafičnih blokov. Na pogled je podoben shemam integriranih vezij.

- Sekvenčni funkcijski diagram (ang. sequential function chart – SFC): zadnji programski jezik, ki ga opisuje standard je sekvenčni funkcijski diagram. V osnovi je predstavljen v grafični obliki, vendar je lahko uporabljen tudi v tekstovni obliki. Opisuje strukturo programa, podobno kot to prikažejo diagrami stanj. Torej program podamo kot zbirko točno določenih stanj in prehodov med temi stanji. Stanja in prehodi so lahko v celoti opisani z uporabo sekvenčnega funkcijskega diagrama ali pa z uporabo enega izmed zgoraj opisanih jezikov iz standarda.

Standard IEC 61131-3 gre v mnogih primerih globoko v podrobnosti. Zato je mišljen kot zbirka smernic in ne kot zbirka pravil. Večina sistemov zato ne podpira vseh programskih jezikov iz standarda [11], kar velja tudi za krmilnik CyBro-2.

3.3 CyBro-2: Glavne lastnosti



Slika 3.2: Krmilnik CyBro-2 [15].

Nekaj ključnih specifikacij krmilnika CyBro-2, prikazanega na sliki 3.2 [3]:

- 10 digitalnih vhodov in 8 digitalnih izhodov,
- 4 analogni vhodi in 1 analogni izhod,
- možnost mrežne povezave (vmesnik Ethernet),
- možnost serijske povezave (dva vmesnika RS-232),
- vmesnik IEX-2 za povezavo z razširitvenimi moduli,
- procesor s taktom 24 MHz,

- 512 kB bliskovnega (ang. flash) pomnilnika (64 kB dostopnega za uporabniški program),
- 128 kB RAM pomnilnika (32 kB dostopnega za spremenljivke uporabniškega programa),
- hitri števec (ang. high speed counter – HSC) in ura realnega časa (ang. real time clock – RTC).

3.4 CyBro-2: Razširitveni moduli

Poleg zgoraj omenjenih zmožnosti je na voljo tudi preko 10 razširitvenih modulov, ki dodatno povečajo funkcionalnost krmilnika. Nekaj izmed njih je prikazanih na sliki 3.3. CyBro-2 v veliki meri uporablja odprte in dobro znane protokole, z nekaj izjemami. Ena izmed njih je povezljivost med krmilnikom in vsemi razširitvenimi moduli, za kar so razvili protokol IEX-2, ki je po funkcionalnosti podoben protokoloma CANopen in DeviceNet [3].
V našem proizvodnem procesu se ob vsakem stroju poleg krmilnika CyBro-2 uporablja tudi razširitveni modul COM-PRN, ki je prikazan na sliki 3.3, na skrajni desni. Ta ponuja podporo za protokola RS-232 in RS-485. Protokol RS-232 uporabljamo za serijsko komunikacijo s tiskalnikom.

Slika 3.3: Razširitveni moduli za krmilnik CyBro-2 [15].

3.5 CyBro-2: Programska oprema

Ob opisani strojni opremi nam podjetje ponuja tudi orodja v obliki programske opreme. Ta so prosto dostopna in dosegljiva na uradni strani podjetja. V času izdelave te naloge je bilo na voljo 18 programskih orodij. Najpomembnejše izmed teh programskih orodij je bržkone orodje CyPro, ki je integrirano razvojno okolje (ang. integrated development enviroment – IDE) namenjeno krmilnikom CyBro-2. Vsebuje vse osnovne funkcije klasičnega integriranega razvojnega okolja, kot so urejevalnik kode, prevajalnik in monitor spremenljivk. Povezava med integriranim razvojnim okoljem CyPro in krmilnikom CyBro-2 se lahko vzpostavi bodisi z uporabo serijske povezave (vmesnik RS-232) ali z uporabo mrežne povezave (vmesnik Ethernet). Z uporabo vmesnika Ethernet se lahko povežemo na krmilnik preko lokalne mreže, preko interneta ali z direktno povezavo med krmilnikom in osebnim računalnikom [4].
Krmilnik CyBro-2 je osnovan na standardu IEC 61131-3 in vključuje kar nekaj smernic standarda. Zato krmilnik in razvojno okolje podpirata dva od petih jezikov iz standarda. To sta strukturiran tekst (ST) in seznam ukazov (IL), torej oba tekstovna jezika [4]. Za rešitev, ki jo bomo opisali znotraj tega dela, bomo uporabili strukturiran tekst, saj omogoča pisanje krajše in preglednejše kode, kar je v našem primeru ključnega pomena.
Poleg orodja CyPro je ključnega pomena tudi orodje CyBro HTTP Server. To je orodje za komunikacijo s krmilnikom z uporabo protokola HTTP. Z njim dostopamo (branje in pisanje) do spremenljivk, shranjenih na krmilniku. Deluje v lokalnem omrežju in uporablja protokol TCP ter vrata 80, odgovor pa je v obliki dokumenta XML. V podjetju je uporabljen za komunikacijo s proizvodnim informacijskim sistemom (ang. manufacturing execution system – MES).

Poglavje 4

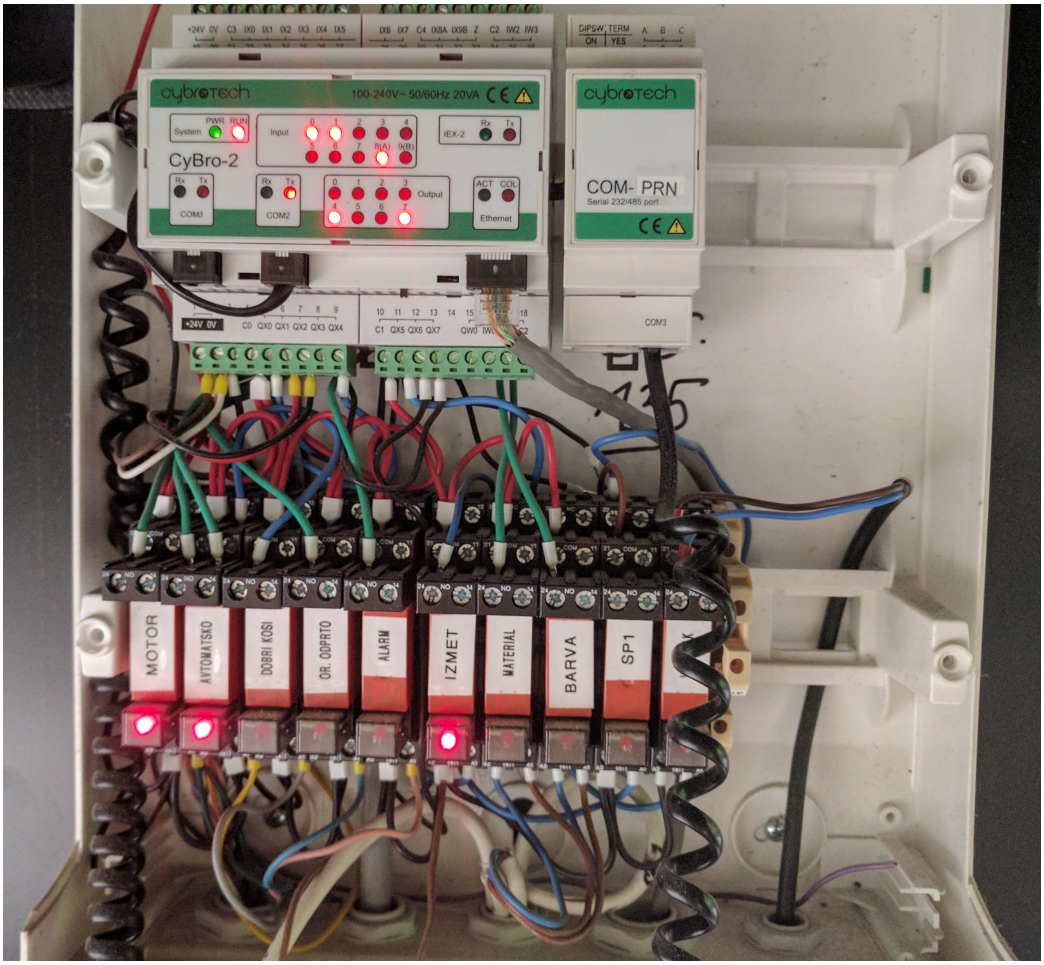
Načrtovanje novega sistema

4.1 Razlogi

Program, ki se trenutno izvaja na krmilnikih v proizvodnji in ga bomo nadomestili, je bil izdelan s strani zunanjih izvajalcev ob postavitvi proizvodnje. Čeprav deluje, vsebuje nekaj negativnih lastnosti, ki povzročajo težave pri procesu proizvodnje in pri spreminjanju le-tega. Ena večjih težav je, da je koda nerazumljiva, kar lahko pripišemo dejstvu, da program nima nobene dokumentacije, je praktično brez komentarjev in brez opisov spremenljivk. Poleg tega prvotno ni bil izdelan za proizvodni proces v obliki, kot se izvaja danes. Zaradi tega je bil program velikokrat dopisan in popravljen, kar pa ni bilo ustrezno dokumentirano. Programska koda je torej zaradi neustreznega vzdrževanja postajala še bolj neberljiva in nerazumljiva.
Te lastnosti povzročajo težave predvsem ko je prišlo do napak v delovanju ter ko so bile v teku menjave raznih sistemov, odvisnih od delovanja krmilnika, kot na primer pri menjavi proizvodnega informacijskega sistema. Izkazalo se je, da je program težko učinkovito nadgraditi ali popraviti.
Poleg opisanega, pa so bile zahtevane tudi spremembe v nekaterih algoritmih delovanja (na primer kasneje opisana nadgradnja tiskanja nalepk), s čimer se bo še dodatno optimiziral proces proizvodnje.

4.2 Krmilna omarica

Na sliki 4.1 vidimo primer krmilne omarice ob stroju. Zgoraj je pritrjen krmilnik, z razširitvenim modulom COM-PRN, spodaj pa so vgrajeni ločilni releji. Releji so namenjeni izmenjavi signalov med strojem in periferijo. V tabeli 4.1 je natančneje prikazana konfiguracija vhodov in izhodov krmilnika ter razširitvenega modula COM-PRN. Pri tem lahko poudarimo, da navodila krmilnika priporočajo, da se imena spremenljivk vhodov in izhodov krmilnika ne spreminjajo [4].



Slika 4.1: Krmilna omarica ob stroju.

Tabela 4.1: Uporaba vhodov in izhodov krmilnika in razširitvenega modula.

Naziv	**Tip**	**Uporaba**
cybro_ix00	Digitalni vhod krmilnika	Glavno stikalo stroja
cybro_ix01	Digitalni vhod krmilnika	Stikalo ročno/avtomatsko na stroju
cybro_ix02	Digitalni vhod krmilnika	Signal, ki pove ali je brizg kvaliteten
cybro_ix03	Digitalni vhod krmilnika	Signal, ki pove ali je orodje odprto
cybro_ix04	Digitalni vhod krmilnika	Signal, ki pove ali je stroj v alarmu
cybro_ix05	Digitalni vhod krmilnika	Ni uporabljen
cybro_ix06	Digitalni vhod krmilnika	Signal, ki pove ali je dozirni sistem v alarmu
cybro_ix07	Digitalni vhod krmilnika	Signal, ki pove ali je periferija v alarmu
cybro_ix08	Digitalni vhod krmilnika	Stikalo periferije
cybro_ix09	Digitalni vhod krmilnika	Tipka periferije
cybro_qx00	Digitalni izhod krmilnika	Ni uporabljen
cybro_qx01	Digitalni izhod krmilnika	Luč alarma
cybro_qx02	Digitalni izhod krmilnika	Hupa alarma
cybro_qx03	Digitalni izhod krmilnika	Ni uporabljen
cybro_qx04	Digitalni izhod krmilnika	Izmetna loputa
cybro_qx05	Digitalni izhod krmilnika	Vrtiljak s škatlami
cybro_qx06	Digitalni izhod krmilnika	Spustna loputa
cybro_qx07	Digitalni izhod krmilnika	Tekoči trak
COM2	Vrata RS-232 krmilnika	Terminal tehtnice
COM3	Vrata RS-232 razširitvenega modula COM-PRN	Tiskalnik

4.3 Programiranje na osnovi končnih avtomatov

Programiranje na osnovi končnih avtomatov (ang. automata-based programming) je ena izmed programerskih paradigem, pogosto uporabljena pri načrtovanju in izdelavi programov za krmilnike. Ideja je, da program ali del programa predstavimo v obliki končnega avtomata. Torej za program določimo vsa stanja, ki jih lahko zaseda, ter prehode med temi stanji.
V strukturiranem tekstu lahko končne avtomate opišemo s pomočjo stavka `case...of`. Na sliki 4.2 je predstavljen primer diagrama končnega avtomata ter realizacija v razvojnem okolju CyPro, s pomočjo programskega jezika strukturiran tekst. Primer predstavlja preprosto prižiganje luči. Luč je vezana na digitalni izhod `cybro_qx00` in se prižge ob pritisku na tipko, ki je vezana na digitalni vhod krmilnika `cybro_ix00`. Nato ostane prižgana 3 sekunde ter se samodejno ugasne. Algoritem lahko predstavimo z dvema stanjema:

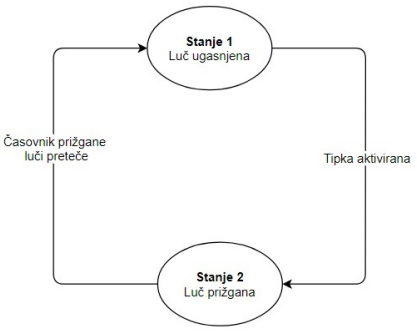
1. čakanje na zahtevo po prižiganju luči (pritisk tipke) – luč ugasnjena in
2. luč ostane prižgana 3 sekunde – luč prižgana.

Pri tem lahko definiramo tudi dva prehoda med stanji:

1. tipka aktivirana in
2. časovnik prižgane luči je potekel.

Tak način programiranja je pri krmilnikih še posebej primeren zaradi cikličnega izvajanja programa.
V našem primeru gre za sekvenčni proizvodni proces, ki je še posebej primeren za opisovanje s končnimi avtomati. Dodatni razlogi za delo s končnimi avtomati:

- Če uporabimo končne avtomate lahko v dokumentaciji vključimo diagrame delovanja celotnega sistema, v obliki diagramov stanj, z upo-




```
1 // AVTOMAT LUČI
2 case avtomatLuc of
3      // POČAKAJ NA AKTIVACIJO S TIPKO
4      1: if cybro_ix00 then
5            cybro_qx00 := 1;
6            avtomatLuc := 2;
7         end_if;
8      // DRŽI LUČ PRIŽGANO
9      2: casovnikPrizganaLuc := casovnikPrizganaLuc + fp(clock_100ms);
10        if casovnikPrizganaLuc > 3000 then
11           casovnikPrizganaLuc := 0;
12           cybro_qx00          := 0;
13           avtomatLuc          := 1;
14        end_if;
15 end_case;
```

Slika 4.2: Primer diagrama avtomata (levo) in njegova realizacija (desno).

rabo poenotenega jezika modeliranja (ang. unified modeling language – UML). Več o dokumentaciji bomo predstavili v poglavju 7.

- So lahki za razumevanje in omogočajo preproste nadgradnje.
- V primeru, da se v prihodnosti podjetje odloči za menjavo krmilnikov in/ali programskega jezika, nam bodo izdelani diagrami stanj omogočali lažji prenos sistema.

Poglavje 5

Algoritmi in opisi delovanja

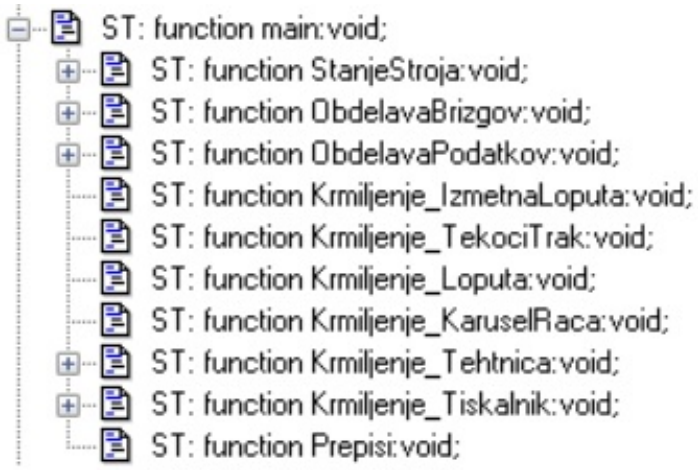
5.1 Struktura programa

Na sliki 5.1 je prikazana osnovna struktura izdelanega programa. V integriranem razvojnem okolju CyPro se ob kreiranju programa vedno ustvari funkcija `main`, ki služi kot glavna funkcija in je klicana ob vsakem ciklu krmilnika. V našem primeru vsebuje nastavitve spremenljivk, ki jih operater nastavi glede na lastnosti stroja na katerem teče program, ter klice ostalih funkcij.
Nekatere pomembnejše funkcije in dele programa bomo podrobneje razdelili in natančneje opisali v nadaljevanju, kar bomo naredili v treh razdelkih:

1. razne funkcije (`StanjeStroja` in `Prepisi`),
2. glavne kontrolne funkcije (`ObdelavaBrizgov` in `ObdelavaPodatkov`),
3. krmilne funkcije (funkcije, ki v imenu vsebujejo predpono `Krmiljenje_`).

5.2 Funkciji StanjeStroja in Prepisi

Funkcija `StanjeStroja` opravlja dve glavni nalogi: ugotavlja status stroja ter krmili luč in hupo alarma. Stroj lahko zaseda pet različnih stanj: avtomatsko delovanje, ročno delovanje, v alarmu, v stanju čakanja ter izklopljeno



Slika 5.1: Struktura izdelanega programa.

stanje. Krmilnik ustrezno prebere ta stanja ter jih zapiše v eno celoštevilsko vrednost, ki jo iz krmilnika nato prebere in izpiše proizvodni informacijski sistem, kar je prikazano na sliki 5.2.

```
 9 // ZBIRANJE STATUSA STROJA
10 statAvto    := not ix04_AlarmStroj and ix01_StrojAvtomatsko;
11 statCakanje := not ix04_AlarmStroj and ix01_StrojAvtomatsko and not qx04_StrojIzmetnaLoputa;
12 statRocno   := ix04_AlarmStroj or not ix01_StrojAvtomatsko;
13 statAlarm   := ix04_AlarmStroj or ix06_AlarmDozSistem or ix07_AlarmPeriferija;
14
15 // ZLAGANJE STATUSOV V ENO IZHODNO VREDNOST
16 outStatusStroja  := 1*statAvto + 4*statRocno + 8*statAlarm + 16*statCakanje + 128*ix00_StrojVklop;
```

Slika 5.2: Preračun statusa stroja.

Funkcija `Prepisi` je začasna funkcija, ki vse spremenljivke (relevantne za proizvodni informacijski sistem) prenese v spremenljivke, katerih imena so enaka kot v starem programu. Obratno stori za spremenljivke, v katere proizvodni informacijski sistem piše – prenese vrednosti spremenljivk s starim imenom v spremenljivke z novim imenom. S tem omogočimo, da je program kompatibilen z vsemi nadrejenimi sistemi, dokler ti ne bodo ustrezno posodobljeni.

5.3 Glavne kontrolne funkcije

Glavni kontrolni funkciji (`ObdelavaBrizgov` in `ObdelavaPodatkov`) opravljata vse glavne preračune ter na podlagi teh (in vhodnih signalov) prožita izvajanje večjega dela kasneje opisanih krmilnih funkcij.

5.3.1 Funkcija ObdelavaBrizgov

Funkcija `ObdelavaBrigov` se proži ob odpiranju orodja. Ko se orodje odpre, se najprej izvede zamik, tekom katerega ima stroj čas, da pošlje signal ali so kosi kvalitetno izdelani. Zamik je nastavljiv na začetku programa, v funkciji `main`, in je odvisen od lastnosti stroja. Po tem zamiku se proži krmiljenje izmetne lopute in na koncu štetje dobrih ali slabih kosov, glede na pozicijo izmetne lopute. Poleg tega funkcija ob odpiranju orodja izračuna še čas cikla stroja – čas od zadnjega odpiranja orodja. Čas cikla stroja je pomemben za proizvodni informacijski sistem, ki podatek prebere in uporabi za analizo proizvodnje.

5.3.2 Funkcija ObdelavaPodatkov

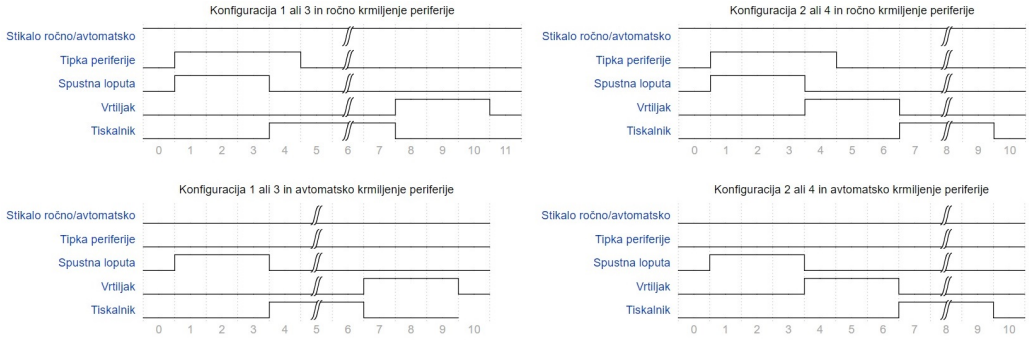
Funkcija `ObdelavaPodatkov` je glavna kontrolna funkcija. Izračuna vse ciljne in trenutne vrednosti ter preverja ali so te ciljne vrednosti dosežene. Glede na te ugotovitve in vhodne signale nato ustrezno proži krmiljenje spustne lopute, vrtiljaka ter tiskalnika. Glede na konfiguracijo sistema (glej poglavje 2.3) ter glede na pozicijo stikala za izbiro med ročnim in avtomatskim krmiljenjem periferije, imamo na tem mestu šest možnih kombinacij. Na sliki 5.3 so prikazani časovni diagrami za proženje spustne lopute, vrtiljaka in tiskalnika, glede na izbran način delovanja v primeru, da poleg odpiranja lopute pride tudi do polne škatle. V vseh primerih se najprej proži odpiranje lopute, kar dosežemo, ko se izpolni eden izmed naslednjih pogojev:

- dosežemo zadostno težo v tehtnici – pri konfiguraciji 1 ali 2 (tehtanje),

- dosežemo zadostno število brizgov – pri konfiguraciji 3 ali 4 (štetje kosov) ali

- sprožimo odpiranje ročno s pritiskom na dodeljeno tipko – katerakoli konfiguracija v ročnem načinu krmiljenja periferije.

Če smo v avtomatskem načinu in krmilnik ugotovi, da je škatla polna, potem proži krmiljenje vrtiljaka ter tiskalnika, v vrstnem redu, odvisnem od konfiguracije sistema. Če pa smo v ročnem načinu, potem se krmiljenje vrtiljaka ter tiskalnika proži, če držimo tipko dokler se loputa ne zapre.



Slika 5.3: Časovni diagrami proženja periferije.

5.4 Krmilne funkcije

5.4.1 Krmiljenje izmetne lopute

Krmiljenje izmetne lopute se proži v funkciji `ObdelavaBrizgov` po kratkem zamiku ob odpiranju orodja in pred štetjem kosov. Njeno delovanje je sledeče: Loputa se prestavi na izmet (izhod na krmilniku = 0), če stroj preide na ročni način, če je stroj v alarmu ali ob slabem brizgu. Nato algoritem izmetne lopute počaka na dva zaporedna dobra brizga. Pri prvem kose pošlje v izmet, pri drugem pa kose že preusmeri med dobre kose.

5.4.2 Krmiljenje tekočega traku

Krmiljenje tekočega traku ne potrebuje proženja in se izvaja stalno. Krmili se z enim pogojnim stavkom, v katerem so zbrani vsi pogoji za obratovanje traku.

5.4.3 Krmiljenje spustne lopute in vrtiljaka

Tako krmiljenje spuste lopute, kot tudi vrtiljaka se proži v funkciji `ObdelavaPodatkov`, ko so dosežene ustrezne vrednosti v loputi in/ali škatli. Obe delujeta na isti način: po prejetem signalu za krmiljenje krmilnik pošlje spustni loputi signal za odpiranje oziroma pošlje vrtiljaku signal za obračanje (izhod na krmilniku = 1) . Nato signal ostane aktiven toliko časa, kolikor ga nastavimo v funkciji `main`. Ta čas pri spustni loputi pomeni koliko časa je odprta, pri vrtiljaku pa z njim ustvarimo zamik pred proženjem nadaljnjih akcij (na primer tiskanje nalepke).

5.4.4 Krmiljenje tehtnice

Krmiljenje tehtnice se izvaja, če izberemo konfiguracijo 1 ali 2 (glej poglavje 2.3). Sestavljeno je iz dveh delov: inicializacija tehtnice in komunikacija s tehtnico. Ob zagonu stroja, ob prehodu iz ročnega krmiljenja periferija na avtomatsko ali ob menjavi konfiguracije je vedno zahtevano nekaj sekundno umirjanje tehtnice in za tem inicializacija terminala. Inicializacijo lahko uporabimo za nastavljanje raznih parametrov tehtnice preko krmilnika. Terminal, ki je obravnavan v tem delu (DPA 3T), ne potrebuje inicializacije, zato je ta korak preskočen. Funkcija je zgolj pripravljena za uporabo z drugo tehtnico, ki to potrebuje. Ko je inicializacija končana, se začne ciklična komunikacija med terminalom tehtnice in krmilnikom za branje vrednosti tehtnice. Terminal tehtnice za komunikacijo s krmilnikom uporablja serijsko povezavo s protokolom RS-232. Za serijsko komunikacijo moramo najprej na krmilniku nastaviti vmesnik RS-232 na prosto programirljiv način (ang. free programmable) in ustrezno nastaviti parametre komunikacije, kar storimo z uporabo

uporabniškega vmesnika okolja CyPro. Ti parametri so [16]:

- hitrost prenosa v bitih na sekundo (ang. baudrate) – v našem primeru 9600 bitov na sekundo,
- uporaba paritete (ang. parity bit) – preprosto nizkonivojsko preverjanje napak. V našem primeru je ne potrebujemo, zato izberemo možnost N,
- število bitov v enem paketu (ang. data bits) – terminal tehtnice uporablja 8-bitne znake ASCII, zato izberemo možnost 8,
- število končnih bitov (ang. stop bit) – krmilnik omogoča samo 1 končni bit.

V splošnem so lahko ti parametri drugače nastavljeni, vendar se morajo ujemati s parametri naprave, ki komunicira s krmilnikom.
Krmilnik za uporabo vmesnika RS-232 ponuja razne funkcije za pošiljanje, prejemanje in razčlenitev odgovora. Na sliki 5.4 je prikazan izsek iz programa za komunikacijo med terminalom tehtnice in krmilnikom, realiziran z uporabo nekaj izmed ponujenih funkcij. Terminalu tehtnice moramo najprej poslati ukaz `$DA00?\r`, s katerim zahtevamo prebrano vrednost. Nato počakamo na odgovor, ki ga pričakujemo v obliki `$00 xxxxxxx\r`, kjer `x` predstavlja prebrano vrednost.

5.4.5 Krmiljenje tiskalnika

Ob polni škatli se pred ali po obračanju vrtiljaka natisne nalepka. Tiskalnik za komunikacijo s krmilnikom uporablja serijsko povezavo s protokolom RS-232 preko razširitvenega modula COM-PRN. Pred komunikacijo moramo ustrezno nastaviti parametre serijske komunikacije, kot je to že opisano v poglavju 5.4.4.
Krmiljenje poteka v treh delih. Najprej se vsi podatki potrebni za tisk nalepke pretvorijo v zapis ASCII ter zapišejo v tabelo numeričnih vrednosti (en

```
16      // PREBERI VREDNOST VSAKIH 100ms
17      0: if fp(clock_100ms) then
18           // PRIPRAVI UKAZ ZA BRANJE VREDNOSTI
19           dprns(0,0,0,'$DA00?\r');
20           // POŠLJI 7 ZNAKOV
21           tx_start(7);
22           // ZAČNI SPREJEMANJE ODGOVORA, KI SE ZAČNE Z $ IN KONČA Z \r, TIMEOUT PO 1s
23           rx_start('$','\r',0,1000,0);
24           avtomatTehtnica_kom := 1;
25         end_if;
26      // POČAKAJ NA ODGOVOR
27      1: if not rx_active() then
28           avtomatTehtnica_kom := 2;
29         end_if;
30      // OBDELAJ PREJET ODGOVOR
31      2: vrednostTehtnica := rx_strtoi(3);
```

Slika 5.4: Izsek iz programa za komunikacijo s terminalom tehtnice.

znak v eno polje). V drugem koraku se izvaja pošiljanje podatkov (tabele znakov ASCII) na tiskalnik, na koncu pa se izvede še zamik, ki zagotovi, da se nalepka uspešno natisne pred izvajanjem nadaljnjih akcij (na primer obračanje vrtiljaka s škatlami). Več o podatkih, ki se morajo poslati na tiskalnik, bomo opisali v poglavju 6.

Delo z vmesnikom RS-232 na razširitvenem modulu COM-PRN je drugačno kot delo z vmesnikom RS-232 na krmilniku. Na razširitvenem modulu nimamo na voljo funkcij, ampak vmesnik krmilimo z nastavljanjem in branjem integriranih spremenljivk. Na sliki 5.5 je prikazan izsek iz kode za pošiljanje vsebine tabele `inPodatkiZaTiskalnik` na tiskalnik. V tem pri-

```
11 // AVTOMAT POŠILJANJA
12 case avtomatPosiljanjeNaTiskalnik of
13      // INICIALIZACIJA
14      0: indeks := 0;
15         avtomatPosiljanjeNaTiskalnik := 1;
16      // POŠILJANJE NA TISKALNIK
17      1: if prn00_byte_count = 0 then
18           //PRIPRAVI NASLEDNJI ZNAK
19           prn00_byte_data := inPodatkiZaTiskalnik[indeks];
20           //ZAPIŠI ZNAK NA MEDPOMNILNIK (BUFFER)
21           prn00_byte_req  := 1;
22           //POŠLJI VSEBINO MEDPOMNILNIKA NA TISKALNIK
23           prn00_print_req := 1;
24           indeks := indeks + 1;
25         end_if;
26         if indeks > inDolzinaPodatkov then
27           avtomatPosiljanjeNaTiskalnik := 0;
28         end_if;
29 end_case;
```

Slika 5.5: Izsek iz programa za komunikacijo s tiskalnikom.

meru se pošiljanje izvaja znak po znaku, alternativno pa bi lahko najprej napolnili medpomnilnik in nato posredovali vse podatke tiskalniku. Po naših testih se slednja možnost izkaže za enako hitro, saj ozko grlo predstavlja čas cikla krmilnika in ne čas pošiljanja na tiskalnik.

Poglavje 6

Posodobitev tiskanja nalepk

V tem poglavju bomo opisali nadgradnjo tiskanja nalepk ob polni škatli. Najprej bomo predstavili delovanje starega sistema in zakaj je bila potrebna nadgradnja, nato pa še idejo in realizacijo novega sistema.

6.1 Star sistem

Sistem pred posodobitvijo je dovoljeval tiskanje ene pred definirane nalepke. Tiskalnik je imel v spominu shranjeno eno predlogo, za katero je krmilnik poslal vse potrebne podatke. Ker pa so kupci vedno bolj pogosto zahtevali na škatlah drugačne nalepke, je bila potrebna nadgradnja sistema za tiskanje različnih nalepk, glede na podatke na delovnem nalogu. V nadaljevanju bomo podrobneje opisali delovanje starega sistema, kar bomo storili v dveh delih. Najprej bomo opisali tiskalnik ter predloge nalepk, nato pa še vlogo krmilnika v tem sistemu.

6.1.1 Tiskalnik in predloge nalepk

Predloge nalepk za tiskalnik Zebra GX420d lahko ustvarimo z uporabo enega izmed dveh programskih jezikov za opis strani [20]: programski jezik Eltron (ang. Eltron programming language – EPL) ali novejši programski jezik Zebra (ang. Zebra programming language – ZPL). Star sistem uporablja EPL,

in ker zadostuje vsem potrebam, bo uporabljen tudi v novem sistemu.
S pomočjo ukazov EPL lahko opišemo izris besedila, nekaterih grafičnih elementov ter različnih črtnih kod. Poleg tega lahko upravljamo z nastavitvami tiskalnika ter definiramo spremenljivke, katerih vrednosti določimo pred tiskom in tako omogočimo spreminjanje izgleda nalepke.

6.1.2 Krmilnik

Krmilnik je ob zahtevi za tiskanje (torej ob polni škatli) zbral vse potrebne vrednosti spremenljivk (kot sta na primer datum izdelave in teža škatle) in ukaz za izbiro shranjene predloge nalepke ter ukaz za tiskanje. Vse podatke je pretvoril v zapis ASCII ter jih zapisal v tabelo (ang. array) numeričnih vrednosti. Nato je vsebino tabele znak po znaku poslal na tiskalnik, ki je ukaze in spremenljivke interpretiral ter ustrezno reagiral – stiskal nalepko. Na sliki 6.1 je prikazan primer kode EPL, primer pripadajočega programa v krmilniku (sestava tabele, ki se pošlje na tiskalnik, ter koda za pošiljanje na tiskalnik) ter stiskana nalepka.

6.2 Nov sistem

6.2.1 Ideja

V kolikor bi bil program v krmilniku krajši (bi zasedel manj prostora) in bi se nove nalepke redko pojavljale, potem bi lahko na tiskalnik naložili vse različne predloge nalepk ter v programu krmilnika omogočili različne načine polnjenja tabele s podatki za tiskalnik, glede na zahtevano nalepko. Tak način delovanja bi zahteval, da ob vsaki novi nalepki na vse tiskalnike naložimo novo dodatno predlogo nalepke ter da dopolnimo program v krmilniku (omogočimo dodaten način polnjena tabele s podatki za tiskalnik). V našem primeru opisana rešitev ni primerna, saj na krmilniku nimamo več dovolj prostora za takšno razširitev programske kode ter ker takšen sistem ni dovolj dinamičen in zahteva preveč dela ob vsaki novi nalepki.

Program na krmilniku

New Program - ST: function main:void;

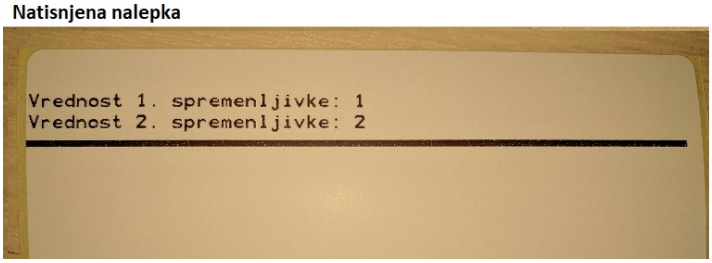
```
// AVTOMAT POŠILJANJA
case avtomatPosiljanjeNaTiskalnik of
    // PROŽI TISKANJE
     0: if fp(cybro_ix00) then
          avtomatPosiljanjeNaTiskalnik := 10;
        end_if;
    // PRIPRAVA PODATKOV
    10: indeks := 0;
        dolzinaPodatkov := 14;
        // IZBERI ETIKETO 5 (FR"5") TER ZAHTEVAJ SPREMENLJIVKE (?)
        podatkiZaTiskalnik[0]  := 70;  //F
        podatkiZaTiskalnik[1]  := 82;  //R
        podatkiZaTiskalnik[2]  := 34;  //"
        podatkiZaTiskalnik[3]  := 53;  //5
        podatkiZaTiskalnik[4]  := 34;  //"
        podatkiZaTiskalnik[5]  := 10;  //\n
        podatkiZaTiskalnik[6]  := 63;  //?
        podatkiZaTiskalnik[7]  := 10;  //\n
        // VREDNOST PRVE SPREMENLJIVKE
        podatkiZaTiskalnik[8]  := 49;  //1
        podatkiZaTiskalnik[9]  := 10;  //\n
        // VREDNOST DRUGE SPREMENLJIVKE
        podatkiZaTiskalnik[10] := 50;  //2
        podatkiZaTiskalnik[11] := 10;  //\n
        // NATISNI ENO ETIKETO
        podatkiZaTiskalnik[12] := 80;  //P
        podatkiZaTiskalnik[13] := 49;  //1
        podatkiZaTiskalnik[14] := 10;  //\n
        avtomatPosiljanjeNaTiskalnik := 20;
    // POŠILJANJE NA TISKALNIK (ZNAK PO ZNAKU)
    20: if prn00_byte_count = 0 then
          prn00_byte_data := podatkiZaTiskalnik[indeks];
          prn00_byte_req  := 1;
          prn00_print_req := 1;
          indeks := indeks + 1;
        end_if;
        // POŠILJANJE ZAKLJUČENO
        if indeks > dolzinaPodatkov then
          avtomatPosiljanjeNaTiskalnik := 0;
        else
        end_if;
end_case;
```

Koda predloge nalepke (EPL)

untitled1 *

```
N
OC

FK"5"
FK"5"

FS"5"

V00,10,N,"Prva spremenljivka"
V01,10,N,"Druga spremenljivka"

LE0,80,800,8

A25,25,0,3,1,1,N,"Vrednost 1. spremenljivke: "V00
A25,50,0,3,1,1,N,"Vrednost 2. spremenljivke: "V01

FE
```

Natisnjena nalepka



Slika 6.1: Primer kode EPL, pripadajočega programa v krmilniku ter stiskane nalepke.

Osnovna ideja novega sistema je, da proizvodni informacijski sistem ob prijavi delovnega naloga poleg podatkov naloga na krmilnik avtomatsko naloži tudi predlogo nalepke, ki ustreza izbranemu delovnemu nalogu. Krmilnik nato posreduje prejeto predlogo nalepke na tiskalnik. Poleg tega se na krmilnik prenesejo tudi vse dodatne vrednosti, ki jih krmilnik v osnovi ne hrani in so potrebne za izpis na željeno nalepko. Za prikaz delovanja ideje bomo izdelali prototipno spletno aplikacijo, ki jo bomo uporabili za pošiljanje predlog nalepk in dodatnih vrednosti na krmilnik.

V nadaljevanju bomo opisali izdelavo predlog nalepk, ki bodo ustrezale no-

vemu sistemu. Opisali bomo tudi spremembe na krmilniku za delovanje z novim sistemom ter izdelavo prototipne spletne aplikacije.

6.2.2 Tiskalnik in predloge nalepk

Vse predloge nalepk morajo imeti definirane iste spremenljivke. Torej moramo pred izdelavo predvideti največje možno število spremenljivk na eni nalepki ter okviren format teh spremenljivk. V delu kode EPL, kjer opišemo izris teh spremenljivk, pa nato uporabimo le tiste spremenljivke, ki jih potrebujemo. S tem dosežemo, da je lahko program na krmilniku vedno isti – pošilja na tiskalnik vedno iste spremenljivke. Med spremenljivkami so lahko najprej tiste, ki so se tiskale na nalepke pred posodobitvijo, oziroma se hranijo ali izračunajo na krmilniku, ter nekaj novih generičnih spremenljivk za vse dodatne podatke, ki se morajo natisniti na nove nalepke. Več o generičnih spremenljivkah bomo opisali v nadaljevanju.

6.2.3 Krmilnik

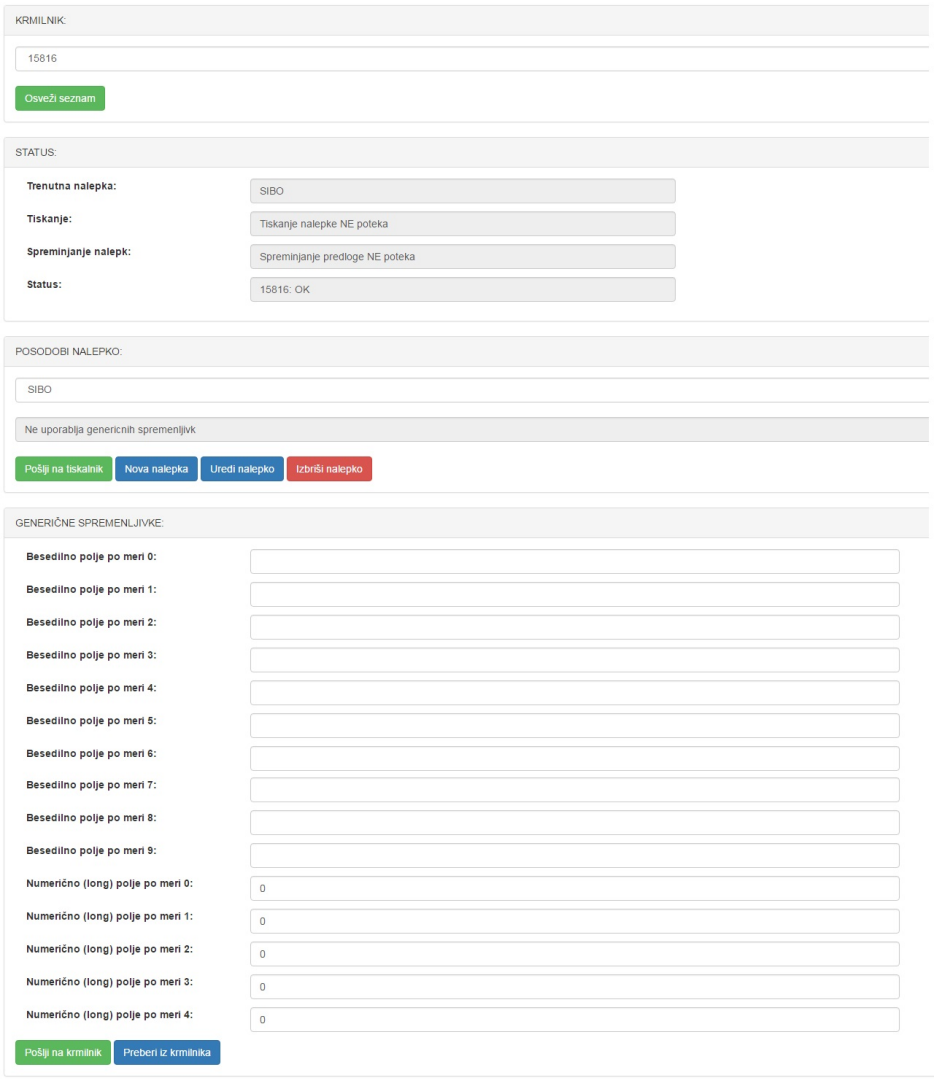
Ko krmilnik prejme celotno kodo EPL s strani proizvodnega informacijskega sistema, oziroma v našem primeru iz prototipne spletne aplikacije, mora te podatke posredovati na tiskalnik. Za posredovanje teh podatkov smo uporabili enak sistem kot za pošiljanje podatkov za tiskanje nalepke.
Poleg tega moramo na krmilniku pripraviti generične spremenljivke za dodatne vrednosti na novih nalepkah. V našem primeru smo se odločili za 10 tabel numeričnih vrednosti dolžine 60, ki služijo za tiskanje besedilnih vrednosti ali decimalnih števil (vsako polje tabele bo vsebovalo en znak ASCII) ter 5 spremenljivk tipa `long` za tiskanje celih števil.
Z opisanim smo dosegli, da krmilnik uporabimo kot vmesnik za pošiljanje kode nalepke na tiskalnik ter da lahko z ustrezno predvidenimi generičnimi spremenljivkami uporabimo vedno isti program, ne glede na nalepko, ki se mora natisniti. Pri tem je potrebno poudariti še, da pošiljanje na tiskalnik z uporabo novega sistema poteka dlje, saj moramo zaradi generičnih

spremenljivk poslati na tiskalnik več podatkov. V našem primeru smo pred posodobitvijo na tiskalnik pošiljali 202 znaka, kar je trajalo približno od 3 do 5 sekund, po posodobitvi pa 873 znakov, kar traja od 15 do 20 sekund. Ker pa se škatle polnijo dlje kot poteka tiskanje, nam v našem primeru to ne predstavlja težave.

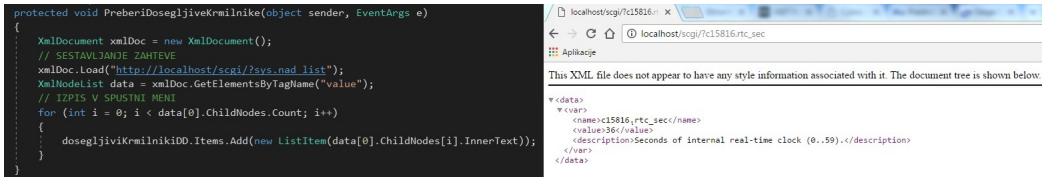
6.2.4 Prototipna spletna aplikacija

Namen spletne aplikacije je shranjevanje predlog nalepk v obliki kode EPL ter pošiljanje teh predlog in vrednosti za generične spremenljivke na krmilnik. Aplikacijo smo razvili z uporabo orodja Microsoft Visual Studio Enterprise 2017. Visual Studio je integrirano razvojno okolje, ki omogoča razvoj različnih vrst aplikacij (spletnih, mobilnih in namiznih), spletnih storitev in podatkovnih baz [18]. Za razvoj aplikacije smo uporabili programski jezik C#, za shranjevanje kode EPL smo uporabili podatkovno bazo Microsoft SQL Server, za mrežno komunikacijo s krmilnikom pa orodje CyBro HTTP server. Za boljši izgled smo poleg jezikov HTML in CSS uporabili tudi ogrodje Bootstrap. Bootstrap je ogrodje za razvoj čelnega dela (ang. front end) spletnih aplikacij. Ponuja predloge za izpopolnjevanje izgleda spletnih aplikacij ter olajša razvoj odzivnih spletnih aplikacij [2].
Na sliki 6.2 je prikazan uporabniški vmesnik izdelanega prototipa. Prvi razdelek (KRMILNIK) nam poda spustni seznam z vsemi dosegljivimi krmilniki. Ob izbiri enega izmed njih, se v naslednjem razdelku (STATUS) prikažejo podatki o statusu krmilnika (katera nalepka je naložena, ali poteka tiskanje, ali poteka spreminjanje nalepke in ali je krmilnik sploh kompatibilen s tem sistemom). Potem imamo na voljo razdelek POSODOBI NALEPKO, ki vsebuje spustni seznam vseh prebranih nalepk iz podatkovne baze. Pod spustnim seznamom je še polje opomb ter gumbi za manipuliranje z nalepkami. Še zadnji razdelek je razdelek GENERIČNE SPREMENLJIVKE. Omogoča branje in pisanje generičnih spremenljivk na izbrani krmilnik in iz njega.
Orodje CyBro HTTP server omogoča nastavljanje in branje spremenljivk krmilnika. Spletna aplikacija torej ob vsaki zahtevi za komunikacijo sestavi

in pošlje ustrezno zahtevo HTTP. Krmilnik nato vrne datoteko XML, katero spletna aplikacija ustrezno razčleni. S programskim jezikom C# lahko pošiljanje zahteve in prejemanje odgovora realiziramo z uporabo imenskega prostora `System.Xml`. Na sliki 6.3 na levi vidimo sestavljanja zahteve ter razčlenjevanje odgovora za primer iskanja vseh dosegljivih krmilnikov. Na isti sliki na desni pa vidimo primer branja spremenljivke z uporabo spletnega brskalnika.



Slika 6.2: Uporabniški vmesnik spletne aplikacije.



Slika 6.3: Primer pošiljanja zahteve in razčlenjevanja odgovora (levo) in primer branja spremenljivke v brskalniku (desno).

Poglavje 7

Dokumentacija

7.1 Osnovna dokumentacija

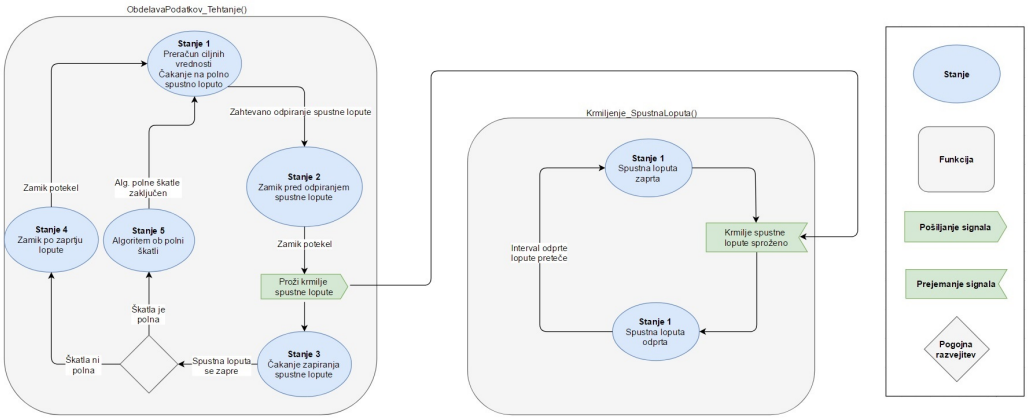
Da je programska koda razumljiva in omogoča učinkovite nadgradnje, potrebujemo ustrezno dokumentacijo. Najbolj osnoven način dokumentiranja kode so ustrezni komentarji ter opisi uporabljenih spremenljivk. V našem primeru smo poleg tega dodali vsaki funkciji tudi glavo, ki vsebuje kratek opis funkcije, morebitne dodatne informacije ter zgodovino sprememb. Primer glave je prikazan na sliki 7.1.

```
/*************************************************************
   Krmiljenje izmetne lopute

   Info: Ob prehodu na dobre brizge je prvi dober brizg še
         poslan v izmet. Drug dober brizg sproži preklop
         izmetne lopute.
         Avtomat se izvaja samo ob zahtevi (ob odprtem orodju)

   Zgodovina sprememb:
   - Ustvarjeno marec 2017
*************************************************************/
```

Slika 7.1: Primer glave funkcije.

7.2 Diagrami

Poleg opisanega smo podali tudi grafično dokumentacijo v obliki diagramov. Za izdelavo teh smo uporabili poenoten jezik modeliranja (ang. unified modeling language – UML). UML ponuja mehanizme za vizualiziranje, načrtovanje in dokumentiranje programskih sistemov [5]. Čeprav je objektno orientiran, lahko nekatere elemente uporabimo tudi pri dokumentiranju programske kode krmilnikov. Na najvišjem nivoju lahko s pomočjo razrednih diagramov predstavimo strukturo programa, torej funkcije in odvisnosti med njimi. Znotraj vsake funkcije pa lahko podrobneje predstavimo algoritme delovanja s pomočjo diagramov stanj (za dele kode, ki so izdelani na osnovi končnih avtomatov) ter diagramov aktivnosti (za dele kode, ki niso izdelani na osnovi končnih avtomatov) [19]. V našem primeru struktura programa ni tako obsežna da bi potrebovala svoj diagram. Zato smo strukturo prikazali na diagramih stanj in diagramih aktivnosti, kot to vidimo na izseku diagrama na sliki 7.2. Prikazano je delovanje dveh avtomatov, njuna odvisnost ter kateri funkciji pripadata.



Slika 7.2: Izsek iz diagrama programa.

Ob opisanem smo podali tudi dokumentacijo v obliki seznama vseh spremenljivk s pripadajočimi podatki (ime, tip, atributi in opis) ter v obliki seznama vseh uporabljenih vhodov in izhodov, kar smo že prikazali v tabeli 4.1.

Poglavje 8

Sklepne ugotovitve

V tem diplomskem delu smo predstavili načrtovanje in izdelavo novega programa za avtomatizacijo perifernih naprav v procesu proizvodnje. Tekom dela smo predstavili proces proizvodnje, delo s krmilniki, uporabljen krmilnik CyBro-2 ter standard IEC 61131-3. Pri načrtovanju smo predstavili glavne razloge za menjavo programa. Poleg tega smo predstavili tudi programersko paradigmo programiranja na osnovi končnih avtomatov ter njeno uporabnost pri pisanju programov za krmilnike. Uporaba te paradigme nam je omogočala lahko razumljiv, grafičen način dokumentiranja programa ter nam bo v prihodnosti olajšala nadgradnje programa. Na kratko smo opisali tudi nekaj glavnih algoritmov delovanja ter serijsko komunikacijo med krmilnikom in terminalom tehtnice oziroma tiskalnikom. Pomemben del predstavlja tudi nadgradnja sistema za tiskanje nalepk. S tem smo drastično zmanjšali dodatno ročno delo, ki je bilo potrebno pred posodobitvijo. Učinkovitost novega sistema za tiskanje nalepk je sicer do neke mere odvisna od tega, kako dobro predvidimo maksimalno število različnih spremenljivk ter format teh spremenljivk. Če pa spremenljivke ustrezno predvidimo, potem sistem omogoča visoko stopnjo dinamičnosti.

V času testiranja v učinkovitosti delovanja med novim in starim sistemom nismo opazili večje razlike, z izjemo tiskanja nalepk. V prihodnosti pa pričakujemo bolj tekoče obratovanje procesa proizvodnje ter veliko lažje nad-

gradnje katerihkoli sistemov, odvisnih od delovanja krmilnika, kar je bil tudi glavni razlog za menjavo sistema.
Največjo pomanjkljivost sistema in posledično možnost za nadgradnjo predstavljajo nekatere enosmerne komunikacije med krmilnikom in perifernimi napravami, kar v nekaterih primerih lahko privede do napak. Primera take komunikacije:

- Komunikacija krmilnik – tiskalnik. Razširitveni modul COM-PRN omogoča samo pošiljanje podatkov na tiskalnik, ne pa tudi sprejemanja. Zato moramo predlogo nalepke naložiti ob vsaki prijavi delovnega naloga.

- Komunikacija krmilnik – vrtiljak. Kot smo omenili, se vrtiljak začne vrteti ob prehodu izhoda krmilnika iz 0 na 1. Nato se vrtiljak krmili sam na podlagi lastnih senzorjev. Krmilnik nima informacije o tem ali se je vrtiljak dejansko obrnil ali ne, kar lahko privede do napak.

Da bi rešili problem enosmerne komunikacije, bi morali sistem razširiti z dodatno strojno opremo, za kar se podjetje ni odločilo.
Testiranje je bilo do sedaj izvedeno samo na enem stroju. V prihodnosti bo potrebno izpeljati še testiranje v večjem obsegu, torej na različnih strojih v proizvodnji ter tekom daljšega obdobja.

Literatura

[1] W. Bolton. *Programmable Logic Controllers*. Elsevier Science, 2015.

[2] Bootstrap Get Started. Dosegljivo: `https://www.w3schools.com/bootstrap/bootstrap_get_started.asp`. [Dostopano 27. 6. 2017].

[3] CyBro Hardware Manual version 1.0 rev. 2. Dosegljivo: `http://www.cybrotech.com/wp-content/uploads/2016/11/CyBro-Hardware-Manual-v1.0-rev-2.pdf`, 2016. [Dostopano 1. 6. 2017].

[4] CyPro User Manual rev. 35. Dosegljivo: `http://www.cybrotech.com/wp-content/uploads/2016/11/CyPro-User-Manual-v35.pdf`, 2016. [Dostopano 1. 6. 2017].

[5] H.E. Eriksson, M. Penker, B. Lyons, and D. Fado. *UML 2 Toolkit*. OMG. Wiley, 2003.

[6] V. Goodship and Arburg (Firm). *Practical Guide to Injection Moulding*. Rapra Technology, 2004.

[7] D.H. Hanssen. *Programmable Logic Controllers: A Practical Approach to IEC 61131-3 Using CoDeSys*. Wiley, 2015.

[8] Hard-Wired vs. Programmable Logic. Dosegljivo: `http://www.pcguide.com/ref/cpu/rootsProgrammable-c.html`. [Dostopano 15. 6. 2017].

[9] IEC 61131 Standards. Dosegljivo: `http://www.plcopen.org/pages/tc1_standards/`. [Dostopano: 1. 6. 2017].

[10] Tim Jager. IEC 61131-3 Choosing a Programming Language. Dosegljivo: https://www.dmcinfo.com/latest-thinking/blog/id/111/iec-61131-3-choosing-a-programming-language, 2009. [Dostopano 15. 6. 2017].

[11] K.H. John and M. Tiegelkamp. *IEC 61131–3: Programming Industrial Automation Systems: Concepts and Programming Languages, Requirements for Programming Systems, Aids to Decision-Making Tools*. Springer, Berlin Heidelberg, 2013.

[12] LIBELA ELSI, Merilne naprave. Dosegljivo: http://www.libela-elsi.si/merilne_naprave.htm. [Dostopano: 31. 5. 2017].

[13] J. Mallon and J.M. Mallon. *Advances in Automation for Plastics Injection Moulding*. RAPRA review reports: RAPRA Technology Limited. Rapra Technology, 2001.

[14] F. Petruzella. *Programmable Logic Controllers*. McGraw-Hill Education, 2016.

[15] Products – Cybrotech. Dosegljivo: http://www.cybrotech.com/product/. [Dostopano 15. 6. 2017].

[16] Serial Communication. Dosegljivo: https://learn.sparkfun.com/tutorials/serial-communication/rules-of-serial. [Dostopano 13. 6. 2017].

[17] SIBO G., poslanstvo. Dosegljivo: https://www.sibo-group.eu/o-nas/poslanstvo/. [Dostopano: 15. 5. 2017].

[18] Visual Studio. Dosegljivo: https://sl.wikipedia.org/wiki/Visual_Studio. [Dostopano 27. 6. 2017].

[19] Roland Wagner. UML in controller programming. Dosegljivo: http://www.sks.fi/www/images/aud_UML_en.pdf/$FILE/aud_UML_en.pdf. [Dostopano 29. 6. 2017].

[20] Zebra GX420d/GX430d Desktop Thermal Printer User Guide. Dosegljivo: https://www.zebra.com/content/dam/zebra/manuals/en-us/printer/gx420d-gx430d-ug-en.pdf, 2016. [Dostopano 22. 6. 2017].