UNIVERZA V LJUBLJANI
Fakulteta za elektrotehniko

Matija Dirjec

Pametna hiša

UNIVERZITETNO DIPLOMSKO DELO

Mentor: doc. dr. Boštjan Murovec

Ljubljana, december 2005

ZAHVALA

Zahvalil bi se doc. dr. Boštjanu Murovcu za vso podporo, nasvete in pomoč pri pisanju diplomskega dela ter Vesni Peršič za izkazano podporo in pomoč.

KAZALO:

Povzetek....6
Abstract....7
1 Uvod....8
2 Predstavitev možnosti avtomatizacije....11
2.1 Delitev glede na različne načine komunikacije....11
2.1.1 Prenos po posebnih žicah....11
2.1.2 Prenos informacij po električnem omrežju....12
2.1.3 Brezžični prenos....12
2.2 Standard X-10....13
2.3 Standard CEBus....13
2.4 Standard EIB - European Instalation Bus....13
3 Predstavitev možnih sistemov avtomatizacije, ki se pojavljajo na slovenskem trgu....14
3.1 Domača podjetja in sistemi....14
3.1.1 Robotina....14
3.1.2 Indata....14
3.1.3 GOAP sistemi....14
3.2 Uporaba tujih sistemov....15
3.2.1 Dexus....15
3.2.2 Prelog....15
3.2.3 Liko Pris....15
3.2.4 Silon....15
4 Izbira PLK-ja....16
4.1 SyBro PLK....18
4.1.1 Tehnični podatki SyBro PLK....18
4.2 Predstavitev programskega paketa CyPro....20
4.2.1 Načini programiranja....20
4.2.2 Izbira tipa spremenljivk....24
5 Upravljanje stanovanjske hiše s pomočjo PLK-ja....26
5.1 Upravljanje s hišo....26
5.1.1 Prižiganje in ugašanje luči....26
5.1.1.1 Razlaga programa za luč....26
5.1.1.2 Razlaga programa za luč na stopnicah s časovnikom....28
5.1.2 Upravljanje peči za centralno kurjavo ali ogrevanje....29
5.1.2.1 Razlaga programa za krmiljenje ogrevanja....29
5.1.3 Upravljanje rolet....31
5.1.3.1 Razlaga programa za upravljanje rolet za mraz....31
5.1.3.2 Razlaga programa za upravljanje rolet ob vročini....33
5.2 Varnost pred vlomi....34
5.2.1 Alarm....34
5.2.1.1 Razlaga programa za alarm....34
5.2.1.2 Razlaga programa za utripanje luči ob alarmu....36
5.3 Varčevanje z energijo in vodo....38
5.3.1 Varčevanje s toploto....38
5.3.2 Varčevanje z električno energijo....38
5.3.3 Varčevanje z vodo....38
5.4 Varovanje stanovanjske hiše in stanovalcev pred naravnimi nesrečami....39
5.4.1 Varovanje pred neurji....39
5.4.1.1 Razlaga programa za zaznavanje vremenskih pogojev....39
5.4.1.2 Razlaga programa za ravnanje ob dežju....40

5.4.1.3 Razlaga programa za ravnanje ob močnem vetru....41
5.4.2 Varovanje pred poplavami....42
5.4.2.1 Razlaga programa za ravnanje ob primeru poplave....42
5.4.2.2 Razlaga programa za upravljanje črpalk....42
5.5 Nadgradnja sistema z nadzornim SCADA sistemom....43
5.6 Nadgradnja s krmiljenjem na daljavo z pomočjo GSM SMS sporočil....44
6 Sklepne ugotovitve....45
7 Viri....46
Literatura....46
Internetni viri....46

KAZALO SLIK:

Slika 1: Iskanje izgubljenih predmetov....8
Slika 2: Celostna zasnova pametne hiše....9
Slika 3: SyBro PLK....18
Slika 4: Način programiranja instruction list....20
Slika 5: Način programiranja v struktuiranem tekstu....22
Slika 6: Program z lestvičnim diagramom....23
Slika 7: Program za programiranje PLK-ja....24
Slika 8: Okno za vstavljanje nove spremenljivke....24
Slika 9: Pulzni časovnik....25
Slika 10: Časovnik z zakasnitvijo....25

KAZALO TABEL:

Tabela 1: Tabela numeričnih ukazov....20

SLOVAR:

PLK
Programabilni logični krmilnik

SCADA
Nadzorni grafični program

PC
Osebni računalnik

RS-232
Komunikacijski standard, ki nam določa komunikacijska vrata in način komuniciranja

RS-422
Komunikacijski standard, ki nam določa komunikacijska vrata in način komuniciranja

LADDER
Programski jezik, ki ga uporabljajo PLK-ji

RV
Relativna vlažnost

Timer
Časovnik

I/O
Vhodi in izhodi (inputs and outputs)

Reset
Ponastavitev

PID
proporcionalni, integrirni in diferencialni

Povzetek

V okviru diplomskega dela smo raziskali možnosti za izvedbo avtomatizirane stanovanjske hiše z uporabo programabilnega logičnega krmilnika (PLK). Na začetku so predstavljene opcije, s katerimi se da avtomatizirati hišo. To so namenski sistemi za avtomatizacijo hiš, ki se jih ne da nadgraditi, hotelski moduli za avtomatizacijo hotelskih sob, ki so izpeljanke PLK-jev in industrijski PLK. Predstavljena je tudi ponudba na slovenskem trgu in podjetja, ki v Sloveniji samo implementirajo tuje sisteme ter podjetja, ki imajo svoj razvoj in vizijo za prodor na tuje trge. Nekaj teh slovenskih podjetij je na tujih trgih že zelo uspešnih.
Predstavljen je programski paket CyPro, ki se ga uporablja za programiranje PLK-jev znamke CyBro, ki jih delajo v koprski Robotini.
Predstavljeno je upravljanje s hišo, kamor spada prižiganje in ugašanje luči, regulacija ogrevanja in upravljanje z roletami. V drugem delu je predstavljena varnost pred vlomilci, kamor spada alarm in upravljanje hiše med alarmom. Sistema za upravljanje hiše in varnost pred vlomilci sta nadgrajena z varčevanjem z energijo in z varčevanjem z vodo. Poleg tega je predstavljeno tudi varovanje hiše in stanovalcev pred vremenskimi nevšečnostmi, kot so viharji in poplave. Pri vseh teh primerih so dodani izseki programskih kod, ki bi se jih lahko uporabilo v praksi. Kode so narejene za po eno luč, črpalko, roleto, itd.. Če bi želeli avtomatizirati hišo s pomočjo teh programskih kod, bi morali samo dodati vhodno izhodne spremenljivke in jih dopisati v programsko kodo na mesta, ki so za to predstavljena. Lahko bi imeli do 256 spremenljivk, ki bi jih samo vstavili v programsko kodo. Če bi želeli več spremenljivk, bi morali dodati še en PLK, kar pomeni, da bi morali malo predelati programsko kodo, saj bi morali dodati komunikacijo med obema PLK-jema.
Na koncu je predstavljena še možnost nadgradnje z nadzornim SCADA sistemom in krmiljenje na daljavo s pomočjo GSM SMS sporočil.
S pomočjo tega diplomskega dela smo prišli do ugotovitev, da je avtomatizacija smiselna in rentabilna samo do neke meje. Večja stopnja avtomatizacije stanovanjske hiše bi bila lahko tudi problematična, saj se tak sistem lahko pokvari in onemogoči normalno uporabo stanovanjskega objekta.

Abstract

The main focus of this thesis is researc of possibilities for implementation of an automated dwelling house by using programmable logical controller (PLC). At the beginning options for automatic control of a house are presented. This are systems specially designed for automatization of houses that are upgradeable, hotel modules for automatization of hotel rooms. This systems are derivatives of PLC and industrial PLC. Also the supply on Slovene market is presented, including companies that only implements foreign systems in Slovenia and those Slovene companies that have own development and vision how to be succesful on foreign markets as well. Some of those Slovene companies are already very successful abroad.

CyBro programme packet that is used for programming PLCs variety CyBro made in Robotina company from Koper is presented in detail.

Furthermore house managing that includes switching the lights on an off, heating regulation and managing window blinds is presented. In the second part security from burglars that includes alarm and managing the house during it is presented. Systems for house managing and security are upgraded with energy and water saving. Besides protecting the house and its residents from weather inconveniences like storms and floods are presented The examples are accompanied with programming excerpts that can be used in practice. Namely, one example code for controlling light, pump, window blind, etc. is included. If the automation of a house is to be done with these program excerpts, the slight adoption to the real situation needs to be done by adding proper input-output variables at the places that are indicated in the code. Specifically, up to 256 variebles can be added. If more variables are needed, we can expand the presented system with additional PLC which also implies slight adoption of the program code and inclusion of a mechanism for communication among both PLCs.

Thirdly, possibilities for upgrading the described functions with controlling SCADA system and remote access with GSM SMS messages are described.

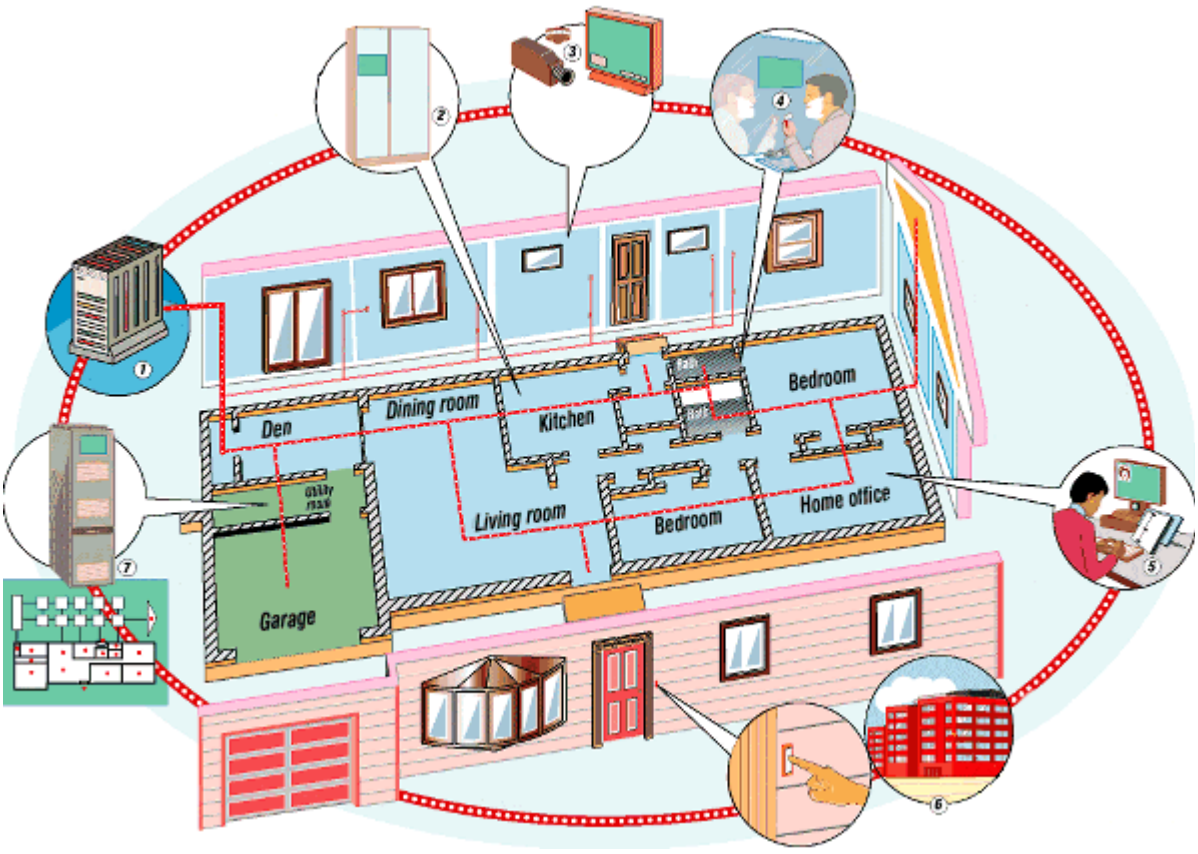
With the help of this diploma work we found out that automatization makes sense and is profitable only up to a certain point. More automatization of a dwelling house could cause problems because system like that can break down and makes normal use of a dwelling house impossible.

1 Uvod

V diplomskem delu je prikazana realizacija avtomatizirane pametne stanovanjske hiše. Realizirana je s pomočjo CyPro [1] PLK-ja in je dovolj inovativna ter cenovno sprejemljiva za povprečnega slovenskega lastnika stanovanjske hiše. PLK, ki ga uporabljamo v tem diplomskem delu, je standardni manjši industrijski PLK, ki je zanesljiv, modularen, cenovno ugoden in lahko dostopen na slovenskem trgu.
Za avtomatizacijo stanovanjske hiše smo imeli na voljo tudi module za avtomatizacijo hotelskih sob, ki so namensko razviti za funkcije, potrebne v hotelskih sobah. Bili smo mnenja, da ti moduli niso primerni, poleg tega so tudi cenovno neracionalni za uporabo pri avtomatizaciji stanovanjske hiše. Problem bi bil z njihovimi odvečnimi funkcijami in tudi s tem, da bi morali za vsako sobo uporabiti po en modul, ki bi mu ukazoval nadzorni modul.
Problem teh modulov je tudi, da je nadzorna plošča, ki jo ima večina teh sistemov, zelo toga. Primer: na nadzorni plošči za temperaturo je ponavadi izpisana temperatura v črticah, zaradi česar ne moremo nastaviti temperature na 23 °C, ampak samo po občutku po črticah. Razlog je v tem, da se lahko stranka v hotelu pritoži in zahteva povrnitev stroškov, če v sobi ni tako toplo, kot piše na nadzorni plošči. Če je temperatura v črticah, nihče ne more trditi, da ni toplo za dve črtici.
Proizvajalci hotelskih modulov zagovarjajo uporabo takih sistemov s tem, da bi bilo drugače preveč stroškov ožičenja. Po našem mnenju pri avtomatizaciji ene stanovanjske hiše še ne pride do problema z ožičenjem, saj je povprečna stanovanjska hiša v Sloveniji velika kot večji apartma v boljših hotelih. Pri teh modulih tudi ni vseh funkcij, ki jih lahko narediš in nadomestiš z industrijskim PLK-jem.
Sistemi inteligentnih hiš se glede na zahtevnost in namen med seboj precej razlikujejo. Razlikujejo se po namenih in sredstvih, ki se jih uporabi za avtomatizacijo. Najbolj so uveljavljene enostavne različice, namenjene ljudem z raznimi zdravstvenimi težavami, kot so pozabljivost in razne oblike invalidnosti. To so predvsem enostavnejši načini avtomatizacije, ki se jih da realizerati že z manjšimi mikrokrmilniki.
En primer takšnega projekta je bil predstavljen leta 2000 v ZDA kot sodelovanje dobrodelne organizacije Dementia Voice [II], združenja Housing 21 in Bath Institute of Medical Engineering. Poudarek je bil na varnosti hiše in njene pomoči pri iskanju različnih predmetov, kot so torbice, očala in ključi. Ostale predstavljene funkcije so bile še izklop kuhalnika, ko izhlapi vsa voda, preprečevanje puščanja odprtih pip, regulacija temperature v kopalni kadi, sledenje osvetlitve pri gibanju po hiši in podobno.

Slika 1: Iskanje izgubljenih predmetov.

Večji projekti, kot ga prikazuje slika 2, se koncepta pametne hiše lotevajo precej bolj celovito. Sistem je zasnovan na nivoju osebnega računalnika in ne na nivoju mikrokrmilnika, ki ima premajhno procesorsko zmogljivost. Človek je v kontaktu s svojim domovanjem tudi ko je v službi ali na dopustu, ker komunikacija poteka preko interneta.



Slika 2: Celostna zasnova pametne hiše.

Nekaj poglavitnih lastnosti takšnega objekta:

- pametni hladilnik z zaslonom na dotik, bralnikom črtne kode živil, bazo receptov, naročanjem artiklov prek interneta in drugo,
- senzorji in kamere za ugotavljanje prisotnosti oseb, z namenom sprožitve alarma ali regulacije osvetljenosti prostorov,
- zasloni s podatki, razporejeni na raznih mestih, tudi v ogledalu kopalnice,
- posebni obeski, ki sistemu omogočajo sledenje oseb in prilagajanje naprav posebej zanje,
- možnost opravljanja dela od doma s pomočjo sistema videokonference, elektronske pošte ali drugih mehanizmov komunikacije,

- varčno upravljanje z energijo: toplotno in električno,
- ko nisi v stanovanjski hiši lahko sprejmeš sporočilo, da je nekdo pozvonil na hišni zvonec. Preko vgrajene kamere je mogoče identificiranje osebe in jim odpreti vrata stanovanjske hiše.

Projekt avtomatizacije stanovanjske hiše smo delno postavili na že obstoječe zasnove, ki smo jih primerno predelali in dopolnili tako, da so primerni za realizacijo s PLK-jem.
Naš projekt smo razdelil v štiri funkcijske bloke. V prvem bloku je upravljanje hiše za čim boljše počutje stanovalcev, kamor spada uravnavanje temperature po sobah in večjih sklopih ogrevanja, kot so talno gretje in radiatorji; v tem sklopu upravljamo tudi z roletami. V drugem sklopu skrbimo za varnost pred vlomi. Tukaj imamo alarm in upravljanje stavbe ob primeru vloma. V tretjem sklopu skrbimo za varčevanje z energijo in posredno z denarjem lastnikov. Varčujemo s toploto, električno energijo in vodo. V četrtem sklopu varujemo stanovanjsko hišo pred naravnimi nesrečami, kot so neurja (dež, veter) in poplave.
Predstavili bomo tudi, kako bi se dalo sistem nadgraditi z nadzornim SCADA sistemom in krmiljenjem na daljavo s pomočjo GSM SMS sporočil.

2 Predstavitev možnosti avtomatizacije

Pametno hišo bi lahko opredelili kot hišo, ki poseduje znanje o svojih parametrih, se zna nanje odzivati in jih zna posredovati v zunanji svet. Za njeno uresničitev rabimo inteligentni računalniški sistem. Na svetu že obstajajo različni sistemi za realizacijo inteligentnih hiš, neprestano pa se pojavljajo tudi novi sistemi, ki ponujajo nove možnosti in s tem dopolnjujejo že obstoječe sisteme. Glavne razlike izvirajo iz tehnologije, na osnovi katere je sistem zgrajen, kar tudi določa nekatere njegove funkcionalnosti, ki jih je v določenih sistemih lažje implementirati. Tako so enostavnejši sistemi osnovani na 8 bitni mikrokrmilniški osnovi, zapletenejši pa na osnovi zmogljivejših mikrokrmilnikov, ki sestavljajo osebni računalnik.

2.1 Delitev glede na različne načine komunikacije

Razlike obstajajo tudi glede na realizacijo fizičnega nivoja prenosa podatkov (komunikacijski kanal) med posameznimi elementi. Tako lahko podatke prenašamo v grobem na tri različne načine: po električnem omrežju, po posebnih žicah in brezžično. Od izbire prenosnega medija so zelo odvisni način implementacije, varnost, maksimalen pretok podatkov, največja možna kompleksnost in cena, ki je poleg tehničnih značilnosti pomemben dejavnik pri odločitvi za vgradnjo takega sistema.

2.1.1 Prenos po posebnih žicah

Vsaka naprava, ki je avtomatizirana ali služi za nadzor, je priklopljena na poseben kabel (parica, koaksialni kabel ali optični kabel), ki ji omogoča komuniciranje z ostalimi napravami v sistemu. Tak način se uporablja v računalniških mrežah, tako lokalnih kot na internetu. Način komunikacije (nižji in višji protokoli) je specifičen za vsak sistem, pogojen pa je tudi z izbiro fizičnega nivoja (kabla).
Pri vgradnji takega sistema je potrebno v hišo inštalirati dodatno omrežje (ožičenje), na katerega se bodo priklapljale naprave, zaradi česar je tak sistem primeren predvsem za novogradnje in večje temeljitejše obnove. Prednosti tega sistema so popolnoma ločena komunikacija od zunanjega sveta, kar otežuje vdore v sistem, in neodvisnost sistema od stanja električnega omrežja. Prednost je tudi v veliko večji zanesljivost komuniciranja. Vdor v sistem je mogoč samo preko računalnika, ki je priklopljen na internet in na naše omrežje. V okviru te fizične prenosne poti se uporablja predvsem parica s protokoli kot so CAN bus, Lonworks, Fieldbus, CEBus [X] in EIB [XI] .

2.1.2 Prenos informacij po električnem omrežju

Vse naprave, ki so del inteligentnega sistema, se priključujejo neposredno na električno omrežje, ki je že razpeljano po hiši, ter ga uporabljajo tako za medsebojno komunikacijo kot tudi za napajanje. Pri komunikaciji med posameznimi moduli se na oddajni strani, na nosilno frekvenco 50 Hz, superponira komunikacijski signal, moduliran z višjimi frekvencami, na sprejemni strani pa se te višje frekvence izloči ter signal demodulira nazaj v prvotno obliko.
Za vgradnjo takega sistema ni potrebna inštalacija novih kablov, kar ga naredi primernega tudi za že zgrajene objekte. Slabost teh sistemov je širjenje komunikacijskih signalov izven hiše, kar pomeni veliko nevarnost za vdor v sam sistem. V električnem omrežju se velikokrat pojavljajo razne motnje (nihanje in izpad napetosti), ki lahko vplivajo na delovanje samega sistema. V svetu je uveljavljenih nekaj sistemov, ki podpirajo omenjeni način prenosa podatkov. To so X-10 [IX], EIB in CEBus. Tak prenos se lahko uporablja tudi za lokalne računalniške mreže. Prenos podatkov je počasnejši kot pri normalnih računalniških mrežah, ima pa seveda to prednost, da nam ni treba vleči novih kablov po hiši.

2.1.3 Brezžični prenos

Tudi ta pristop je pogosto uporabljen pri inteligentnih sistemih. Predvsem se pojavlja odkar je prišlo do razcveta brezžičnih komunikacij v računalništvu. Tak način prenosa informacij je primeren predvsem v starejših hišah, kjer je inštalacija novih kablov zapletena in draga. Za prenos podatkov se uporablja radijski medij, protokoli in frekvence pa se razlikujejo od sistema do sistema. Najpogosteje uporabljene frekvence so: 433 MHz, 1 GHz in 2 GHz (Bluetooth).
Prednosti takega sistema so predvsem enostavna vgradnja in možna uporaba v že zgrajenih hišah, pomanjkljivost pa je ponovno širjenje radijskih signalov izven hiše, kar povečuje nevarnost vdora v sistem. Seveda je tu zelo pomembno kako daleč se te informacije prenašajo. Tako lahko recimo že pri Bluetooth tehnologiji uporabimo dva različna modula, pri katerih en omogoča komunikacijo do 10 m daleč drug pa do 100 m. Da se temu izognemo, je tako kot pri prenosu po električnem omrežju, potrebno poskrbeti za zanesljivo šifriranje. Seveda pa tudi z najboljšim šifriranjem ne moremo zagotoviti imunosti na razne vdore v sistem.
Problem naprav, ki spadajo v to skupino, je predvsem napajanje, še posebej, če uporabimo brezžično komunikacijo zaradi nedostopnosti dodatnih ali omrežnih kablov. V primeru, da imamo na voljo tudi električno energijo, lahko preko teh kablov tudi komuniciramo. V primeru, da do naprav potegnemo dodatni napajalni kabel, pa lahko le tega uporabimo za prenos inforamcij. Uporaba brezžičnih modulov je tako le malokrat upravičena, je pa res, da se bodo ti moduli z razvojem tehnologije Bluetooth vse bolj uveljavljali. Take komunikacije se v glavnem uporablja samo za prenos manjše količine informacij iz oddaljenih delov sistema avtomatizirane hiše.

2.2 Standard X-10

X-10 [IX] je razmeroma star standard, ki za prenos podatkov med posameznimi napravami uporablja električno omrežje. Prvotno je bil razvit za ameriški trg (110V), kasneje pa predelan tudi za evropskega (230V).
Dobra stran tega sistema je razmeroma nizka cena posameznih modulov in naprav, ima pa veliko pomanjkljivost, da je v skupen inteligentni sistem možno priključiti samo 256 naprav, kar onemogoča njegovo implementacijo v profesionalna okolja in je s tem primeren samo za manjše domače inteligentne sisteme. Sistem se nam ni zdel primeren zaradi modulov, ki se jih ne da programirati, tako da se jih lahko samo poveže med seboj in je s tem vsa funkcionalnost že določena. Pri diplomskem delu je bil cilj v tem, da se hišo krmili z algoritmom, ki ni vnaprej določen.

2.3 Standard CEBus

CEBus [X] je podoben EIB standardu in prav tako vsebuje več fizičnih prenosnih poti: parica, koaksialen kabel, električno omrežje in RF brezžična komunikacija. Podpira ga kar nekaj proizvajalcev strojne opreme.

2.4 Standard EIB - European Instalation Bus

EIB [XI] je odprt standard, ki ga je pod okriljem telesa EIB Association sprejelo več različnih podjetij in posameznikov. Standard ustreza OSI razporeditvi nivojev in vključuje vse nivoje, od fizičnega do aplikacijskega. Za fizično prenosno pot podatkov se lahko uporablja parico, električno omrežje, brezžično (RF) ali Ethernetu podobno komunikacijo; odvisno od potreb.
Celoten sistem je zgrajen modularno in podpira več kot 65.000 naprav, ki so logično razdeljene v manjše skupine, kar omogoča lažje upravljanje in nadzorovanje tako posamezne naprave, kot tudi celotnega sistema.
Prednost EIB sistema je definitivno veliko število proizvajalcev, ki stojijo za tem standardom in ponujajo različne produkte, primerne za implementacijo v inteligentne sisteme. Med slabosti bi lahko šteli razmeroma visoko ceno za produkte, ki podpirajo EIB standard. To je razlog, da se ti sistemi še niso tako uveljavili v hišnih sistemih, kot so se v večjih poslovnih zgradbah.

3 Predstavitev možnih sistemov avtomatizacije, ki se pojavljajo na slovenskem trgu

V Sloveniji je že nekaj ponudnikov inteligentnih sistemov, ki pa ponujajo predvsem prodajo in vgradnjo tujih sistemov v naše domove in druge objekte. Med najbolj zastopane sisteme sodita X-10 in EIB, med industrijskimi inteligentnimi sistemi pa predvsem Lonworks.
Nekaj podjetij se je odločilo tudi za lasten razvoj inteligentnih sistemov, kar je sigurno dobrodošla novost na tržišču, saj uporaba tujih cenovno dokaj neugodnih sistemov ni vedno upravičena.

3.1 Domača podjetja in sistemi

3.1.1 Robotina

Podjetje Robotina [XII] je razvilo lasten inteligentni sistem pod imenom Integra BM (Integra building management system) [3], ki ga sedaj uspešno vgrajuje predvsem v večje objekte. En takšnih objektov je Portoroški hotel Palace. Sistem je bil razvit na Hrvaškem, v Zagrebu. Razvila ga je tamkajšnja podružnica Robotine, ki je zadolžena za večino Robotininega razvoja. Zasnovan je na dveh komunikacijskih protokolih. Za nižji protokol so izbrali CAN bus, ki je namenjen komunikaciji med elementi v enem podsklopu (prostor ali nadstropje). Za komunikacijo med omenjenimi podsklopi je uporabljen protokol RS-232 ali Ethernet, ki omogoča tudi transparentno priključitev sistema na internet. Sami procesorski moduli nimajo integrirane Eternet komunikacije. Robotina kot dodatek ponujajo poseben modul, ki nam pretvori RS-232 v Eternet komunikacijo. S to funkcionalnostjo in z dodatnim GSM modulom so poskrbeli, da je mogoče njihov sistem upravljati praktično od kjerkoli. Tudi GSM modul ni implementiran v procesorski modul, ampak se ga dobi kot dodatek, ki se preko RS-232 poveže s procesorskim modulom.

3.1.2 Indata

Tudi podjetje Indata razvija svoj lasten inteligentni sistem, ki naj bi predvsem temeljil na modularnosti. Podrobnejših informacij o njihovem sistemu trenutno ni v obtoku, prav tako pa ni razvidno ali imajo sistem že dokončan ali ne. Zaradi zaupnosti poslovnih informacij do več podatkov nismo mogli priti.

3.1.3 GOAP sistemi

To je slovensko podjetje, ki se je usmerilo predvsem na trg luksuznih križark. Razvit ima že celovit sistem za upravljanje hotelskih sob. Poleg tega že razvitega sistema nadaljuje z razvojem novih naprednejših rešitev. GOAP razvija module v Sloveniji, kjer jih tudi proizvaja. Čeprav se vse razvije in naredi v Sloveniji, imajo težavo v tem, da nimajo svojega lastnega močnega razvoja in proizvodnje. Razvoj in proizvodnja njihove opreme je pri drugih podjetjih v Sloveniji, pri katerih GOAP naroči te storitve. Na trgu luksuznih križark so si uspeli v kratkem času pridobiti dobro ime in velik del svetovnega posla.

3.2 Uporaba tujih sistemov

Uporaba tujih sistemov na žalost ne vključuje lastnega razvoja, tako da so ponudniki tovrstnih sistemov v večini odvisni od ponudbe tujih podjetij in njihovih cen. Ta skupna kombinacija zna biti v določenih okoliščinah zelo neugodna, kar zavira hitrejšo implementacijo tovrstnih sistemov v slovenske zgradbe.

3.2.1 Dexus

To je podjetje, ki na slovenskem trgu ponuja produkte sistema X.10, ki so prilagojeni za evropsko električno omrežje 230 V.

3.2.2 Prelog

Podjetje ponuja produkte sistema EIB in jih vgrajuje v objekte na področju Slovenije. Sama zasnova sistema je razvidna pri opisu tujih sistemov, kako uspešni so pri sami implementaciji pa ne navajajo, saj nimajo nobenih referenc.

3.2.3 Liko Pris

Podjetje se je v zadnjem času usmerilo tudi na področje implementacije inteligentnih sistemov v večje poslovne stavbe. Za inteligentni sistem uporabljajo elemente različnih tujih proizvajalcev, ki jih povežejo v skupni sistem. Uporabljajo tudi sisteme slovenskega podjetja GOAP sistemi.

3.2.4 Silon

Podjetje se v veliki meri ukvarja z vgradnjo inteligentnih sistemov v večje objekte in pri tem uporablja tuji sistem Lonworks.

4 Izbira PLK-ja

PLK smo izbrali zaradi dobrih možnosti za razširitev in zelo dobre prilagodljivost, ki se jo da izkoristiti za prilagoditev v vsako stanovanjsko hišo. Tudi če bi potrebovali zelo velike potrebe po vhodih in izhodih, ki bi presegle zmogljivost enega PLK-ja, bi lahko problem rešili s povezovanjem večih PLK-jev.

PLK so razvili v 60-ih in 70-ih letih za avtomobilsko industrijo.
Prve specifikacije zahtev so bile:

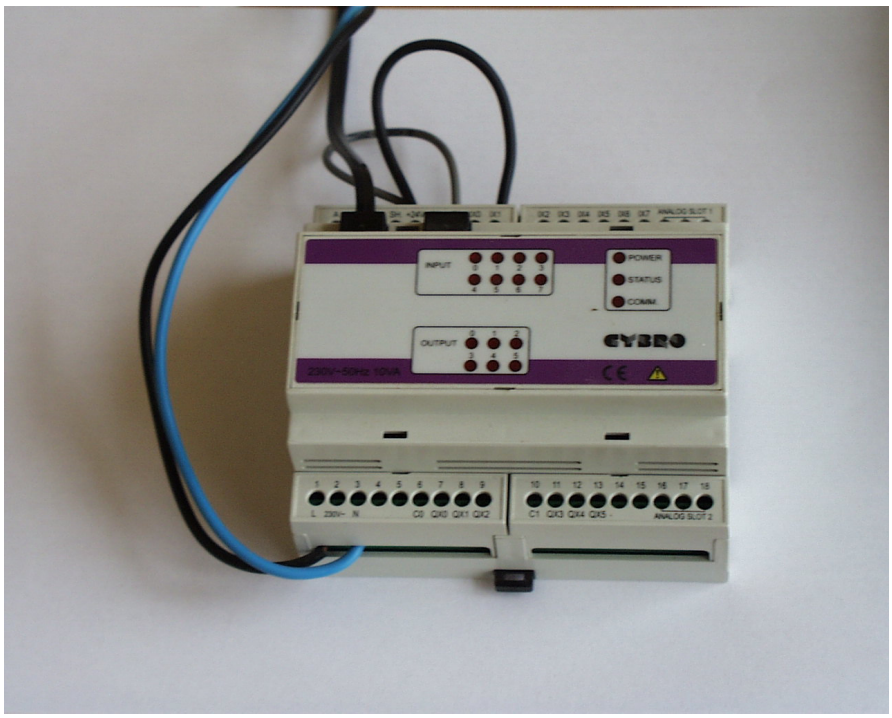
- naprava brez gibljivih delov (elektronska izvedba),
- računalniška fleksibilnost (možnost reprogramiranja),
- zanesljivo delovanje v industrijskem okolju (vibracije, temperatura, prah),
- enostavno programiranje in vzdrževanje (uporabljajo elektroinženirji in tehniki).

Prva naprava, ki je ustrezala zahtevam, je bila Gould Modicon. Modicon je sedaj pod okriljem francoske multinacionalke Schneider. Prvi pravi uspehi PLK-ja so se pojavili po razvoju lestvičnega diagrama (Ladder Logic) za programiranje, kot so jih bili že prej vajeni elektroprojektanti in uporabniki.
Od teh časov so se razvili v eno najbolj univerzalnih orodij za industrijsko avtomatizacijo. So osnova skoraj vseh avtomatiziranih proizvodnih procesov. Uporabljajo se v raznih aplikacijah, od avtomatiziranega zvonjenja v cerkvenih zvonovih do krmiljenja električnega omrežja in elektrarn.
Mikrokrmilniški ali procesni modul je najvažnejši del sistema, saj deluje kot njegovi možgani. Sestavljajo ga mikrokrmilnik, pomnilnik, integrirana vezja in vezja, potrebna za shranjevanje in uporabljanje informacij in pomnilnika. Sestavljajo ga tudi komunikacijska vrata in druga periferija. PLK-ji lahko kontrolirajo od samo 6 vhodno izhodnih (I/O) sponk, pa do 40 000 in več. En procesor lahko kontrolira več kot en proces ali proizvodno linijo. Ponavadi so procesni moduli povezani med seboj tako, da skrbijo za kontinuiran nadzor nad procesom. Število vhodov in izhodov je omejeno s skupno kapaciteto PLK-sistema, predvsem strojnih in pomnilniških kapacitet. Procesor skrbi za stanja vhodov, ki jih nato logično obdela in posreduje vrednosti na izhode.
Imamo več tipov vhodnih modulov med katerimi lahko izbiramo. V glavnem se razdelijo na digitalne in analogne. Naprej se delijo še glede na različne električne nivoje, na katerih delujejo. Pri digitalnih imamo tako lahko različne nivoje napetosti, ki jih zaznavajo. Pri analognih pa je lahko razlika tudi v tem, ali merijo napetost ali tok. Samo električno vezje vhodnega vezja je precej enostavno, saj prenese zgolj vrednost iz enega konca vezja na drug konec vezja. Moduli imajo ponavadi do 32 vhodov. Vsak vhod ima svoj naslov, ki ga uporablja procesorski modul. Analogni moduli so manjši, saj imajo do 6 vhodov v enem modulu. V njih je analogno digitalni pretvornik (A/D), ki nam lahko spremeni signal, ki nosi informacijo z zaznano temperaturo, vlago, hitrost, tlak ali pozicijo, v digitalno obliko. Signal je ponavadi 4 mA - 20 mA ali 0 V – 10 V.
Tudi izhodni bloki imajo podobne lastnosti. Za nadzor nad izhodi se uporabljajo triaki ali releji. Triaki ponavadi preklapljajo napetosti do 24 V in tokove do 500 mA. Releji pa lahko preklapljajo tudi 220 V in tokove do 3 A. Seveda obstajajo tudi posebni moduli, ki lahko zelo odstopajo od teh vrednosti. Analogni izhodni moduli so posebna vrsta izhodnih modulov, ki uporabljajo digitalno analogno pretvorbo (D/A) informacije. Analogni izhodni modul nam pretvori do 12 bitno vrednost v analogni signal. Ponavadi je ta signal v območju 0 V – 10 V

ali 4 mA – 20mA. Tak analogni signal je ponavadi uporabljen pri motorskih ventilih ali pnevmatskih pozicijskih napravah.
Pri izbiri smo na začetku pregledali ponudbo in različne možnosti, ki so trenutno na razpolago. Tako je bila ena opcija tudi izbira namenskih modulov za avtomatizacijo hotelskih sob, kot so npr. so Integra BM. Ta sistem je že izpopolnjen, saj je zanj obstaja vse od SCADA programa do multi senzorja za požar in prisotnost osebe.
Sistem Integra BM je zasnovan tako, da imamo v nadzornem nivoju postavljeno SCADA-o, njej pa je podrejen sobni krmilnik (Room controller). To je nivo našega PLK-ja. Na sobni krmilnik je predvidena priključitev prikazovalnika z IR sprejemnikom (HMI with IR receiver), modul za hotelske sobe (Hotel room module), standardni vhodno izhodni modul (Standard I/O module), krmilnik toplotnega pretvornika (FanCoil controller), krmilnik luči (Light controller) in multi senzor (Multisensor).
Po našem mnenju bi se pri teh modulih pojavil problem pri redundanci implementiranih funkcij, ki so v hotelu potrebne, v stanovanjski hiši pa ne, tako da bi ti moduli precej podražili sam sistem, ki ne bi bil 100 % izkoriščen. Hotelski moduli so namreč predprogramirani. Da se jih tudi programirati. Problem je v tem, da če se lotiš programiranja, izgubiš vse prednosti hotelskih modulov pred PLK-ji. Hotelski moduli so izpeljanke PLK-jev. Hotelski moduli so po naših izkušnjah strogo namenski, saj je programiranje modula za eno hotelsko sobo približno enako zahtevno, kot za stanovanjsko hišo pri podobnih funkcijah.
Pri hotelskih sistemih so prednosti bolj očitne, saj ima hotel po več sto enakih sob, v katerih lahko uporabimo isti program. Za razliko od tega ima vsaka stanovanjska hiša svojo specifiko, kar pomeni veliko programerskega truda za vsak posamezni primer izvedbe. Zelo težko bi namreč našli več deset enakih hiš, pa tudi v tem primeru je malo verjetno, da bi se več lastnikov odločilo za enake funkcije avtomatizirane hiše. Poleg hotelskih modulov obstajajo tudi moduli za stanovanjske hiše, ki pa imajo vse funkcije že predprogramirane, tako da se jih ne da razširiti. Poleg tega bi bil problem tudi pri nadgradnji z nadzornim sistemom.
Zaradi boljše izrabe vseh priključnih sponk v modulih je bila pomembna tudi modularnost, ki nam prek izkoriščenosti modulov omogoča optimalni izkoristek.

4.1 SyBro PLK

Odločili smo se za SyBro PLK, ki je plod domačega znanja, saj ga delajo v Robotini v Kopru. Cena osnovnega modula je zelo ugodna, medtem ko je problem pri razširitvi, saj je cena vsakega vhoda in izhoda za razširitev zelo visoka. Za avtomatizacijo celotne hiše bi se odločili za Schneider-jev Twido PLK, ki je precej zmogljivejši (že v osnovi je možna LAN povezava) in cenejši v večjih konfiguiracijah. PLK, ki smo ga izbrali, je dovolj za avtomatizacijo ene družinske hiše. Kljub temu, da je manjši in cenejši, to ne pomeni, da je zaradi tega manj zanesljiv, saj mora še vedno zadovoljevati vse industrijske standarde.



Slika 3: SyBro PLK.

4.1.1 Tehnični podatki SyBro PLK

Programski pomnilnik: 128 K flash EEPROM
Podatkovni pomnilnik: 128 K static RAM
Sistemska ura: 24 MHz
Čas procesiranja: 250 ns osnovni ukazi
Dosegljive časovne baze: 10 ms, 100 ms in 1 s

Tipi podatkov: bit, word, int, long, real
Čas po katerem se podatki izgubijo: 15 dni
Točnost ure (RTC): 30 s/mesec

Digitalni vhodi: 24 V, 7 mA ponor/izvor
Zakasnite digitalnih vhodov: 5 ms
Digitalni izhodi: 1 A, 250 V relejski kontakti in 0,5 A, 30 V tranzistorski izhodi

Analogni vhodi: Pt100/Pt1000, Ni100/Ni1000, 0(4) mA .. 20 mA in 0 V..10 V

Analogni izhodi: 0(4) mA .. 20 mA in 0 V..10 V

Natančnost: 0.1 %

IEX kapaciteta: 16 enot (512 I/O točk)
IEX bus razdalja: 5 m max.
IEX hitrost prenosa (baud rate): 100 kbps

A-bus kapaciteta: 32 naprav
A-bus razdalja: 1 000 m max.
A-bus hitrost prenosa (baud rate): 19 200 bps
A-bus odzivni čas: 25 ms x število naprav

High Speed Counter: 10 kHz max.
HSC nivoji: 24 VDC

Napajalna napetost:
85 V – 264 V, 50/60 Hz (PLK)
24 VDC (PLK, Bio-16, PS)

5V (IEX) izhodni tok:
600 mA max. (PLK)
1 000 mA max. (Bio-16)
1 200 mA max. (PS)

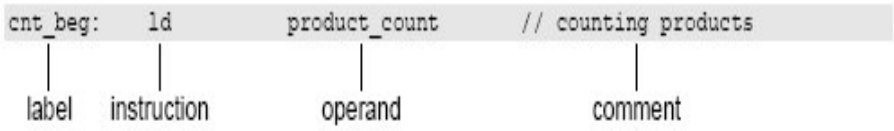
24 V izhodna moč: 120 mA max. (samo PLK)

Delavna temperatura: 0 °C..50 °C, 85 % RV (relativna vlažnost)
Temperatura shranjevanja: -10 °C..75 °C, 85 % RV
Ohišje: IP20, nastavki za DIN letvico

4.2 Predstavitev programskega paketa CyPro

4.2.1 Načini programiranja

S tem programskim paketom se lahko programira v dveh načinih. Prvi je instruction list, drugi pa strukturni tekst (structured text) V načinu instruction list je programiranje podobno programiranju mikrokrmilnikov, saj izgleda struktura ukazne vrstice tako, kot je predstavljeno na sliki 4. S pomočjo programiranja v načinu instruction list se lahko program zelo dobro optimizira. To pomeni, da je program zelo hiter in ima zelo kratko kodo. Da pridemo do tega optimalnega izkoristka moramo znati zelo dobro programirati. Če nam programiranje ne gre najbolje od rok, se nam zelo hitro zgodi, da postane program dolg, počasen in zelo nepregleden.



Slika 4: Način programiranja instruction list.

Zaradi vsega naštetega smo instruction list način programiranja prepustil drugim, mi pa smo izbrali način programiranja v strukturiranem tekstu, ki je predstavljen na sliki slika 6. Ta način programiranja je precej lažji in veliko bolj pregleden kot pri instruction list načinu programiranja, medtem ko je koda počasnejša in daljša od zelo dobre kode instruction list. Hitrost kode pri avtomatizaciji hiše ni ključna, saj so vsi procesi v hiši zelo počasni in z velikimi histerezami. Možni ukazi za programiranje v struktuiranem tekstu so predstavljeni v tabeli tabela 1 in v nadaljevanju besedila [I in 1, str. 36 - 40]].

operator	alias	unary	binary	bit	word	int	long	real	result
+			•			•	•	•	same
-		•	•			•	•	•	same
*			•			•	•	•	same
/			•			•	•	•	same
mod	%		•			•	•		same
not	!	•		•	•				same
and	&		•	•	•				same
or	|		•	•	•				same
xor	^		•	•	•				same
=	==		•	•	•	•	•	•	bit
<>	!=		•	•	•	•	•	•	bit
<			•			•	•	•	bit
<=			•			•	•	•	bit
>			•			•	•	•	bit
>=			•			•	•	•	bit
:=			•	•	•	•	•	•	same

Tabela 1: Tabela numeričnih ukazov.

if...then...else Pogojni stavek če(if). Če je pogoj izpolnjen se izvrši prvi programski stavek. Če ni se preveri drug pogoj. Če je drugi pogoj izvršen se izvrši drugi programski stavek.

```
if <expression> then
<statements>;
elsif <expression> then
<statements>;
else
<statements>;
end_if;
```

case...of Pogojni stavek »v primeru«. Tu se nam izvrši programski stavek glede na vrednost spremenljivke.

```
case <expression> of
<value1>: <statements>;
<value2>: <statements>;
...
<valuen>: <statements>;
else
<statements>;
end_case;
```

for...do To je programska zanka, ki se izvršuje pri določenih vrednosti spremenljivke.

```
for <var>:=<expression> to <expression> do
<statements>;
end_for;
```

while...do To je programska zanka, ki se izvršuje dokler je pogoj v njej izpolnjen.

```
while <expression> do
<statements>;
end_while;
```

Zaznavanje pozitivnega (prednjega) roba
fp(b:bit):bit;

Zaznavanje negativnega (zadnjega) roba
fn(b:bit):bit;

Tipi spremenljivk:

High speed counters;	Števci, ki lahko štejejo visoke frekvence vhodnih signalov
int(expression):int;	Celoštevilske vrednosti
word(expression):word;	Spremenljivke sestavljene iz znakov
long(expression):long;	Celoštevilske vrednosti, ki grejo do višjih števil
real(expression):real;	Realne vrednosti

Funkcije za prikaz

clear display
dclr(slot:int); Brisanje prikazovalnika

print ASCII character
dprnc(slot:int, x:int, y:int, c:char); Prikaz ASCII znaka na prikazovalniku

print string
dprns(slot:int, x:int, y:int, str:string); Prikaz besedila

print binary value
dprnb(slot:int, x:int, y:int, c0:char, c1:char, value:bit); Prikaz binarne vrednosti

print integer value

dprni(slot:int, x:int, y:int, width:int, zeroblank:bit, value:int);	Prikaz celoštevilske vrednosti

print long value

dprnl(slot:int, x:int, y:int, width:int, zeroblank:bit, value:long);	Prikaz visoke celoštevilske vrednosti

print real value

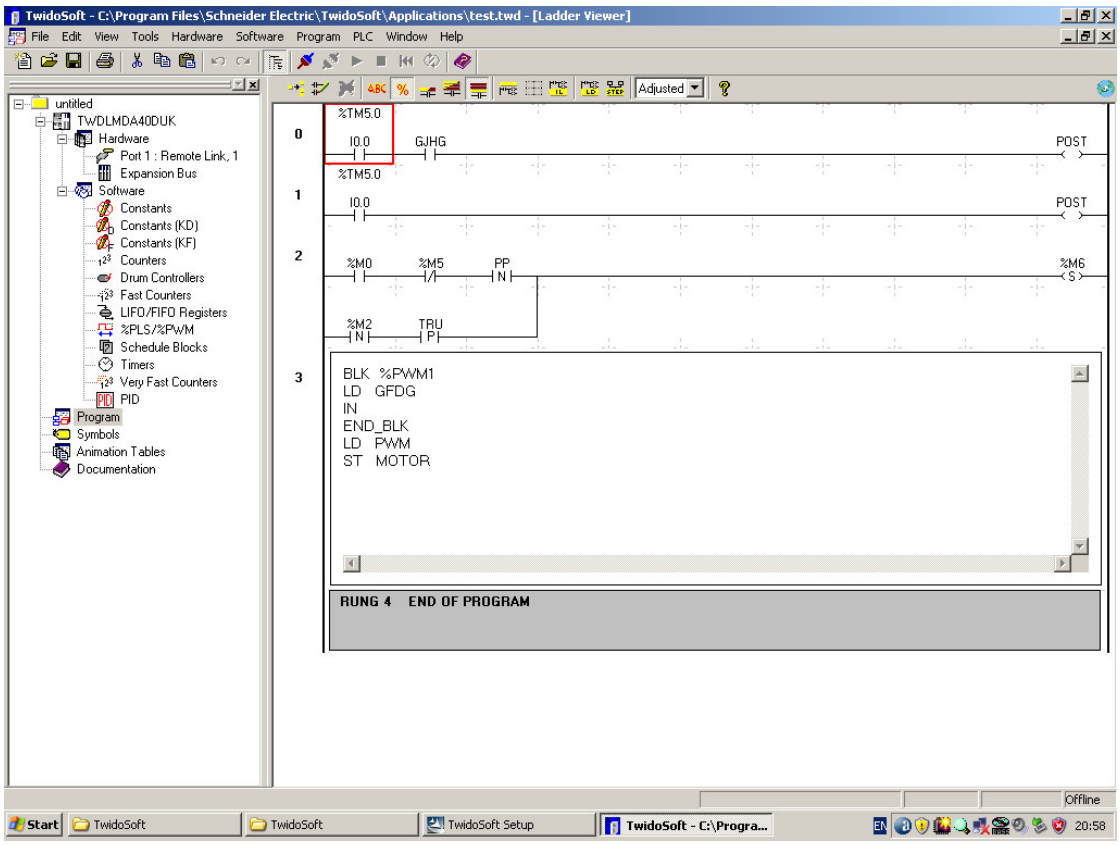
dprnr(slot:int, x:int, y:int, width:int, dec:int, value:real);	Prikaz realne vrednosti

Manjša legenda za lažje razumevanje ukazov:
slot .....slot number
x .....x position (0-left)
y .....y position (0-top)
width .....print width
zeroblank ....zero blanking (0-no, 1-yes)
dec.....decimal places
c .....single character
str.....array of characters
value .....value to print

Vidimo, da programski jezik spominja na Pascal in je posebej prirejen za aplikacije v industriji. Struktura jezika je enaka kot pri Pascalu. Dodane so samo posebne funkcije, ki so potrebne za programiranje industrijskih PLK-jev. Način, na katerem je ponavadi napisan program, je zelo podoben lestvičnemu diagramu (ladder), saj je velika večina programerjev PLK-jev pred tem delala z lestvičnim diagramom.

```
y position:=5;
down_timer.pt:=15000;
circle area:=r*r*3.14;
case_counter:=case_counter+1;
volts:=amps*ohms;
start:=(oil press and steam and pump) and not emergency stop;
valid_value:=(value = 0) or ((value > 10) and (value <= 60));
```

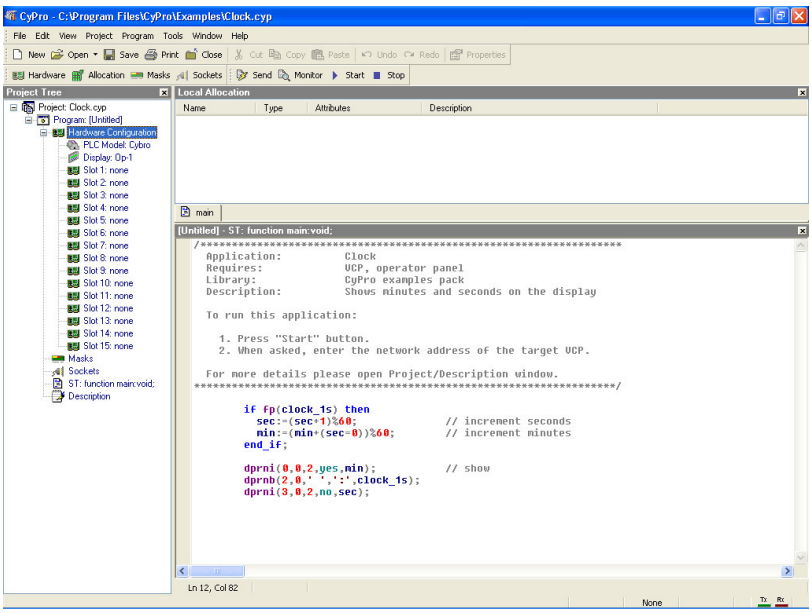
Slika 5: Način programiranja v struktuiranem tekstu.



Slika 6: Program z lestvičnim diagramom.

Programiranje v lestvičnem diagramu je lažje in preglednejše predvsem pri analiziranju delovanja in iskanju napak v programski kodi (slika 6). Pri strukturnem tekstu ki smo ga uporabili mi imamo za analizo na voljo samo okno s spremenljivkami, pri lestvičnem diagramu pa je analiza olajšana saj se nam spremenljivke različno obarvajo v lestvičnem diagramu glede na vrednost spremenljivke. Tudi pri programih z lestvičnim diagramom imamo okno s spremenljivkami. Ker se nam spremenljivke v lestvičnem diagramu obarvajo, hitro vidimo, kje se nam ob napaki program ustavi, tako da moramo samo poiskati razlog napake.

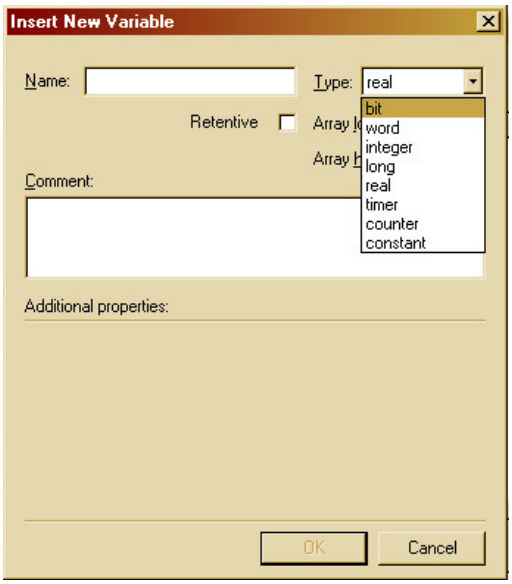
Programsko okno pri programu je standardne oblike, kot pri vseh programskih jezikih, in je predstavljeno na sliki 6.



Slika 7: Program za programiranje PLK-ja.

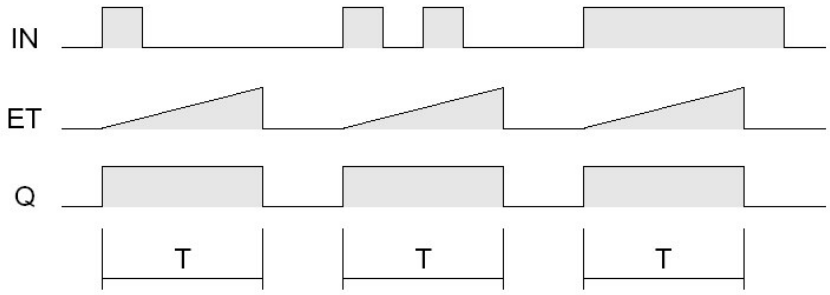
4.2.2 Izbira tipa spremenljivk

Pri programiranju PLK-jev je izbira pravilnih spremenljivk zelo pomembna, zaradi česar bomo v tem poglavju predstavili tipe spremenljivk in njihove funkcije. Spremenljivke se določa v oknu, ki je prikazano na sliki 8. Najprej določimo ali je spremenljivka notranja oziroma vhodno/izhodna. Vhodno/izhodne spremenljivke so lahko tipa bit in integer. Pri bitnih spremenljivkah se nastavi ali spremeni posamezen bit. Pri vhodno/izhodnih spremenljivkah to pomeni, da imamo informacijo o tem, ali je stikalo sklenjeno ali razklenjeno, oziroma ali je na vhodu napetost ali je ni. Pri vhodno/izhodnih spremenljivkah tipa integer se v spremenljivko PLK-ja shrani vrednost iz vhoda. Na vhod je lahko vezan Pt100, Pt1000, Ni100/Ni1000, 0 – 10V ali 4 - 20mA. Spremeni se nam v vrednost, ki je odvisna od bitov A/D pretvornika.



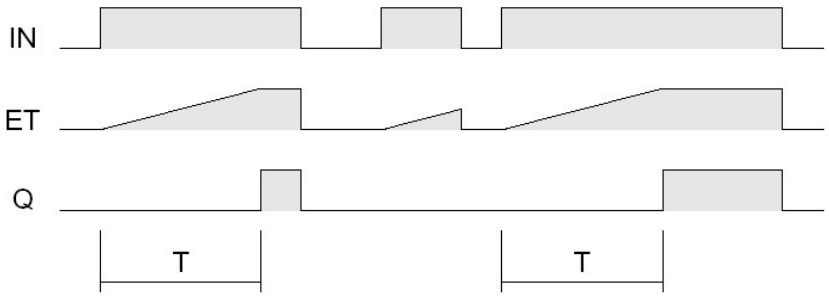
Slika 8: Okno za vstavljanje nove spremenljivke.

Kot notranja spremenljivka ima lahko word katerokoli skupino vrednosti 16-ih bitov. Word nam ne predstavlja vrednosti. Uporablja se samo kot priročna metoda za logične operacije na 16 bitih istočasno. Iz spremenljivke word se da s pravilnim naslavljanjem dobiti bitno vrednost. Poleg tega lahko tudi več spremenljivk tipa bit sestavimo v word. Na spremenljivki word se ne da izvajati logičnih operacij. Imamo še notranje spremenljivke integer, long in real. Te se uporabljajo pri računanju stanj v programu PLK-ja. Ti tipi spremenljivk so enaki kot pri vseh programskih jezikih in se tudi uporabljajo na enak način in za enak namen. En važnejših tipov spremenljivk pri PLK-jih je časovnik (timer). V SYBRO PLK-ju ima lahko dve obliki. Lahko je pulzni časovnik ali časovnik z zakasnitvijo. Kot pulzni časovnik je v stanju ON nek v naprej določen časovni interval. Delovanje pulznega časovnika je predstavljeno na sliki 9.



Slika 9: Pulzni časovnik.

Drug tip, časovnik z zakasnitvijo, je predstavljen na sliki 10, kjer se vidi, da se nam vrednost spremeni v ON šele po nekem časovnem intervalu. Te časovnike se uporablja npr. pri preverjanju napak na napravah. Vrednost na izhodu spremenimo trenutno in če bi istočasno tudi preverili, ali se nam je naprava vklopila, bi vedno dobili napako, saj se nam naprava ne more priključiti tako hitro. To opazimo predvsem pri mehanskih napravah, ki imajo večje časovno konstanto kot računalnik.



Slika 10: Časovnik z zakasnitvijo.

5 Upravljanje stanovanjske hiše s pomočjo PLK-ja

5.1 Upravljanje s hišo

5.1.1 Prižiganje in ugašanje luči

Prižiganje in ugašanje luči je možno na dva načina. Pri vseh predstavljenih tipih stikal za luči uporabljamo pulzna stikala, kar pomeni, da logično stanje ena nastopi samo, ko je stikalo pritisnjeno.
Prvi način uporabe je podoben običajnemu delovanju luči: ko stikalo pritisnemo, se luč prižge, in ko pritisnemo še enkrat, se luč ugasne. Pri tem načinu lahko preko večih stikal prižigamo in ugašamo eno luč. Za to ne potrebujemo menjalnih in križnih stikal, saj lahko več stikal vežemo na isti vhod PLK-ja. PLK ne ve, katero stikalo smo pritisnili, in za PLK tudi ni pomembno. PLK nam ob prvem pozitivnem robu na vhodu, kamor je priklopljeno stikalo, luč prižge, ob drugem pozitivnem robu pa jo ugasne, nakar se to ponavlja. Pri prižiganju in ugašanju luči je potrebno paziti, da ponavadi pri preklopu stikala dobimo oscilirajoči signal, ki ga moramo izničiti, zato vedno vzamemo prvi rob in potem držimo signal na logični 1 vsaj pol sekunde. Na ta način se nam ne zgodi, da z enim pritiskom na stikalu, luč prižgemo in ugasnemo.
Pri drugem tipu uporabe nam luč gori nek poljubno dolg časovni interval. To je enako kot stikalo v blokih za prižiganje luči na hodnikih. Tudi tukaj moramo paziti na oscilacije ob pritisku, kar zopet naredimo s prvim robom pulza in časovnikom. Ob prvem robu aktiviramo pulzni časovnik, ki nam drži izhod na 1, kar nam prižge luč za nek v naprej določeno dolg časovni interval. Luč se nam ugasne, ko ta časovni interval preteče. Poleg tega bi bilo možno ugasniti luč s ponovnim pritiskom na stikalo, če bi imeli drugačni PLK, ki bi imel ponastavitev (reset) časovnika. Tako bi lahko ugasnili luč, če je ne potrebujemo. S tem ko luč ugasnemo predčasno, naredimo nekaj za naš žep in za ekologijo.

5.1.1.1 Razlaga programa za luč

`Tipka2` nam predstavlja vhod v PLK. S priredilnim ukazom := se nam vrednost prenese na notranjo časovno spremenljivko `Luc2`. Končnica `.IN` nam predstavlja vhod v časovnik. Notranja časovna spremenljivka je tukaj uporabljena zaradi elektronskih oscilacij, ki se pojavijo ob preklopu stikalnega peresa. Ob preklopu se nam pojavi več prehodov skozi nič, kar bi nam lahko luč v zelo kratkem času prižgalo in ugasnilo; v tem primeru bi luč ostala prižgana. Časovnik `Luc2` ima konstanto 0,5 s. Trajanje 0,5 s smo ocenili za dovolj majhno konstanto v kateri se uporabnik še ne odločiti, da bi luč ponovno ugasnil. Poleg tega je ta konstanta dovolj dolga, da se oscilacije pri preklopu izničijo. Če bi konstanto časovnika povečali na 5 s, bi to pomenilo, da ko prižgemo ali ugasnemo luč, naslednjih pet sekund ne moremo spremeniti stanja.
V vrstici `If (Luc22==1 and fp(Luc2.Q)) then` preverjamo pogoj za ugašanje luči. Tu nam spremenljivka `Luc22` predstavlja luč, ki jo prižigamo in ugašamo. Spremenljivka `Luc22` je lahko notranja spremenljiva ali direktni naslov izhoda iz PLK-ja;. v tem primeru je zunanja spremenljivka. Navadno se nam v takih situacijak pojavlja notranja spremenljivka, ki se na koncu programa vsakič prepiše v izhodno spremenljivko. Tak način se uporablja zaradi tega, da se lahko poljubne skupine izhodov blokira v enem stanju s pomočjo ene spremenljivke. Z logičnim ukazom == preverjamo, če je vred `Luc22` enaka 1, kar pomeni, da

luč gori. Logični ukaz in (and) nam da izhod ena, ko sta izpolnjena oba pogoja. Ukaz `fp(Luc2.Q)` je sestavljen iz dveh komponent. `Fp(xxxx)` del ukaza nam pove, da je pogoj izpolnjen samo enkrat ob prehodu pozitivne fronte. To je samo takrat, ko se vrednost spremeni iz 0 v 1. Če tega ne bi preverjali samo ob prehodu iz 0 v 1, bi to pomenilo, da bi program ob vsakem prehodu skozi kodo prižigal in ugašal luč. V našem primeru ko je prižiganje luči v kodi za ugašanjem, to pomeni da luči ne bi mogli ugasniti. Končnica `.Q` pri spremenljivki `Luc2` pomeni, da preverjamo izhod iz časovnika. Če so pogoji v oklepajih izpolnjeni, se nam izvrši stavek `Luc21:=0;`, ki je v pogojnem stavku. Pri izvrševanju tega stavka se nam samo prepiše vrednost 0 v spremenljivko `Luc21`, kar nam ugasne luč.
V drugem delu se nam podobni pogoji preverijo za prižiganje luči. Tudi tukaj se preverja stanje ob pritisku na tipko za luč. Razlika je v tem, da mora biti luč pred tem ugasnjena in da se vrednost `Luc21` da v 1 in ne v nič, kot je bilo pri ugašanju luči.
V zadnji vrstici imamo ukaz `Luc22:=Luc21 and AlarmLuc;` za kontrolo luči ob zaznanem vlomu. Ta vrstica nam notranjo spremenljivko za prižiganje luči ob izpolnjenem pogoju prepiše na izhod PLK-ja. Izhod PLK-ja oziroma izhodna spremenljivka se nam ne postavi na 1, če nista spremenljivki `Luc21` in `AlarmLuc` na 1. `AlarmLuc` je 0, ko je zaznan vlom, kar bomo opisali v nadaljevanju.

```
    /*  Za navadno luč  */

    Luc2.IN:=Tipka2;              /*Tipka je vezana na IX2   */
                                  /*Luc1   je   pulzni   Časovnik   s
                                  konstanto  0.5s  Na  sprednji  rob
                                  spreminjamo stanje
                                  tako da luč gori dokler ponovno ne
                                  pritisnemo tipke   */

    If (Luc22==1 and fp(Luc2.Q))  then     /*   Če luč gori se ob
                                  prvem robu ugasne  */
       Luc21:=0;
       End_if;

    If (Luc22==0 and fp(Luc2.Q))  then     /*  Če luč ne gori se ob
                                  prvem robu prižge  */
       Luc21:=1;
       End_if;

  Luc22:=Luc21 and AlarmLuc;      /*   Če je AlarmLuc==1 potem luči
                                  delajo drugače so ugasnjene */
```

5.1.1.2 Razlaga programa za luč na stopnicah s časovnikom

Prvo se prepiše zunanja spremenljivka v notranjo časovno spremenljivko. Tudi tukaj je pomembna velikost časovne konstante, saj je od tega odvisno, koliko časa bo luč gorela. Ker je vhod direktno vezan na časovnik, nam oscilacije ob preklopu ne delajo problemov. Časovnik deluje tako, da se ob prvem robu vhoda v časovnik začne odštevati časovna konstanta. Ob prvem robu na vhodu `.IN` se nam postavi `.Q` v stanje 1. Ko časovna konstanta preteče, se nam `.Q` nastavi na 0: ko se odšteje celotna konstanta se luč ugasne. Ta tip luči je dosti enostavnejši za realizacijo, saj nam ni potrebno ugašati luči. Luč bi lahko ugasnili, če bi imel naš časovnik ponastavitev (reset). V tem primeru bi ob pritisku na tipko najprej preverili, če luč že gori. Če luč ne gori, bi normalno vezali vhod na časovnik, v nasprotnem primeru bi se vhod vezal na reset, kar bi nam postavilo časovnik na nič in luč bi nam ugasnila. Na ta način bi lahko prihranili nekaj električne energije saj luč ne bi gorela po nepotrebnem.
Pri tem programu za luč bi lahko na isto luč vezali različne vhode. Tako bi luč prižgali za različno dolge časovne intervale. Če bi luč prižgali pri vznožju stopnic bi nam recimo gorela dlje, kot če bi jo prižgali pri vrhu stopnic. To bi bilo uporabno saj za hojo po stopnicah navkreber potrebujemo več časa kot za spuščanje po njih.
V programu je zopet dodan pogoj alarma, tako da luč deluje samo ob nezaznanem vlomu.
V program je dodan ukaz za blokado luči. Tako se na ta način lahko ugasne vse luči ali samo izbrane skupine. To bi se lahko uporabljalo v otroških sobah. Tako bi lahko to predstavljalo pomoč pri pošiljanju otrok k spanju. S tem aktiviranim ukazom se ne da več prižgati luči v določenem sektorju. Ta sektor je lahko samo ena luč, ena soba ali cela hiša. To aktivacijo bi bilo dobro vezati na časovnik, tako da bi luči po nekem pretečenem času zopet začele delati.

```
/* Za luč na stopnicah  */

Luc1.IN:=Tipka1;                  /*Tipka je vezana na IX0   */
                                  /*Luc1 je pulzni časovnik tako da
                                  luč po pritisku na tipko gori eno
                                  minuto   */
  Luc11:=Luc1.Q and AlarmLuc;     /*   Če je AlarmLuc==1 potem luči
                                  delajo drugače so ugasnjene */

    UgasniLuc:=UgasniLucIn;       /* Vhod za luč */

    If UgasniLuc==1 then          /*Dokler je UgasniLuc enak 1 je luč
                                  ugasnjena   */
       Luc11:=0;
    End_If;
```

5.1.2 Upravljanje peči za centralno kurjavo ali ogrevanje

Ogrevanje bi lahko krmilili s pomočjo PID regulatorja. V glavnem imajo vsi PLK-ji že integriran nek PID regulator, tako da je potrebno nanj samo priklopiti ustrezne vhode in izhode. Če PLK nima integriranega PID regulatorja, ga lahko naredimo sami, kot je prikazano v spodnjih vrsticah.

5.1.2.1 Razlaga programa za krmiljenje ogrevanja

Na začetku imamo nastavitev konstant. Konstanta `Zelvred` nam pove, kakšna je želena vrednost temperature. V bolj izpopolnjenem programu bi se to vrednost lahko nastavljalo s pomočjo operacijske plošče. Operacijska plošča pri CyBro PLK-ju je prikazovalnik z vgrajenimi tipkami. Če želimo bolj zapletene operacijske plošče, v katere lahko dodamo tudi kak program, lahko uporabimo prikazovalnike, občutljive na dotik. Na takih prikazovalnikih lahko teče tudi SCADA (Nadzorni grafični program). Prikazovalniki, občutljivi na dotik, se pri avtomatizaciji hiše ne bi uporabljali, razen v izjemnih primerih. Uporaba ne bi bila finančno upravičena, saj je cena prikazovalnika, občutljivega na dotik, enaka ceni parih PLK-jev, kar pomeni, da bi nam tak prikazovalnik zvišal ceno brez povečanja funkcionalnosti. Taki prikazovalniki bi se uporabljali zgolj zaradi prestiža.
Spremenljivka Temperatura je tukaj uporabljena kot konstanta. V resnici bi bila to temperatura, ki bi jo brali iz analognega vhoda. V tem primeru smo dodali to notranjo spremenljivko, ker nismo imeli modula z analognimi vhodi.
V vrstici `if retentive_fail or Ojacanje=0 then` se nastavijo konstante samo ob posebnih priložnostih, ko je katerikoli izmed pogojev izpolnjen.
Prvi pogoj `retentive_fail` je izpolnjen, ko nam odpove retentiv pomnilnik. To je pomnilnik v PLK-ju, ki se nam ob prenehanju napajanja ne izbriše. Ohranja ga baterija, ki se nahaja v PLK-ju. Ker je ta pomnilnik zelo omejen, se vanj vpisuje samo spremenljivke, ki jih je nujno potrebno ohraniti.
Poleg tega da je retentiv pomnilnik omejen, ima še en zelo velik problem. Zaradi njega se na terenu pojavi veliko problemov pri okvarah PLK-jev.
Če zamenjamo PLK in na novega naložimo enak program, se nam lahko zgodi, da le-ta ne bo deloval. To se zgodi takrat ko preberemo program iz PLK-ja ker nimamo zadnje verzije programa na svojem računalniku. Problem je v tem da je potrebno posebej prebrati iz spomina tudi spremenljivke iz retentivnga pomnilnika. Če tega ne storimo, ob ponovnem nalaganju programa na nov PLK, naložimo samo program in ne tudi retentivnih spremenljivk.
Poleg tega, da pozabimo na program in retentivne spremenljivke, mora biti tudi program zelo slabo napisan, saj se retentivni pomnilnik spreminja s programom.
To se nam lahko zgodi, če isti program predeluje več ljudi, ki ne poznajo celotnega programa.
Pri drugem pogoju `Ojacanje=0` se nam ponastavijo tudi konstante, saj ojačanje PID regulatorja nikoli ne more biti 0.
Nastavitvi konstant sledi PID algoritem. Vsi ukazi se izvršujejo iz desne proti levi, kot je normalno za vse programske jezike. Najprej se izračuna vrednosti vseh konstant PID regulatorja, nakar se vrednosti teh konstant uporabi v izračunu.
Nato se izračuna `reg_dev`, ki nam predstavlja napako med izmerjeno in želeno vrednostjo temperature. Temu sledi izračun izhoda, ki se ga naredi z numeričnim odvajanjem spremenljivk. V tem izračunu se sedaj uporabi konstante, ki smo jih izračunali na začetku.
Po končanem izračunu se prestavi vrednosti za eno mesto, kar nam predstavlja časovno premaknitev spremenljivk.

Čisto na koncu je potrebno izračunati še vrednost izhodne spremenljivke (y:=y+dy;). Nova vrednost y je enaka stari vrednosti, kateri prištejemo spremembo, ki se ju zgodila v zadnji časovni periodi.

```
/* Nastavitev konstant */
  ZelVred:=25;
  Temperatura:=25;

  if retentive_fail or Ojacanje=0 then
    Ojacanje:=10;
    ti:=300;
    td:=60;
  end_if;

/* PID algoritem */

  Ka:=td/(td*20*0.01+1);
  Kb:=1/(td*20*0.01+1);
  Kc:=1/(ti*(td*20*0.01+1));
  Kd:=td*20*0.01/(td*20*0.01+1);
                          /*   ZelVred   je naša želena vrednost
                          (Setpoint)  */
                          /*     Temperatura   je   naša   izmerjena
                          vrednost  */

  reg_dev:=((ZelVred- Temperatura)*100)/(SP_HI-SP_LO);

  dy:=Ka*(reg_dev-2*reg_dev1+reg_dev2)+Kb*(reg_dev-reg_dev1);
  if (y>0 or reg_dev>0) and (y<100 or reg_dev<0) then
    dy:=dy+Kc*reg_dev;
  end_if;
  dy:=Kd*dy1+Ojacanje*dy;

  reg_dev2:=reg_dev1;
  reg_dev1:=reg_dev;
  dy1:=dy;
  y:=y+dy;
```

Da dobimo prave vrednosti konstant PID regulatorja, je potrebno sistem postaviti in potem s pomočjo meritev in modeliranja narediti PID regulator. Za to bi nam lahko zelo pomagal SCADA sistem, iz katerega bi lahko dobili vse potrebne podatke za modeliranje. PID nam ne bi krmilil peči. Po naših izkušnjah bi bilo bolje, če bi nam PID krmilil mešalni ventil in črpalke, za peč pa bi uporabili kak enostavnejši tip regulacije.

5.1.3 Upravljanje rolet

Upravljanje rolet je lahko narejeno tako, da le-te dajejo vtis prisotnosti stanovalcev. To dosežemo na tak način, da se rolete premikajo preko tedna v nekem naključnem zaporedju. Policija nas vedno opozarja, da naj se med počitnicami navzven ne vidi, da nas ni doma, kar lahko dosežemo s tem, da se rolete spustijo in dvignejo tudi takrat, kadar nas ni doma.
Rolete lahko poleg tega uporabimo tudi pri zniževanju stroškov ogrevanja. To naredimo tako, da imamo zunaj hiše senzor svetlobe in senzor temperature. Za senzor temperature lahko uporabimo senzor, ki je uporabljen že za regulacijo gretja. S senzorjem svetlobe pa zaznamo kdaj se znoči, kar nam pove, da lahko spustimo rolete in s tem ne izgubimo dnevne svetlobe. S senzorjem temperature si pomagamo tako, da ne spustimo rolet vedno, ko se stemni ampak samo, ko se temperatura spusti pod nek prag. Rolet ne spuščamo vedno avtomatsko, ker se stanovalci počutijo bolj utesnjene v stanovanju s puščenimi roletami in tudi ponoči je lahko pogled skozi okno lep, še posebej ob lepi jasni noči.

5.1.3.1 Razlaga programa za upravljanje rolet za mraz

Na začetku programa je potrebno najprej preveriti vhod za temperaturo. Če je ta manjši od 205, pomeni, da je s senzorjem nekaj narobe. To vemo po tem, ker pri taki vrsti komunikacije vrednost ne sme biti pod 4 mA, kar pri nas predstavlja 205, saj imamo na vhod pripeljano vrednost od 4 do 20mA.
Po preverjanju napake vhoda lahko začnemo s preračunom. Imamo 10 biten A/D (analogno/digitalni) pretvornik, kar pomeni, da imamo na razpolago 1024 vrednosti. Problem teh 1024 vrednosti je v tem, da se razporedijo na vrednosti vhoda od 0 do 20mA. Tako imamo vrednosti od 0 do 4 mA neizkoriščene za prenos informacije o temperaturi. Povedo nam samo, ali je komunikacija prisotna. Na ta način izgubimo 20% vrednosti A/D pretvornika, ki niso izkoriščene za prenos informacij.
Ker smo si za termometer izbrali vrednosti od -50 do +50°C, nam za pretvorbo razpona 100°C ostane samo 819 vrednosti A/D pretvornika. Tako dobimo ločljivost manjšo od 0,1°C. To ločljivost bi lahko izboljšali z 12 bitnim A/D pretvornikom, ki ima 4096 vrednosti.
Podobno kot je pripravljena temperatura za uporabo v PLK-ju, je potrebno pripraviti tudi osvetljenost.
Ko imamo ti vrednosti pripravljeni, jih lahko uporabimo v programu. Tako so uporabljene vrednosti v `fp (TemperaturaZunaj<450) and (Svetlost<450) and TempError<>1 and RoletaDol<>1 then,` kjer preverjamo, če je »temperatura pod +5°C«, »osvetljenost pod določeno mejo«, »ni napake vhoda« in »roleta ni spuščena«. Pri pogoju za temperaturo preverjamo pogoje samo na pozitivnem robu, tako da dobimo signal za spuščanje rolet samo ob prehodu čez prag, ki ga določimo v pogojih.
Če tukaj ne bi preverjali ob prvem robu, bi imeli ob izpolnjenih pogojih ob vsakem prehodu programa čez kodo nov ukaz za spuščanje rolet. To bi lahko škodilo modulu za rolete.
Osvetljenost preverjamo zato, ker smo se odločili, da želimo imeti v stanovanju čim več naravne svetlobe. Poleg tega se nam, če sveti sonce v stanovanje, stanovanje tudi ogreva, kar še dodatno zmanjša stroške ogrevanja.
Če so zgornji pogoji izpolnjeni, se nam v drugem delu programskega stavka spustijo rolete (`RoletaVeter.IN:=1;`).
Poleg tega stavka bi lahko dodali tudi podobne stavke za vse rolete, ki bi jih želeli ob teh pogojih spustiti.
Spremenljivka `RoletaVeter` je časovnik, kar pomeni, da je potrebno signal za spuščanje rolet držati v stanju 1, nek, določeno dolg impulz.

Rolete se v našem programu avtomatsko tudi vzdignejo. Ta del programa je predstavljen v razlagi programa za upravljanje rolet ob vročini.

```
/* Upravljanje rolet mraz/noc */

    If Temp<205 then
       TempError:=1;
       Else
       TempError:=0;
       End_if;

    TemperaturaZunaj:=Temp-205;        /* Prvo odštejemo 4 mA  */
                                       /* Območje je 819 Nič stopinj
                                       celzija je 410*/
                                       /* 8,2 je ena stopinja */
                                       /*  Temperatura ima 4-20mA 10
                                       biten A/D (1024) območje -50
                                       do +50*/
    Svetlost:=Svet-205;                /* Isto kot za temperaturo */
      If   fp   (TemperaturaZunaj<450)   and   (Svetlost<450)   and
     TempError<>1 and RoletaDol<>1 then
                                       /*  Če  je  pod  +5  Tukaj  bi
                                       dodali  se  pogoj  če  bi  imeli
                                       uro v PLK-ju*/
        RoletaVeter.IN:=1;             /* Rolete spustimo če je zunaj
                                       temno in mrzlo */
        End_if;
```

5.1.3.2 Razlaga programa za upravljanje rolet ob vročini

Program za spuščanje rolet ob hudi vročini je zelo podoben programu za spuščanje rolet ob mrazu. Razlika je samo v pogojih, ki morajo biti izpolnjeni, da se rolete spusti.
Rolete se tukaj spustijo, če je »zunaj zelo toplo«, »ni napake vhoda za temperaturo« in »rolete niso spuščene«. V tem primeru ne preverjamo kako svetlo je, saj smo predvidevali, da se nam ponoči ti pogoji ne morejo izpolniti. Če bi bili mnenja, da se lahko ti pogoji izpolnijo tudi ponoči in takrat ne bi želeli, da se rolete spustijo, bi dodali pogoj za osvetljenost, podobno kot je to narejeno v programu za upravljanje rolet ob mrazu.
Drugi del programa je namenjen dvigovanju rolet ob izboljšanju vremenskih pogojev. Tukaj preverjamo, če je »temperatura znotraj pogojev«, če »rolete niso vzdignjene« in če »je zunaj svetlo«.
Ta del programa vpliva tudi na prejšnji program za spuščanje rolet ob mrazu. Dodan je predvsem za olajšanje bivanja v stanovanju in ne iz energetskih vidikov.
Ostali deli programov za upravljanje rolet so v osnovi namenjeni olajšanju bivanja ali večjemu udobju v hiši, saj nam ni potrebno uporabljati raznih klim in ogrevanj, pa imamo kljub temu primerno temperaturo za bivanje.
Ker nam izničijo potrebo po energetsko potrošnem uravnavanju temperature, nam privarčujejo tudi nekaj denarja. Mnenja smo, da bi lahko na ta način, s spuščanjem rolet, tudi znatno zmanjšali potrebo po klimatiziranju prostorov poleti.
Za vse te predpostavke mora biti hiša, seveda, tudi primerno toplotno izolirana.

```
/* Upravljanje rolet vročina/dan */

      If fp (TemperaturaZunaj>656) and TempError<>1 and RoletaDol<>1
then
                                    /* Če je nad +30 Tukaj bi
                                    dodali se pogoj če je ura*/
         RoletaVeter.IN:=1;         /* Rolete spustimo če je
                                    vroče*/
         End_if;

               If fp (Svetlost>450) and RoleteZgoraj<>1 and
TemperaturaZunaj<656 and TemperaturaZunaj>450 and RoletaVeter.Q<>1
then
         RoleteGor.IN:=1;
         End_if;
```

5.2 Varnost pred vlomi

5.2.1 Alarm

Pri alarmu je sistem narejen tako, da nas v primeru sprožitve zvočno in svetlobo opozori. Ko se aktivira alarm, na primer ob vlomu, se v nekaj sekundah vklopi sirena in začnejo utripati luči. Utripanje luči je dodano zaradi tega, da se vlomilec prestraši, saj naj bi sirena in utripanje luči izvale paniko pri vlomilcu in ga pregnale.
Pri alarmu lahko tudi spustimo rolete in tako zapremo vse poti. Spuščanje rolet bi bilo bolj smiselno v primeru, če bi imeli alarm že na ograji in bi s tem vlomilcu otežili vhod v hišo. V tem primeru bi torej spustili rolete, ko bi zaznali vlomilca pred hišo in ne, ko je že v hiši.
Zapiranje vlomilca v hišo bi namreč lahko naredilo več škode kot koristi. Vlomilec bi lahko med poskusom pobega poškodoval notranjost hiše. Zapiranje bi imelo smisel, če bi hoteli na vsak način preprečiti, da nam vlomilec ne odnese kake večje dragocenosti. Pri tem pa nam ne bi veliko pomenila ostala škoda, ki bi pri tem nastala.

5.2.1.1 Razlaga programa za alarm

V primeru programa za alarm imamo prve nastavitve konstant narejene ob prvem prehodu programa skozi kodo. To pomeni takrat, ko je `first_scan` enak 1 in se nam izpolni pogoj iz pogojnega (IF) stavka.
Ob prvem prehodu skozi program tako nastavimo spremenljivke AlarmLuc in Tretjina. AlarmLuc nastavimo na 1, kar pomeni, da alarm ni vključen.
Če alarma tukaj ne bi nastavili na 1, bi se nam ob vsakem zagonu vključil alarm in z njim vse sirene in druge nastavitve, ki so predvidene ob alarmu.
Spremenljivko Tretjina nastavimo na 1, ker jo potrebujemo pri periodičnem spreminjanju spremenljivke iz 0 v 1, ki jo uporabljamo za utripanje luči.
Spremenljivka `AlarmOn` je uporabljena za vklop alarma. Če je enaka 0, je alarm izključen, če je 1, pomeni da je alarm vključen. Alarm vključimo s pomočjo zunanje tipkovnice, ki nam spreminja vhod `AlarmOn` glede na kodo, ki jo vtipkamo.
Ko preverimo, če je alarm vključen, sledijo pogoji za okna, vrata in senzorje premikanja. Tako se alarm vključi, če se odpre okno, kar pomeni, da se vhodna spremenljivka `Okno` postavi v 1. Enako se alarm vključi, če se odprejo vrata, kjer se spremenljivka `Vrata` spremeni v 1. Enako je tudi za senzorje premikanja.
Pri vratih in oknih bi zaznali odprtost s pomočjo induktivnega senzorja vgrajenega v okvir. Tako bi morali postaviti senzor k okovju in bi zaznali, da se je del okovja premakni. To bi veljalo za aluminijasta, plastična in lesena okna ali vrata. Pri železnih, protivlomnih vratih, bi bila montaža še enostavnejša, saj bi bilo vseeno kam postavimo senzor.
Ker induktivni senzor zazna železo, mora biti montiran v okovje, kjer zaznamo položaj zapaha, ki je železen. Pri železnih oknih ali vratih, bi bila važna le vrsta senzorja, saj bi bil senzor montiran v železen okvir, kar lahko dela nekaterim senzorjem probleme pri zaznavanju. Če bi se želeli izogniti montiranju senzorja v okovje in bi imeli plastične, aluminijaste ali lesene okvirje, bi lahko montirali v okvir železno ploščico in bi s senzorjem na ta način zaznali, ali je okno odprto ali zaprto.
Če se katerikoli od pogojev »okno«, »premikanje« ali »vrata« izpolni, se nam sproži spremenljivka `Alarm` in `AlarmRdLuc`. `Alarm` je časovnik z zakasnitvijo, ker bi drugače ob vsakem prihodu skozi vrata začel delovati alarm, tako pa imamo čas, da alarm izključimo.

Poleg tega smo dodali še spremenljivko `AlarmRdLuc,` ki se vključi takoj, ko se alarm sproži. Ta spremenljivka je potrebna kot opozorilo, da bo alarm začel delovati.
Nadaljnji koraki programa za alarm se začnejo izvajati po določenem časovnem zamiku, ko se začnejo prižigati in ugašati luči in se vklopi sirena. Prižiganje in ugašanje luči traja do izklopa alarma. Sirena ima časovno omejitev delovanja. To smo dodali predvsem zaradi moteče sirene, ki bi lahko motila okolico.
K sireni in lučem bi lahko dodali še spuščanje rolet ali katero koli drugo operacijo, za katero bi želeli, da se vklopi ob alarmu.
Ko alarm izključimo, se vse spremenljivke, ki smo jih uporabili v programu alarma, nastavijo nazaj v mirovno stanje. To je stanje, v katerem je spremenljivka ob inicializaciji PLK-ja.

```
 if first_scan then          /* Ko je first_scan 1 to pomeni da gre
                             program prvič čez programsko kodo. Ob
                             prvem izvrševanju kode je potrebno
                             nastaviti vrednosti osnovnih konstant in
                             spremenljivk */
       AlarmLuc:=1;
       Tretjina:=1;
       end_if;

    /*  Alarm  */

                             /*  Ce je alarm je AlarmLuc=0  */
            If AlarmOn==1 and (Okno==1 or Premikanje==1 or Vrata==1)
then
                             /* AlarmOn je pri priključitvi alarma,
                             Okno je 1, če je odprto*/
           Alarm.IN:=1;
                             /*  Premikanje je 1, če je senzor
                             premikanja, Vrata je 1, če je odprto*/
           AlarmRdLuc:=1;
                             /* Prižge se lučka da se bo aktiviral
                             alarm */
           End_if;

           If Alarm.Q==1 then
                             /* Alarm je časovnik z zakasnitvijo */
                AlarmLuc:=0;
                             /*  Sirena in Luci začnejo delovati po
                             zakasnitvi */
                AlarmSirena.IN:=1;
                             /*  Sirena tuli samo določen čas nato
                             utihne */
     /* Tukaj lahko dodamo pulze za spuščanje rolete */
              End_if;

          If AlarmOn==0 then
                        Alarm.IN:=0;
                        AlarmRdLuc:=0;
                        AlarmLuc:=1;
                        AlarmSirena.IN:=0;
                        End_if;
```

5.2.1.2 Razlaga programa za utripanje luči ob alarmu

Ta del programa je v osnovi zelo enostaven, saj je vse v enem programskem stavku. Tako najprej preverimo, če je `AlarmLuc` enak 0. Če je ta pogoj izpolnjen, se nam na izhodno spremenljivko prepiše spremenljivka `Pulzi`. Luči delujejo, ko je spremenljivka `Pulzi` enaka ena. Tukaj lahko dodamo vse luči, za katere bi želeli, da nam utripajo ob alarmu.
Razlaga programa za periodično spreminjanja stanja spremenljivke je bolj zapletena. Za začetek fiksiramo stanje v nekem položaju s pomočjo pogoja `Tretjina`. To je potrebno, ker se `Tretjina` lahko uporabi tudi kje drugje v programu, in ne samo pri alarmu. Ta opcija je uporabna pri razhroščevanju programa na objektu. Tako lahko `Tretjino` uporabimo v programski kodi ali pa je že predhodno kje uporabljena in bi jo radi blokirali.
`Pulzi` so izhod iz tega dela programa. Če je spremenljivka `Pulzi` enaka 0, postavimo spremenljivko `Tretjina_z`, ki je časovnik, v 1. Ko ta časovnik spremeni stanje v 1, v drugem stavku tega dela programa nastavimo `Tretjina_z` v 0 in `Pulzi` v 1. To so pogoji, da se `Pulz` postavi v 1. Temu sledi del, kjer nastavimo `Pulz` v 0. S tem preverimo, če je `Pulz` enak 1 in če je, zaženemo časovnik `Tretjina_o`. Za tem preverimo prehod spremenljivke v stanje 1. Ko se to zgodi, nastavimo spremenljivki `Pulzi` in `Tretjina_o` v 0.
Na koncu programa je potrebno še nastavljanje spremenljivk ob blokadi impulza. Ob blokadi se vse spremenljivke, uporabljene pri pulzu, nastavijo na 0, kar nam predstavlja mirovno stanje.

```
/*  Utripanje luči ob alarmu  */

If AlarmLuc==0 then          /*  Ko  je  AlarmLuc  0  potem  luči
                             utripajo */
   Luc22:=Pulzi;
   End_if;



 If Tretjina==1 and Pulzi==0 then /* Utripanje luči */
               Tretjina_z.IN:=1;       /*  S  tretjino  se  da
                                       blokirati utripanje */
               End_if;
     If Tretjina==1 and fp (Tretjina_z.Q==1) then
               Pulzi:=1;
               Tretjina_z.IN:=0;
               End_if;
     If Tretjina==1 and Pulzi==1 then
               Tretjina_o.IN:=1;
               End_if;
     If Tretjina==1 and fp (Tretjina_o.Q==1) then
               Pulzi:=0;
               Tretjina_o.IN:=0;
               End_if;
If Tretjina==0 then
               Pulzi:=0;
               Tretjina_z.IN:=0;
               Tretjina_o.IN:=0;
               End_if;



/* Za luc na stopnicah  */
Luc1.IN:=Tipka1;
  Luc11:=Luc1.Q and AlarmLuc
                                       /*   Če  je  AlarmLuc==1  potem
                                       luči   delajo   drugače   so
                                       ugasnjene */
UgasniLuc:=UgasniLucIn;
If UgasniLuc==1 then
   Luc11:=0;
End_If;

/*  Za navadno luc  */
Luc2.IN:=Tipka2;
If (Luc22==1 and fp(Luc2.Q))  then
   Luc21:=0;
   End_if;
If (Luc22==0 and fp(Luc2.Q))  then
   Luc21:=1;
   End_if;
Luc22:=Luc21 and AlarmLuc;           /*   Če  je  AlarmLuc==1  potem
                                       luči   delajo   drugače   so
                                       ugasnjene */
```

5.3 Varčevanje z energijo in vodo

5.3.1 Varčevanje s toploto

Ker je ogrevanje vodeno s PLK-jem, je lahko prilagojeno vsaki hiši. Merimo lahko zunanjo in notranjo temperaturo. Tako lahko z PID regulatorjem optimiziramo porabo energije za ogrevanje. Poleg tega lahko z manjšimi stroški razdelimo hišo na več ogrevalnih vej (sektorjev), kar pomeni, da bi lahko teoretično ogrevali vsako sobo v drugem temperaturnem režimu.
Za krmiljenje vsake veje bi potrebovali najmanj termometer in ventil. Za boljše izkoristke toplotne energije bi potrebovali več termometrov, in sicer enega ali več za temperaturo prostorov, enega za temperaturo ogrevalne vode, enega za temperaturo povratne vode in enega za temperaturo okolice.
Termometri v prostoru bi nam povedali trenutno temperaturo, s pomočjo katere bi lahko izračunali odstopanje od želene temperature. Iz razlike temperatur ogrevalne vode, povratne vode ter pretoka bi lahko izračunali toplotno energijo, ki jo dovedemo v prostor, kar bi nam olajšalo regulacijo. Za to bi potrebovali še toplotne izgube stavbe.

5.3.2 Varčevanje z električno energijo

Z elektriko bi lahko varčevali pri lučeh, ki ob pritisku na tipko gorijo že vnaprej določen čas, saj bi se jih dalo ugasniti pred iztekom tega časa. Poleg tega lahko varčujemo z elektriko tudi tako, da se vse vtičnice, ki ne potrebujejo stalnega napajanja, kot ga na primer hladilnik, ura in televizija, ob vklopu alarma izključijo. Poleg varčevanja z elektriko nam lahko ta način prepreči tudi marsikatero skrb, saj bi vedeli, da se nam je štedilnik ali likalnik izključil, ko smo odšli od doma, tudi če smo ga sami pozabili ugasniti.
S povezavo senzorjev premikanja v sistem varčevanja z elektriko bi lahko ugašali luči in druge porabnike električne energije, ko petnajst minut ne bi zaznali gibanja. Tako bi se na primer sama izključila luč, ki bi jo pozabili ugasniti že po petnajstih minutah, in ne bi nezavedno trošili električne energije.
Vezava električnih porabnikov na avtomatsko izklapljanje pa ne bi bila smiselna v vseh prostorih. Tako bi se nam lahko zgodilo, da bi pri gledanju zelo napetega filma avtomatika izključila luč in televizor, ker bi zaznala, da se določen interval ni nič premaknilo.

5.3.3 Varčevanje z vodo

Z vodo bi lahko varčevali tako, da bi se nam ob vklopu alarma zaprla voda v celi hiši, razen v vnaprej določenih izjemah, kot sta pralni stroj in zalivanje vrta.
Na ta način se nam ne bi moglo zgoditi, da bi voda cele počitnice tekla v nič, ker je puščal kotliček v kopalnici.
Poleg tega bi lahko avtomatizirali zalivanja vrta, saj bi s pomočjo senzorja dežja lahko zaznali, koliko je treba zaliti vrt in ne bi avtomatsko zalivali vrta ob vnaprej določeni uri, kljub temu, da bi bilo to v tistem trenutku nepotrebno.
Dodatno bi lahko merili tudi relativno vlažnost (RV) zemlje. Če bi imeli RV zemlje, bi tako lahko še bolj optimizirali zalivanje vrta in porabili še manj vode.

5.4 Varovanje stanovanjske hiše in stanovalcev pred naravnimi nesrečami

5.4.1 Varovanje pred neurji

V tem sklopu smo se posvetili ravnanju ob dežju in močnem vetru. Ob dežju se nam ob odprtem oknu le-to zapre ali pa se spustijo rolete. Na ta način preprečimo, da bi nam dež padal v hišo.
Pri ravnanju ob vetru je krmiljenje malo bolj zapleteno, saj krmilimo različno glede na jakost vetra. Če veter ni premočan, je naloga krmilja pospraviti tende, da jih veter ne polomi. Ob močnem vetru pa tende vzdignemo in spustimo rolete, da nam kak predmet, ki bi ga veter lahko prinesel s seboj, ne bi razbil okna. Spuščanje rolet ob močnem vetru bi imelo enak učinek, kot ga ima pritrjevanje lesenih plošč na okna. Le-to vidimo po televiziji ob pripravi na vsak orkan v Ameriki.

5.4.1.1 Razlaga programa za zaznavanje vremenskih pogojev

S senzorji lahko zaznamo različne vremenske pogoje, kot so dež, veter in poplave. Pri zaznavanju dežja imamo na vhod v PLK vezan senzor za dež, ki lahko deluje na dva načina. Prvi je uporaba tehtnice za dež, pri drugem pa uporabimo posodo, v kateri sta dve žici priključeni na PLK, kjer spremljamo, kdaj bo voda naredila stik med njima. Pri tej metodi bi se lahko pojavil problem zaradi deževnice, ki podobno kot destilirana voda ne prevaja elektrike, tako da ta metoda ni najbolj zanesljiva.
Zaznavanje vetra bi bilo najbolj enostavno s senzorjem vetra, ki ima 4-20 mA analogni izhod. Tako bi v PLK-ju samo pretvorili ta signal in bi že dobili vrednosti, ki jih potrebujemo. Za izhod iz PLK-ja bi lahko uporabili tudi 0-10V izhod, ki ga podpirajo vsi PLK-ji z analognimi izhodi, saj je pretvorba za merjenje toka ponavadi narejena z 500mΩ uporom, ki je na vhodu v PLK. Tako se za merjenje toka meri padec napetosti na tem uporu.
Za zaznavanje poplave bi lahko uporabili nivojske vilice. To so vilice, ki so vzbujane z namenom, da se tresejo. Ko do njih pride tekočina ali katera druga snov, se frekvenca tresenja spremeni in na izhodu iz senzorja (vilic) dobimo spremenjen signal. Takšne senzorje se uporablja v industriji in so zelo dragi. Druga možnost je potopno stikalo, pri katerem se spremeni električno stanje v stikalu, ko se stikalo v obliki plovca vzdigne. V potopnem stikalu je ponavadi kroglica, ki se premika in da stik, če se potopno stikalo vzdigne v vodoravno lego. Problem tega stikala je v tem, da je potrebno precej vode, da se stikalo sploh postavi v navpično lego. Tretja možnost je podobna kot pri zaznavanju dežja, in sicer s pomočjo dveh elektrod želimo dobiti električni stik preko zbrane vode. Tukaj ne bi imeli problemov zaradi prevodnosti, saj je v taki talni vodi že dovolj raztopin, ki prevajajo elektriko.
V spodnjem programu se vhodne spremenljivke prepišejo v naše notranje.

```
/* Zaznavanje vremenskih pogojev */

    DezPada:=Dez;               /*  Sonda zazna dež in da digitalni
                                signal  */
    VeterPiha:=Veter;           /*    Vetrnica zazna veter in da
                                analogni signal  */
    PoplavaJe:=Poplava;         /*    Sonda zazna poplavo in da
                                digitalni signal  */
```

5.4.1.2 Razlaga programa za ravnanje ob dežju

Za začetek izvajanja programa v primeru dežja, mora biti izpolnjenih precej pogojev. Najprej preverimo, če pada dež, za kar je pogoj `DezPada==1`. Za tem preverimo, če imamo spuščene rolete, kar naredimo s pogojem `RoletaDol<>1`. Nato moramo preveriti, če je okno zaprto (`OknoZaprto<>1`). Če so vsi trije pogoji izpolnjeni, kar pomeni da pada dež, okno je odprto in rolete niso spuščene, se nam spustijo rolete in tako zaščitijo hišo pred dežjem, ki bi v nasprotnem primeru lahko padal skozi odprto okno.
Spremenljivka `Roleta` je časovnik, ker za krmiljenje rolet potrebujemo samo impulz za gor, ali na drug vhod, impulz za dol. S tem stavkom `RoletaSpusti:=Roleta.Q;` prenesemo naš signal časovnika na izhod PLK-ja.
Sledi upravljanje rolet po dežju. Takoj ko preneha padati dež, vzdignemo rolete. Rolete se vzdignejo samo v primeru, če nam program zaradi mraza ali vročine le-tega ne blokira.

```
/*  Ravnanje pri dežju  */

if DezPada==1 and RoletaDol<>1 and OknoZaprto<>1 then
                              /* Če dež pada se zgodi sledeče  */
                              /*  Če je okno odprto in rolete gor
                              se spustijo rolete  */
      Roleta.IN:=1;           /*   Roleta je pulzni časovnik, ki
                              traja 0,5s. Pulz damo roleti da se
                              spusti */
      end_if;                 /* Roleta ima dva vhoda en je za
                              spuščanje drug za vzdigovanje  */
   RoletaSpusti:=Roleta.Q;    /*   Če bi imeli okna, ki bi jih
                              lahko zapirali bi tukaj zaprli okno
                              */

If DezPada==0 then
   Roleta.IN:=0
   end_if;
```

5.4.1.3 Razlaga programa za ravnanje ob močnem vetru

Ob močnem vetru moramo pospraviti tende, ker bi jih v nasprotnem primeru veter lahko uničil. Da nam to ni potrebno narediti ročno, imamo tukaj narejen kratek program, ki to naredi namesto nas.
Za jakost vetra smo določili analogni vhod, ki nam pretvori tokovne vrednosti 4-20 mA v spremenljivke z vrednostmi od 40 do 200. Tako predstavlja 200 najmočnejši veter, 40 pa nam pomeni brezvetrje. Vrednosti spremenljivke so lahko tu poljubne, saj jih lahko pretvorimo, podobno kot smo jih v prejšnjih programih.
V programu preverjamo, če veter piha znotraj določenih mej, ki so 100 in 150. V tem primeru vzdignemo tende, saj bi jih tak veter lahko poškodoval.
Nad zgornjo mejo, 150, je veter že tako močan, da bi nam lahko poškodoval okna, tako da v tem primeru vzdignemo tende in spustimo rolete.
V drugem delu imamo pogoje za primer zmanjšanja moči vetra. Če moč vetra pade pod 150 in je nad 100, se rolete dvignejo. Ko moč vetra pade pod 100, se vzdignejo tudi tende.

```
/*  Ravnanje pri vetru  */

If VeterPiha>100 and VeterPiha<150 then
                                        /*    Če  piha  veter  čez
določeno mejo
                                   vzdigne tendo  */
        TendaGor.in:=1;
  elsif VeterPiha>150 then          /*   Če  piha  veter  čez  drugo
                                    mejo  vzdigne  tendo  in  spusti
                                    rolete  */
        TendaGor.in:=1;
        RoletaVeter.IN:=1;
                 end_if;

        RoletaSpusti:=RoletaVeter.Q;
        TendaVzdigni:=TendaGor.q;

If VeterPiha<150 and VeterPiha>100 then

        RoletaVeterDol.IN:=1;
  elsif VeterPiha<100 then

        TendaDol.IN:=1;
        RoletaVeter.IN:=0;
                  End_if;
```

5.4.2 Varovanje pred poplavami

Pri varovanju pred poplavami potrebujemo v kleti črpalko in nekaj senzorjev za vodo. Krmilje je zelo enostavno, saj le preverjamo prisotnost vode in glede na to, vključimo črpalko. Ko vode ni več, črpalko izključimo. Poleg tega krmiljenja je dodano tudi preverjanje delovanja črpalke, da se nam ne zgodi, da poganjamo zablokirano ali pregreto črpalko.

5.4.2.1 Razlaga programa za ravnanje ob primeru poplave

Program za primer poplave se nam aktivira, če so izpolnjeni pogoji za poplavo, voda je pri črpalki in črpalka ni v napaki. Ko so izpolnjeni ti pogoji, se črpalka za vodo priključi. Na koncu programa je zopet dodan stavek, ki nam prepiše našo notranjo spremenljivko na izhod za priključitev črpalke.

```
    /*  Ravnanje pri polavi  */
VodaJePriCrpalki:=Poplava;

     If PoplavaJe==1 and VodaJePriCrpalki==1 and CrpalkaEror:=0 then
                                /*  Če je poplav in voda tudi pri
                                črpalki se zažene črpalka  */

          CrpalkaPriklop:=1;    /*  Če  ni  vode  pri  črpalki  se
                                črpalka ustavi  */


     end_if;

Crpalka:=CrpalkaPriklop;        /*  Tukaj se zažene črpalka  */
```

5.4.2.2 Razlaga programa za upravljanje črpalk

Napako črpalke preverjamo po spodnjem postopku. Najprej počakamo nekaj časa, ker se nam črpalka ne zažene v istem trenutku, kot se nam postavi izhod za priklop črpalke.
Tako ob priklopu črpalke ne moremo takoj preveriti, če je črpalka v napaki. Ko nam ta čas, ki je nastavljen v `CrpalkaZakasnitev`, poteče, in imamo priključeno črpalko, preverimo, če je črpalka v napaki. Če je črpalka v napaki se nam postavi `CrpalkaEror` v 1, drugače je v 0.

```
/* Napaka črpalke */

CrpalkaZakasnitev.IN:=Crpalka;
                                /* Čez neko časovno zakasnitev šele
                                preverimo če črpalka dela */

                  If  Crpalka==1  and  CrpalkaZakasnitev.Q==1  and
CrpalkaBimetal==0 then
            CrpalkaEror:=1;
          else
            CrpalkaEror:=0;
       end_if;
```

5.5 Nadgradnja sistema z nadzornim SCADA sistemom

Sistem ni težko nadgraditi s sistemom SCADA, saj so vsi industrijski PLK-ji nadgradljivi s SCADA sistemom. Potrebujemo samo OPC strežnik, ki nam poveze PLK in SCADA sistem. PLK-ji imajo navadno vgrajeno komunikacijo RS-232. OPC strežnik dela na računalniku, kjer je tudi SCADA.
OPC strežnik programsko poveže izhode PLK-ja in vhode SDADA sistema. Novejši PLK-ji, imajo tudi komunikacijo RS-485 in LAN. Uporabljeni PLK ima samo komunikacijo RS-232 tako, da bi za nadgradnjo s SCADA sistemom potreboval dva modula. Prvi bi spremenil komunikacijo v RS-485 ali LAN. Drugi modul pa bi nam spremenil komunikacijo nazaj v RS-232. Tako bi lahko prenašali komunikacijo na večjo razdaljo. To je potrebno zato, ker je PLK navadno v »elektro« omari nekje v kleti, računalnik pa v delovni sobi ali nekje drugje v stanovanju, tako da je razdalja večja od 10 m.
Komunikacija je problematična tudi zaradi fizične bližine komunikacijskih povezav in močnostnih linij.
SCADA bi nam lahko delala večje in zahtevnejše izračune, ki jih PLK ni sposoben narediti. Poleg tega, bi lahko s pomočjo SCADA sistema našo hišo povezali na internet, kar bi nam omogočilo krmiljenje hiše preko svetovnega spleta.
Osnovni namen SCADA sistemov bi v našem primeru ostal ob strani. Beleženje podatkov o hiši nam ne bi preveč pomagalo, saj bi jih uporabljali bolj iz radovednosti kot za kaj uporabnega.
Podatke bi se dalo z veliko dela in preračunov uporabiti tudi za optimizacijo ogrevanja hiše, kar bi nam lahko prineslo še kak odstotek izboljšav pri ogrevanju. Drug osnovni namen je krmiljenje sistema, ki v našem primeru prav tako ne bi preveč koristil, saj bi to pomenilo, da bi namesto na stikalo ob oknu, za spuščanje rolet, šli do računalnika in tam pritisnili z miško na virtualno tipko, ki bi nam spustila roleto.
SCADA sistem po našem mnenju ne bi občutneje izboljšal delovanja našega krmilja, tako da ne bi opravičil stroškov, ki bi jih imeli z nabavo. Stroški so zelo visoki, saj je vsaka točka, ki jo lahko gledamo s SCADA sistemom, zelo draga. Programi so tudi zelo dobro zaščiteni, saj potrebujemo za normalno delovanje programa strojni ključ, ki ga damo v LPT ali USB vrata na računalniku.

5.6 Nadgradnja s krmiljenjem na daljavo z pomočjo GSM SMS sporočil

Kot je bilo v prejšnjem poglavju ugotovljeno, je nabava SCADA sistema nerentabilna, saj omogoča veliko preveč funkcij, ki jih v resnici ne rabimo. Zaradi tega bi bilo bolj smotrno, če bi krmilili hišo preko SMS sporočil. SMS sporočila nam po našem mnenju omogočajo ravno pravšnjo mero nadzora nad hišo.
Preko SMS sporočil bi lahko dobili podatke o temperaturah, splošnem stanju v hiši, vlomu, poplavi ali kaki drugi ujmi.
Omejeni bi bili samo na dolžino SMS sporočila. Vsebina bi bila takšna, kot bi želeli.
SMS sporočila bi lahko uporabili tudi za krmiljenje hiše na daljavo. Lahko bi na primer na daljavo vzdignili temperaturo v stanovanju in se tako po nepričakovano hitri vrnitvi vseeno vrnili v toplo hišo.
Poleg tega bi lahko preverjali, če imamo priklopljen alarm in podobno.

6 Sklepne ugotovitve

Dejstvo je, da se sistemi avtomatiziranih hiš gradijo in tudi vgrajujejo. Drugo dejstvo je, da so ti sistemi še vedno cenovno prevelik zalogaj za veliko večino graditeljev novih hiš ali stanovanj. Problem je tudi, da se večji sistemi hitreje amortizirajo kot manjši. Tako je bila naša prva ugotovitev, da so taki sistemi predvsem luksuznega značaja.
Ugotovili smo, da je avtomatizacija smiselna in rentabilna samo do neke meje. Po naših ugotovitvah ni dobro avtomatizirati celotne hiše, saj nikoli ne vemo, kdaj se nam lahko tak sistem pokvari. V nas uporabnikih je še vedno prisoten nek zadržek do tako visoke tehnologije.
Pri preveliki stopnji avtonomnosti avtomatizirane hiše bi se nam ob morebitni napaki v sistemu lahko zgodilo, da ne bi mogli več v hišo brez velikega kladiva in spremljajoče materialne škode.
Pri odpravi zadržkov bi lahko pomagal psihološki učinek, ki ga dobimo iz tega, da je hiša avtomatizirana z industrijskim PLK-jem. Pridevnik industrijski nam kot uporabniku daje neko notranjo sigurnost v zanesljivost sistema, saj imamo nekje v glavi, da če je industrijsko, potem ne more iti nič narobe. To bi bil lahko nek nasprotni učinek, kot se pojavi, ko nam v ušesu zazveni ime iz področja zabavne elektronike, kjer je znano, da se aparati pogosto kvarijo.
Pravilnost teh asociacij se je pokazala tudi pri tem diplomskem delu, saj se nikoli niso pojavili problemi pri PLK-ju. Nasprotno pa je bilo zelo veliko problemov pri urejanju tega besedila.
Po našem mnenju bi bilo smiselno avtomatizirati osnovne funkcije, ki so prikazane v tem delu. Večji obseg avtomatizacije bi bil predrag in brez stalnega nadzora tudi preveč tvegan za napake. Za večji obseg avtomatizacije bi potrebovali skrbnika sistema, kar nas pripelje čez meje, ki smo si jih zadali pri tem projektu, saj je naš glavni cilj avtomatizacija običajne družinske hiše.
Po našem mnenju bo sčasoma prišlo na trgu do velikega razcveta opisanih sistemov, kar bo zelo znižalo njihovo cene in jim s tem povečalo dostopnost, kar se je že zgodilo v avtomobilski industriji. Tu so namreč še pred desetimi leti veljali dodatki, kot so ABS, SRS, zračne blazine, električni pomik stekel, avtomatska klimatska naprava in podobno, kot izključno luksuzni dodatki, ki so si jih lahko privoščili samo najbogatejši. Sčasoma so se stvari tako spremenile, da sedaj vse to pričakujemo že v vseh avtomobilih.
Enak proces se sedaj odvija na trgu hišne opreme. Cene se počasi znižujejo in tudi negativni psihološki učinki bodo počasi izginili.
Če se bodo stvari spreminjale, kot so se v avtomobilski industriji, lahko pričakujemo, da bodo najprimernejši posebni sistemi, namenjeni prav stanovanjskim hišam. Hotelski moduli bodo ostali neprimerni, saj bodo funkcionalno neustrezni. PLK-ji pa bodo najzanimivejši za tiste, ki vidijo v sistemu avtomatizirane pametne hiše veselje in bodo želeli sistem nadgrajevati in funkcionalno spreminjati po lastnih željah.

7 Viri

Literatura

1. CyPro User Manual V211

2. CyBro User Manual

3. Integra BM (Reklamno gradivo)

4. Esad Jakupović, *Življenje v pametni hiši*, Moj Mikro, Delo revije, Ljubljana 2001, letnik 17, str.70-71

5. Nataša Zoltar-Bolce *Digitalno življenje v hiš*, Kapital, 23 avgust 2004, str.61-63

Internetni viri

I. Robotina PLK:
http:// www.cybro-plc.com-downloads-Special-Epipack-CyPro-CyPro%20manual%20v201.mdi

II. Dementia voice:
http://www.dementia-voice.org.uk/Projects.htm

III. Smart house unveiled, BBC news,
http://news6.thdo.bbc.co.uk/hi/english/health/newsid_799000/799128.stm

IV. Sci/Tech House of the future, BBC news,
http://news.bbc.co.uk/hi/english/sci/tech/newsid_194000/194560.stm

V. Products:
http://www.electronichouse.com/categories.html

VI. Event Control System(ECS):
http://www.hometoys.com/htinews/aug00/reviews/ecs/ecs.htm

VII. Smart House Web Survey:
http://www.cc.gatech.edu/fce/seminar/fa98-info/smart_homes.html

VIII. PLCMAN
http://www.plcman.co.uk/theplc/

IX. X-10 Organization:
http://www.x-10.org/

X. CEBus Industry Council, Inc.:
http://www.cebus.org/

XI. EIB Association:
http://www.eiba.org/

XII. Robotina:
http:// www.robotina.si

IZJAVA

Izjavljam, da sem diplomsko delo izdelal samostojno pod vodstvom mentorja doc. dr. Boštjana Murovca. Izkazano pomoč drugih sodelavcev sem v celoti navedel v zahvali.

V Ljubljani, Podpis: