UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Anže Leban

Telegrami za komunikacijo med napravami v industrijskih omrežjih

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU

Mentor: izr. prof. dr. Uroš Lotrič

Ljubljana, 2013

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01928/2013
Datum: 12.04.2013

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

Kandidat: **ANŽE LEBAN**

Naslov: **TELEGRAMI ZA KOMUNIKACIJO MED NAPRAVAMI V INDUSTRIJSKIH OMREŽJIH**

TELEGRAMS FOR COMMUNICATION BETWEEN DEVICES IN INDUSTRIAL NETWORKS

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Omrežja, zasnovana na tehnologiji Ethernet vedno bolj prodirajo tudi v industrijska okolja. Zmogljivejši industrijski krmilniki imajo danes že vgrajene vmesnike za priklop na omrežje Ethernet. To omrežje zato lahko uporabimo tudi za komunikacijo med napravami v industrijskih omrežjih. Ker standardiziranih vmesnikov za komunikacijo še ni, predlagajte in izdelajte svojega. Programske vmesnike pripravite za spekter najbolj uporabljanih krmilnikov.

Mentor:

izr. prof. dr. Uroš Lotrič

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Nikolaj Zimic

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Anže Leban, z vpisno številko 63030312, sem avtor diplomskega dela z naslovom:

Telegrami za komunikacijo med napravami v industrijskih omrežjih.

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 21. junija 2013 Podpis avtorja:

Zahvala

Zahvaljujem se podjetju Iskra Impuls, ki mi je omogočilo izvedbo diplomskega dela. Za pomoč in nasvete sem hvaležen vsem zaposlenim.

Zahvaljujem se mentorju, izr. prof. dr. Urošu Lotriču, za strokovno pomoč, svetovanje in spodbudo.

Zahvaliti se želim svojim staršem, ki so mi študij omogočili in me spodbujali ter moji družini za potrpežljivost.

Povzetek

Razvoj informacijskih tehnologij je pustil svoj vpliv tudi v industrijskih omrežjih, saj se za komunikacijo med programirljivimi logičnimi krmilniki in višjimi sistemi vodenja vedno bolj uveljavlja komunikacija na vodilu Ethernet, po protokolu TCP/IP. V okviru diplomske naloge smo razvili programsko kodo, ki omogoča komunikacijo med krmilnikom in višjenivojskim sistemom po vodilu Ethernet. Posvetili smo se delu na strani krmilnika.

V okviru rešitve je bilo potrebno določiti tudi zgradbo sporočil – telegramov, ki si jih komunikacijska partnerja pošiljata. Telegrami so sestavljeni iz dveh delov, kontrolnega in podatkovnega. Za nemoteno delovanje sistema je ključno, da se telegrami med komunikacijo ne izgubljajo. Programsko kodo smo razvili za različne tipe strojne opreme, uporabili smo dva Siemensova krmilnika; krmilnik družine Simatic S7-300 in krmilnik Simotion D425. Uporabili pa smo tudi modul CP 343-1 Lean, ki je razširitveni modul namenjen krmilnikom družine S7-300.

Programska koda je bila razvita za uporabo v avtomatskih visokoregalnih skladiščih, za komunikacijo med krmilniki dvigal in višjenivojskim sistemom.

Ključne besede:
avtomatizacija, visokoregalno skladišče, telegrami, procesno vodilo.

Abstract

The development of information technology has also made its impact on the industrial networks, as the TCP/IP Ethernet bus communication is ever more frequently used for communicating between programmable logic controllers and high-level systems. In this thesis we have developed a program code that enables communication between the controller and the higher-level system on Ethernet bus. Our focus was on the controller side.

To complete the task, it was necessary to determine the structure of the message - telegram, that you send between communication partners. Telegrams are build of two parts, control and data. Key for the system smooth functioning is that the telegrams are not lost during communication path. The program code was developed for different types of hardware, we used two Siemens controllers, controller Simatic S7-300 and Simotion D425. We also used the module CP 343-1 Lean which is an expansion module for the controller S7-300.

The program code was developed for use in automated high-bay warehouse for communication between crane controllers and higer-level systems.

Key words:
automation, high-bay warehouse, telegrams, communication.

1 Uvod

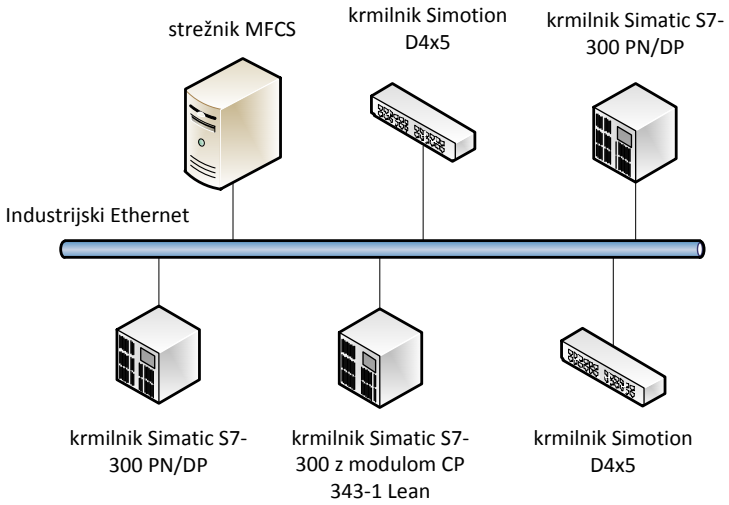
Razvoj informacijskih tehnologij je pustil svoj pečat tudi v industrijskih okoljih, zato je potreba po povezovanju med programirljivimi logičnimi krmilniki (v nadaljevanju krmilniki) in višjimi sistemi vodenja vedno bolj prisotna. Krmilnike lahko povezujemo z višjimi sistemi vodenja na več načinov. Lahko se odločimo za izbiro namenskih razvojnih orodij s katerim izdelujemo poljubne nadzorne sisteme. Taka orodja imajo že vnaprej pripravljene gradnike, ki nam olajšajo delo (vodenje dnevnika napak, vizualizacija elementov...), vendar moramo upoštevati omejitve in zahteve razvijalca ter plačati licenčnino. Lahko izdelamo tudi lasten sistem vodenja z uporabo programerskih razvojnih okolji, kot je na primer Visual Studio, v katerem uporabimo programske jezike C#, C++, VB in druge. Ko razvijamo lastno aplikacijo, potrebujemo sistem za izmenjavo podatkov med aplikacijo in krmilnikom. Uporabimo lahko različne plačljive ali prosto dostopne knjižnice, standard OPC, ki je industrijsko podprt in standardiziran (OPC Fundation) [5], vtičnike za razvojna okolja ali pa komuniciramo prek mrežnih vtičnikov, ko za komunikacijo uporabljamo vodilo Ethernet.

V okviru diplomskega dela smo se ukvarjali s komunikacijo med krmilnikom in višje nivojskim sistemom v visokoregalnem skladišču, posvetili smo se delu na strani krmilnika. V skladišču imamo lahko več dvigal, ki jih krmilimo vsakega s svojim krmilnikom in nadzorni sistem vodenja, ki ga imenujemo strežnik MFCS (ang. Material Flow Control System). Strežnik MFCS nadzira in vodi delovanje vseh krmilnikov v omrežju. Primer sheme omrežja je prikazan na sliki 1.

Vodilo Ethernet v industrijskih okoljih izpodriva ostala vodila, zato se je pojavila želja po komunikaciji med krmilnikom in strežnikom po protokolu TCP/IP na vodilu Ethernet, prek mrežnih vtičnikov. Programsko kodo za komunikacijo smo želeli zasnovati sami, saj smo želeli biti programsko neodvisni in brez vmesnega sistema, kot je na primer strežnik OPC. S tem smo dosegli platformno neodvisnost, nadzor nad prenosom podatkov in neodvisnost od uveljavljenih razvijalcev, izognili smo se tudi plačilu licenčnine.

V okviru diplomske naloge smo uporabili tri različne tipe krmilne strojne opreme, to so krmilnik serije Siemens Simatic S7-300, modul Siemens Simatic CP 343-1 Lean in krmilnik Siemens Simotion D425. Trudili smo se, da bi bila programska koda enotna na vseh tipih

strojne opreme, vendar je kljub temu prišlo do razlik v kodi. Do razlik je prihajalo predvsem zaradi različnih vgrajenih funkcij, ki se uporabljajo za komunikacijo po vodilu Ethernet.



Slika 1: Shema omrežja.

Za izmenjavo informacij in ukazov med krmilnikom in strežnikom MFCS smo uporabili sporočila, ki jih imenujemo telegrami. V okviru diplomskega dela smo morali določiti zgradbo telegramov. Telegrami so sestavljeni iz kontrolnega dela, ki mu pravimo glava, in podatkovnega dela, ki mu pravimo telo. Glava telegrama je pri vseh tipih telegramov enake dolžine, medtem ko je dolžina telesa odvisna od tipa telegrama.

Za seznanitev s področjem industrijskih komunikacij smo v naslednjem poglavju umestili industrijske komunikacije v plasti. Opisali smo tudi komunikacijo med krmilnikom in strežnikom MFCS po vodilu Ethernet.

V tretjem poglavju smo predstavili zgradbo glave in telesa telegrama. Predstavili smo možne dogodke med prenosom telegrama, opisali kako odkrijemo telegram v toku podatkov, in opisali oddajno in sprejemno stran komunikacije.

Uporabljeno programsko in strojno opremo smo predstavili v četrtem poglavju. Opisali smo vse tipe strojne opreme, ki smo jih uporabili in predstavili osnovno zgradbo krmilnika. Predstavili smo Siemensovi programski okolji Siemens Step7 v5,5 in Siemens Scout v4,2, ki smo ju uporabljali pri svojem delu.

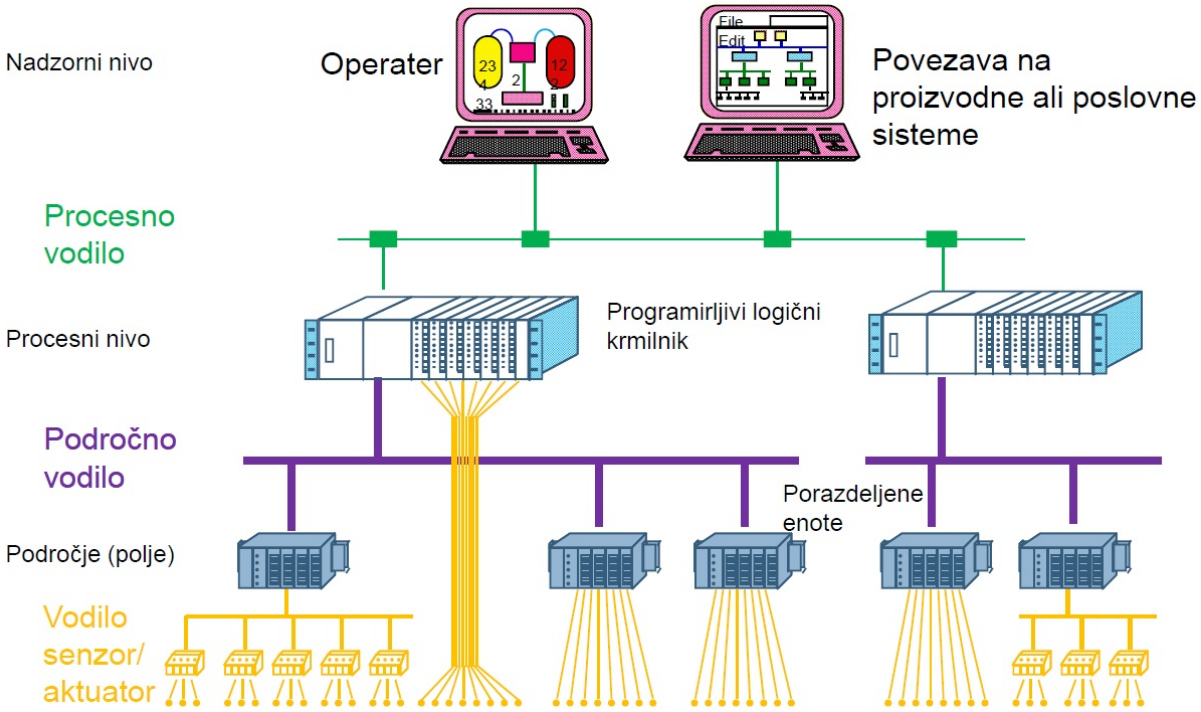
V petem poglavju smo opisali izvedbo programske opreme na različnih tipih strojne opreme. Predstavili smo vgrajene funkcije za komunikacijo po protokolu Ethernet in razlike med njimi. Opisali smo tudi, kako nadziramo potek komunikacije s kontrolnimi zastavicami, predstavili smo uporabljene časovnike in zanimivosti v programski kodi. Predstavili smo tudi primer uporabe programske kode v praksi.

2 Komunikacije v industriji

2.1 Umestitev industrijskih komunikacij

Za začetek računalniškega vodenja procesov lahko štejemo prvo aplikacijo procesnega računalnika v petrokemični tovarni Texaco v Port Arthurju (ZDA) v letu 1959. V šestdesetih letih se je potem razvil koncept direktnega digitalnega vodenja (DDC), ki je temeljil na zamenjavi analognih regulatorjev s centralnim digitalnim računalnikom. Pravi razcvet pa je računalniško vodenje procesov doživelo v sedemdesetih letih s pojavom mikroprocesorjev in mikroračunalnikov, katerih uporaba je omogočila skokovito povečanje funkcionalnosti in učinkovitosti, žal pa tudi kompleksnosti sistemov za vodenje [7].

Razvoj informacijskih tehnologij je tako pustil svoj vpliv tudi v industrijskih okoljih. To se odraža na vseh plasteh industrijskih komunikacij. Plasti in ustrezna vodila so prikazana na sliki 2.



Slika 2: Umestitev industrijskih komunikacij.

2.1.1 Vodilo senzor/aktuator

Vodilo senzor/aktuator prestavlja najnižjo plast v industrijskih komunikacijah. Z njim povezujemo merilne (senzorji ali tipala) in izvršne (ventil, loputa, elektromotor) sisteme z lokalnimi vozlišči in programirljivim logičnim krmilnikom. Za vodilo senzor/aktuator je značilen prenos velikega števila majhnih paketov z majhnimi zakasnitvami. Vodila so prilagojena za neprijazne razmere, kot so visoka in nizka temperatura, tresljaji, elektromagnetne motnje, prisotnost tekočin in soli. Najpogostejša vodila, ki jih uporabljamo na nivoju senzor/aktuator, so AS-interface, CAN in DeviceNet.

2.1.2 Področno vodilo

Na področnem nivoju poteka komunikacija med porazdeljenimi vhodno/izhodnimi enotami, vmesniki človek-stroj in frekvenčnimi regulatorji. S področnim vodilom zmanjšamo število fizičnih povezav. Modularnost sistema se nam poveča, lažje je tudi iskanje in odpravljanje napak. Zaradi modularne zgradbe je lažje tudi razširjanje in krčenje sistema. Najpogosteje je na tem nivoju uporabljeno vodilo Siemens Profibus DP in Modbus.

2.1.3 Procesno vodilo

Procesno vodilo povezuje krmilnike, industrijske računalnike, proizvodne procese in višje nivojske poslovne sisteme. Po procesnem vodilu prenašamo večjo količino podatkov, zakasnitve so na procesnem nivoju lahko večje, saj čas ni primarnega pomena. Danes na tej plasti vodilo Ethernet izpodriva ostala vodila.

2.2 Izvedba komunikacije

Komunikacija med krmilniki in strežnikom MFCS poteka po Industrijskem Ehternetu. Izvedene so tri različice programske opreme za dve različni družini krmilnikov in dodaten modul CP:

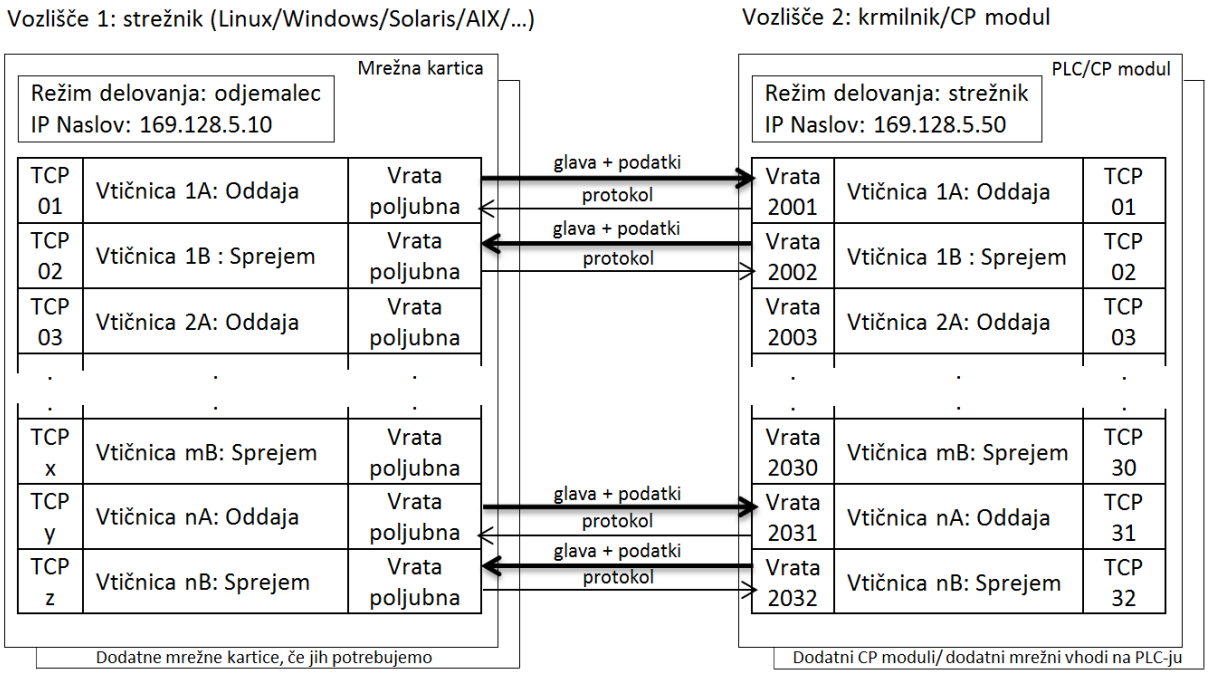
- Simatic S7 CP443-1 Lean modul,
- Siemens Simatic S7-300 CPU,
- Siemens Simotion D4x5.

Za povezavo med krmilnikom in strežnikom je bil uporabljen protokol TCP.

Tabela 1: Plasti OSI model – definicija TCP/IP.

Plast	Enota podatkov	Aplikacija	Naloga	Primer
Aplikacijska	**podatki**	**-**	**povezave modela OSI z uporabnikom**	**prenos datotek**
Predstavitvena	**podatki**	**-**	**predstavitve podatkov**	**skupni jezik**
Plast seje	**podatki**	**-**	**nadzor komunikacije, sinhronizacija**	**nadzor povezave**
Transportna	**segmenti**	**TCP**	**vzpostavitev/končanje povezave**	**varovan prenos podatkov**
Omrežna	**paketi**	**IP**	**naslavljanje drugih omrežij**	**komunikacija med dvema omrežjema**
Povezavna	**okviri**	**Ethernet**	**fizično naslavljanje**	**CSMA/CD**
Fizična	**biti**	**Ethernet**	**medij, signali in binaren prenos podatkov**	**koaksialen kabel**

Programska oprema sloni na transportni plasti na logični povezavi od točke do točke, ki poteka med vrati, določenimi za komunikacijo. Plasti OSI modela so predstavljene v tabeli 1.



Slika 3: Povezava med partnerjema, ki komunicirata [4].

Slika 3 prikazuje povezavo med partnerjema, ki komunicirata po protokolu TCP/IP. Velik okvir predstavlja komunikacijsko opremo (mrežna kartica strežnika, mrežna kartica modula Siemens S7 CP343-1). Vsaka komunikacijska oprema ima omejeno število hkratnih povezav. Strežnik omogoča več kot 1000 hkratnih povezav, medtem ko modul CP omogoča največ 16 povezav po protokolu TCP. Če imamo potrebo po večjem številu povezav, moramo vgraditi dodatne module. Za potrebe naše programske opreme potrebujemo dve povezavi TCP za komunikacijo z enim partnerjem.

Za opis povezave med dvema TCP vtičnicama potrebujemo podatke zbrane v tabeli 2.

Tabela 2: Opis povezave med dvema vtičnicama [1, 2].

Parameter	Pogled s strani krmilnika	Pogled s strani strežnika
Lokalen IP naslov	**169.128.5.50**	**169.128.5.10**
Oddaljen IP naslov	**nepomembno**	**169.128.5.50**
Lokalna številka vrat za pošiljanje	**2001**	**neznano**
Oddaljena številka vrat za pošiljanje	**neznano**	**2001**
Lokalna številka vrat za prejemanje	**2002**	**neznano**
Oddaljena številka vrat za prejemanje	**neznano**	**2002**
Lokalno obnašanje povezave	**pasivno**	**aktivno**
Oddaljeno obnašanje povezave	**aktivno**	**pasivno**

2.3 Nizkonivojska komunikacija

V primeru, da imamo v sistemu več krmilnikov, strežnik MFCS vzpostavi komunikacijo TCP/IP z vsakim krmilnikom posebej. Strežnik MFCS je aktivni partner in skrbi za vzpostavitev in prekinitev povezave, krmilniki so pasivni partnerji, zato čakajo na zahtevo po vzpostavitvi povezave strežnika MFCS, ki deluje kot TCP odjemalec. Hkrati sta odprti dve vtičnici (ang. socket) – prva za oddajanje in druga za sprejemanje telegramov. Številke odhodnih vrat (ang. port) in številke dohodnih vrat so vedno definirane. Primer povezave med dvema vtičnicama je prikazan v tabeli 2 (uporabljena vrata 2001, 2002). Strežnik MFCS za komunikacijo z različnimi krmilniki uporablja različna vrata.

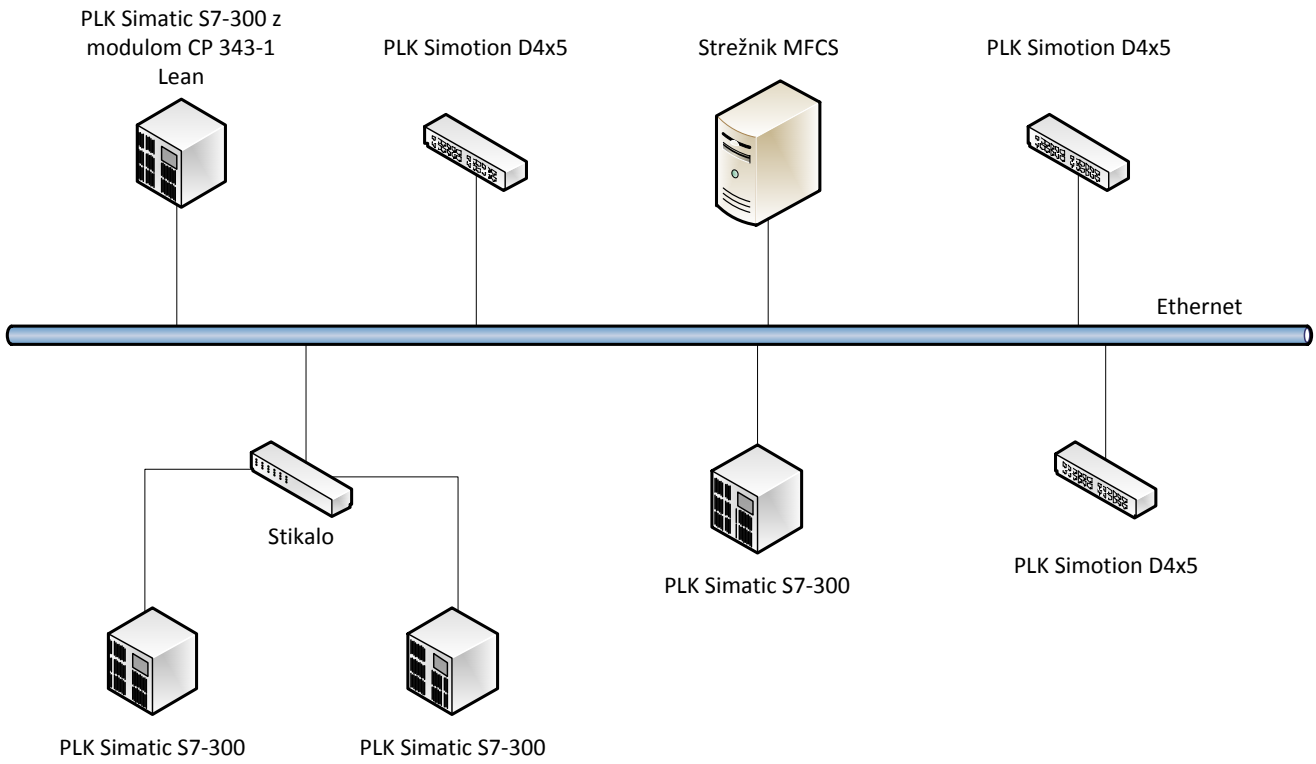
Smer potovanja telegramov (oddajanje, sprejemanje) gledamo iz strani nadrejenega sistema. Vsakršno oddajanje telegramov poteka preko odhodnih vrat MFCS po odhodnem kanalu, vsakršno sprejemanje pa preko dohodnih vrat MFCS po dohodnem kanalu. Izjema so le potrditveni telegramov (ang. acknowledge), ki morajo biti poslani po tistem kanalu, ki je bil uporabljen za njihov sprejem.

Potrebno je preprečiti morebitne izgube telegramov. Pošiljanje telegrama je zaključeno šele takrat, ko sprejmemo potrditveni telegram. Dokler oddajnik ne sprejme potrditvenega telegrama, ne smemo začeti pošiljati naslednjega telegrama. V primeru, da potrditve ni, čez nekaj časa ponovno pošljemo isti telegram. Podatki, ki jih je potrebno določiti za vsako povezavo strežnika MFCS in krmilnika, so na primeru prikazani v tabeli 2 [1, 2].

3 Telegram

3.1 Kaj je telegram?

Telegram je sporočilo, ki ga uporabljamo za izmenjavo informacij in ukazov med programirljivim logičnim krmilnikom in nadzornim sistemom - strežnikom MFCS. V omrežju imamo običajno več krmilnikov in en strežnik MFCS, ki nadzira dogajanje in hrani zahtevane podatke. Primer omrežja prikazuje slika 4. Komunikacija poteka med krmilnikom in strežnikom MFCS.



Slika 4: Shema omrežja.

Razvoj lastnih telegramov je bil potreben, ker smo želeli biti neodvisni pri uporabi programske opreme. Za komunikacijo med krmilnikom in strežnikom MFCS zaradi uvedbe lastnih telegramov ne potrebujemo vmesnega sistema, kot so strežniki OPC, standardne knjižnice, ali vmesnik Siemens Prodave.

3.2 Zgradba telegrama

Za potrebe komunikacije smo morali določiti zgradbo telegrama. Telegram smo razdelili na dva dela, na kontrolni del telegrama in podatkovni del telegrama. Kontrolnemu delu telegrama pravimo glava. Glava telegrama je vedno enake dolžine – 8 besed. Podatkovni del telegrama, ki ga imenujemo telo, pa je od telegrama do telegrama različen. Primer telegrama dolžine sto besed je prikazan na sliki 5.

Beseda (ang. word)	Element
0	start 0x0A0B
1	zaporedna številka telegrama (255, 0..99)
2	oznaka pošiljatelja
3	oznaka prejemnika
4	tip telegrama (telegram aktivnosti, transportni sistem, sistem dvigal)
5	dolžina telegrama v bajtih (dolžina glave(16 bajtov) + dolžina telesa)
6	normalni telegram 0x0000/potrditveni telegram 0xFFFF
7	parnost
8	1. beseda podatkov
9	2. beseda podatkov
.	.
.	.
81	74. beseda podatkov
82	75. beseda podatkov
83	76. beseda podatkov
.	.
.	.
99	92. beseda podatkov

Slika 5: Zgradba telegrama dolžine 100 besed.

Zgradba glave telegrama:

3.2.1 Beseda 0: Start 0x0A0B

Zaradi lažjega odkrivanja začetka telegrama v toku podatkov je prva beseda vsakega telegrama enaka 0x0A0B.

3.2.2 Beseda 1: Zaporedna številka telegrama (255, 0..99)

Zaradi kontrole zaporedja telegramov ima vsak telegram svojo zaporedno številko. Prvi telegram po vzpostavitvi komunikacije ima rezervirano posebno številko 255. Ta pove sprejemniku, da se štetje začenja na novo. Vsak naslednji telegram pa ima zaporedno številko od 0 do 99 naraščajoče, s korakom ena. Po zaporedni številki telegrama 99 ima naslednji telegram zopet zaporedno številko 0.

Z zaporedno številko imamo kontrolo nad telegrami in pravilnostjo prenosa, saj sprejemna stran pričakuje telegram, ki bo imel zaporedno številko, ki bo za ena višja od prejšnje (izjema sta zaporedni številki 255 in 99).

3.2.3 Beseda 2: Oznaka pošiljatelja

Vsaka naprava, ki v našem omrežju komunicira, ima svojo unikatno oznako. Oznake določimo sami glede na zahteve projekta.

Oznake pošiljatelja in prejemnika so predstavljene v tabeli 3, na primeru visokoregalnega skladišča s transportnim sistemom.

Tabela 3: Oznake pošiljatelja/prejemnika.

Enota	Oznaka
strežnik MFCS	1
krmilniki dvigal	100-111
krmilniki transportnega sistema	200, 201

Na napravi, ki bo poslala telegram, v polje oznaka pošiljatelja vpišemo unikatno oznako lokalne naprave, ki jo enkrat določimo, potem pa se ne spreminja.

Oznako pošiljatelja potrebujemo, da prejemnik lahko ugotovi kdo mu je telegram poslal.

3.2.4 Beseda 3: Oznaka prejemnika

Na napravi, ki bo poslala telegram v polje oznaka prejemnika vpišemo unikatno oznako naprave, ki ji bomo poslali telegram.

Oznako pošiljatelja potrebujemo, da prejemnik lahko preveri, če je telegram res namenjen njemu.

3.2.5 Beseda 4: Tip telegrama

Koliko in kakšne tipe telegramov potrebujemo, je odvisno od naprav, s katerimi komuniciramo in potreb projekta.

Različni tipi telegramov so predstavljeni v tabeli 4, na primeru visokoregalnega skladišča s transportnim sistemom.

Tabela 4: Tipi telegramov.

Tip telegrama	Tip telegrama
telegram aktivnosti povezave	1
telegrami dvigal	1201, 1202
telegrami transportnega sistema	1101, 1102

Telegram aktivnosti povezave (ang. life sign telegram) je telegram tipa 1. Aktivni partner mora ves čas vedeti ali je povezava še aktivna. Zato mora oddajnik (enkrat je to strežnik MFCS, drugič krmilnik) v primeru, da ni potrebe po pošiljanju pravih telegramov, 20 sekund po zadnjem poslanem telegramu sporočiti, da je še aktiven. Ta tip telegrama je sestavljen samo iz glave. Ob pošiljanju telegrama povečamo zaporedno številko telegrama, sprejemnik pa mora telegram tudi potrditi. V primeru, da strežnik MFCS več kot 1 minuto ne sprejme nobenega telegrama oziroma ne dobi potrditve za svoje telegrame, poskrbi za zaprtje in ponovno vzpostavitev problematičnega kanala, bodisi sprejemnega ali oddajnega.

Telegram dvigala transportni nalog tipa 1201 se uporablja za izdajanje ukazov dvigalu in za sporočanje o izvedenih ukazih. Telegram, ki ga pošlje strežnik MFCS, predstavlja ukaz za dvigalo. Ko je ukaz izveden, uspešno ali neuspešno, pošlje dvigalo ustrezen povratni telegram. Povratni telegram se od telegrama, ki ga je dvigalo dobilo od strežnika MFCS, razlikuje samo v kodi potrditve ali kodi napake. Ena od njiju je različna od nič.

S kontrolnim telegramom dvigala tipa 1202 strežnik MFCS nadzira delovanje dvigala in zahteva ter sprejema njegovo stanje. Strežnik MFCS ga pošlje, kadar želi izvedeti stanje

dvigala, ali spremeniti režim delovanja dvigala. Krmilnik na dvigalu pošlje telegram s podatki o statusu kot odgovor na zahtevo strežnika MFCS, ali pa samoiniciativno ob vzpostavitvi povezave s strežnikom MFCS in ob pomembnih spremembah stanja dvigala.

Telegram transportnega sistema za transportni nalog tip 1101 se uporablja za premikanje materiala in za sporočanje o izvedenih premikih. Ta telegram pošilja strežnik MFCS.

Kontrolni telegram transportnega sistema tip 1102 pošilja strežnik MFCS, kadar ga zanima stanje transportnega sistema.

3.2.6 Beseda 5: Dolžina telegrama v bajtih

V besedi 5 hranimo celotno dolžino telegrama v bajtih, dolžino glave in telesa skupaj.

Tabela 5: Dolžine telegramov glede na tip

Tip telegrama	Dolžina telegrama v bajtih
telegram aktivnosti povezave – tip 1	16
telegrami dvigala – tip 1201	50
telegrami transportnega sistema – tip 1101	46

Vsak tip telegrama ima definirano dolžino. Telegram aktivnosti povezave nima telesa, zato je njegova dolžina 16 bajtov, kolikor je dolžina glave. Dolžina ostalih tipov je odvisna od količine podatkov, ki jih določen tip telegrama prenaša. Dolžine nekaterih telegramov so predstavljene v tabeli 5.

3.2.7 Beseda 6: Normalni telegram/potrditveni telegram

Pri normalnemu telegramu je beseda 6 v glavi enaka 0x0000, pri potrditvenem telegramu pa 0xFFFF. Vsi tipi telegramov, ki so opisani v tabeli 4, so normalni telegrami. Potrditveni telegrami se uporabljajo za potrditev sprejema normalnih telegramov. Ko sprejemnik prejme normalen telegram, vedno pošlje potrditveni telegram. Tako pošiljatelj ve, da je sprejemnik njegov telegram res dobil.

3.2.8 Beseda 7: Parnost

Protokol TCP/IP ne zagotavlja, da bo celoten telegram sprejet naenkrat. Na primer, če strežnik MFCS oddaja telegram dolg 100 bajtov, lahko krmilnik sprejme najprej 40 bajtov, nato pa še preostalih 60 bajtov, zato je večja možnost, da pride do napake med prenosom. Nujno je, da so telegrami opremljeni s testom paritete (ang. parity check). Rezultat računanje paritete vstavimo v besedo 7 glave telegrama.

```
FUNCTION FC_TelegramCheck: WORD
// XOR se računa na zaporednih WORDih

VAR_INPUT
     data: ARRAY[0..735] OF WORD;
     length: INT; // v WORDih
END_VAR
VAR_TEMP
     control: WORD;
     i: INT;
END_VAR

BEGIN
     control := w#16#0;
        FOR i := 1 TO length DO
           control := control XOR data[i];
        END_FOR;
     FC_TelegramCheck := control;
END_FUNCTION
```

Slika 6: Funkcija računanja paritete telegrama FC_TelegramCheck.

Funkcija na sliki 6 računa pariteto nad celotnim telegramom skupaj - glavo in telesom.

3.2.9 Zgradba telesa

Telo telegrama je odvisno od tipa telegrama, saj smo za različne tipe definirali različne zgradbe telesa. Primer zgradbe telesa telegrama bomo prikazali na telegramu transportni nalog, ki je tipa 1201 in ga uporabljamo za izdajanje ukazov dvigalu ter za sporočanje o izvedenih ukazih. Telo telegrama tipa 1201 je dolgo 16 besed, prikazano je v tabeli 6. Sestavlja ga oznaka telegrama, ki jo generira strežnik MFCS. Druga beseda telesa predstavlja ukaz, s katerim dvigalu pošljemo nalogo, ki jo želimo izvesti. Z besedami enajst, dvanajst, trinajst in štirinajst enolično določimo paletno mesto, na katero se ukaz nanaša. Polji potrditev in napaka potrebujemo pri pošiljanju povratnega telegrama strežniku MFCS, saj mora biti ena

od njiju vedno različna od nič. V primeru, ko se ukaz normalno izvrši, v polje potrditev zapišemo vrednost 101. Kodi potrditve 102 ali 103 pa uporabimo, ko je ukaz uspešno izbrisan bodisi zaradi zahteve strežnika MFCS bodisi zaradi ročnega posredovanja. V primeru napak v polje napaka zapišemo kodo napake. Pri telegramu transportni nalog dvigala smo definirali dvajset kod napak, kot so napaka v glavi zaradi napačnega pošiljatelja, napačna naloga, napačen cilj, polje potrditev je različno od nič, neustrezna višina palete, ciljna lokacija je zasedena, neustrezna črtna koda, vilice so zasedene in podobno.

Tabela 6: Zgradba telegrama transportni nalog dvigala - tip 1201.

Beseda	Ime	Opis
0—7	glava	glava telegrama
8—9	oznaka telegrama MFCS	številčna oznaka telegrama, ki jo generira MFCS, 0 … 4294967295
10	naloga (ukaz)	11 – pojdi na cilj (običajen ukaz) 12 – pojdi na cilj in naloži paleto (običajen ukaz) 13 – pojdi na cilj in odloži paleto (običajen ukaz) 21 – pojdi na cilj (popravni ukaz) 22 – pojdi na cilj in naloži paleto (popravni ukaz) 23 – pojdi na cilj in odloži paleto (popravni ukaz) 99 – izbriši vse ukaze
11	cilj: R	oznaka regala
12	cilj: X	položaj palete v smeri vožnje (X)
13	cilj: Y	položaj palete v smeri dviga (Y)
14	cilj: Z	globina palete (Z)
15	obvladovanje napak	+1: ob napaki vilic sproži napako na PLC +2: ob prevzemu palete ne preverjaj črtne kode
16—20	podatki o paleti	tip palete, črtna koda, višina, napake
21	potrditev	Ko je ukaz izvršen, vrnemo telegram, v katerem ustrezno spremenimo glavo in to polje.
22	napaka	V primeru, da je ob branju telegrama opažena nekonsistentnost, ali da je telegram neizvedljiv, ga zavrnemo z ustrezno kodo.
23	rezerva 1	rezervno polje, vrednost postavimo na 0
24	rezerva 2	rezervno polje, vrednost postavimo na 0

V programski opremi smo omejili največjo dolžino telegrama na 100 besed, ker potrebe za daljše telegrame nismo imeli. Na največjo dolžino telegrama je vezanih več polj, tako na oddajni, kot sprejemni strani, zato smo z omejitvijo prihranili na prostoru.

3.3 Pošiljanje telegrama na primeru

Vzemimo, da krmilnik z oznako 207 pošilja telegram strežniku MFCS, ki ima oznako 1. Telegram ima poleg glave vpisani še dve besedi 0xBEEF in 0xFEED tako, kot je prikazano v tabeli 7.

Tabela 7: Pošiljanje telegrama na primeru.

Beseda	Pomen besede	Telegram nastavek (Dec)	Telegram nastavek (Hex)	Telegram popoln (Hex)	Potrditev nastavek (Hex)	Potrditev popolna (Hex)
0	start	2571	0A 0B	0A 0B	0A 0B	0A 0B
1	zaporedna številka telegrama	42	00 2A	00 2A	00 2A	00 2A
2	oznaka pošiljatelja	207	00 CF	00 CF	00 01	00 01
3	oznaka prejemnika	1	00 01	00 01	00 CF	00 CF
4	tip telegrama	1201	04 B1	04 B1	04 B1	04 B1
5	dolžina telegrama	20	00 14	00 14	00 10	00 10
6	normalni/potrditve ni telegram	0	00 00	00 00	FF FF	FF FF
7	parnost	0	00 00	4E 48	00 00	F1 B1
8	podatki	48879	BE EF	BE EF		
9	podatki	65261	FE ED	FE ED		

Glede na naše potrebe najprej generiramo nastavek telegrama, ki ga želimo poslati. Nato vstavimo nastavek v funkcijo FC_TelegramCheck(). Funkcija nam v našem primeru, prikazanem v tabeli 7, vrne vrednost 0x4E48. To vrednost vpišemo v glavo na mesto besede 7. Tako zgrajen telegram krmilnik odpošlje strežniku MFCS.

Ko strežnik MFCS sprejme telegram, najprej s funkcijo FC_TelegramCheck() preveri pravilnost prenosa. Če funkcija vrne vrednost 0x0000, je telegram pravilno sprejet in začne s pošiljanjem potrditve. V primeru, da funkcija vrne rezultat različen od 0x0000, je prišlo do napake pri prenosu, zato prejete podatke zavrže in čaka na sprejem novih.

Potrditveni telegram sestavimo tako, da zamenjamo polji pošiljatelja in sprejemnika (besedi 2 in 3 v glavi), prilagodimo dolžino telegrama na velikost glave, besedo 6 nastavimo na vrednost 0xFFFF, saj nam ta sporoča, da gre za potrditveni telegram. Tako smo pripravili nastavek potrditvenega telegrama. Potrditveni telegram ne vsebuje podatkov. Na enak način kot prej s pomočjo funkcije FC_TelegramCheck() izračunamo pariteto, ki je v našem primeru 0xF1B1 in jo vpišemo v besedo 7 ter potrditveni telegram odpošljemo.
Krmilnik ob sprejemu potrditvenega telegrama preveri, če se le-ta nanaša na zadnji oddani telegram. V primeru, da se nanaša na zadnji oddan telegram, je prenos telegrama zaključen in lahko začne s pošiljanjem novega telegrama. Če potrditev ni ustrezna, čaka na pravo potrditev, ali pa po izteku časovnika ponovno pošlje telegram.

3.4 Razmere med prenosom telegrama

Protokol TCP/IP je orientiran na delo s tokom podatkov, zato morajo biti zahteve po preverjanju integritete podatkov višje kot pri paketno orientiranih protokolih, kot so Profibus, TCP/IP z RFC1006 ter protokol ISO. Paketno orientirani protokoli nekatere probleme že sami v zasnovi odpravijo. Na primeru vodila Profibus prenos vedno pomeni prenos enega, celotnega telegrama [6]. Če oddajnik pošlje 2 telegrama, vsakega velikosti 10 bajtov, prejemnik zazna 2 dogodka, vsakega dolžine 10 bajtov. V primeru protokola TCP/IP to ni zagotovljeno. Sprejemnik lahko zazna le en dogodek in sprejem 20 bajtov, zato mora sprejemnik znati zgraditi in pregledati strukturo telegrama. Možno je tudi, da prejemnik najprej sprejme 15 bajtov, nato pa še preostalih 5 bajtov, zato mora znati sestaviti telegram.
Kako deluje algoritem za detekcijo glave telegrama, je opisano v poglavju 3.6.

3.5 Dogodki med prenosom

3.5.1 Legenda - telegram

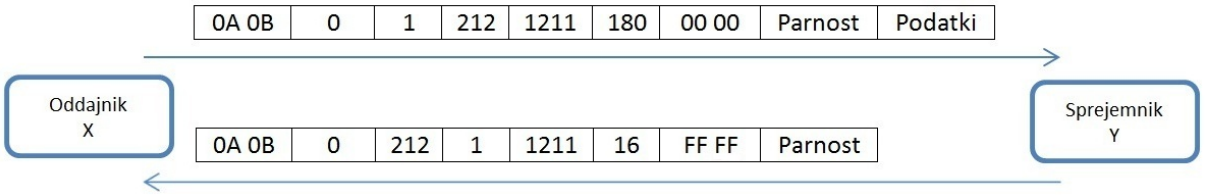
Na sliki 7 je predstavljena legenda telegrama.

0A 0B	255, 0..99	X	Y	T	L	Vrsta telegrama	Pariteta	Podatki
Start	Zaporedna št.	Pošiljatelj	Prejemnik	Tip	Dolžina			

Slika 7: Legenda – telegram.

3.5.2 Prenos brez napak

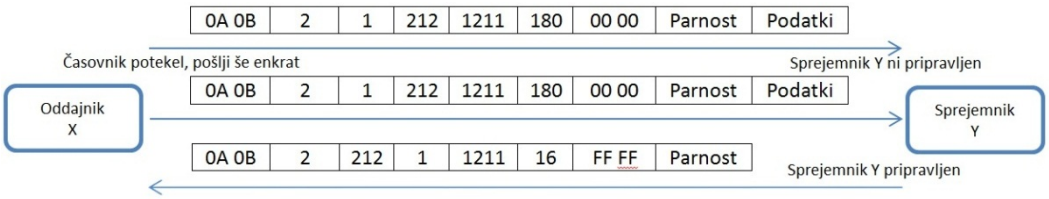
Na sliki 8 je prikazan prenos telegrama brez napak. Oddajnik X pošlje telegram, sprejemnik Y telegram sprejeme in pošlje oddajniku potrditveni telegram [3]. Ko oddajnik sprejme potrditveni telegram, se prenos uspešno zaključi.



Slika 8: Prenos brez napak.

3.5.3 Sprejemnik ni pripravljen

Oddajnik pošlje telegram, vendar sprejemnik ni pripravljen, ker je v okvari ali v režimu stop, zato telegrama ne sprejme. Ker oddajnik ne sprejme potrditvenega telegrama, po pretečenemu času 5 s zopet pošlje enak telegram. Oddajnik pošilja telegrame v intervalu 5 s dokler sprejemnik ni pripravljen. Ko je sprejemnik pripravljen, sprejme telegram in nanj odgovori s potrditvenim telegramom, kot je prikazano na sliki 9.

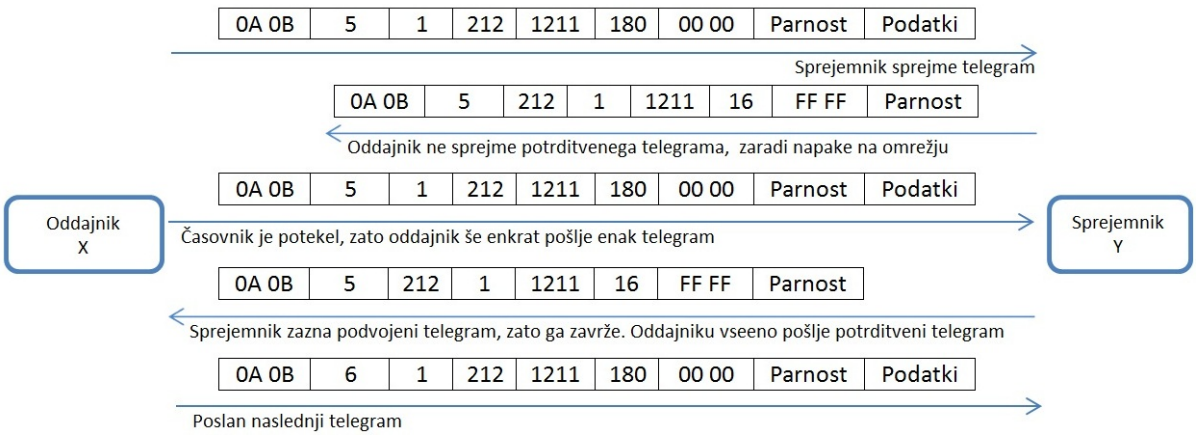


Slika 9: Sprejemnik ni pripravljen.

3.5.4 Napaka na omrežju med čakanjem oddajnika na potrditveni telegram

Oddajnik pošlje telegram in sprejemnik ga uspešno sprejme. Oddajnik odda potrditveni telegram, vendar zaradi napake ne omrežju, ne doseže sprejemnika. Ker oddajnik ni dobil potrditve, po določenem času zopet pošlje enak telegram. Sprejemnik telegram sprejme in ugotovi, da je enak prejšnjemu. Zadnji prejeti telegram, ki je enak prejšnjemu, imenujemo podvojeni telegram. Sprejemnik podvojeni telegram razveljavi, oddajniku pa pošlje

potrditveni telegram, kot je prikazano na sliki 10. Ko oddajnik sprejme potrditveni telegram, se prenos uspešno zaključi.

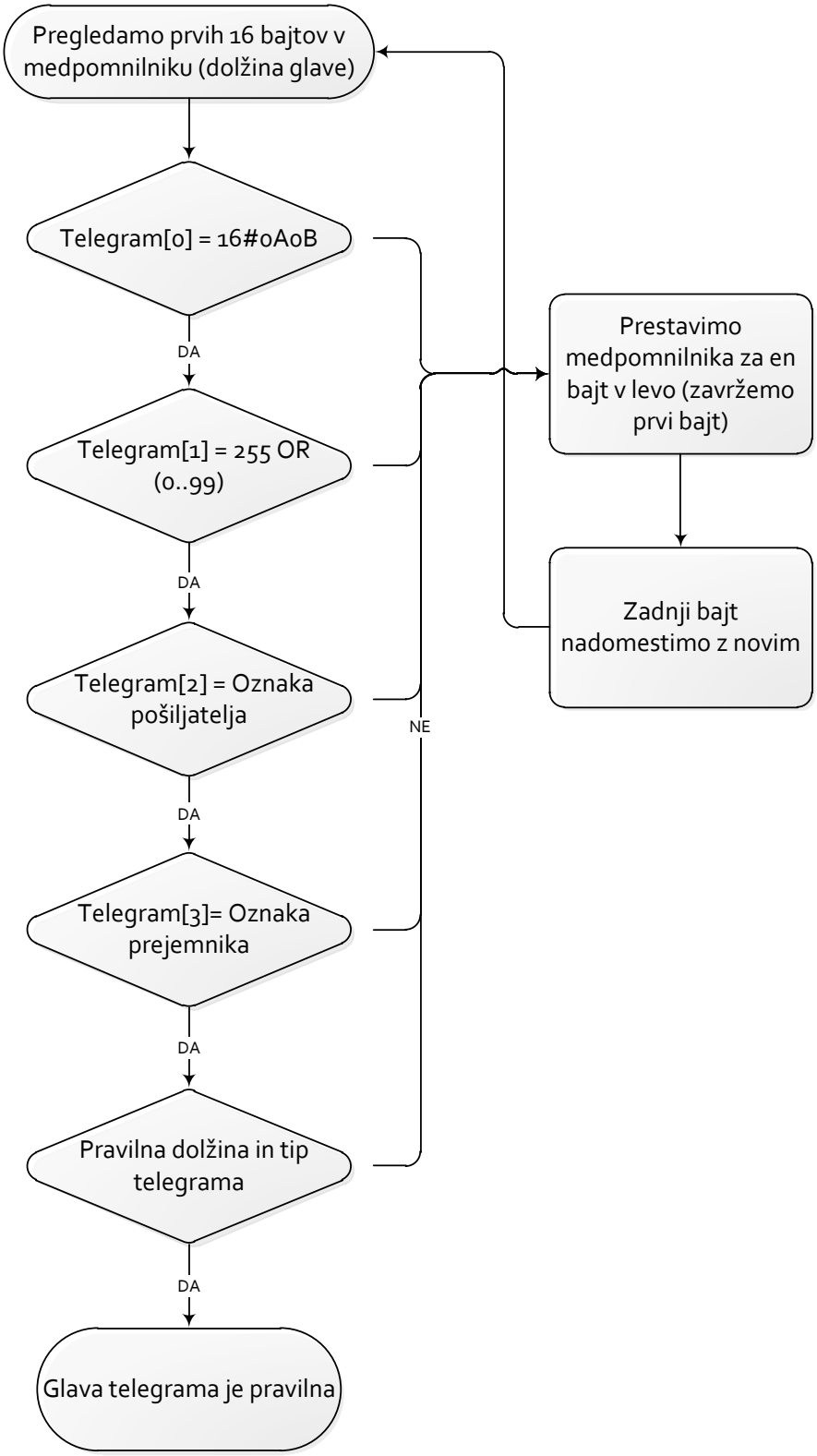


Slika 10: Napaka na omrežju med čakanjem na potrditev.

3.6 Odkrivanje telegrama v toku podatkov

Zaradi problemov pri prenosu TCP/IP, mora sprejemnik iz toka podatkov znati ugotoviti, kje se telegram nahaja. Algoritem smo razvili tako, da najprej odkrijemo, kje se nahaja glava telegrama. Ko dobimo glavo, iz nje preberemo informacijo o dolžini celotnega telegrama in tako sestavimo celoten telegram. V diagramu poteka na sliki 11 je prikazano, kako zaznamo glavo telegrama.

Za zaznavo glave telegrama, se osredotočimo na prvih 8 besed, ki se nahajajo v medpomnilniku prejetih podatkov, saj ta dolžina ustreza dolžini glave. Preverimo, če je prva beseda enaka dogovorjenemu začetku telegrama 0x0A0B. Pogledamo tudi, če je zaporedna številka telegrama v dogovorjenem območju (255, 0...99). Ker komunikacija poteka samo med strežnikom MFCS in krmilnikom, imamo vnaprej znano oznako pošiljatelja in prejemnika, zato preverimo tudi to. Preverimo tudi, če je dolžina telegrama v dogovorjenem območju. Dolžina telegrama mora biti enaka ali večja dolžini glave ter manjša, kot je dolžina najdaljšega telegrama. Preverimo tudi tip telegrama. Če so vse navedene besede v skladu s pričakovanji, potem predvidimo, da smo dobili glavo telegrama. Če katerakoli beseda ni v skladu s pričakovanji, premaknemo glavo za eno besedo v levo. S tem prvo besedo zavržemo, zadnjo pa nadomestimo z novo. Nato po enakem postopku zopet preverimo, ali novi podatki ustrezajo glavi telegrama.

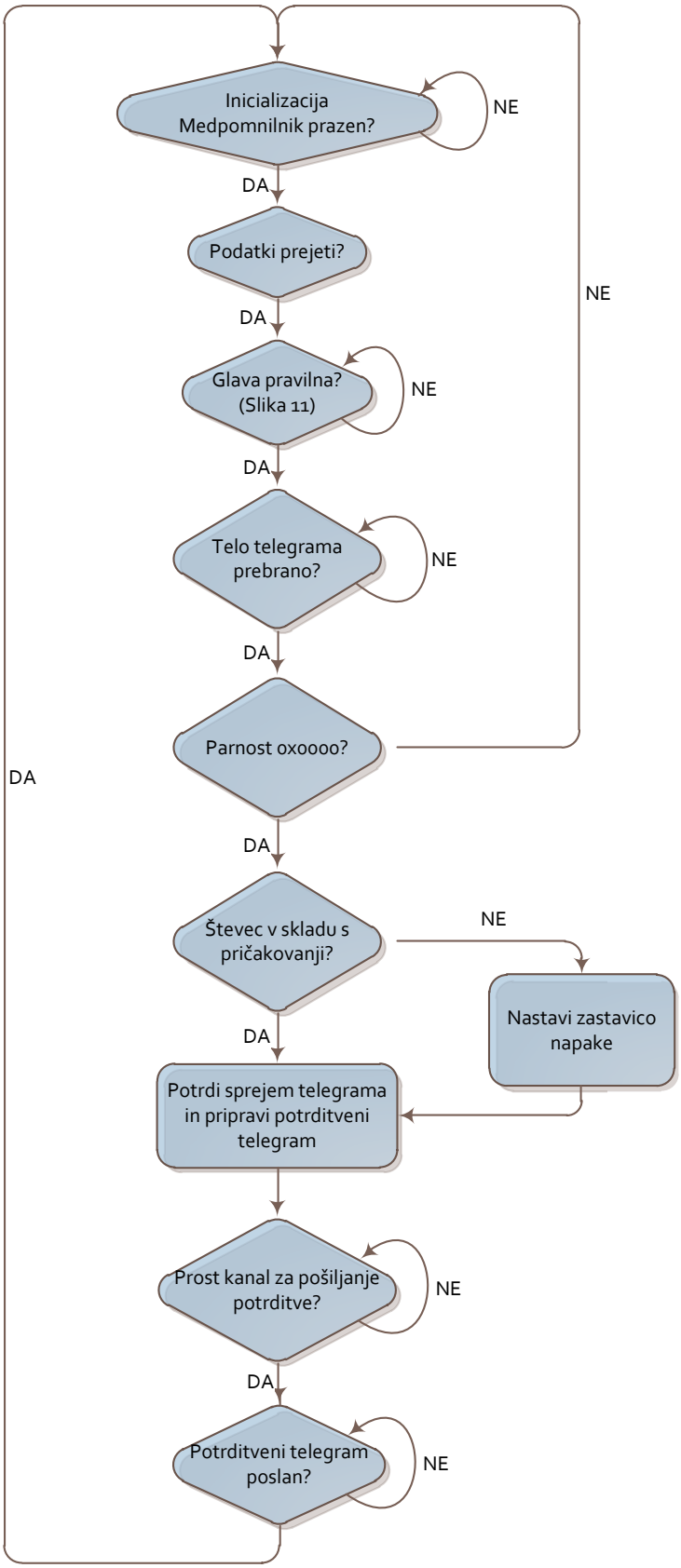


Slika 11: Odkrivanje glave telegrama v toku podatkov.

3.7 Sprejemna stran komunikacije

Na sprejemni strani krmilnik sprejema normalne telegrame in pošilja potrditvene. Algoritem je napisan v obliki avtomata, ki ga prikazuje slika 12.

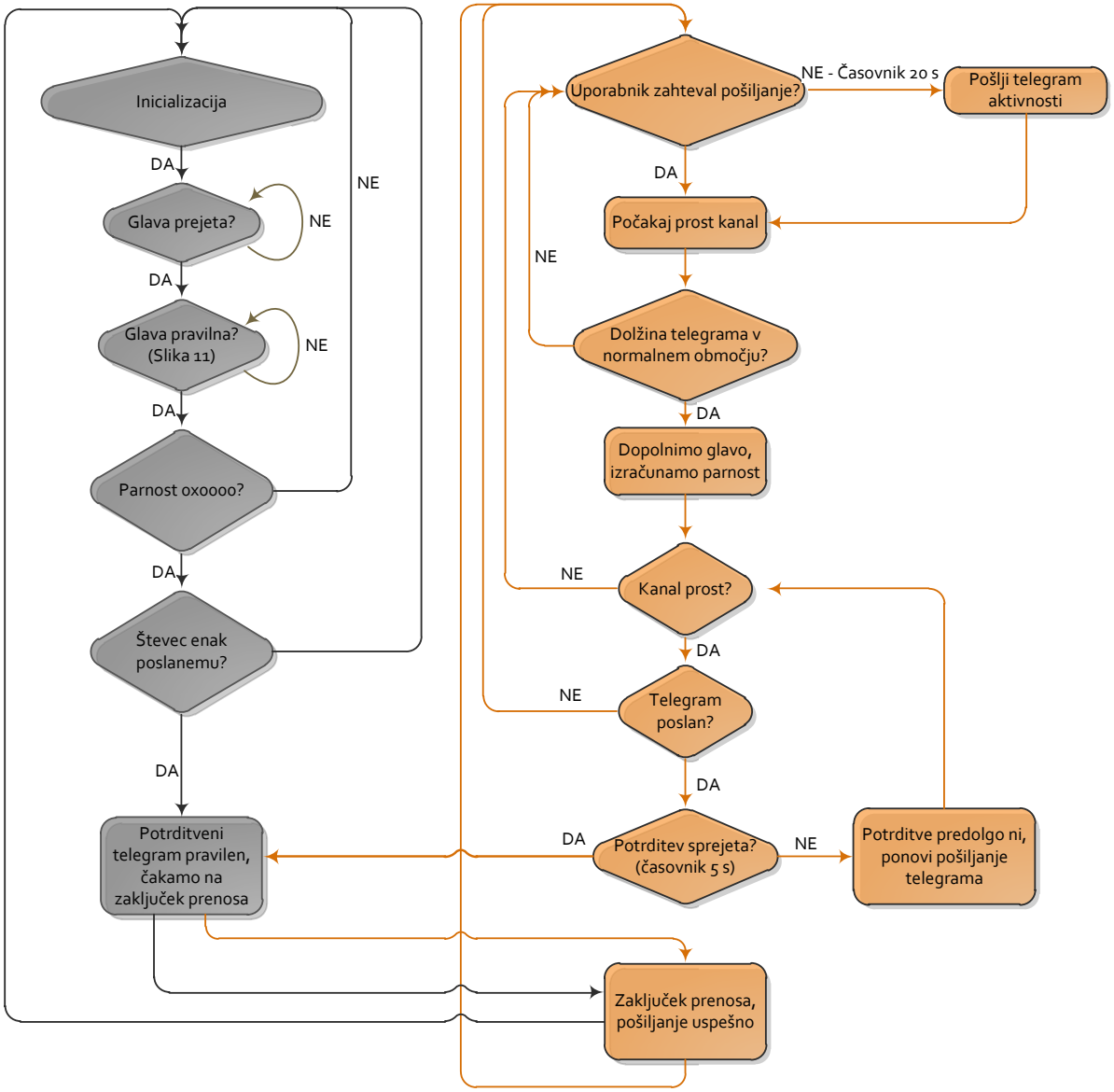
Začetno stanje avtomata je inicializacija, v katerem ponastavimo določene spremenljivke. Ko prejmemo nove podatke, skočimo v naslednje stanje, kjer preverimo glavo (podroben opis preverjanja glave je opisan v prejšnjem polavju). Ko domnevamo, da smo dobili pravilno glavo, preverimo še podatkovni del telegrama - telo. Dolžina telesa je od telegrama do telegrama različna, zato jo moramo za vsak telegram izračunati. Dolžino celotnega telegrama preberemo iz glave telegrama ter ji odštejemo dolžino glave, da dobimo dolžino telesa. Če v medpomnilniku nimamo toliko podatkov, kot je dolžina celotnega telegrama, ponovno kličemo funkcijo prejema. Ko dobimo celoten telegram, s funkcijo FC_TelegramCheck() preverimo njegovo parnost. Če je parnost enaka 0x0000 potem domnevamo, da smo dobili pravilen telegram. Zaradi možnosti prejema podvojenih telegramov in sledenja sekvenci, preverimo zaporedno številko telegrama. Če se zaporedna številka ujema z vrednostjo našega števca zaporedja telegramov, potem smo dobili nov telegram. Potrdimo sprejem telegrama in pripravimo potrditveni telegram. Potrditveni telegram zajema samo glavo. Pripravimo ga na podlagi prejetega telegrama tako, da zamenjamo oznako pošiljatelja in prejemnika ter tip telegrama nastavimo na 0xFFFF. Ostala polja, razen parnosti, ostanejo enaka glavi prejetega telegrama. Za tako izdelan nastavek telegrama izračunamo parnost, tako dobimo potrditveni telegram, ki ga poskušamo poslati. Ko je kanal za pošiljanje prost, telegram odpošljemo, avtomat pa skoči v začetno fazo inicializacija.



Slika 12: Sprejem telegrama.

3.8 Oddajna stran komunikacije

Na oddajni strani krmilnik na našo zahtevo pošilja normalne telegrame in sprejema potrditvene telegrame. Algoritem je napisan v obliki dveh avtomatov, ki sta odvisna drug od drugega. Algoritem lahko razdelimo na avtomat, ki skrbi za sprejem potrditvenega telegrama, in avtomat, ki skrbi za pošiljanje telegrama strežniku MFCS. Avtomata sta predstavljena na sliki 13.



Slika 13: Oddajanje telegrama, del diagrama poteka sive barve skrbi za sprejem potrditvenih telegramov, oranžni del diagrama poteka pa skrbi za pošiljanje normalnih telegramov strežniku MFCS.

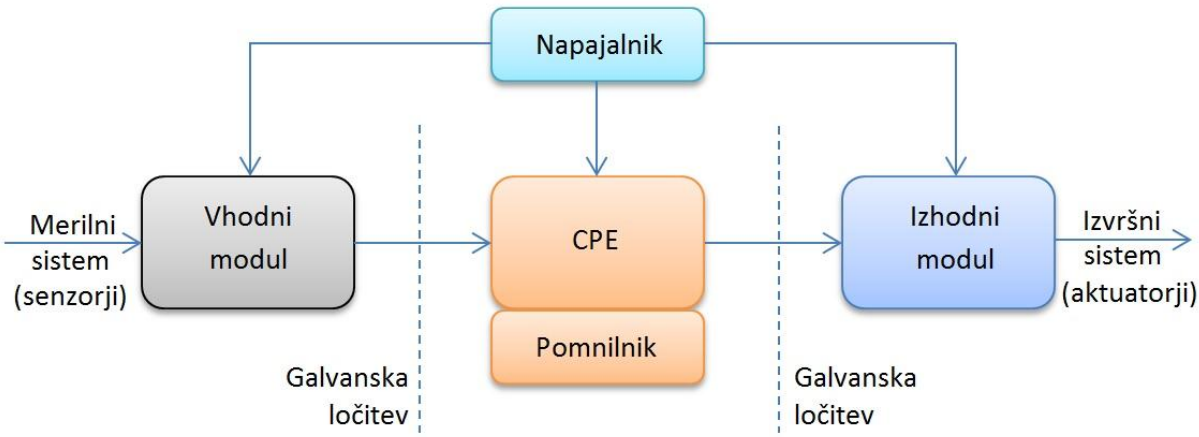
Če želimo poslati telegram strežniku MFCS, moramo najprej napolniti telo telegrama s podatki. To mora programer storiti sam. Ko zahtevano pomnilniško strukturo napolni z želenimi podatki, odda zahtevo za pošiljanje telegrama tako, da postavi vrednost kontrolne zastavice za pošiljanje v visoko logično stanje 1. Če dlje časa ne pošiljamo normalnih telegramov (20 s), sistem sam pošilja telegrame aktivnosti povezave in s tem ohranja povezavo aktivno. Ko dobimo zahtevo za pošiljanje pogledamo, ali imamo prost kanal za pošiljanje telegrama. Preden telegram odpošljemo preverimo, da je njegova dolžina v definiranih okvirjih. V naslednjem koraku preverimo zaporedno številko, dopolnimo glavo in izračunamo parnost celotnega telegrama. Še enkrat preverimo, če imamo prost kanal za pošiljanje in telegram odpošljemo. Ko nam funkcija pošiljanje sporoči, da je telegram poslan, čakamo sprejem potrditvenega telegrama. Če ga po izteku časovnika dolžine 5 s ne dobimo, telegram ponovno pošljemo. Za sprejem potrditvenega telegrama skrbi avtomat, predstavljen na sliki 13. Najprej pogledamo ali smo dobili zahtevanih 16 bajtov podatkov, ki ustrezajo dolžini potrditvenega telegrama, nato preverimo ali je prejeti telegram res potrditveni in glava ustreza zahtevam. Telegramu s funkcijo FC_TelegramCheck() izračunamo parnost, ki mora biti enaka 0x0000. Preverimo tudi, če je zaporedna številka potrditvenega telegrama enaka zaporedni številki oddanega telegrama. Če je potrditveni telegram pravilen, potem avtomat, ki skrbi za sprejem potrditvenega telegrama počaka avtomat, ki skrbi za pošiljanje telegrama na zaključek prenosa, nato pa skoči v stanje inicializacija. Avtomat, ki skrbi za pošiljanje telegrama, čaka na prejem potrditvenega telegrama. Ko mu avtomat, ki skrbi za prejem potrditvenih telegramov sporoči, da je bil odposlani telegram potrjen, uspešno zaključi pošiljanje, ponastavi kontrolne zastavice in spremenljivke.

4 Uporabljena programska in strojna oprema

4.1 Strojna oprema

4.1.1 Programirljiv logični krmilnik Siemens Simatic S7-300

Krmilnik je računalnik namenjen vodenju procesov v industrijskem okolju (slika 14). Krmilnike nameščamo v industrijskem okolju v bližini sistema, zato morajo biti zaščiteni pred umazanijo, vodo in mehanskimi poškodbami. Odporni morajo biti na elektromagnetne motnje, vibracije in temperaturo. Naloge krmilnika so merjenje, vodenje in regulacija, omogoča komunikacijo z drugimi krmilniki in napravami ter vodi dnevnik napak. Modularna izvedba krmilnikov in vhodno/izhodnih modulov omogoča enostavno sestavljanje. Digitalni in analogni vhodi ter izhodi krmilnika podpirajo standardne napetostne in tokovne nivoje.



Slika 14: Arhitektura programirljivega krmilnika.

Senzorji lahko zajamejo marsikatero fizikalno količino, ki jo podajo kot binarno (0—24 V) ali analogno vrednost (4—20 mA, 0—20 mA). Analogne vrednosti krmilnik filtrira in skalira, potem pa jih vzorči in s pretvornikom A/D pretvori v binarno obliko, da jih lahko procesiramo. Binarne vrednosti filtrira in vzorči, da jih lahko procesiramo. Na izhodnem modulu s krmilnikom krmilimo tranzistorje ali releje.

Slika 15: Krmilnik Siemens CPU 314C-2 PN/DP.

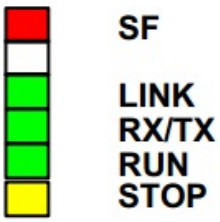
Siemens S7 314C-2 PN/DP (slika 15) je kompakten krmilnik z vgrajenimi digitalnimi in analognimi vhodi/izhodi in tehnološkimi funkcijami, visoko procesorsko zmogljivostjo za binarne operacije, omogoča pa tudi operacije s plavajočo vejico. Izvajalni čas binarnega ukaza je približno 0,06 µs, za ukaz s plavajočo vejico pa 0,59 µs [10]. Povezava oddaljenih vhodov/izhodov je možna preko vmesnikov Profibus ali Industrijski Ethernet. Krmilnik ima kombiniran vmesnik MPI/Profibus DP in dva vmesnika Ethernet, ki delujeta tudi kot dvovratno stikalo po protokolu Ethernet TCP/IP. Krmilnik je primeren za srednje velike aplikacije. Vgrajenega ima 192 kB delovnega pomnilnika, program shranjujemo na pomnilniško kartico SD velikosti do 8 MB. Krmilnik konfiguriramo in programiramo v Siemensovem programskem okolju Simatic Step7.

4.1.2 Modul CP 343-1 Lean

Modul CP 343-1 Lean (v nadaljevanju modul CP) je razširitveni modul namenjen krmilnikom družine S7-300 (slika 16). Omogoča komunikacijo krmilnika prek vodila Industrijski Ethernet ter hitrost prenosa 10 Mb ali 100 Mb. Lahko ga uporabimo za razširitev komunikacijskih zmogljivosti tudi pri krmilnikih, ki imajo vmesnik Ethernet že vgrajen. Konfiguracijske nastavitve modula naredimo v programskem okolju Step7 z orodjem HW config in NetPro.

Slika 16: Siemens modul CP 343-1 Lean.

Pet statusnih led diod prikazuje stanje naprave in komunikacije modula CP (slika 17). Luč SF nam sporoča, da je modul ustavljen zaradi napak. Luč RUN sveti, ko modul obratuje. Luč STOP nam sporoča, da je modul zaustavljen ali pa se na njem izvaja diagnostika ali konfiguracija. Za prenos podatkov sta pomembni luči LINK ter RX/TX. Luč LINK nam sporoča, da je povezava vzpostavljena, RX/TX pa utripa, ko modul CP podatke oddaja, ali sprejema.



Slika 17: Statusne LED diode modula CP-343 1 Lean.

Modul CP omogoča največ 12 hkratnih povezav po Industrijskem Ethernetu. Za potrebe naše komunikacije med krmilnikom in strežnikom MFCS potrebujemo dve povezavi po protokolu TCP, zato kapacitete modula ne izkoristimo v celoti. Modul CP lahko hkrati uporabimo tudi za povezavo z vmesnikom človek stroj in za komunikacijo z drugimi krmilniki ali moduli CP.

Za komunikacijo po protokolu TCP se uporabljata vgrajeni funkciji AG_SEND za pošiljanje in AG_RECV za prejemanje podatkov. Funkciji po protokolu TCP omogočata prenos podatkovne strukture velikosti od 1 B do 8192 B. Čas obdelave funkcije na krmilniku znaša od 2,5 ms do 5 ms [9].

4.1.3 Krmilnik Simotion D425

Krmilnik Simotion D425 ima poleg običajnih enot krmilnika vgrajeno tudi nadzorno enoto za krmilni del frekvenčnih regulatorjev. Potrebno je kupiti samo močnostni del frekvenčnih regulatorjev, zato je nakup krmilnika smotrn pri projektih, kjer uporabljamo veliko frekvenčnih regulatorjev, z manj logike. Krmilnik D425 podpira regulacijo največ 16 osi. Čas cikla srednje obremenjenega krmilnika je 2 ms [8]. Krmilnik lahko povežemo preko vodil Profibus in Ethernet [12].

Slika 18: Krmilnik D425.

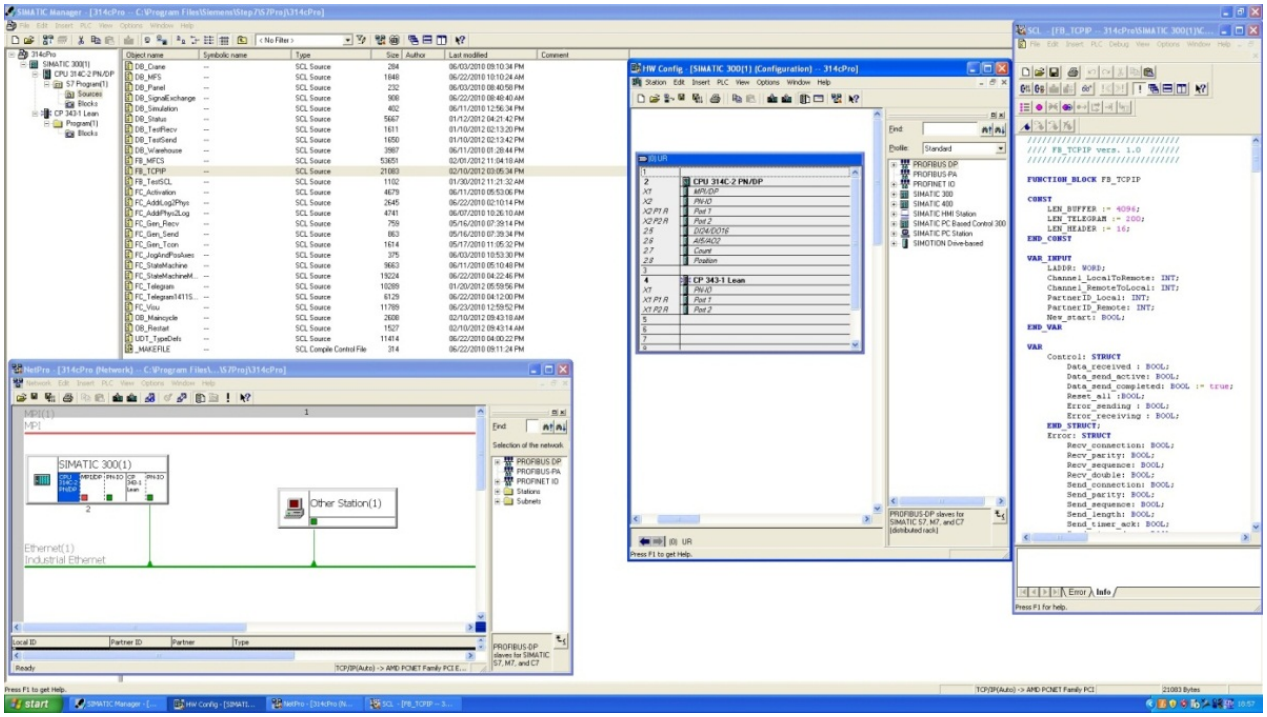
Krmilnik programirano in konfiguriramo v Siemensovem orodju Simotion Scout, ki je opisano v nadaljevanju.

4.2 Programska oprema

4.2.1 Simatic Step7 v5,5

Simatic Step7 je Siemensovo programsko okolje za delo s krmilniki družin Simatic S7-300/S7-400, Simatic M7-300/M7-400 in Simatic C7.

Okolje omogoča programiranje krmilnikov, konfiguriranje, načrtovanje povezav strojne opreme, testiranje, servisiranje, dokumentiranje, diagnosticiranje in arhiviranje. Programsko okolje je predstavljeno na sliki 19.



Slika 19: Programsko okolje Siemens Step7.

Programsko okolje Siemens Step7 omogoča programiranje v več programskih jezikih:

- Siemens Graph je jezik, ki temelji na Petrijevih mrežah;
- lestvični diagram LAD je grafični jezik, ki omogoča enostaven prehod na programiranje iz električnih shem;
- funkcijski načrt FBD je standardni grafični jezik, kjer grafično povezujemo funkcije in funkcijske bloke;

- lista ukazov STL je nizkonivojski programski jezik, podoben zbirniku;
- strukturiran tekst SCL je jezik podoben Pascalu, primeren za kompleksne obdelave podatkov.

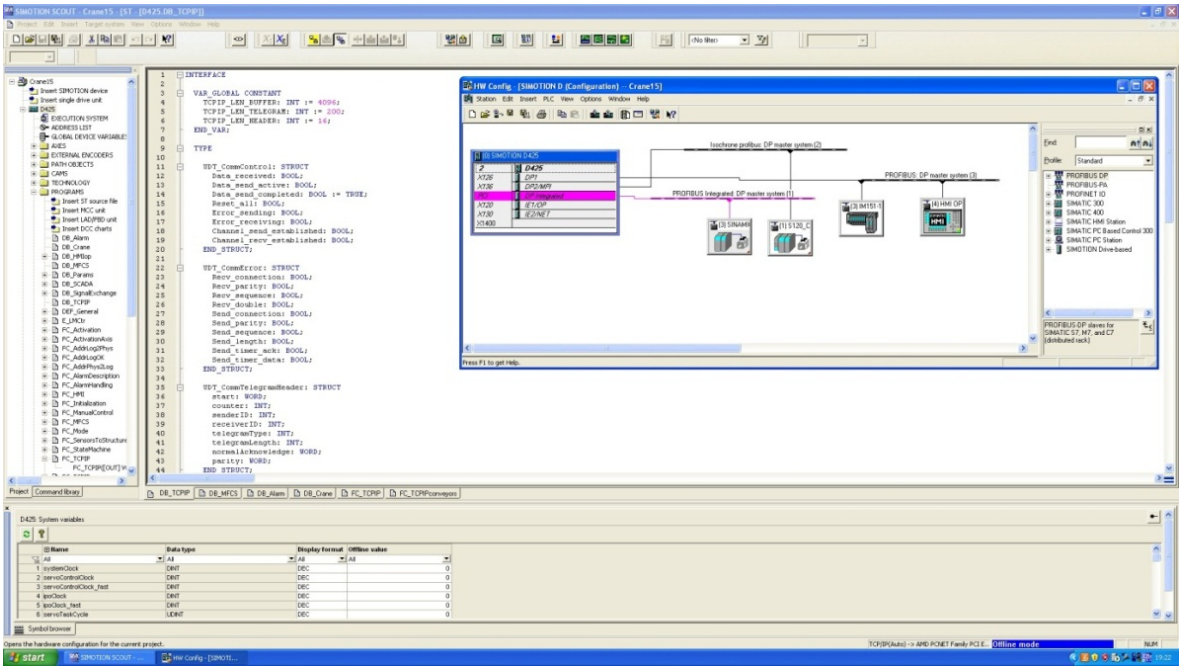
V našem primeru smo programirali v programskem jeziku SCL. Programsko okolje smo uporabili za delo s krmilnikom Simatic in modulom CP. Poleg pisanja kode smo z orodjem nastavili tudi konfiguracijo povezave med krmilnikom in strežnikom MFCS ter spremljali in nadzirali dogajanje.

4.2.2 Simotion Scout v4,2

Simotion Scout je Siemensovo razvojno okolje za krmilnike družin Simotion D4xx, Simotion C2xx in Simotion P350. Orodje omogoča konfiguracijo omrežja in strojne opreme, nastavljanje parametrov, programiranje, testiranje in diagnostiko. Okolje je predstavljeno na sliki 20.

Postopek dela izvedemo podobno v obeh programskih okoljih. Potrebno je:

1. kreiranje novega projekta,
2. izbira naprave (Simotion D, Simotion C, Simotion P, Simatic S7, Simatic),
3. konfiguracija sistema (struktura topologije):
    - izbiranje strojne opreme (HW config),
    - povezovanje strojne opreme (NetPro),
4. izbira programskega jezika (MCC, ST, LAD/FBD, SCL) in programiranje,
5. nalaganje aplikacije na krmilnik,
6. testiranje aplikacije.



Slika 20: Programsko okolje Simotion Scout v4.2.

5 Izvedba komunikacije s telegrami na vseh tipih strojne opreme

5.1 Krmilnik serije S7-300 z vgrajenim vmesnikom Ethernet

Za komunikacijo prek protokola TCP smo uporabili štiri vgrajene Siemensove funkcije, ki skrbijo za vzpostavitev in prekinitev komunikacije ter pošiljanje in prejemanje podatkov:

- TCON (FB65),
- TDISCON (FB66),
- TSEND (FB63),
- TRCV (FB64).

Funkcijski bloki z vhodnimi in izhodnimi parametri so predstavljeni na slikah 22 in 23.

5.1.1 Funkcija TCON (FB65)

Za pošiljanje ali prejemanje podatkov je potrebno povezavo med odjemalcem in strežnikom najprej vzpostaviti. Za vzpostavitev komunikacije poskrbi vgrajena funkcija TCON. Za vzpostavitev komunikacijskega kanala je potrebno nastaviti parametre vmesnika Ethernet. Tega v primeru vgrajenega vmesnika Ethernet ne počnemo v okolju Step7 z orodjem NetPro, ampak ima funkcija TCON kot parameter podano podatkovno strukturo UDT65 TCON_PAR (slika 21), s podatki o konfiguraciji povezave.

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	block_length	WORD	W#16#40	Length of the UDT (64 bytes)
+2.0	id	WORD	W#16#2	Reference to this connection (valuerange: W#16#0001 to W#16#FFFF)
+4.0	connection_type	BYTE	B#16#11	B#16#11: TCP/IP native; B#16#12: ISO on TCP; B#16#13: UDP; B#16#01: TCP/IP comp
+5.0	active_est	BOOL	FALSE	FALSE: passive connection establishment; TRUE: active connection establishment
+6.0	local_device_id	BYTE	B#16#2	Allowed values: B#16#0, B#16#2, B#16#3, B#16#5. See online help.
+7.0	local_tsap_id_len	BYTE	B#16#2	Used length of the parameter local_tsap_id
+8.0	rem_subnet_id_len	BYTE	B#16#0	Unused; must be B#16#00
+9.0	rem_staddr_len	BYTE	B#16#0	Meaning of parameter rem_staddr: B#16#00: is irrelevant; B#16#04: valid address
+10.0	rem_tsap_id_len	BYTE	B#16#0	Used length of the parameter rem_tsap_id
+11.0	next_staddr_len	BYTE	B#16#0	B#16#1 if local_device_id = 0; else B#16#0
+12.0	local_tsap_id	ARRAY[1..16]	B#16#7, B#16#(	Depending on parameter connection_type: local port no. / local TSAP-ID
*1.0		BYTE		
+28.0	rem_subnet_id	ARRAY[1..6]	B#16#0	Unused; must be B#16#00
*1.0		BYTE		
+34.0	rem_staddr	ARRAY[1..6]	B#16#0	IP address of the remote connection end point, e. g. 192.168.0.1
*1.0		BYTE		
+40.0	rem_tsap_id	ARRAY[1..16]	B#16#0	Depending on connection type: remote port no. / remote TSAP-ID
*1.0		BYTE		
+56.0	next_staddr	ARRAY[1..6]	B#16#0	Depending on local_device_id: rack / slot no. of the associated CP / irrelevant
*1.0		BYTE		
+62.0	spare	WORD	W#16#0	Unused; must be W#16#0000
=64.0		END_STRUCT		

Slika 21: Podatkovna struktura UDT65 - TCON_PAR.

Imamo dva možna principa povezave. Lahko naš krmilnik deluje kot odjemalec in je aktiven partner, ali kot strežnik, in je pasiven komunikacijski partner. V našem primeru bo krmilnik pasivni komunikacijski partner.

Seznam pomembnih parametrov podatkovne strukture UDT65 TCON_PAR je naveden v nadaljevanju.

- BLOCK_LENGHT je dolžina strukture TCON_PAR in znaša 64 bajtov.
- ID je unikatna številka povezave, ki določa komunikacijski kanal. V našem primeru potrebujemo dve unikatni številki povezave, eno za oddajno stran, drugo pa za sprejemno stran.
- CONNECTION_TYPE določa tip povezave, ki ga lahko nastavimo na vrednost 0x11, kar pomeni povezavo po protokolu TCP, ali vrednost 0x13, kar pomeni, da želimo komunicirati po protokolu UDP. V našem primeru uporabljamo protokol TCP, zato ima ta parameter vrednost 0x11.
- LOCAL_DEVICE_ID je parameter, ki funkciji sporoča model Siemensovega krmilnika. Za krmilnika Simatic S7 315-2 PN/DP in 317-2 PN/DP moramo parameter nastaviti na vrednost 0x02.
- LOCAL_TSAP_ID določa številko lokalnih vrat za komunikacijo. Podana je v obliki dveh bajtov, najbolj uteženega in najmanj uteženega. Če želimo nastaviti številko lokalnih na 2001, vpišemo v najbolj utežen bajt vrednost 0x07, v najmanj utežen bajt pa vrednost 0xD1, saj je vrednost 0x7D1 v desetiškem sestavu enaka 2001.

- LOCAL_TSAP_ID_LEN je parameter, ki določa dolžino parametra local_tsap_id. Ker je krmilnik pasiven komunikacijski partner, parameter nastavimo na vrednost 0x02.
- REM_STADDR predstavlja naslov IP oddaljenega partnerja, v našem primeru je nedoločen.
- REM_STADDR_LEN je dolžina naslova IP oddaljenega komunikacijskega partnerja, v našem primeru je dolžina naslova 0x00, ker je naslov IP oddaljenega partnerja nedoločen.
- ACTIVE_EST določa ali je krmilnik aktivni ali pasivni komunikacijski partner, v našem primeru je vrednost parametra 0x00, saj je krmilnik pasivni komunikacijski partner.

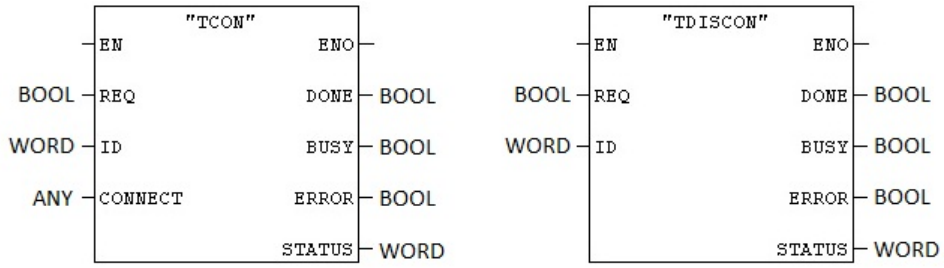
Vhodni parametri bloka TCON:

- REQ – je parameter, s katerim zahtevamo vzpostavitev povezav – ko parameter postavimo v logično stanje 1, potem krmilnik poskuša vzpostaviti povezavo s komunikacijskim partnerjem z nastavitvami, ki smo jih nastavili v podatkovni strukturi UDT65 TCON_PAR;
- ID – unikatni identifikator povezave oz. referenca na povezavo,
- CONNECT – parameter, s katerim podamo podatkovni blok, ki vsebuje podatkovno strukturo UDT65 TCON_PAR.

Izhodni parametri bloka TCON so:

- DONE,
- BUSY,
- ERROR in
- STATUS.

Ko je vrednost parametra DONE enaka logični 1, nam funkcija sporoča, da smo vzpostavili povezavo brez napak. Ko je vrednost parametra BUSY enaka logični 1, je vzpostavitev komunikacije še v teku. Ko je vrednost parametra ERROR enaka logični 1, nam funkcija sporoča, da je prišlo do napake pri vzpostavitvi povezave. Parameter STATUS je tipa beseda (ang. word), ki vrača trenutno stanje na povezavi ali natančnejši opis napake.



Slika 22: Komunikacijska bloka TCON in TDISCON.

5.1.2 Funkcija TDISCON (FB66)

Funkcijo TDISCON uporabljamo za prekinitev aktivne povezave. S parametrom ID funkciji sporočimo katero povezavo želimo prekiniti. Za prekinitev povezave moramo parameter REQ postaviti v logično stanje 1. S pomočjo parametrov DONE, BUSY in ERROR spremljamo dogajanje in vidimo, če je bila povezava uspešno prekinjena. Uporabnik dobi podrobnejše informacije v primeru napak ali aktivnosti na povezavi v parametru STATUS.

5.1.3 Funkcija TSEND (FB63)

Ko je povezava med partnerjema vzpostavljena, lahko pošiljamo in prejemamo podatke. Za pošiljanje podatkov uporabljamo vgrajeno funkcijo TSEND.

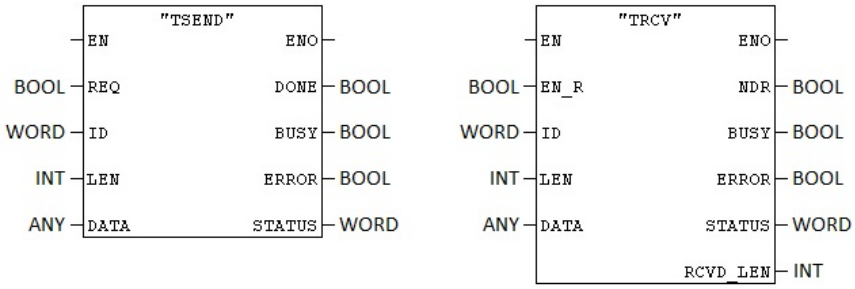
Funkcija TSEND ima štiri vhodne parametre:

- REQ - s postavitvijo parametra REQ v logično stanje 1, zahtevamo pošiljanje podatkov;
- ID - unikatni identifikator povezave oz. referenca na povezavo;
- LEN – število podatkov, ki jih želimo poslati, v bajtih;
- DATA – kazalec na podatkovno strukturo, ki vsebuje naslov podatkov in dolžino strukture podatkov, v enem pošiljanju lahko pošljemo toliko podatkov, kolikor je dolga struktura, vendar je največja dolžina omejena na 8192 bajtov.

Funkcija TSEND ima štiri izhodne parametre, ki nam sporočajo stanje funkcije:

- DONE – vrednost parametra DONE je enaka logični 1, ko je pošiljanje končano brez napak;

- BUSY - ko je vrednost parametra enaka logični 1, potem je pošiljanje podatkov še v teku;
- ERROR - ko je vrednost parametra enaka logični 1, je bilo pošiljanje neuspešno, saj je prišlo do napake;
- STATUS je parameter tipa beseda (ang. word), ki vrača trenutno stanje na povezavi ali natančnejši opis napake.



Slika 23: Komunikacijska bloka TSEND in TRCV.

5.1.4 Funkcija TRCV (FB64)

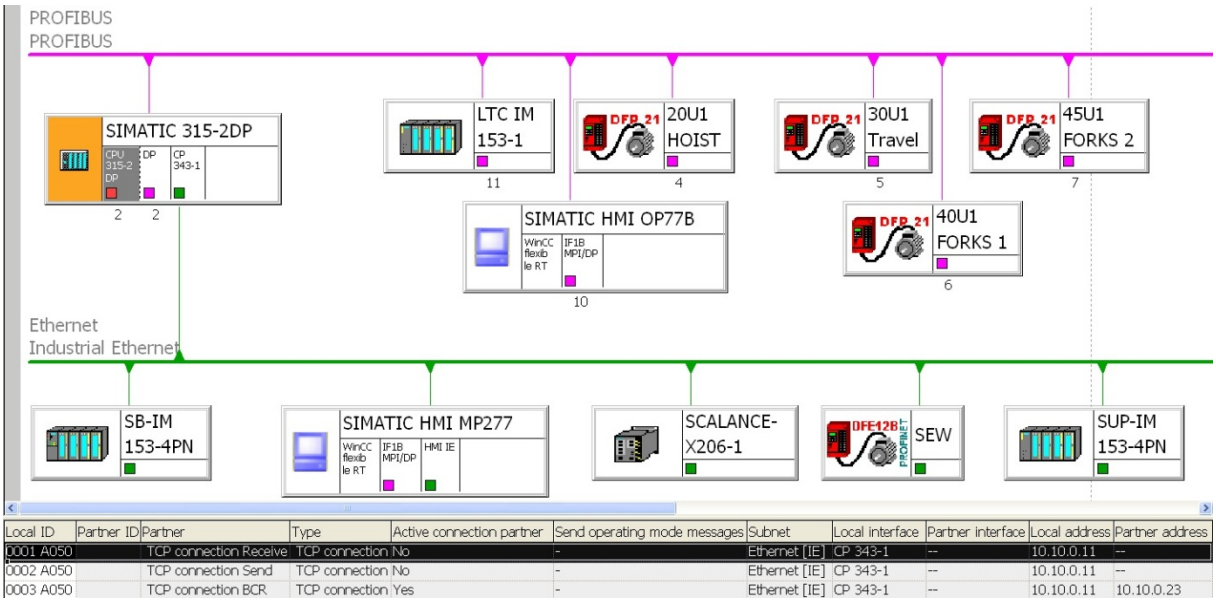
Za sprejem podatkov, ko je povezava že vzpostavljena, uporabljamo funkcijo TRVC. Sprejem nadziramo s parametrom EN_R. Ko je vrednost parametra EN_R enaka logični vrednosti 1, podatke lahko prejemamo.

S parametrom LEN lahko določamo dva principa delovanja funkcije. Če je vrednost parametra LEN enaka 0, potem je pričakovana dolžina podatkov določena s kazalcem na podatkovno strukturo, podano pri parametru DATA. Če je vrednost parametra LEN različna od nič, nam parameter LEN sporoča število podatkov, ki jih želimo sprejeti. V našem primeru je parameter LEN enak nič. Zahtevano število podatkov za sprejem sporočamo tako, da spreminjamo dolžino podatkovne strukture, podane s parametrom DATA.

Sprejem novih podatkov nam funkcija sporoči s parametrom NDR (ang. new data received), ki ga postavi v logično stanje 1. Dolžino prejetih podatkov preberemo iz parametra RCVD_LEN, ki je lahko manjša ali enaka kot dolžina, določena v parametru DATA. Parametri DONE, BUSY in ERROR nam sporočajo uspešnost sprejema. S parametrom STATUS nam funkcija vrača informacijo o dogajanju na povezavi in podrobnejšo informacijo v primeru napak.

5.2 Krmilnik serije S7-300 z modulom CP 343-1 Lean

Da bi bilo mogoče vzpostaviti povezavo med modulom CP in strežnikom MFCS, je potrebno modul pravilno nastaviti. To smo naredili v okolju Step7 z orodjem NetPro.

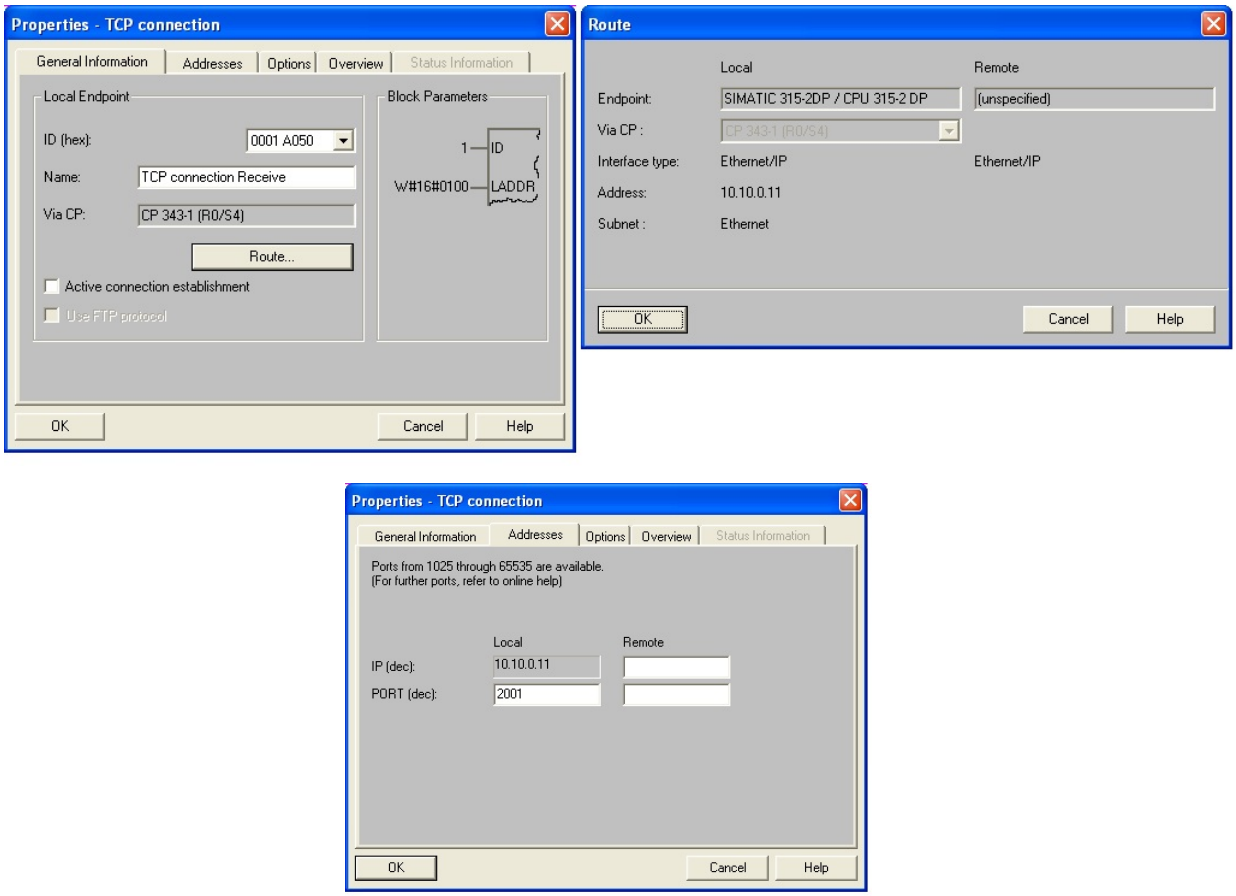


Slika 24: Orodje NetPro, nastavitev povezav s krmilnikom in napravami v omrežju Profibus in Industrijski Ethernet.

Na sliki 24 je prikazan primer konfiguracije omrežja v orodju NetPro. Gre za dvigalo v visokoregalnem skladišču. V konfiguraciji imamo poleg vodila Industrijski Ethernet (zelene barve) tudi vodilo Profibus (vijolične barve). Krmilnik prek omrežja Profibus komunicira s frekvenčnimi regulatorji za vožnjo, dvig in premik vilic. Na omrežju Profibus imamo priključeno tudi porazdeljeno vhodno/izhodno enoto IM153-1 in vmesnik človek-stroj OP77B. Prek omrežja Industrijski Ethernet krmilnik komunicira z dvema porazdeljenima vhodno/izhodnima enotama, mrežnim stikalom, frekvenčnim regulatorjem in vmesnikom človek-stroj MP277.

Povezavo med modulom CP in strežnikom MFCS, ki poteka po Industrijskem Ethernetu, je potrebno na krmilniku nastaviti na sledeč način.

Potrebno je dodati dve povezavi po protokolu TCP, kot je predstavljeno v spodnjem okviru na sliki 25. Eno povezavo potrebujemo za sprejemanje, drugo za oddajanje telegramov.



Slika 25: Nastavitev TCP povezave na modulu CP 343-1 Lean.

Na sliki 25 je prikazano tudi, kako nastavimo povezavo TCP za sprejemanje telegramov [11]. Pomembno je, da polja, ki se nanašajo na oddaljenega partnerja, v našem primeru je to strežnik MFCS, pustimo prazna. Vnesemo lokalni naslov IP modula CP, ki smo ga v prikazanem primeru nastavili na 10.10.0.11 in lokalna vrata za prejemanje telegramov, ki so 2001. Potrdimo izbrane nastavite in povezava za sprejemanje je ustvarjena. Ustvariti je potrebno tudi povezavo za oddajanje telegramov. Naredimo jo na enak način kot povezavo za prejemanje, podatke o oddaljenem strežniku ne vpisujemo, saj je oddaljen partner neznan (ang. Remote unspecified). Potrebno pa je nastaviti druga lokalna vrata za oddajanje telegramov (oznaka 2002).

Za komunikacijo po Industrijskem Ethernetu z modulom CP uporabljamo vgrajeni Siemensovi funkciji AG_SEND in AG_RECV, ki sta predstavljeni na sliki 26.

5.2.1 Funkcija AG_SEND

Funkcija AG_SEND ima pet vhodnih parametrov, navedenih v nadaljevanju.

- ACT – Parameter, s katerim sprožimo pošiljanje, če njegovo vrednost nastavimo na logično vrednost 1.
- ID – Unikatni identifikator povezave, referenca na povezavo.
- LADDR – Parameter tipa beseda (ang. word), v kateri nastavimo začetek oddajnega naslovnega polja modula CP. Naslovno polje nastavimo v orodju Step7, kjer konfiguriramo strojno opremo, v našem primeru je začetni naslov modula CP nastavljen na 255 oziroma 0x100.
- SEND – Kazalec na strukturo, ki jo bomo poslali prejemniku.
- LEN – Parameter, s katerim v bajtih podamo dolžino podatkov, ki jih želimo poslati, največja dovoljena dolžina je 8192 bajtov.

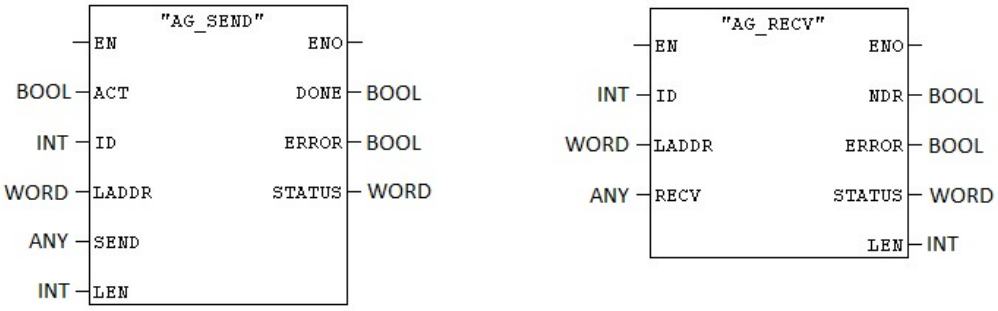
Funkcija AG_SEND ima tri izhodne parametre, navedene v nadaljevanju.

- DONE – Ko je vrednost parametra DONE v logičnem stanju 0, je pošiljanje v teku, zato ne moremo sprožiti novega pošiljanja. Stanje parametra 1 nam sporoča, da je pošiljanje končano brez napak.
- ERROR – Vrednost parametra ERROR v logičnem stanju 1 nam sporoča, da je prišlo do napake med pošiljanjem, zato podatki niso bili poslani.
- STATUS je parameter tipa beseda (ang. word), ki nam skupaj s parametroma DONE in ERROR podrobneje predstavi stanje pri pošiljanju in podrobnejši opis napake. Primer parametrov DONE, ERROR in STATUS je prikazan v tabeli 8.

Tabela 8: Pomen uporabljenih statusnih oznak funkcije AG_SEND.

DONE	ERROR	STATUS	Pomen
1	0	0x0000	Pošiljanje izvedeno brez napak.
0	1	0x7000	Pošiljanje je še v teku, ob klicu funkcije je bila vrednost parametra ACT 0.
0	1	0x8304	Povezava ni vzpostavljena.

Ko pokličemo funkcijo AG_SEND z vrednostjo parametra ACT = 1, pošljemo LEN bajtov podatkov iz strukture, ki smo jo navedli v parametru SEND. Če funkcijo pokličemo z vrednostjo ACT = 0, se vrednosti parametrov DONE, ERROR in STATUS osvežijo. Takrat jih lahko ovrednotimo in ustrezno ukrepamo.



Slika 26: Komunikacijska bloka AG_SEND in AG_RECV uporabljena za komunikacijo po Industrijskem Ethernetu z modulom CP 343-1 Lean.

5.2.2 Funkcija AG_RECV

Funkcija AG_RECV ima tri vhodne parametre:

- ID – Unikatni identifikator povezave, referenca na povezavo;
- LADDR je parameter tipa beseda (ang. word), s katerim nastavimo začetek sprejemnega naslovnega polja modula CP – naslovno polje nastavimo v orodju Step7, kjer konfiguriramo strojno opremo, v našem primeru je začetni naslov modula CP nastavljen na 255 oziroma 0x100;
- RECV je parameter, ki vsebuje kazalec na strukturo, v katero želimo zapisati prejete podatke.

Funkcija AG_RECV ima štiri izhodne parametre:

- NDR – (ang. new data received) Vrednost parametra je enaka logični vrednosti 1, ko prejmemo nove podatke;
- ERROR – parameter v logičnem stanju 1 nam sporoča, da je prišlo do napake pri sprejemu podatkov;
- STATUS je parameter tipa beseda (ang. word), ki nam s pomočjo parametrov NDR in ERROR podrobneje predstavi stanje pri sprejemu podatkov, primer je predstavljen v tabeli 9;

- LEN – v parameter LEN funkcija vrne število prejetih podatkov v bajtih.

Tabela 9: Pomen statusih oznak funkcije AG_RECV

NDR	ERROR	STATUS	Pomen
1	0	0x0000	Novi podatki prejeti.
0	1	0x8304	Povezava ni vzpostavljena.
0	1	0x80C4	Napaka v komunikaciji.

5.3 Krmilnik Simotion D425

Za komunikacijo med krmilnikom Simotion D425 in strežnikom MFCS so uporabljene funkcije _tcpOpenServer(), _tcpSend(), _tcpReceive() in _tcpCloseServer().

5.3.1 Funkcija _tcpOpenServer

Funkcijo uporabljamo za vzpostavitev pasivne komunikacije. Klic funkcije v uporabniškem programu je predstavljen na sliki 27.

```
status := _tcpOpenServer(                    //StructRetTcpOpenServer
   port := 2001                              //UINT, 1024-65535
  ,backLog := 1                              //DINT
  ,nextCommand := IMMEDIATELY                //EnumTcpNextCommandMode
);
```

Slika 27: Funkcija _tcpOpenServer.

Vhodni parametri funkcije.

- Port – Vrata prek katerih bo funkcija komunicirala, v našem primeru so nastavljena na vrednost 2001.
- BackLog – Največje dovoljeno število vzporednih povezav na izbranih vratih, v našem primeru imamo dvoje vrat in dva kanala, zato je dovolj, da dovolimo eno povezavo na vratih.
- NextCommand – parameter, ki nam omogoča nastaviti, kdaj se bo funkcija konfigurirala. Imamo dve možnosti:
  (1) IMMEDIATELY – takojšnja konfiguracija,

(2) WHEN_COMMAND_DONE – konfiguracija, ko je ukaz izveden.

V našem primeru vedno uporabljamo takojšnjo konfiguracijo – IMMEDIATELY.

Izhodna parametra funkcije sta sledeča:

- Status je parameter funkcije, ki nam sporoča stanje funkcije in morebitne napake.
- ConnectionId je unikatni identifikator povezave, ki se uporablja kot vhodni parameter funkcij _tcpSend() in _tcpReceive(), s katerim sporočimo, po katerem komunikacijskem kanalu bomo podatke pošiljali ali prejemali.

5.3.2 Funkcija _tcpSend()

Funkcijo _tcpSend() uporabljamo za pošiljanje podatkov. Predstavljena je na sliki 28.

```
status := _tcpSend(                                  // DINT
    connectionId := _tcpOpenServer.ConnectionId      // DINT
   ,nextCommand := IMMEDIATELY                       // EnumTcpNextCommandMode
   ,dataLength := 4096                               // UINT, 0-4096
   ,data := Send_bufferS                             // ARRAY [0..4095] OF BYTE
);
```

Slika 28: 5Funkcija _tcpSend().

Parametri funkcije so navedeni v nadaljevanju.

- ConnectionId – unikatni identifikator povezave, vrednost dobimo iz funkcije _tcpOpenServer(), ter tako povežemo funkcijo _tcpSend() z odprtim komunikacijskim kanalom.
- NextCommand – uporabljamo takojšnja konfiguracijo, IMMEDIATELY, kot je razloženo pri funkciji _tcpOpenServer().
- DataLength je parameter, s katerim sporočamo funkciji koliko bajtov podatkov želimo poslati.
- Data – ime strukture polja podatkov tipa bajt, kjer imamo podatke, ki jih želimo poslati. Dolžina polja je omejena na največ 4096 bajtov.
- Status je izhodni parameter funkcije, ki nam sporoča ali je pošiljanje podatkov potekalo brez napak. V primeru napak je vrednost parametra status negativna, moramo poklicati funkcijo _tcpCloseServer(), da prekinemo povezavo.

5.3.3 Funkcija _tcpReceive()

Funkcijo _tcpReceive() uporabljamo za prejemanje podatkov. Parametre ima enake kot funkcija _tcpSend(). Za razliko od funkcije _tcpSend(), bomo pri funkciji _tcpReceive() v polje, ki ga navedemo pri parametru data, podatke prejeli.

5.3.4 Funkcija _tcpCloseServer()

Funkcijo _tcpCloseServer() uporabimo za prekinitev povezave, ki je bila pasivno vzpostavljena.

```
status := _tcpCloseServer(port := 2001);   //1024 – 65535
```

Slika 29: Funkcija _tcpCloseServer().

Pri klicu funkcije moramo podati vrata, na katerih je vzpostavljena povezava, ki jo želimo prekiniti (slika 29). Funkcija nam v parameter status sporoči, če je prišlo med izvajanjem funkcije do napak ali pa je bila povezava uspešno prekinjena.

5.4 Kontrolne zastavice

Za upravljanje in nadzor nad komunikacijo smo implementirali pet kontrolnih zastavic, ki so skupne za oddajno in sprejemno stran. Zastavice smo poimenovali telegram prejet, pošlji telegram, pošiljanje ni v teku, napaka pri oddajanju in napaka pri sprejemanju.

Kontrolna zastavica - telegram prejet - nam v logičnem stanju 1 sporoča, da smo v podatkovno strukturo za sprejem telegrama prejeli nov telegram. Ko preberemo prejeti telegram iz podatkovne strukture, moramo sami postaviti zastavico v logično stanje 0, s tem funkciji sporočimo, da je telegram prevzet in lahko v sprejemno podatkovno strukturo sprejmemo novega.

Ko želimo telegram poslati, napolnimo podatkovno strukturo za pošiljanje z želenimi podatki in nastavimo zastavico - pošlji telegram - v logično stanje 1, da telegram odpošljemo. Funkcija sama poskrbi za postavitev zastavice v logično stanje 0.

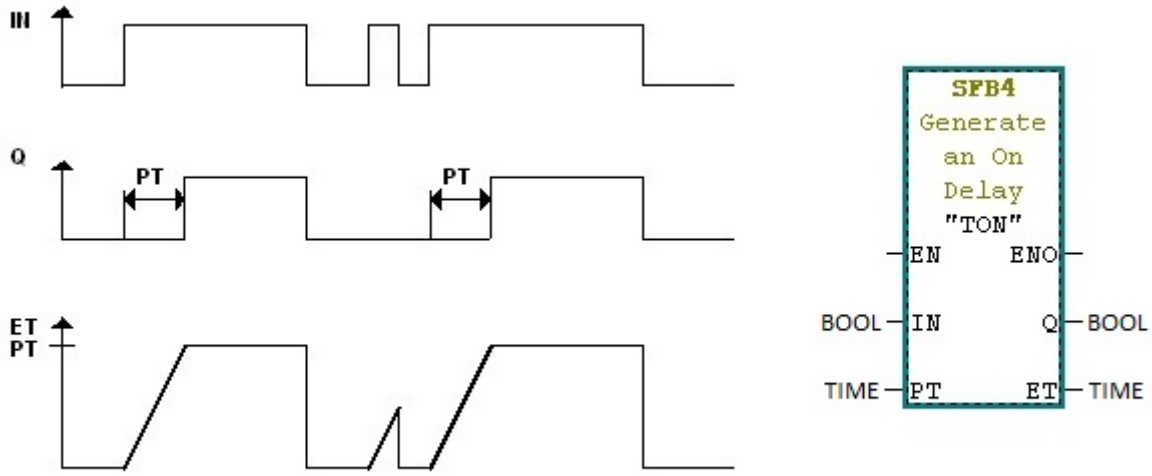
Zastavica - pošiljanje ni v teku - ima logično vrednost 0 med pošiljanjem telegrama. Ko funkcija ne pošilja in čaka na pošiljanje, ima zastavica logično vrednost 1.

Na oddajni strani lahko pride do različnih napak, v primeru katerekoli napake, se zastavica - napaka pri pošiljanju - postavi v logično stanje 1. Napake, ki se lahko pojavijo so: napaka, ki nam jo sporoči vgrajena funkcija za pošiljanje, potrditveni telegram ima lahko napačno parnost in zaporedno številko, časovnik za potrditveni telegram se izteče, če potrditvenega telegrama predolgo ne dobimo, napako imamo tudi v primeru, če je kanal za pošiljanje predolgo zaseden. Zastavica napaka pri pošiljanju nam sporoča, da je prišlo do napake pri oddaji telegrama. Ker telegram v primeru napake ni poslan, ne povečamo števca zaporedne številke telegrama.

Na sprejemni strani lahko pride do napake pri pošiljanju potrditvenega telegrama, kar nam sporoči vgrajena funkcija za pošiljanje, prejeti telegram ima lahko napačno parnost in zaporedno številko. Na sprejemni strani nam napako povzroči tudi prejeti podvojeni telegram, to je telegram enak prejšnjemu z enako zaporedno številko. V primeru katerekoli navedene napake, se zastavica - napaka pri sprejemanju - postavi v logično stanje 1, tako nam funkcija sporoča, da je prišlo do napake pri sprejemanju telegrama. V primeru napake pri sprejemu posebni ukrepi niso potrebni, zato samo ponastavimo vrednosti parametrov in čakamo, da nam bo strežnik MFCS poslal nov telegram.

5.5 Časovniki

Časovniki ob izpolnjenem pogoju merijo čas. V našem projektu smo uporabili tri časovnike z zakasnjenim vklopom – TON (ang. Timer ON delay). Funkcija TON ima dva vhodna in dva izhodna parametra, prikazana je na sliki 30. Vhodni parameter IN je pogoj za merjenje časa, ko je le-ta v visokem logičnem stanju 1, časovnik odšteva nastavljen čas. Koliko časa bo odšteval, nastavimo s parametrom PT. Ko nastavljen čas preteče, se izhodni parameter Q postavi v visoko logično stanje 1. Tako nam funkcija sporoči, da se je časovnik iztekel. V izhodnem parametru ET funkcija meri pretečeni čas, pove nam, koliko časa je pogoj za merjenje časa izpolnjen. V primeru, da parameter IN pred iztekom merjenja časa pade v nizko logično stanje, se vrednost pretečenega časa ET ponastavi. Ob ponovnem prehodu v visoko logično stanje 1, se začne ponovno odštevanje časa od začetka za celoten čas nastavljen v parametru PT.



Slika 30: Časovnik z zakasnjenim vklopom – TON.

5.5.1 Časovnik za vzpostavitev komunikacijskega kanala

Preden telegram odpošljemo vedno preverimo, če je komunikacijski kanal prost. V primeru, da kanal ni prost, smo programsko opremo zasnovali tako, da imamo na voljo 100 ms, da kanal vzpostavimo. V primeru, da kanala v razpoložljivem času ne vzpostavimo, postavimo zastavico - napaka pri oddajanju - v stanje 1. Časovnik smo nastavili tako, da smo parameter PT nastavili na vrednost 100 ms.

5.5.2 Časovnik potrditvenega telegrama

Časovnik potrditvenega telegrama uporabljamo za merjenje časa od trenutka, ko smo normalni telegram poslali, do prejema potrditvenega telegrama. Časovnik smo nastavili tako, da smo parameter PT nastavili na vrednost 5 s, kar je prikazano na sliki 31. V primeru, da v nastavljenem času 5 s potrditvenega telegrama ne prejmemo, funkcija ponovno pošlje enak telegram.

```
Timer_Ack(IN := Send_stateS = 6, PT := T#5s);

CASE Send_stateS OF

6:  // potrditve predolgo ni, ponovimo prenos
    IF Send_StateR = 4 THEN
      Send_StateS := 7;
    ELSIF Timer_Ack.Q THEN
      Send_transmit_data := false;
      Error.Send_timer_ack := true;
      Send_StateS := 4;
    END_IF;
```

Slika 31: Uporaba časovnika potrditvenih telegramov.

Pogoj, da časovnik začne odštevati čas je, da je avtomat na oddajni strani v stanju SEND_STATES = 6, saj v tem stanju čaka, da sprejemni avtomat sprejme potrditveni telegram in preide v stanje SEND_STATER = 4.

5.5.3 Časovnik za ohranjanje aktivne povezave

V času, ko normalnih telegramov ne pošiljamo, moramo še vedno poskrbeti, da je povezava s komunikacijskim partnerjem aktivna. Povezavo vzdržujemo aktivno tako, da pošiljamo telegrame aktivnosti. Na strani krmilnika skrbimo za aktivnost oddajne strani. Neaktivnost povezave merimo s časovnikom dolžine 20 s, ko se časovnik izteče, sistem pošlje telegram aktivnosti povezave.

5.6 Posebnosti v programski kodi

Da bi bila programska oprema na vseh platformah enotna, smo razvili del kode za prejemanje števila podatkov glede na zahtevo, ki jo prejmemo iz avtomata stanj na sprejemni strani. Ko na sprejemni strani v avtomatu zahtevamo nove podatke, bodisi glavo, telo ali samo dodaten bajt zaradi pomika glave v levo, v spremenljivko Recv_request_bytes zapišemo število zahtevanih bajtov.

```
IF Recv_request_bytes > 0 THEN
  // poberemo že prebrane podatke podatkov
  IF Recv_bufferR_index < Recv_Rlength THEN
    IF Recv_bufferR_index + Recv_request_bytes > Recv_Rlength THEN
        bytes_to_copy := Recv_Rlength - Recv_bufferR_index;
    ELSE
        bytes_to_copy := Recv_request_bytes;
    END_IF;
    IF (Recv_telegram_index + bytes_to_copy - 1) < LEN_TELEGRAM THEN
      FOR i := 0 TO bytes_to_copy - 1 DO
        Recv_telegram[Recv_telegram_index+i] := Recv_bufferR[Recv_bufferR_index+i];
      END_FOR;
      Recv_telegram_index := Recv_telegram_index + bytes_to_copy;
      Recv_bufferR_index := Recv_bufferR_index + bytes_to_copy;
      Recv_request_bytes := Recv_request_bytes - bytes_to_copy;
    END_IF;
  END_IF;
  // zmanjkalo podatkov
  IF Recv_Rlength = 0 OR Recv_bufferR_index >= Recv_Rlength THEN

    ptr := Recv_bufferR;
    ptr_struct.Count := INT_TO_WORD(Recv_request_bytes);

    temp_length := 0;
    AG_RECV(ID :=  Channel_RemoteToLocal,
            LADDR :=  LADDR,
            RECV :=  ptr,   //Recv_bufferR,
            NDR :=  Recv_Rndr,
            ERROR :=  Recv_Rerror,
            STATUS :=  temp_status,
            LEN :=  temp_length);

    IF Recv_Rerror THEN
      Recv_Rstatus := temp_status;
    ELSIF Recv_Rndr THEN
      Recv_Rndr := false;
      Recv_Rlength := temp_length;
      Recv_Rstatus := temp_status;
      Recv_bufferR_index := 0;
      Error.Recv_connection := false;
    END_IF;
  END_IF;
END_IF;
```

Slika 32: Del kode, ki glede na potrebe, sprejema podatke.

Programska koda na sliki 32 poskrbi, da se zahtevano število bajtov prenese v strukturo prejetega telegrama Recv_telegram[].

Če imamo v medpomnilniku prejetih podatkov Recv_bufferR[] zahtevano število podatkov, potem jih prenesemo v strukturo prejetega telegrama Recv_telegram[], drugače pa kličemo

funkcijo AG_RECV() in nastavimo dolžino strukture Recv_bufferR[] na število podatkov, ki jih potrebujemo.

Funkciji AG_SEND() in AG_RECV() vračata telegrame celotne dolžine, zato sestavljanje telegramov ni potrebno, vendar smo se zaradi poenotenja kode na vseh tipih strojne opreme odločili, da tudi pri delu z modulom CP uporabimo enak pristop.

5.7 Uporaba programske opreme na primeru

Programerju, ki uporablja našo programsko opremo, ni potrebno poznati vseh njenih podrobnosti. Na sliki 33 je prikazan del kode, ki skrbi za sprejem telegrama 1201. V kodi spremljamo stanje zastavice telegram prejet (ang. data received). Ko nam zastavica sporoči, da smo prejeli nov telegram, preverimo, če je ta tipa 1201. Če ugotovimo, da je tip pravi, telegram prenesemo iz strukture programske kode za sprejem telegrama v strukturo, ki jo imamo rezervirano v uporabniškem programu. Sprejemno polje funkcije sprostimo tako, da zastavico telegram prejet ponastavimo v logično stanje 0.

```
IF TCPIP.Control.Data_received THEN
  retval := BLKMOV( SRCBLK := TCPIP.Recv_telegram,
                    DSTBLK := MFCS.Telegram.headerRecv);
  // 1201
  IF MFCS.Telegram.ReadyFor1201 AND MFCS.Telegram.headerRecv.telegramType = 1201 THEN
    retval := BLKMOV( SRCBLK := TCPIP.Recv_telegram,
                      DSTBLK := MFCS.Telegram.Crane.TransportOrder);
    TCPIP.Control.Data_received := FALSE;
    FC_Telegram1201Syntax();
    Crane.MFCS.TOSyntaxBad := MFCS.Telegram.Crane.TransportOrder.ErrorCode <> 0;
    IF NOT Crane.MFCS.TOSyntaxBad THEN
      IF MFCS.Telegram.Crane.TransportOrder.Command = 99 THEN // delete
        MFCS.Command.Crane.TODelete := TRUE;
      ELSIF Crane.MFCS.TOBuffer THEN // buffer full
        Crane.MFCS.TOBufferFull := TRUE;
      ELSE
        Crane.MFCS.TOTelegramReceived := TRUE;
      END_IF;
    END_IF;
  END_IF;
```

Slika 33: Primer sprejema telegrama 1201 v programski kodi.

Nadaljnja obdelava je odvisna od tipa telegrama in je od telegrama do telegrama različna. Funkcija FC_Telegram1201Syntax() preveri zgradbo telegrama in v primeru napake v polje vpiše dogovorjeno kodo napake. Pri telegramu 1201 preverimo tudi, če je prišel ukaz 99, ki pomeni brisanje vseh nalog dvigala. Če prenesemo telegram brez napak, dvigalu s spremenljivko telegram prejet (ang. telegram received) sporočimo, da ga čaka nova naloga.

6 Sklepne ugotovitve

Cilj diplomske naloge je bil izdelati programsko kodo, ki omogoča komunikacijo med krmilnikom in višjenivojskim sistemom, na strani krmilnika. V okviru rešitve je bilo potrebno določiti tudi zgradbo telegramov, ki se uporabljajo za komunikacijo med partnerjema. Zagotoviti je bilo potrebno, da se telegrami ne izgubljajo. Delo je potekalo na različnih tipih strojne opreme. Za potrebe diplomske naloge smo uporabili dva Siemensova krmilnika različnih družin in razširitveni komunikacijski modul CP. Siemensovi orodji, Simatic Manager in Simotion Scout, sta se izkazali kot učinkoviti in zmogljivi. Kljub temu, da sem se ukvarjal s komunikacijo med krmilniki dvigal in višjim sistemom vodenja, sem dobil veliko znanj tudi s področja načrtovanja in razvoja visokoregalnih skladišč ter programiranja krmilnikov dvigal.

Zastavljeno nalogo smo uspešno rešili in komercialno uporabili na dveh projektih. Programska koda na krmilnikih Simotion D425 deluje v praksi, v visokoregalnem skladišču brezcarinske prodajalne v Dubaju na enajstih dvigalih. Skladišče uspešno obratuje od avgusta 2012. Programska koda deluje tudi na krmilniku Simatic S7 CPU315-2 PN/DP, z modulom CP, v Turčiji, v visokoregalnem skladišču podjetja Üçge, od aprila 2012.

Programska koda je bila zasnovana z namenom uporabe v visokoregalnih skladiščih, vendar bi jo kljub temu lahko uporabili kjerkoli, kjer bi bila potreba po komunikaciji med krmilniki in višjenivojskimi sistemi. Za komunikacijo v drugih industrijskih procesih, bi bilo potrebno le prirediti zgradbo telegramov, da bi vsebovali podatke, ki bi jih v zahtevanem procesu potrebovali.

Na krmilniku Simatic, z vgrajenim vmesnikom Ethernet, programska koda ne deluje, kot bi si želeli, saj v primeru prekinitve povezave sistem sam ne more ponovno vzpostavi komunikacijskega kanala med krmilnikom in višjenivojskim sistemom. Težava je nastala zaradi vgrajenih funkcij, ki se uporabljajo za komunikacijo, zato na to nimamo vpliva. Zaradi težave krmilnika, ki ima vgrajenim modulom Ethernet, le-tega ne uporabljamo na projektih.

V praksi se je pokazalo, da bi bilo programsko kodo možno izboljšati. Glavo telegrama bi v toku podatkov hitreje in z večjo verjetnostjo našli, če bi imeli podatek o pariteti izračunan samo nad podatki glave. Glavo telegrama bi bilo potrebno podaljšati za besedo, ki bi vsebovala izračunano pariteto. Telegram bi imel tedaj dve besedi s pariteto, eno besedo, ki bi vsebovala pariteto podatkov glave in drugo, ki bi vsebovala pariteto podatkov celotnega telegrama.

7 Literatura

[1] Iskra Impuls, Tehnična dokumentacija projekta Dubaj DDF, marec, 2012.

[2] Iskra Impuls, Tehnična dokumentacija projekta Üçge, februar, 2012.

[3] Iskra Impuls, TCP/IP – Driver for Ethernet with NGKP2, marec, 2005.

[4] Iskra Impuls, General Issues of Telegram Specification, november, 2009.

[5] Iwanitz, F., Lange, J. (2001). OLE for process control: fundamentals, implementation, and application, Hüting, Heidelberg.

[6] Killian, G., Weingmann, J. (2003). Decentralization with PROFIBUS DP/DPV1: architecture and fundamentals, configuration and use with SIMATIC S7, Publicis Corporate Pub, Erlangen.

[7] Strmčnik, S. (1988). Celostni pristop k računalniškemu vodenju procesov, Fakulteta za elektrotehniko, Ljubljana.

[8] (2/2012) SIMOTION SIMOTION D4x5-2 Dosegljivo na: https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objID=16512438&subtype=133300

[9] (6/2004) S7-CPs for Industrial Ethernet – CP 343-1 Lean Dosegljivo na: http://cache.automation.siemens.com/dnl/Tk/Tk2MDMwMwAA_19308657_HB/mn_cp343-1-lean_76.pdf

[10] (2008) CPU 314C-2 PN/DP Dosegljivo na: https://eb.automation.siemens.com/goos/catalog/Pages/ProductData.aspx?nodeID=10088673&language=en&regionUrl=/&activeTab=product#activetab=product&

[11] (9/2007) S7-CPs for Industrial Ethernet Configuring and Commissioning Dosegljivo na: https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=21555710&caller=view

[12] (11/2010) SIMOTION SCOUT Communication System Manual Dosegljivo na: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CC4QFjAA&url=http%3A%2F%2Fsupport.automation.siemens.com%2FWW%2Fllisapi.dll%2Fcsfetch%2F51786340%2FCommunication_en-US.pdf%3Ffunc%3Dcslib.csFetch%26nodeid%3D51786348%26forcedownload%3Dtrue&ei=2wNbUYmBIcrGtAa5p4CQBw&usg=AFQjCNH637onTcqNK7yuZZzLTvNw9KfDnw&sig2=xitLvRNI6aAql67ITiCxoQ&bvm=bv.44697112,d.Yms&cad=rja