KAUNO TECHNOLOGIJOS UNIVERSITETAS
ELEKTROS IR ELEKTRONIKOS FAKULTETAS

Aldis Adomavičius

P&ID SCHEMOS TRANSFORMAVIMAS Į UML ĮTERPTINĖMS SISTEMOMS

Baigiamasis magistro projektas

Vadovas
Prof. dr. V. Deksnys

KAUNAS, 2016

KAUNO TECHNOLOGIJOS UNIVERSITETAS

ELEKTROS IR ELEKTRONIKOS FAKULTETAS

ELEKTRONIKOS INŽINERIJOS KATEDRA

P&ID SCHEMOS TRANSFORMAVIMAS Į UML ĮTERPTINĖMS SISTEMOMS

Baigiamasis magistro projektas

Elektronikos inžinerija (kodas 621H61002)

Vadovas
Prof. dr. Vytautas. Deksnys

Recenzentas
Doc. dr. Žilvinas Nakutis

Projektą atliko
Aldis Adomavičius

KAUNAS, 2016

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Elektros ir elektronikos fakultetas

(Fakultetas)

Aldis Adomavičius

(Studento vardas, pavardė)

Elektronikos inžinerija 621H61002

(Studijų programos pavadinimas, kodas)

Baigiamojo projekto „P&ID schemos transformavimas į UML įterptinėms sistemoms“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 16 m. gegužės 30 d.

Kaunas

Patvirtinu, kad mano **Aldžio Adomavičiaus** baigiamasis projektas tema *„P&ID schemos transformavimas į UML įterptinėms sistemoms“* yra parašytas visiškai savarankiškai, o visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

(vardą ir pavardę įrašyti ranka) *(parašas)*

Adomavičius A. P&ID schemos transformavimas į UML įterptinėms sistemoms. *Elektronikos inžinerijos magistro b*aigiamasis projektas / vadovas prof. dr. Vytautas Deksnys; Kauno technologijos universitetas, Elektros ir elektronikos fakultetas, Elektronikos inžinerijos katedra.

Kaunas, 2016. 62 psl.

SANTRAUKA

Šiame darbe pateikiama P&ID schemos, automatizavimo schemos plačiai naudojamos pramonėje, transformavimo į UML metodikas, sudarant konvertavimo žemėlapį, naudojamą sudarinėjant UML diagramas.

Transformacijai atlikti, P&ID diagramose esantiems elementams surasti atliekamas vaizdo atpažinimo šablonų radimo metodas, o punktyrinėms linijoms – Hough transformacija, tekstui – OCR programa „*Tesseract*“. Sugrupuotą informacija perduodama UML diagramų braižymo programai „Visual Paradigm“. Sudarytos diagramos keičiamos programiniu kodu pagal elementų keitimo taisykles.

Transformacijos, sukurtos naudojant šią metodiką, adekvatumui patikrinti kuriamas naujas produktas bendradarbiaujant su įmone AB „Axis Industries“.

Sukurta dalinai automatizuota keitimo metodika transformuojanti P&ID schemas į UML ir generuojanti programinį kodą, kai P&ID diagramos yra konvertuojamos iš kitų kompiuterinių programų į nuotraukas, naudojnati vienodo dydžio standartizuotus elementus.

Reikšminiai žodžiai (iki 8 žodžių):

UML, P&ID, Vaizdo atpažinimas, Linux, Įterptinės sistemos, Matlab

Adomavičius A. P&ID Diagram Transformation to UML for Embedded Systems. Final project of *Electronics engineering master degree* / supervisor doc. dr. Vytautas Degsnys; Kaunas University of Technology, Faculty of Electrical and Electronics Engineering, department of Electronics Engineering.

Kaunas, 2016. 62 psl.

SUMMARY

In this paper the methodology of P&ID diagram, automation schematics widely used in industry, transformation to UML is introduced, creating map of conversion, which is used to create UML diagrams.

To perform transformation, for identification of elements in P&ID image recognitions template matching is performed, for dashed lines – Hough transformation, text – OCR program „*Tesseract*“. Grouped information is transfered to UML design tool „Visual Paradigm“. Created diagrams are exported to software code, based on element transformation rules.

The adequacy of transfomations using this methodology is tested by developing new product, cooperating with „Axis Industries“ firm.

In this paper partialy automated system for transformation from P&ID to UML diagrams and generating source code is developed, where P&ID diagrams are converted from other computer programs to pictures, using same size of standart elements.

Keywords (up to 8 words):

UML, P&ID, Image recognition, Linux, Embedded systems, Matlab

Turinys

Sutrumpinimai

Sutrumpinimas	Apibūdinimas
P&ID	Automatizavimo schema (angl. Piping and Instrumentation Diagram/ Drawing)
UML	Unifikuota modeliavimo kalba (angl. Unified Modeling Language)
OMT	Objektinio modeliavimo metodika (angl. Object Modeling Techniques)
OOSE	Objektui orientuotos programinės įrangos inžinerija (angl. Object-Oriented Ssoftware Engineering)
PID	Proporcinis Integruojantis Diferencijuojantis reguliatorius (angl. Proportional-Integral-Derivative controller)
MDA	Modeliu paremta architektūra (angl. Model Driven Architecture)
PIM	Nuo platformos nepriklausomas modelis (angl. Platform Independent Model)
PSM	Platformos specifinis modelis (angl. Platform Specific Model)
OCR	Optinis Teksto atpažinimas (angl. Optical Character Recognition)
dpi	Taškai colyje (angl. dots per inch)
GSM	Globalus mobilių telefonų ryšio standartas (angl. Global System for Mobile communications)
UART	Universalus Asinchroninis Imtuvas-Siųstuvas (angl. Universal Asynchronous Receiver-Transmitter)

1 Įvadas

Kiekvienais metais didėja įterptinėms sistemoms reikiamų įgyvendinti funkcijų sudėtingumas ir jų kiekis, todėl sistemos sukūrimo trukmę tampa sunku prognozuoti ir išlaikyti numatytą biudžetą. Tai lemia sistemos kūrėjų darbą, kurie nuolat ieško lanksčių sprendimų įgyvendinant vartotojų lūkesčius [9]. Kuriant įterptinės sistemos programinę įrangą, autorius [8] siūlo naudoti objektais paremtą sistemos kūrimo būdą. Kai sistema yra objektinė, tuomet jos kūrimui bei analizei patogu naudoti UML, nes šia modeliavimo forma apibrėžiama dauguma objektinės sistemos aspektų [9].

P&ID diagrama yra sistemos įrangos ir ją naudojančios proceso tėkmės diagrama. Joje vaizduojama įranga bei jos poveikis sistemai su reguliavimo kontūrais, apribojimais, paleidimo bei stabdymo algoritmais. Algoritmai dažniausiai yra atvaizduojami supaprastintai, o jeigu jie yra pakeičiami realioje sistemoje ne visada yra pakeičiami diagramose, kadangi neretai programavimo darbus ir P&ID diagramų sudarymą atlieka sričių inžinieriai. Be to sudėtingų algoritmų vizualizavimas paprastai yra labai imlus laikui darbas.

Norint, kad sistema atitiktų profesionalią kokybę ji turi:[23]

- Patenkinti vartotojo poreikius;
- Būti paprastai ir greitai aptarnaujama;
- Būti dokumentuota.

Naudojant UML, taikomam procesui ar metodologijai sistemos kūrimas yra perkeliamas iš vystymo etapo į analizės ir projektavimo etapą. Tai sumažina riziką ir sukuria architektūros aprašymo bei modeliavimo priemonę, prieš pradedant programavimo darbus. Gaunama nauda atsiperka, nes sistema tampa dokumentuota, greitai aptarnaujama ar modifikuojama. UML modelio elementus galima transformuoti į programos kodą. Taip sutaupoma laiko ir sumažėja išlaidos programavimo darbams [22,23].

Šiame darbe autorius pateikia metodus skirtus sudaryti UML diagramoms iš P&ID, kuriose yra atvaizduojami visi valdomi elementai ir ryšiai tarp jų. Šiais metodais siekiama sumažinti laiko sąnaudas, klaidų skaičių rašant programinį kodą, dokumentuoti sistemą.

1.1 Darbo tikslas ir uždaviniai

Šio darbo tikslas: *sukurti metodus P&ID schemos transformavimui į UML, taikomus įterptinėms sistemoms.*

Šiam tikslui pasiekti formuojami tokie uždaviniai:

1. Sukurti metodus P&ID diagramų transformacijai į UML aprašus;

2. Suklasifikuoti proceso tarpusavio ryšių diagramas (P&ID), „TAG“-ų tipus ir valdymo kontūrus taip, kad būtų galima vienareikšmiškai priskirti „TAG“-ams tam tikras funkcijas, bei susieti jas su reguliavimo kontūrais arba blokuočių elementais;
3. Sukurti „TAG“ ryšių ir funkcijų duomenų (bazės) struktūrą;
4. Sukurti P&ID elementų paieškos algoritmą, užpildantį duomenų bazę.

Duomenų bazės struktūroje turi būti numatyta:

- matavimo rezultatų ir matavimo prietaisų sąrašas;
- vykdymo elementų sąrašas;
- signalizacijų ir aliarmų sąrašas;
- valdymo kontūrų sąrašas;
-funkcinis aprašymas (priskiriant tipines funkcijas/blokus konkretiems elementams ar grupėms).

2 UML ir P&ID

2.1 UML

UML (angl. *Unified Modeling Language*, liet. Unifikuota Modeliavimo Kalba) – tai standartinė grafinė kalba, pritaikyta programuojamų sistemų specifikavimui, vizualizavimui, projektavimui ir dokumentavimui. UML sukurta remiantis trimis pagrindiniais objektinio modeliavimo metodais (Booch, OMT ir OOSE), įtraukus keletą dalykų iš modeliavimo kalbos sandaros (angl. *design*), objektinio programavimo ir architektūrinio aprašo kalbų [1].

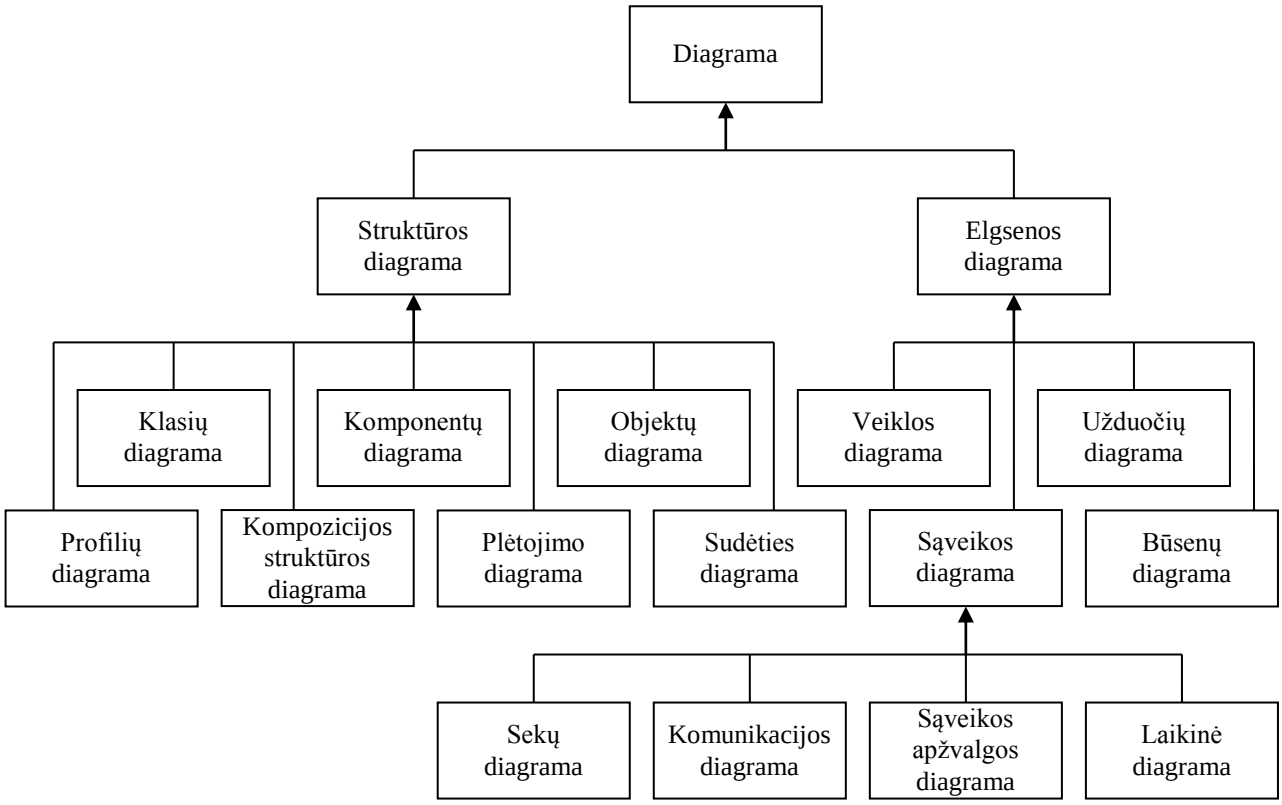
UML sparčiai populiarėja visame pasaulyje ir yra naudojama daugelio IT specialistų, kurie projektuoja programinę įrangą. Sistemos yra abstrakčiai atvaizduojamos modeliais, naudojant gerai apibrėžtą sąvokų žodyną ir taisykles. Jų modeliai aprašomi tiksliai ir vienareikšmiškai. UML yra vizuali kalba, turinti grafinę notaciją, skirtą įvairių programinės įrangos architektūros aspektų modeliavimui. Naudojantis UML modeliais galima greičiau ir lengviau suprasti programinės įrangos struktūrą bei veikimo principus, todėl jie yra efektyviai naudojami programinės įrangos architektūros dokumentavimui ir projektavimo sprendimų aptarimui. UML gali pateikti daug projektuojamos sistemos atvaizdų, pasitelkdama įvairias struktūrines ir elgsenos diagramas [1].

UML modelio sukūrimas naudojamos šiems tikslams:

- Sukūrus sistemos modelį, galima rasti kelis sprendimo variantus;
- Modelio sukūrimas padeda valdyti sistemos kompleksiškumą;
- Problemos modeliavimas prieš sistemos sukūrimą sumažina klaidos tikimybę ir problemos sprendimo kaštus.

2.1.1 UML diagramų tipai

UML versijoje 2.4.1 pateikiama 14 diagramų tipų, kurios suskirstytos į dvi kategorijas. Septyni diagramų tipai apibrėžia struktūrinę informaciją. Kiti septyni – pagrindinius elgsenos tipus, įskaitant keturis diagramų tipus, kurie apibrėžia skirtingus sąveikos aspektus. Šios diagramos gali būti suskirstytos į hierarchines kategorijas kaip parodyta 2.1 paveiksle [1].



2.1pav. UML 2.x versijos diagramų struktūra

Statinis sistemos vaizdas pabrėžia jos statinę struktūrą, naudojantis objektais, atributais, operacijomis ir ryšiais. Jį sudaro [1]:

- Klasių diagrama;
- Kompozicijos struktūros diagrama.

Sistemos dinaminis vaizdas pabrėžia dinaminę sistemos elgseną, parodant tarpusavio bendravimą tarp objektų ir jų vidinių būsenų pakitimus. Jį sudaro [1]:

- Sekų diagrama;
- Veiklos diagrama;
- Būsenų diagrama.

2.1.1.1 Struktūros diagramos

Struktūros diagramos vaizduoja sistemos struktūrą ir yra plačiai naudojamos programinės įrangos sistemų dokumentavimui. Struktūros diagramos apibrėžia tai, kas turi būti modeliuojamoje sistemoje [25]:

- Klasių diagrama (angl. *Class diagram*) paaiškina sistemos struktūrą, nurodydama sistemos klases, jų savybes ir ryšius tarp jų;
- Komponentų diagrama (angl. *Component diagram*) vaizduoja kaip sistema yra padalinta į komponentus ir parodo jų tarpusavio ryšius;

- Kompozicijos struktūros diagrama (angl. *Composite structure diagram*) paaiškina vidinę klasės struktūrą ir jos struktūros galimą bendradarbiavimą;
- Plėtojimo diagrama (angl. *Deployment diagram*) naudojama modeliuojant sistemos techninę įrangą ir šios įrangos sąvybių išsidėstymą;
- Objektų diagrama (angl. *Object diagram*) parodo pilną arba dalinį suprojektuotos sistemos struktūrinį vaizdą tam tikru laiko momentu;
- Sudėties diagrama (angl. *Package diagram*) apibūdina kaip sistema yra padalinta į logines grupes, nurodant priklausomybes tarp jų;
- Profilių diagrama (angl. *Profile diagram*) yra UML išplėtimo mechanizmas leidžiantis pritaikyti arba modifikuoti esantį modelį su konstruktais, kurie yra būdingi tai sričiai, platformai ar metodui.

Pasinaudojant klasėmis ir ryšiais tarp jų, nurodomi kuriamos sistemos struktūriniai aspektai. Klasės gali turėti savybes ir operacijas. Sąsajos bei apibendrinimo ryšiai leidžia sukurti objektiškai orientuotas hierarchijas.

2.1.1.2 Elgsenos diagramos

Elgsenos diagramos yra naudojamos sistemos elgsenos ir funkcionalumo vaizdavimui, paaiškinant, kas turi nutikti modeliuojamoje sistemoje:

- Veiklos diagrama (angl. *Activity diagram*) parodo sistemos komponentų darbo eigos žingsnius;
- Būsenų diagrama (angl. *State machine diagram*) standartizuoja daugelio sistemų notaciją, t.y. nuo kompiuterių iki verslo procesų. Ji gali būti naudojama tinklo komunikacijų protokolams modeliuoti;
- Panaudos atvejų diagrama (angl. *Use case diagram*) parodo funkcionalumą, kurį sistema turi, t.y. kokia sistemos paskirtis ir kas bus jos vartotojas.

Sistemos elgsenos modeliavimas aukščiausiu abstrakcijos lygmeniu prasideda nuo susijusių su ja dalyvių identifikavimo ir panaudos atvejų diagramų sudarymo. Detalesnei elgsenos specifikacijai yra naudojamos sąveikos ir veiklos diagramos. Veiklos, būsenų ir sekų diagramos apibūdina vieno sistemos objekto dinaminį elgesį kaip būsenų kaitą ir yra naudojamos parodyti visas būsenas, kuriose gali būti objektas bei parodo, kokie įvykiai pakeičia būsenas [15].

2.1.1.3 Sąveikos diagramos

Sąveikos diagramos yra elgsenos diagramų pogrupis, apibrėžiantis valdymo ir duomenų tėkmę tarp modeliuojamos sistemos elementų [25]:

- Komunikacijos diagrama (angl. *Communication diagram*) parodo sąveiką tarp objektų ar dalių pagal pranešimų eilę;
- Sąveikos apžvalgos diagrama (angl. *Interaction overview diagram*) yra veiklos diagrama, kurioje mazgai nurodo sąveikos diagramas;
- Sekų diagrama (angl. *Sequence diagram*) parodo kaip objektai komunikuoja tarpusavyje, naudojant pranešimų sekos terminus;
- Laikinė diagrama (angl. *Timing diagram*) yra specifinė sąveikos diagrama, kurioje daugiausia dėmesio skiriama laiko apribojimams.

2.2 P&ID

P&ID (Automatizavimo schema) yra diagrama, naudojama procesų pramonėje, kurioje yra atvaizduojamas technologinių įrenginių ir matavimo prietaisų tarpusavio ryšys, skirtas vykstančiam procesui valdyti. Technologiniai įrenginiai, komunikacijos, vykdymo įtaisai, automatizavimo priemonės ir ryšiai tarp jų žymimi specialiais žymėjimais pagal ISA S5.1 ir ISO 14617-6 standartus [4].

Visiems prietaisams aprašyti yra naudojamas sutartinis automatizavimo prietaisų žymėjimas, apimantis grafinius, raidinius ir skaitmeninius žymeklius. Viršutinėje grafinio žymėjimo dalyje rašomas raidinis prietaiso ir matuojamojo dydžio funkcinis žymėjimas. Apatinėje dalyje – prietaiso pozicinis skaitinis ir raidinis žymėjimas. Funkcinio žymėjimo raidžių paaiškinimai pateikti 2.2.1 lentelėje [4].

2.2.1 lentelė Automatizavimo schemos raidiniai žymėjimai

	Matuojamo dydžio parametrai		Atliekama funkcija		
	Matuojamas arba inicijuojamas kintamasis	Modifikatorius	Parodymas arba pasyvi funkcija	Išėjimo funkcija	Modifikatorius
A	Analizė		Aliarmas		
B	Degiklis, degimas		Pasirenkamas	Pasirenkamas	Pasirenkamas
C	Pasirenkamas			Reguliavimas, valdymas	
D	Pasirenkamas	Skirtumas			
E	Įtampa		Jutiklis		
F	Srautas	Santykis			
G	Pasirenkamas				
H	Rankinis poveikis				Viršutinė riba
I	Elektros srovė		Indikacija		
J	Galia	Skenavimas			
K	Laikas, laiko grafikas	Laikas tarp pokyčio			
L	Lygis		Šviesa		Apatinė riba
M	Pasirenkamas	Momentinis			Vidurinė, tarpinė riba
N	Pasirenkamas		Pasirenkamas	Pasirenkamas	Pasirenkamas
O	Pasirenkamas		Apribojimas		
P	Slėgis, vakuumas		Jungties vieta		
Q	Kiekis	Integravimas, sumavimas			
R	Radiacija		Įrašymas		
S	Greitis, dažnis	Apsauga		Kontaktų valdymas	
T	Temperatūra			Perdavimas	
U	Keli kintamieji				
V	Vibracija, mechaninė analizė			Sklendė, užsklanda	
W	Svoris, jėga				
X	Neklasifikuotas	X-ašis	Neklasifikuotas	Neklasifikuotas	Neklasifikuotas
Y	Įvykis, būsena, būvis	Y-ašis			
Z	Pozicija, dimensija	Z-ašis			

Jeigu standartinių žymėjimų neužtenka, Tuomet galima naudoti vieną iš pasirenkamų (nepanaudotų) raidžių, kurioms galima priskirti savo sugalvotą funkciją, bet tada tą funkciją reikia šalimais paaiškinti.

Remiantis standartai siūloma naudoti didelę įvairovę įrenginių simbolių pateiktų 2.2.2 lentelėje. Visų galimų variantų pateikti fiziškai neįmanoma, kadangi yra be galo daug skirtingų įrengimų ir jų skaičius kinta keičiantis technologijoms [4].

2.2.2 lentelė Įrenginių simboliniai žymėjimai

	Vamzdis		Termiškai izoliuotas vamzdis		Apsauginis vamzdis		Šaldomas arba šildomas vamzdis
	Apsauginis maišymo indas		"Pusės vamzdžio" maišyklė		Horizontalus slėginis indas		Vertikalus slėginis indas
	Siurblys		Vakuminis siurblys arba kompresorius		Maišas		Dujų balionas
	Ventiliatorius		Ašinis ventiliatorius		Radialinis ventiliatoris		Džiovyklė
	"Pakavimo" kolona		Distiliavimo kolona		Krosnis		Šaldymo bokštas
	Šilumokaitis		Šilumokaitis		Šaldiklis		Plokštelinis ir korpuso šilumokaitis
	Dviejų vamzdžių šilumokaitis		Tiesių vamzdžių šilumokaitis		U formos šilumokaitis		Spiralinis šilumokaitis
	Dengta dujų ventiliacija		Lenkta dujų ventiliacija		Filtras		Piltuvas
	Garo gaudyklė		Stebėjimo stiklas		Slėgį mažinanti sklendė		Lankstus vamzdis
	Sklendė		Reguliuojanti sklendė		Rankinė sklendė		Atbulinis vožtuvas
	Adatinė sklendė		Peteliškinė sklendė		Diafragminė sklendė		Rutulinė sklendė

Matavimo prietaisų ir atliekamų funkcijų simboliniai žymėjimai pateikti 2.2.3 lentelėje

2.2.3 lentelė Prietaisų arba funkcijų simboliniai žymėjimai

	Pagrindinėje vietoje	Vietinis prietaisas	Pagalbinėje vietoje
Diskretiniai			
Dalijamasis ekranas, valdymas			
Kompiuterio funkcija			
Programuojama logika			

Dažniausiai pramonėje naudojamus įrenginius galima suskirstyti pagal atliekamą funkciją į:

- Vykdymo įtaisus;
- Jutiklius.

Vykdymo įtaisas – tai įrenginys, kurio darbas keičia sistemos parametrus. Pagal valdymo principą jie išskiriami į dvi grupes:

- Variklius (siurbliai, ventiliatoriai, transporteriai ir pan.);
- Sklendes (užsklanda, vožtuvas ir pan.).

Jutiklis – matavimo ar signalizavimo elementas, kuris pakeičia matuojamąjį dydį į kitą dydį tolimesniam apdorojimui. Pagal signalo pobūdį jie skirstomi į:

- Diskretinius (rėliniai kontaktai) ;
- Analoginius.

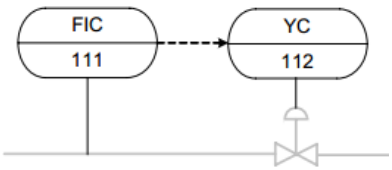
2.2.1 Tipiniai valdymo kontūrai

Įrengimų valdymo dėsnius galima išskirti į:

- Diskretinius
- Tolydinius (analoginius).

Diskretiniai valdymo dėsniai susideda iš nuoseklių skirtingų operacijų, kurias apibūdina tam tikros sąlygos, o tolydiniai dėsniai yra nepertraukiami [13]. Dažniausiai pasitaikantis ir lengviausiai realizuojamas yra tolydinis valdymo dėsnis, nes jų algoritmas yra nepriklausomas nuo jokių sąlygų .

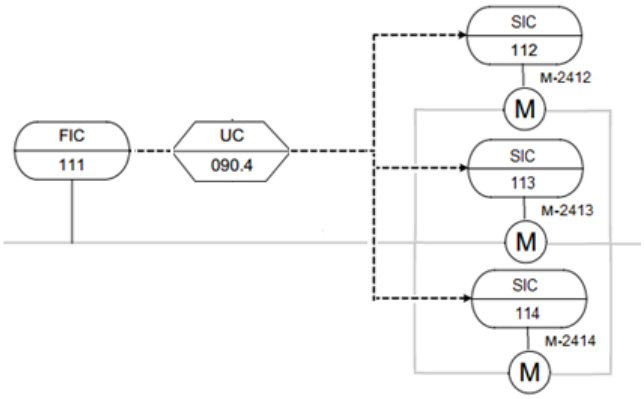
Paprasčiausias tolydinio valdymo kontūras susideda iš vieno matavimo prietaiso ir vieno poveikio elemento, kuris pagal nustatytą dėsnį palaiko užduotą vertę.



2.2.1.1 pav. Srauto matavimo ir valdymo kontūras

2.2.1.1 paveiksle pateiktame pavyzdyje esančiame valdymo kontūre reikia palaikyti nustatytą matavimo vertę, tam yra numatytas valdymo įtaisas bei analoginis matavimo prietaisas. Pastarasis gali būti valdomas analoginiu būdu, tada valdymo signalas apskaičiuojamas matematiškai pagal PID reguliatorių.

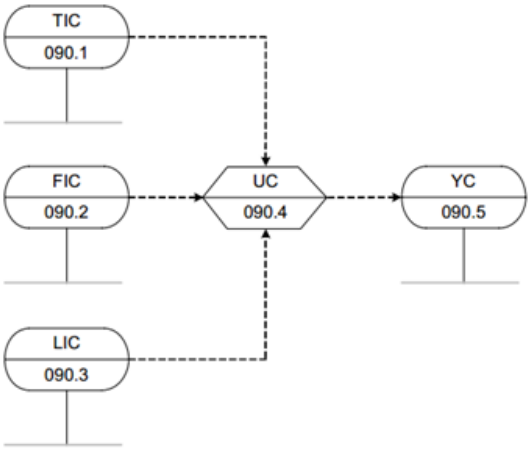
Kai yra naudojami keli įrenginiai, reguliuojantys tą patį parametrą, galima naudoti visiems įrenginiams tą patį PID reguliavimo metodą. Jeigu visi įrenginiai būtų įjungti vienu metu, tai sistema būtų neefektyvi atsižvelgiant į elektros sąnaudas. Todėl yra naudojama papildoma logikos grandis, kuri valdo jų paleidimo bei stabdymo seką.



2.2.1.2 pav. Srauto reguliavimo kontūras, naudojantis 3 siurblius

Šiame kontūre, kai nustatytam parametrui palaikyti nepakanka vieno siurblio našumo, naudojantis papildoma logika yra uždelsiamas kito siurblio paleidimas ir lėtas jo našumo didinimas. Taip yra išvengiama hidraulinio smūgio ir srautas yra didinamas palaipsniui. Kai dirba keli siurbliai mažesniu nei nustatyta našumu, vienas yra palaipsniui stabdomas, kad staigiai nesumažėtų srautas.

Galima valdyti ne tik kelis įtaisus pagal vieną parametrą, bet ir vieną įtaisą pagal kelis parametrus. Taip sukuriant papildomą kintamąjį, kuris būtų susietas matematinėmis operacijomis su kitais parametrais.



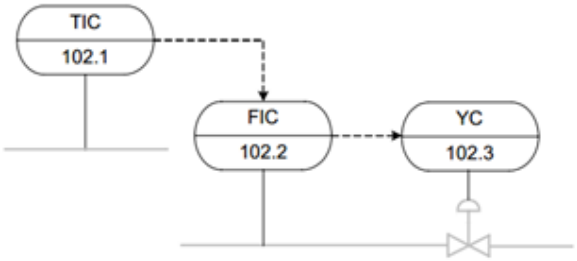
2.2.1.3 pav. Kelių kintamųjų reguliatorius

Paveiksle 2.2.1.3 ysančiame pavyzdyje yra parodytas vandentiekio sistemos papildymo reguliavimo kontūras, atsižvelgiant į esamą lygį talpoje LIC090.3 bei įvertinant pildymo srautą FIC090.2 ir temperatūrinę korekciją TIC090.1.

Kaskadiniai valdymo kontūrai yra naudojami tuomet, kai yra valdoma lėtai besikeičianti matavimo vertė ir greičiau kintanti kita vertė, kuri daro įtaką lėtai besikeičiančiajai. Tokiu atveju naudojami du reguliavimo kontūrai:

- Papildomas – valdo poveikio elementą pagal greitai besikeičiantį parametrą;

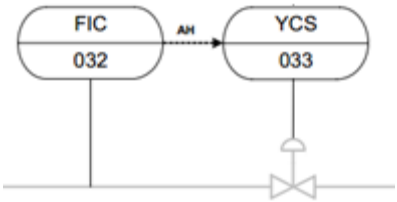
- Pagrindinis – valdo papildomo reguliavimo kontūro užduotį pagal lėtai besikeičiančia matavimo vertę.



2.2.1.4 pav. Kaskadinis temperatūros ir srauto reguliatorius

2.2.1.4 pav. yra kaskadinis temperatūros valdymas, valdant srauto reguliavimo užduotį, pagal kurią yra valdoma sklendės pozicija. Šis metodas taikomas kai reikia sumažinti sistemos trikdžių įtaką ir gauti sistemos valdymą su mažesniais svyravimais.

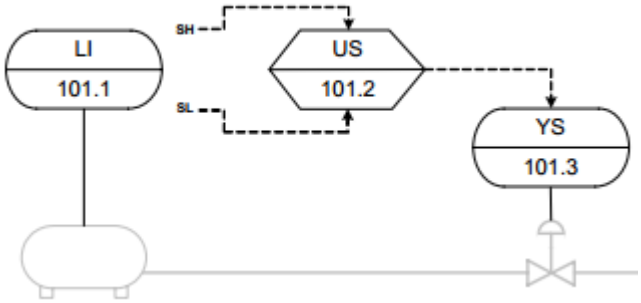
Apribojimai atlieka veiksmus, kurie siekia ar bando išlaikyti saugų proceso darbinį rėžimą, kai parametro reikšmė viršija nustatytas ribas.



2.2.1.5 pav. Srauto reguliatorius su per didelio srauto apribojimu

2.2.1.5 pav. pavaizduotas srauto PID reguliatorius, kai srautas pasiekia nustatytą aukštą ribą yra apribojamas sklendės valdymas.

Paprastoms užduotims atlikti naudojamas diskretinis valdymas.



2.2.1.6 pav. Diskretinis lygio valdymas

2.2.1.6 pav. yra talpos pildymo valdymas, naudojantis vienas lygio signalas su dviejomis aukšto ir žemo lygio ribomis, pagal kurias yra atliekamas loginis sklendės valdymas. Suveikus žemo lygio ribai sklendė yra atidaroma, o pasiekus aukštą ribą uždaroma [18].

2.3 Modelių transformavimas

Sistemos kūrimo metu automatinis modelio perkėlimas ar jo transformavimas tarp modeliavimo etapų ženkliai sumažintų užimamo laiko kiekį bei žmogiškojo faktoriaus įtaką, negu tai darant ne automatizuotu įrankiu. Tarp projektavimo ir įgyvendinimo koncepcijos yra skirtingų abstrakcijos lygių, todėl reikia tinkamai suderinti specifinius įrangos reikalavimus. Modeliavimo elementai, kurie yra reikalingi taikomosioms priemonėms aprašyti kyla iš proceso inžinerijos reikalavimų, turi būti suprantami valdymo sistemos ir proceso valdymo inžinieriams. Modeliavimo koncepcija ir metodai yra reikalingi specifikuojant įvairias valdymo struktūras, valdymo kontūrus, valdymo algoritmus bei prietaisus būdingus sistemai [2].

Modelio transformacija yra būdas, kuris užtikrina kad modelių grupės yra susijusios, naudojant transformacijos aprašus. Transformacijos aprašas yra transformacijos taisyklių rinkinys, aprašantis vieno modelio realizacijos transformaciją į kito modelio realizaciją. Transformacijos taisyklė – aprašas kaip viena ar kelios konstrukcijos pirminio modelio transformuojamos į vieną ar daugiau konstrukcijų kitame modelyje [19].

Modelių transformacijas galima klasifikuoti kaip endogenines ir egzogenines. Endogeninės transformacijos yra modelių aprašytų naudojant tą pačią kalbą transformavimas, naudojama optimizavimui, prastinimui. Egzogeninės transformacijos yra modelių aprašytų naudojant skirtingas kalbas transformavimas, kaip migravimas, kodo generavimas, apgrąžos inžinerija [20].

Iš modelių paremtos architektūros (MDA), jų transformacijos gali būti suskirstytos į:

- Nuo platformos nepriklausančio modelio (PIM) arba nuo platformos priklausančio modelio transformacijos (PSM),
- PSM – kodo transformacijos,

Modelių transformacijos vykdymo eiga:

1. Naudojant metakalbą, specifikuojamos pirminio ir galutinio modelio kalbos, aprašančios jų metamodelius.
2. Kai metamodeliai specifikuoti aprašoma transformacija tarp jų.

2.4 „*AUKOTON*“ UML modelis

Automatizavimo pramonėje nėra vyraujančio standarto modeliavimui ar automatizavimo koncepcijoms. Tai įtakojo projekto „*AUKOTON*“ atsiradimą 2008 metais, kurio tikslas sukurti koncepciją ir metodus proceso valdymo programų kūrimui. Šis projektas atliktas suomių, bendradarbiaujant su tarptautinėmis įmonėmis iš procesų industrijos. Jo tikslas sukurti vientisą kūrimo kelią nuo projektavimo, dokumentavimo iki valdymo programos, remiantis modeliais

paremtais kūrimo metodais. Modelių naudojimo tikslas yra dalinai automatizuoti taikomosios programos kūrimą, kuri suderinama su įranga ir atlieka nustatytą funkciją. Nors pagrindinė modelių naudojimo priežastis yra palengvinti programos kūrimo procesą, bet galima panaudoti valdomo proceso simuliacijai. Tai padeda priimti projektinius spendimus [3].

„*AUKOTON*“ sistemos kūrimas yra paremtas pramonės naudojamomis valdymo praktikomis, koncepcijomis, kurios įgyvendinamos naudojant UML Automatizavimo profilį. „*AUKOTON*“ kūrimo procesas sudaro trys pagrindiniai etapai:

- Nustatomi reikalavimai, apribojimai ir įranga;
- Pagal pirmo etapo duomenis parenkama valdymo koncepcija. Sistemos funkcijos ir komponentų aprašymas yra vykdomas nepriklausomai nuo sistemos platformos. Tai leidžia ateityje pakartotinai panaudoti sprendimus;
- Nepriklausoma nuo platformos koncepcija susiejama su sistemos įranga įgyvendinamoje platformoje bei transformuojama į taikomąją programą [2].

2.4.1 Automatizavimo profilis

Automatizavimo profilis sukurtas remiantis keliais baziniais UML profiliais bei juos praplečiant, kad tenkintų automatizavimo pramonės poreikius. Šis profilis yra sudarytas iš trijų subprofilių, kurie apima daugumą automatizavimo aspektų [12]:

- Reikalavimų (angl. *Requirements*);
- Pasiskirstymo ir tarpusavio sąveikos (angl. *Distribution and Concurrency*);
- Įrangos ir resursų (angl. *Devices and Resources*).

Automatizavimo profilį galima naudoti įvairiose automatizavimo taikomosiose programose, nes jis yra orientuotas programiniams sprendimams aprašyti. Šiuo profiliu kuriami elementai yra nepriklausomi nuo platformos.

2.5 Antro skyriaus išvados

- Apibendendrinant inžinerinę praktiką, valdymo algoritmai gali būti klasifikuojami į grupes pagal atliekamą funkcionalumą. Todėl turint kuo daugiau praktikoje naudojamų pavyzdžių galima sudaryti vienareikšmius objektų ir valdymo kontūrų atitikmenis.
- P&ID diagramose atvaizduojama naudojama įranga ir jos ryšiai, todėl iš šios informacijos galima sukurti pasiskirstymo - tarpusavio sąveikos, įrangos - resursų diagramas.

3 P&ID transformavimas į UML

3.1 Sistemos dekompozicija

Siekiant sistemą geriau suprasti, ji skaidoma į sudedamąsias dalis „iš viršaus į apačią“ (angl. *Top-Down*) dekompozicijos principu. Šio skaidymo principo pasirinkimo motyvai tokie [5,6]:

- Objektinis požiūris į sistemą, kuris įgalina sistemos supratimą kaip objektų visumą;
- Išryškina ryšius tarp komponentų konkrečiame abstrakcijos lygmenyje.

3.1.1 pav. Sistemos dekompozicijos principas pagal objektus

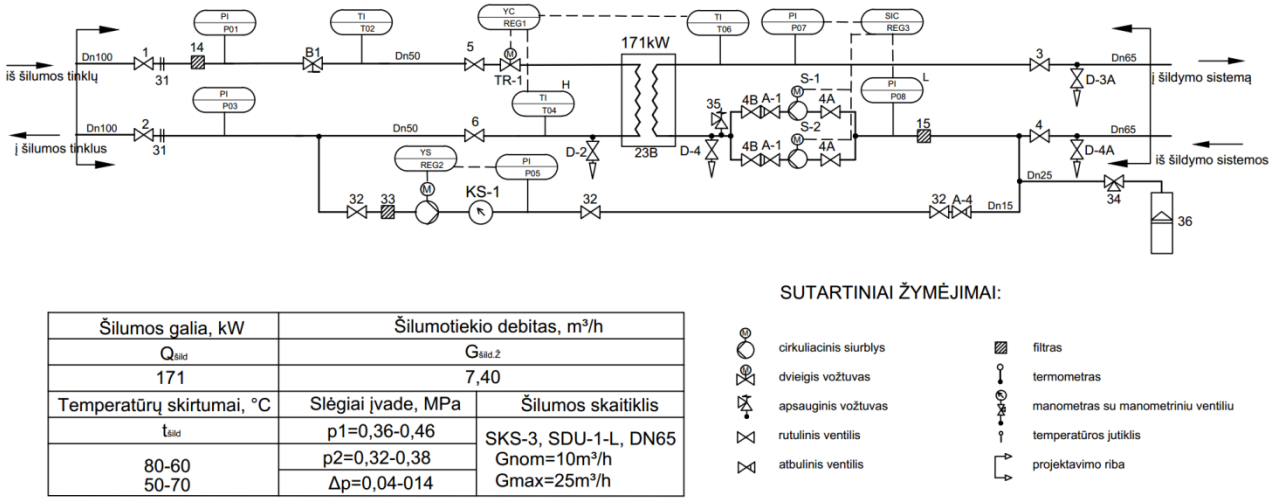
„Iš viršaus į apačią“ sistemos skaidymo idėja pavaizduota 4.1 paveiksle. Skaidant automatizuotas valdymo sistemas, yra tikslinga naudotis tokiomis skaidymo gairėmis: [5,6]

- Dekompozicija atliekama taip, kad sukurti objektai vaizduotų realios sistemos objektus, kurie įtakoja vienas kitą tiek elektriniais signalais, tiek fizikiniais dydžiais vykstančiame procese;
- Kita sistemos objektų dalis yra abstraktesnė, t.y. sukurta remiantis technologiniais reikalavimais (pvz. reguliavimo kontūrai) .
- Toliau sistema skaidoma į nedidelio abstrakcijos laipsnio objektus (pvz. pavaros, vožtuvai ar kiti poveikio elementai).

Nedidelio abstrakcijos laipsnio objektus valdomame procese identifikuoti yra nesudėtinga, o skaidymas į didesnės abstrakcijos objektus, tikslinga skaidyti taip, kad jie vaizduotų tam tikrą, aiškiai apibrėžiamą technologinę operaciją arba tam tikrą technologinių įrenginių funkcijų grupę.

Kita dalis sistemos objektų gali būti labiau abstraktūs, sukurti remiantis technologiniais reikalavimais ir neturintys akivaizdaus fizinio atitikmens valdomame objekte, pvz.: reguliavimo kontūrai, technologinio pobūdžio sąryšiai tarp atskirų gamybos posistemių ir pan.

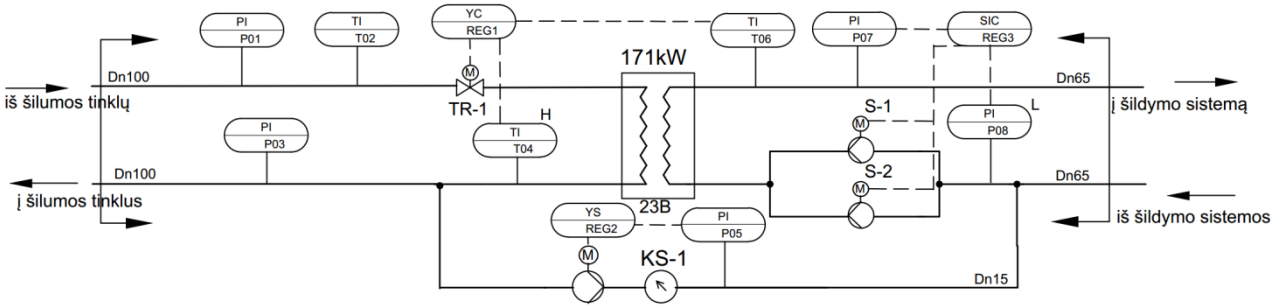
Pagrindinis UML sudarymo iš P&ID uždavinys yra vienareikšmis valdymo kontūro struktūros atpažinimas pagal nagrinėtus pavyzdžius 2.2.1 skyriuje.



Šilumos galia, kW	Šilumotiekio debitas, m³/h	
$Q_{šild}$	$G_{šild.ž}$	
171	7,40	
Temperatūrų skirtumai, °C	Slėgiai įvade, MPa	Šilumos skaitiklis
$t_{šild}$	p1=0,36-0,46	SKS-3, SDU-1-L, DN65 Gnom=10m³/h Gmax=25m³/h
80-60 50-70	p2=0,32-0,38 Δp=0,04-014	

3.1.2 pav. Šilumos punkto P&ID

Kadangi P&ID diagramoje yra atvaizduojami valdomi ir nevaldomi ojektai, sistema yra sunkiau suprantama. Elementai, kurie nėra valdomi sistemos nėra aktualūs, todėl tolimesniam nagrinėjimui yra panaikinami.



3.1.3 pav. Šilumos punkto supaprastinta P&ID

3.1.3 pav. pavaizduota šilumos punkto automatizavimo schema, kurią sudaro:

- Slėgio matavimo prietaisai P01, P03, P05, P07, P08.
- Temperatūrų matavimo prietaisai T02, T04, T06.
- Sklendė TR-1, kuri valdoma pagal temperatūrą T06 ir T04 aukštą ribą.
- Siurlys PS-1, kuris įjungiamas sumažėjus slėgiui P05 ir stadomas jam padidėjus.
- Siurbliai S-1 ir S-2, kurie palaiko vandens slėgį sistemoje P07 ir yra apribojami nuo mažo slėgio P08.

Kituose skyriuose, bus sudaromos šios sistemos UML objektų diagramos, naudojantis keliais skirtingais metodais.

Prieš atliekant transformaciją, reikia sugrupuoti visus galimus elementus į grupes pagal jų atliekamą funkciją. Visose sistemose visus elementus išskaidžius iki įrenginių galima išskirti į tris dažniausiai sutinkamas pagrindines valdymo sistemos grupes:

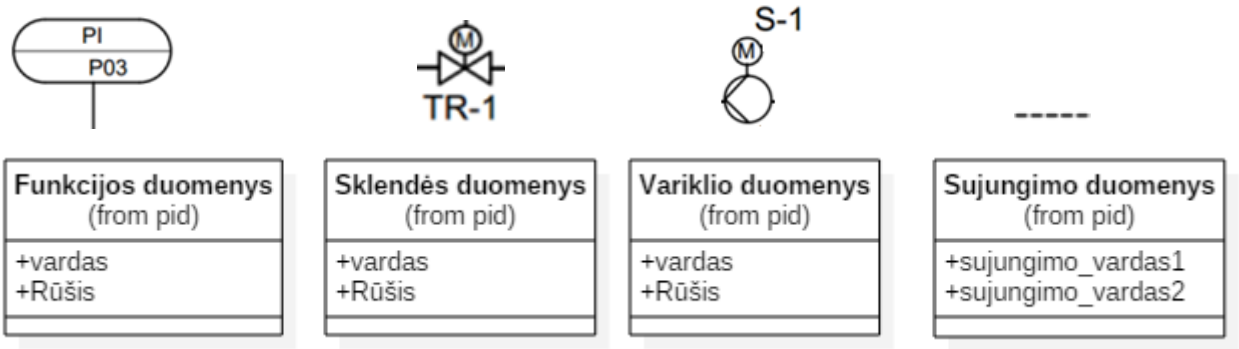
- Aktuatorius (elementus darančius sistemai poveikį)
    - Variklius (ventiliatorius, siurblius, transporterius...)
    - Sklendes (reguliuojančias sklendes, uždarančius vožtuvus...)
- Jutiklius (elementus, kurie avaizduoja sistemos paramerus)
    - Analoginius matavimo prietaisus
    - Diskretinius matavimo prietaisus
- Valdymo funkcijas (elementus nusakančius valdymo dėsnius, bei funkcijas)
    - Tolydinio reguliavimo
    - Diskretinio reguliavimo

Diagramose aktuatorių elementai yra ryškiai išskiriami skirtingomis formomis, o jutikliai bei valdymo funkcijos atvaizduojamos tokiais pat elementais tik skiriasi jų raidiniai žymėjimai. Todėl tie elementai diagramose bus įvardinti funkcijomis su skirtingomis rūšimis.

Ryšiai tarp šių grupių yra pavaizduoti sujungimo elementais:

- Fizinio lygmens (vamzdžiai, transportavimo linijos...)
- Funkcinio lygmens (atvaizduojamos funkcinės priklausomybės)

Valdymo dėsniui bei priklausomybėms nustatyti naudojamas funkcinis lygmuo, o nuo fizinio lygmens priklauso dėsnio bei funkcijų parametrizavimas. Todėl fizinio lygmens nenaudosime UML diagramų sudarymui.



3.1.4 pav. P&ID simbolių parametrai

Iš grafinio P&ID žymėjimo pagal 2.2 skyriuje pateikus standartus galima sudaryti 3.1.3 pav. objektus ir suteikti jiems šiuos parametrus:

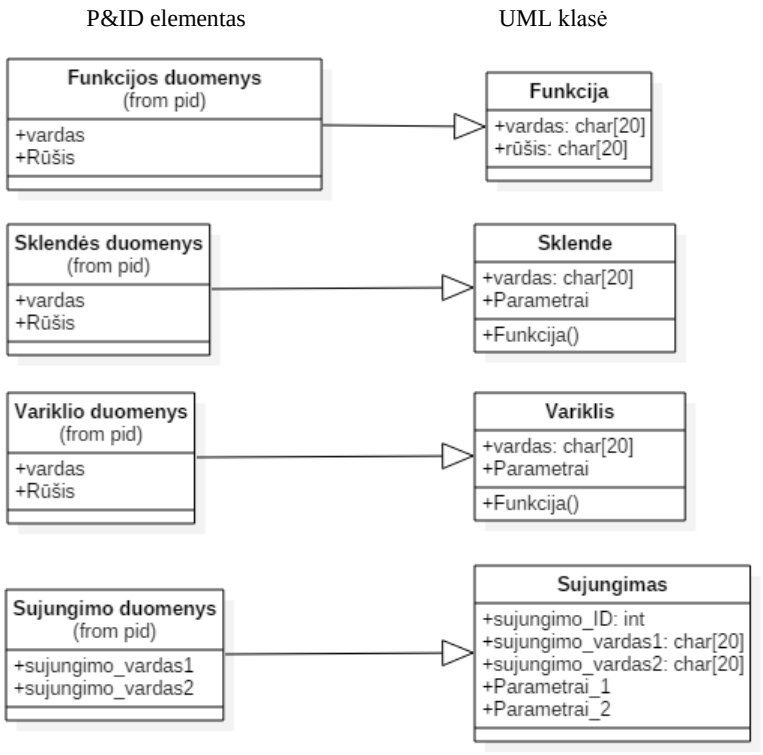
- Vardas – unikalus kiekvieno simbolio raidinis žymėjimas;
- Rūšis – įrenginio ar matavimo prietaiso rūšis, pagal jo raidinį bei grafinį žymėjimą;

Sujungimo elementas yra specifinis neturintis jokio identifikatoriaus ir pagal jo grafinį žymėjimą galima nustatyti apjungiančius elementus. Tuomet jį atitinkančiam UML elementui galima priskirti du parametrus, kurie nurodytų apjungiamų elementų vardus.

3.2 UML diagramų transformacijos žemėlapių ir taisyklių sudarymas

Siekiant konvertuoti P&ID į UML į objektų diagramas, reikia sudaryti transformacijos žemėlapį, kuris apibrėžtų vienareikšmį P&ID objektų ir UML klasių bei objektų atitikmenis. Jie vėliau gali būti naudojami, transformuojant į programinį kodą.

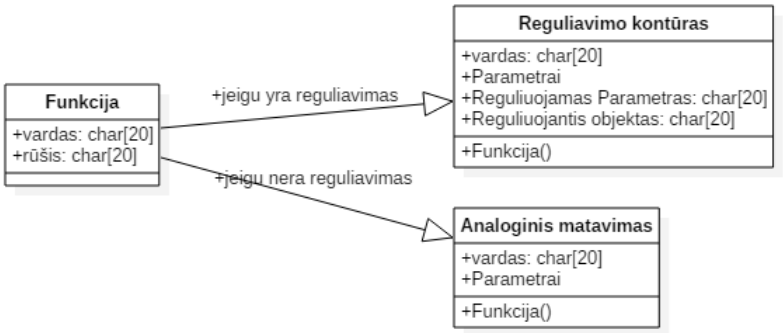
Pirma sukuriamas transformacijos žemėlapis. Jis tipinius schemos elementus keičia UML klasėmis, iš kurių bus kuriami sistemoje esantys objektai.



3.2.1 pav. P&ID objektų ir UML klasių transformacijos žemėlapis

Keičiant elementus, tiesiogiai visi duomenys perduodami UML klasei, sukuriami papildomi klasės parametrai (elemento rūšis yra parametrų rinkinys) ir atliekamos funkcijos. Sujungimo elementams sukuriamas atskiras ID (jis neturi savo atskiro vardo), kad būtų galima visus išskirti ir parametrų rinkiniai, kurie yra susieti su sujungtais elementais.

Funkcijos bei sujungimo elementai nėra galutiniai, todėl reikia atlikti papildomų transformacijų bei ryšių realizavimus. Gaunamas funkcijos elementas yra sudarytas arba iš valdymo dėsnio, arba fizikinio dydžio matavimo. Sujungimo elementas yra tarpinė grandis, kuri perduoda parametrus tarp sujungtų elementų.



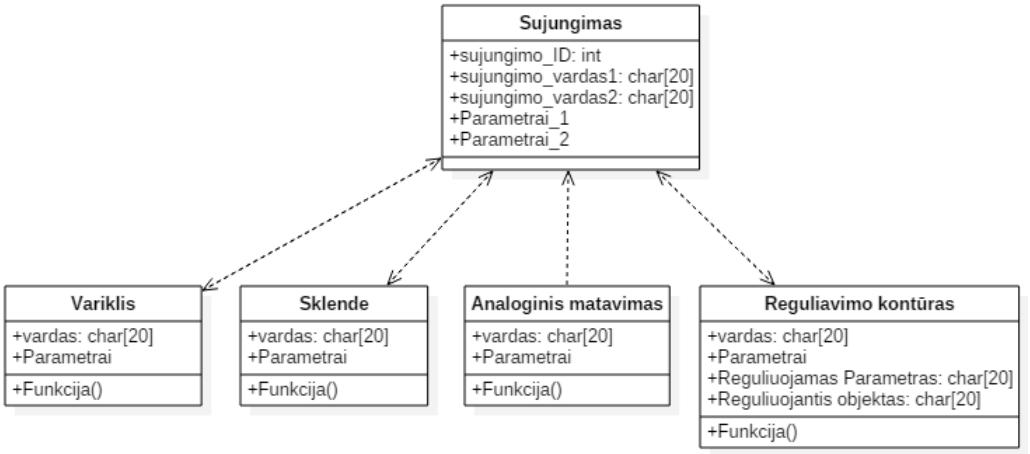
3.2.2 pav. UML funkcijos klasės išskyrimas į dvi klases

Funkcijos elementas keičiamas reguliavimo kontūro arba fizikinio dydžio matavimo elementu priklausomai nuo funkcijos rūšies aprašytos 2.2.1 lentelėje:

- Fizikinio dydžio matavimas – indikacija, įrašymas, perdavimas.
- Reguliavimo kontūras – reguliavimas, valdymas, kontaktų valdymas.

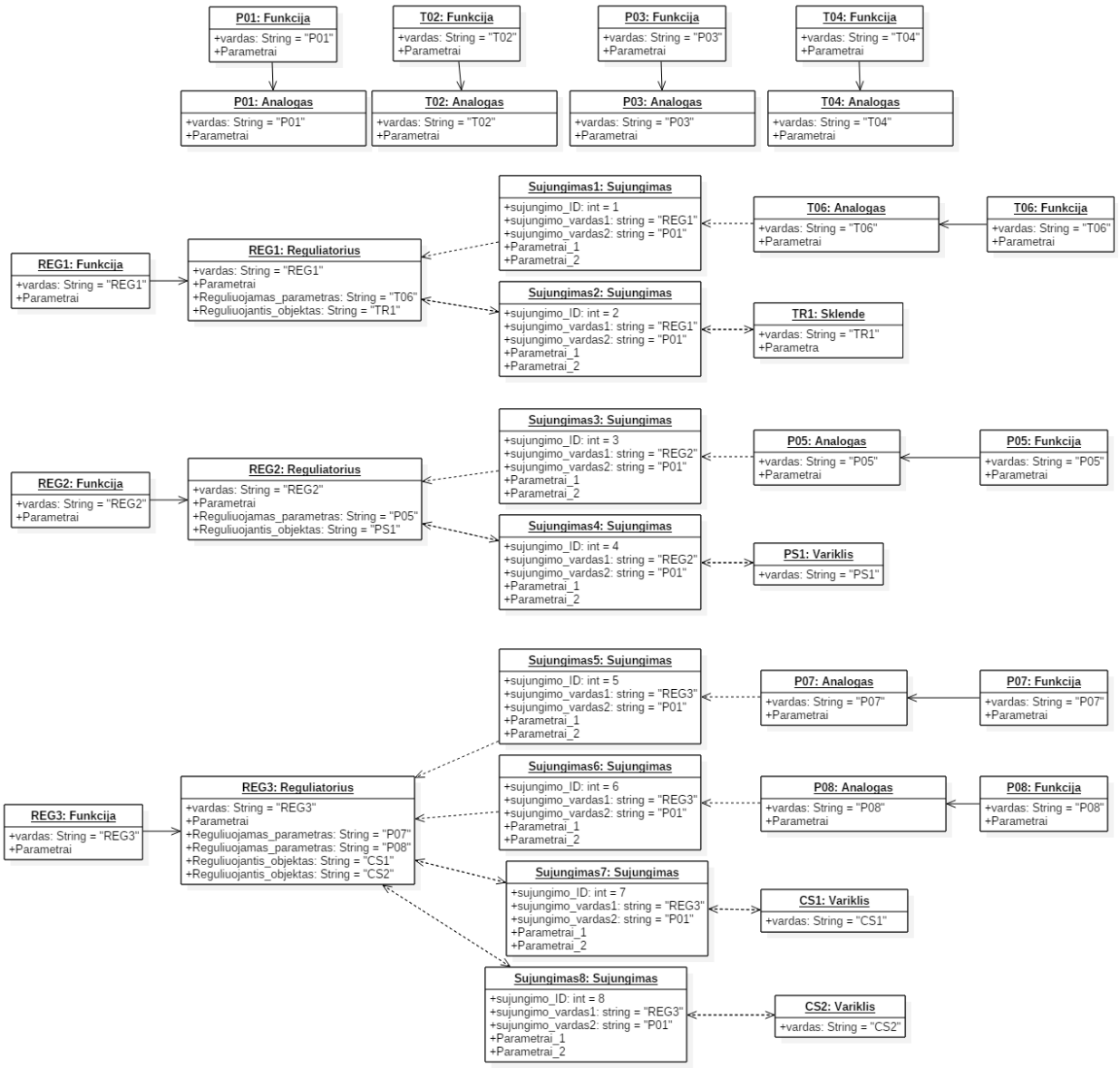
Matavimo ir reguliavimo kontūro klasėms sukuriami papildomi parametrai (elemento rūšis yra parametrų rinkinys) ir atliekamos funkcijos. Reguliavimo kontūrui pridedami parametrai:

- Reguliuojamas parametras – nurodantis kuriam matavimui yra daroma įtaka
- Reguliuojantis objektas – įtaką darantis elementas



3.2.3 pav. UML sujungimo klasės ryšiai

Sujungimo elementas priklausomai nuo sujungiamų objektų duomenis perduoda skirtingomis kryptimis. Perduodami parametrai yra skirtingi kiekvienam objektui, todėl sujungimo elementas tinkantis visiems galimiems sujungimo atvėjams sudaromas iš visų elementų perduodamų parametrų.



3.2.2 pav. Supaprastintos UML objektų diagramos

3.1.2 pav. sistemos UML objektų diagrama, vaizduojanti ryšius tarp elementų, pateikta 3.2.2pav. Objektas keičiamas kitu yra sujungtas linija su rodykle, o darantis poveikį kitam objektui yra sujungtas punktyrinėmis linijomis su rodyklėmis. Rodyklės kryptys nurodo duomenų apsikeitimo arba elementų keitimo puses.

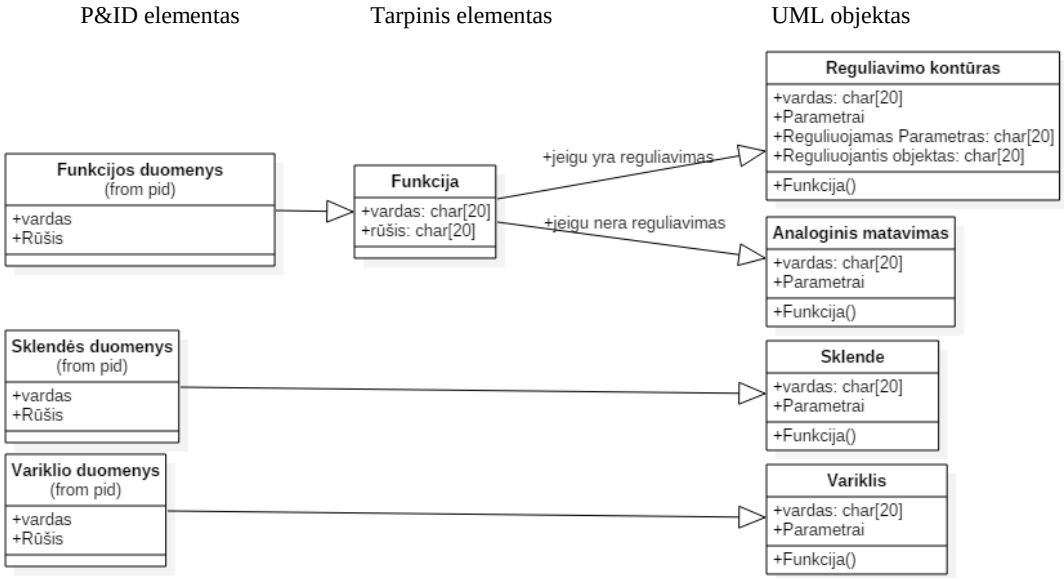
Keičiant P&ID elementus tiesiogiai, sukurtose UML diagramose gaunami papildomi funkcijos objektai. Jie nesuteikia jokios naudingos informacijos, o sujungimo elementai be reikalo apkrauna diagramos vaizdą. Siekiant sumažinti elementų skaičių diagramose, reikia atlikti dalinį duomenų apdorojimą.

3.3 Duomenų apdorojimas, prieš kuriant UML daigramas

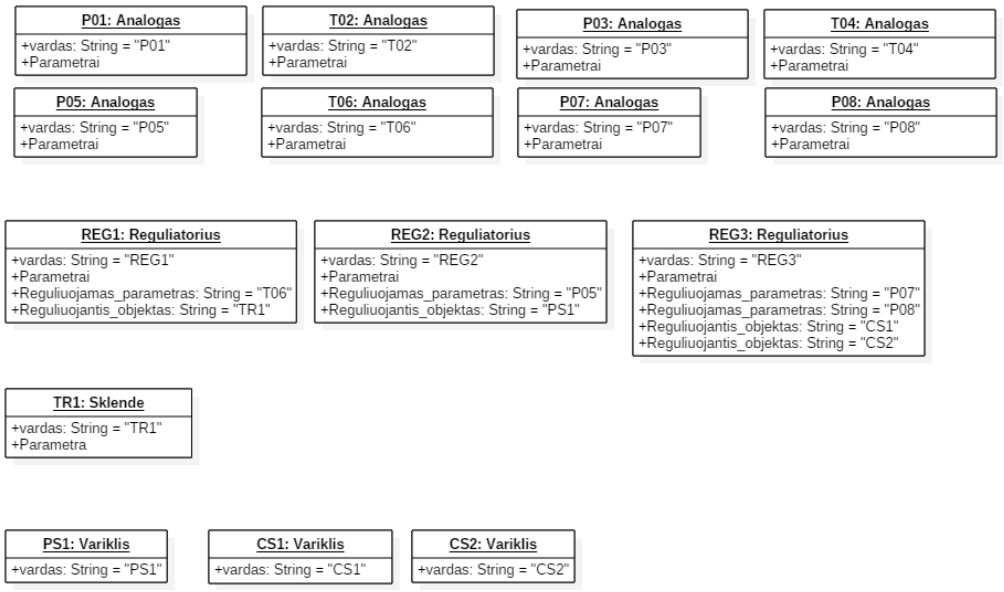
Susidūrus su problemomis keičiant elementus tiesiogiai, atliekamas pradinis duomenų apdorojimas. Ir tik tada sudarinėjami UML objektai. Siekiant sukurti lengviau suprantamą UML diagramą naudojant mažiau elementų, kiekvienas atspindintis aiškią tos sistemos funkciją arba elementą, jos išskiriamos į dvi atskiras diagramas:

- Visų sistemos objektų diagramą
- Ryšių tarp sistemos objektų diagramą

Sudarant visos sistemos objektų diagramą, pirma išskiriamos funkcijos į fizikinių dydžių matavimus matavimus ir reguliavimo kontūrus ir panaikinamas sujungimo elementas.



3.3.1 pav. P&ID objektų ir UML klasių ir objektų transformacijos žemėlapis

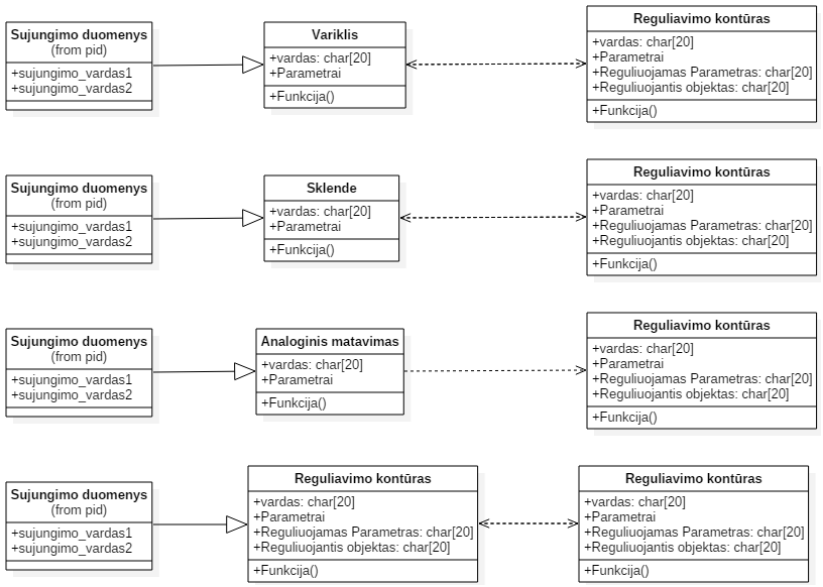


3.3.2 pav. Supaprastinta visų sistemos įrenginių UML diagrama

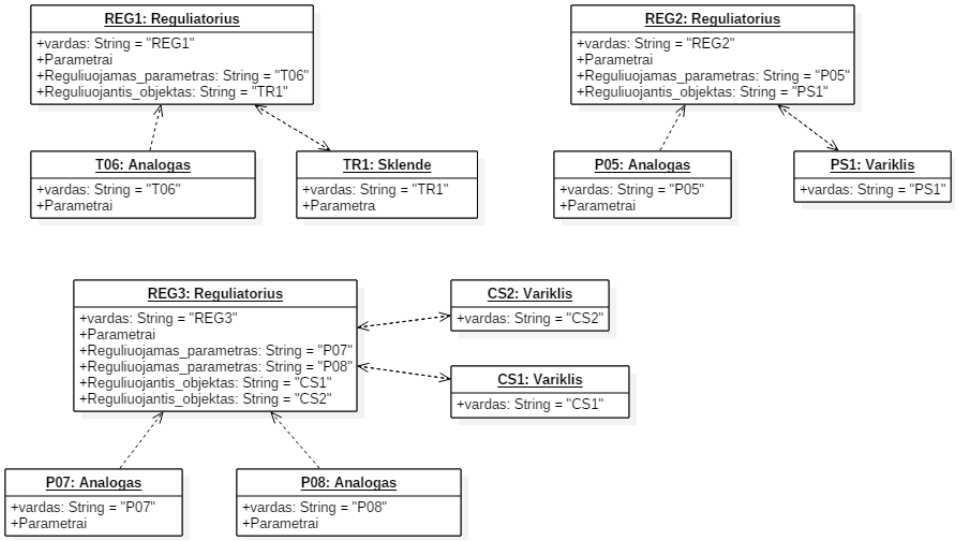
Sukūrus visus sistemos objektus, galima kurti ryšių diagramą. Ši diagrama kuriama antroji, kadangi pirmoje diagramoje elementai yra parametrizuoti atitinkamai pagal P&ID. Taip išvengiami nesutapimai tarp objektų, esančių skirtingose diagramose. Sujungimo elementas turi pakankamą informacijos kiekį sudaryti ryšių diagramai.

Sujungimo elementai gali susieti šių elementų grupes:

- Variklis – Reguliavimo kontūras
- Sklendė – Reguliavimo kontūras
- Reguliavimo kontūras – Reguliavimo kontūras
- Matavimas – Reguliavimo kontūras



3.3.3 pav. P&ID sujungimo ir UML objektų ryšių transformacijos žemėlapis



3.3.4 pav. Supaprastinta objetų ryšių UML diagrama

Sudarius 3.1.2pav. esančios sistemos UML transformacijos žemėlapius dviem metodais, galima sudaryti jų rezultatų palyginimų lentelę.

3.3.5 lent. Metodų palyginimas

Lyginamas parametras	Tiesioginio keitimo	Dalinio apdorojimo
Sistemos objektų perkėlimas	Paprastas	Paprastas
Valdymo kontūrų suradimas	Paprastas	Paprastas
Generuojamos programos sudėtingumas	Sudėtingas	Vidutinis
Valdymo logikos priskyrimas kontūrui	Vidutinis	Paprastas
Sistemos supratimas pagal modelį	Sudėtingas	Vidutinis
Objektų dubliavimas	Yra	Nėra
UML diagramų skaičius	1	2

Abu metodai yra panašūs. Pagrindinis skirtumas yra papildomas duomenų apdorojimas, o ne tiesioginis keitimas. Atlikus dalinį duomenų apdorojimą, vienoje diagramoje išryškinami visi sistemos elementai, o kitoje elementų tarpusavio ryšiai. Kadangi reguliavimo kontūras nėra fizinis elementas, o programinis sprendimas, jį galima pašalinti iš sistemos elementų diagramos. Tada ši diagrama taptų visų sistemos fizinių elementų diagrama.

Toliau darbe bus nagrinėjamas sistemos keitimas, naudojant dalinio apdorojimo metodą. Pagal šį metodą modelio sukūrimas yra lengvesnis ir naudojamų elementų skaičius yra mažesnis.

3.4 Trečio skyriaus išvados

- Norint sudaryti UML diagramas ir elementus, reikia sudaryti tikslų vienareikšmį elementų keitimo žemėlapį, kuris yra sistemos pagrindas. Remiantis šiuo žemėlapiu yra kuriami visi sistemos elementai ir ryšiai tarp jų.
- Atliekant P&ID transformavimą į UML, reikalingas papildomas duomenų apdorojimas, kuris supaprastina gaunamą UML diagramą, sukuriant tik galutinius sistemos elementus.

4 Automatinis P&ID keitimas į UML ir programinio kodo kūrimas

Keičiant iš P&ID į UML ir programinį kodą, visas atliekamas pracedūras galima išskirti į 4 pagrindines dalis:

- Vaizdo atpažinimo
    - Teksto atpažinimo
    - Elementų atpažinimo
    - Punktyrinių linijų radimo
- Dalinio apdorojimo
    - Funkcijų elementų keitimo į kitus elementus
    - Elementų grupavimo
    - Sujungimo elemento keitimo
- UML diagramų sudarymo iš turimos informacijos
- Programinio kodo sudarymo

Šio darbo metu P&ID diagramos yra sudarytos „AutoCAD“ programiniu paketu, naudojant šabloninius vienodo dydžio elementus. Brėžiniai yra eksportuojami į „.PNG“ failus 300dpi. Diagramų apdorojimas atliekamas Matlab programiniu paketu, kurio programos kodas pateiktas 7.1 priedo skyriuje.

Teksto atpažinimui naudojamas atviro kodo OCR programa „Tesseract“, kuri atpažįsta atskiras raides ir nurodo jos koordinates. Siekiant gauti mažiau klaidingų duomenų, atliekamas algoritmo apmokymas skirtingomis kalbomis ir skirtingais šriftais. Vėliau visi šalia esantys simboliai yra grupuojami, taip sudaramos žodžių grupės bei pašalinamos jos iš atpažįstamo vaizdo.

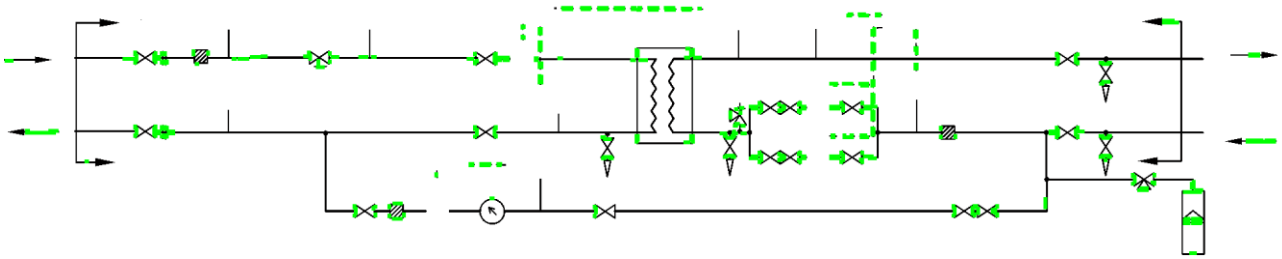
Elementams, esantiems 2.2.2 lentelėje, rasti naudojamas šablonų metodas. Šis metodas yra pakankamas kai:

- Elementai yra horizontalūs arba vertikalūs, šablonus užtenka pakreipti pagal keturias galimas elementų padėtis.
- Standartizuoti elementų dydžiai.

Suradus elementą jis yra panaikinamas iš nuotraukos. Visiems elementams priskiriami žodžių junginiai, kurie yra arčiausiai jų.

Punkyrinėms linijoms surasti naudojama modifikuota Hough transformacija. Ieškomos tik horizontalios ir vertikalios linijos, kurios yra atrenkamos lyginant maksimalų ir minimalų galimą ilgį. Grupuojamos su šalia esančiomis linijomis, kurios yra nutolę ne didesniu negu maksimalus

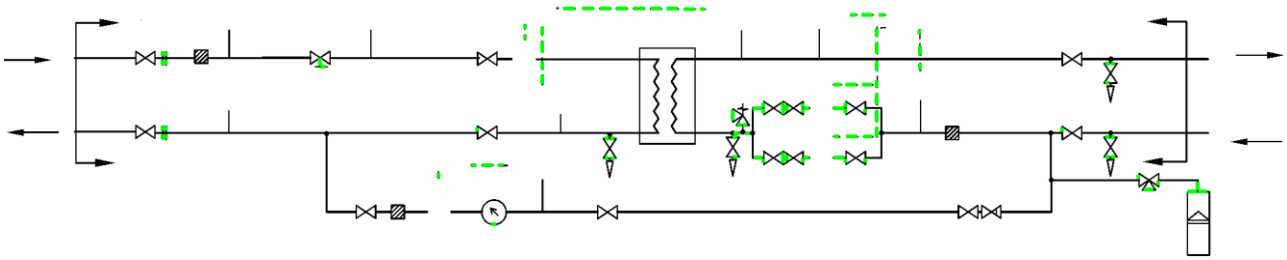
galimas tarpo ilgis (maksimalus punktyro ilgis – 40, minimalus ilgis – 0,3 maksimalaus, tarpo ilgis – 0,8 maksimalaus).



4.1 pav. Hough punktyrų radimas, naudojant „Prewitt“ kraštinių radimo metodą

Kadangi linijos esančios diagramoje yra platesnės negu vieno pikselio, todėl atliekant kraštų ieškojimo algoritmus atsiranda papildomos linijos, kurios sudarytų dubliuotus junginius. Siekiant to išvengti naudojama morfologinė ploninimo erozija.

Diagramoje kai kurios linijos dėl braižymo netikslumų nėra visiškai horizontalios ar vertikalios, todėl atpažįstamos kaip punktyrinių linijų nuotrupos. Siekiant sumažinti šį požymį, papildomai ieškomos linijos, kurios yra ilgesnės negu galimos punktyrinės linijos ir yra panaikinamos.



4.2 pav. Hough punktyrų radimas, naudojant morfologinį ploninimo metodą

Klaidingus atpažinimus lemia linijų persidengimai ir diagramoje likę valdymui neaktualūs elementai. Juos panaikinus, punktyrinės linijos būtų aptinkamos didesniu tikslumu.

Funkcijos elementai keičiami į matavimo arba reguliavimo elementą atitinkamai pagal esamos funkcijos rūšį (pagal 3.2 skyriuje aprašytą metodą).

4.2 pav. Visos surastos punktyrinės linijos yra traktuojamos kaip sujungimo elementai, bet ne visos sujungia elementus. Elementai, kurie sujungia mažiau nei du vykdiklius arba funkcijas, yra pašalinami dėl klaidingų atpažinimų. Kiekvienai likusiai punktyrinei linijai suteikiamas unikalus numeris, kuris yra priskiriamas sujungiamiems elementams.

Po kiekvienos operacijos yra atvaizduojamas paveikslėlis su aptiktais elementais, juos paženklinant apskritimais, o tolimesniai operacijai aptikti elementai yra pašalinami. Atlikus

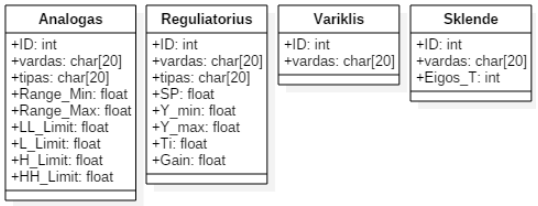
automatizuotą diagramos atpažinimą vartotojui reikia patikrinti ar visa reikalinga informacija yra atpažinta.

Visi duomenys apie elementus yra perkeliami į „Excel“ failą „eksportas.xls“, pagal importavimo struktūrą tinkančią „Visual Paradigm“ programiniam paketui. Failo pavyzdys pateiktas 7.2 priedo skyriuje.

4.3 lent. Veiksmų atlikimo laikų lentelė

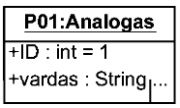
Aliekamas veiksmas	Užimamas laikas
Teksto atpažinimas	6,021 s
Elementų atpažinimas	47,702 s
Punktyrų atpažinimas	13,101 s
Dalinis apdorojimas	14,153 s
Excel failo sukūrimas	10,223s
Viso	91,200 s

Testavimas atliktas naudojant personalini kompiuterį urintį: 4 branduolių 2,8 GHz procesorių, 8GB operatyviosios atminties, GeForce GT730 2GB vaizdo plokštę.



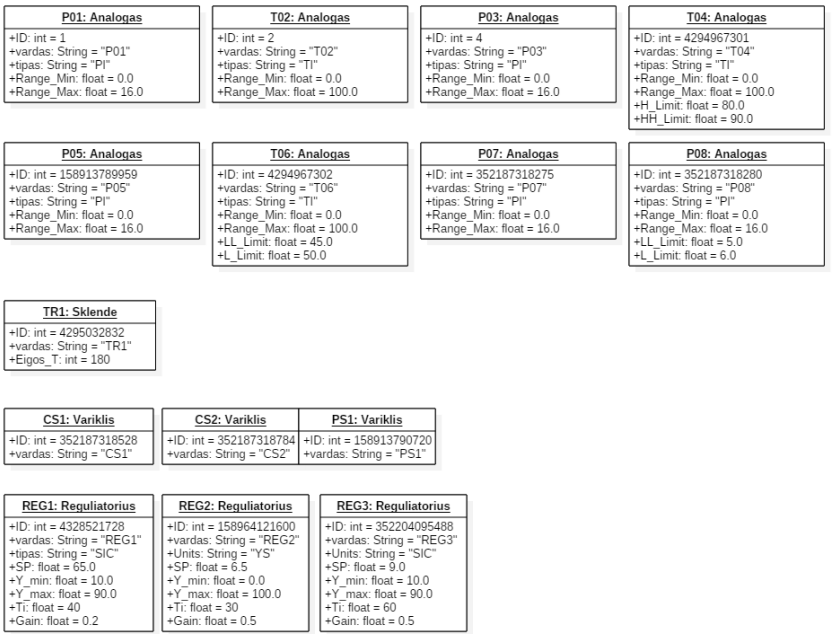
4.4 pav. UML objektai su visais galimais parametrais

Kiekvienas elementas pagal savo rūšį turi skirtingų parametrų komplektą pateiktą 4.4 pav., kuris yra priskiriamas sukuriant kiekvieną elementą.

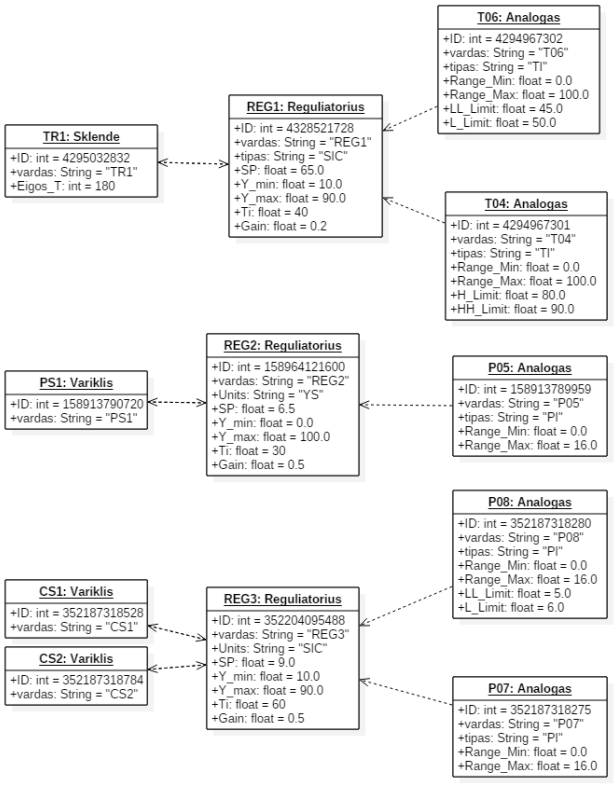


4.5 pav. Importuota UML diagrama

Importavus failą į „Visual Paradigm“ paketą sukuriama nauja objektų diagrama su visais elementais ir ryšiais. Kadangi importuojant nėra nurodomi elementų dydžiai bei kordinatės, todėl reikia juos išdėstyti bei pakeisti jų dydžius rankiniu būdu.



4.6 pav. Objektų UML diagrama (modifikuota)



4.7 pav. Ryšių tarp objektų UML diagrama

Atsižvelgiant į sistemos reikalavimus ir įrangos specifikaciją rankiniu būdu patikrinami visi elementai ir pakeičiamos parametrų vertės ir panaikinami tie parametrai, kurie nėra naudojami (kaip perspėjančios ar reguliavimo ribos LL_Limit, Y_min...).

Programinio kodo generavimas iš UML diagramų (7.3 priedo skyriuje pavyzdys) yra skirtas sukurti visiems sistemos elementams ir struktūrai. Ryšių realizavimas ir funkcijų vykdymo tvarka yra atliekama vartotojo.

4.1 Ketvirto skyriaus išvados

- Sukurtas P&ID diagramos analizės algoritmas, naudojantis UML transformacijos žemėlapiu ir taisyklėmis, keičiantis į UML diagramas.
- Sudarytį šablonai UML diagramų keitimui į programinį kodą
- Kuriant UML diagramas, visi elementai yra sukuriami su visais jiems keičiamais parametrais, todėl reikia įsiterpti vartotojui ir juos surašyti pagal sistemos specifikaciją bei panaikinti keitimo neatitikimus. Siekiant sumažinti vartotojo įsiterpimą, reikalinga papildoma apdorojimo operacija, apdorojanti elementų specifikaciją.

5 Produkto testavimas

Patikrint šio metodo praktinį pritaikomumą buvo atlikti bandymai įmonėje AB „Axis Industries“. Siekiama sukurti produktą, kuris galėtų atlikti valdymo, duomenų perdavimo ir kaupimo funkciją. Šiam produktui buvo suformuoti šie reikalavimai:

- Atlikti realaus laiko valdymo užduotį (nuokrypis ne daugiau 1ms)
- Atlikti temperatūrų matavimą naudojant 1-wire sąsają
- Turėti bent 32Mb laisvos atminties archyvų kaupimui
- Perduoti duomenis GSM modemu į serverį
- Galimybė nusiskaityti duomenis naudojant „Modbus TCP/IP“ protokolu
- Nuskaityti duomenis „Modbus RS485“ protokolu iš firmoje gaminamų modulių
- Galimybė nuskaityti apskaitos prietaisus M-Bus ir RS-232 sąsaja

Prototipo testavimas atliekamas naudojant „Raspberry pi 2“ maketą su papildomu moduliu, kuris išplečia maketo galimybes. Kad tenkintų keliamus reikalavimus, naudojant „Raspbian Jessie“ *Linux* distribuciją, naudojant 4.1.20 kernel versiją.

Papildomo modulio schema pateikta 7.4 priede.

GSM modemui Telit UL865 komunikuoti naudojamas vidinis UART, o RS-232, RS-485 ir M-Bus komunikacijoms naudojamos mikroschemos SC16IS740, naudojant SPI sąsają. Kadangi maketas neturi vidinio laikrodžio, naudojamas išorinis PCF8563 realaus laiko laikrodis, pajungtas prie I2C.

Vykdomos užduoties nuokrypiui skaičiuoti naudojami keli skirtingi apkrovimai, bei kernel versijos, atliekant milijoną ciklų, kurie yra kviečiami kas 10ms.

5.1 lent. Vykdomos užduoties nuokrypių mikrosekundėmis lentelė

Kernel versija	0% apkrovimas			100% apkrovimas		
	Min	Vid	Max	Min	Vid	Max
4.1.20	4	44	530	35	81	2561
4.1.20 Preemt_RT	4	45	243	27	61	321
3.18.20 Xenomai	3	25	61	20	52	135

Naudojant *Preempt_RT* ir *Xenomai* kernel versijas, yra tenkinami užduoties vykdymo apribojimai. Todėl užduotims, kurioms yra svarbūs laiko apribojimai, reikia naudoti modifikuotą *kernel* versiją.

5.1 pav. Web puslapis

Įrenginio diagnostikai, informacijai ir duomenų atvaizdavimui yra sukurtas internetinis puslapis, kuris atvaizduoja duomenis iš programos sukurtų duomenų failų.

Sudarius 3.1.2 paveiksle pavaizduotos sistemos UML diagramas, jos yra keičiamos programiniu kodu, tinkančiu šiam produktui. Šio programinio kodo pilnam sistemos valdymui nepakanka, todėl reikia įsiterpti inžinieriui sudarant programos vykdymo sekas, intervalus, naudojamas funkcijas.

Patikrinti automatizuoto keitimo ir kitų naudojamų praktikų programos sudarymo laikus atlikti sistemos programavimo darbai naudojant skirtingus metodus.

5.1 lent. Programos kūrimo laikų lentelė

Kūrimo metodas	Užimamas laikas	Realityvi programinių klaidų sudarymo tikimybė
Nenaudojant objektinio programavimo sprendimų	7 d.	100%
Naudojant objektinio programavimo sprendimus	5 d.	~30%
Naudojant UML diagramų programinio kodo generavimą	3 d.	~15%

Atliekant sistemų keitimą į UML diagramas ir keičiant jas į programinį kodą yra sumažinama klaidų tikimybė ir laiko sąnaudos. Maksimaliai sumažinti klaidų skaičių galima tik pilnai automatizavus visą procesą ir sudarius programinį paketą, kuriuo būtų galima konfiguruoti sistemos parametrus.

5.1 Penkto skyriaus išvados

- Sukurtas ir užprogramuotas įrenginys, naudojantis darbe aprašytu metodu.
- Palyginti programos kūrimo laikai naudojantis skirtingas praktikoje naudojamais metodais.
- Pilnam keitimui į programinį kodą UML diagramų nepakanka, reikia papildų įrangos ir sistemos valdymo specifikacijos perdavimo metodų.

6 Išvados

- Sudarytas metodas P&ID diagramų transformacijos į UML aprašus.
- Sudarytas algoritmas vienareikšmiškai keičiantis ir klasifikuojantis P&ID elementus.
- Pilnam keitimui į programinį kodą UML diagramų nepakanka, reikia papildų įrangos ir sistemos valdymo specifikacijos perdavimo metodų.

7 Literatūra

1. ISO/IEC 19505-1:2012(E). „Information technology - Object Management Group Unified Modeling Language (OMG UML)“, Infrastructure, 2012 m., 230 p.

2. „Model driven development of industrial process control applications“ , David Hastbacka, Timo Vepsalainen, Seppo Kuikka, 2011 m., 14p

3. http://ir18.ugent.be/cms/files/webform/CODES1-MESMFinal.pdf 2014.03.10

4. http://elektra-ku.lt/attachments/File/Studentams/Automatizavimo_sistem___schemotechnika_2009.pdf 2014.03.11

5. „Batch Processing Systems Engineering: Fundamentals and Applications for Chemical Engineering“ G. V. Reklaitis Springer, 1996 m., 868 psl.

6. http://www.sesam-world.dk/_isa/ISA%2095%20Part%201%20CDV03/ISA950001_Update_CDV03.pdf 2014.03.20

8. Elecia White, “Making Embedded Systems Design Patterns for Great Software”, 2011 m., 330p

9. Rong Chen, Marco Sgroi, Grant Martin, Luciano Lavagno, Alberto Sangiovanni- Vincentelli, Jan Rabaey, „Embedded System Design Using UML and Platforms“, 11p

10. http://www.tutorialspoint.com/uml/uml_class_diagram.htm 2014.10.01

11. http://pic.dhe.ibm.com/infocenter/rsarthlp/v8/index.jsp?topic=%2Fcom.ibm.xtools.modeler.doc%2Ftopics%2Fcclassd.html 2014.10.01

12. http://dspace.cc.tut.fi/dpub/bitstream/handle/123456789/6086/alho.pdf?sequence=3&isAllowed=y 2014.10.01

13. „Technologinių procesų automatizavimas ir valdymas“, Vytautas Aleksa, Vytautas Galvanauskas, 2011m., 284p.

14. http://www.uml-diagrams.org/profile-diagrams.html 2014.11.05

15. http://www.sparxsystems.com.au/downloads/whitepapers/UML_Tutorial_Part_2_Introduction.pdf 2014.11.05

16. http://www.tutorialspoint.com/uml/uml_basic_notations.htm 2015.03.05

17. http://www.uml-diagrams.org/use-case-diagrams.html 2015.04.02

18. „IEC 62424“, International standard, 2008 m., 140p.

19. A. Kleppe, J. Warmer , W. Bast, „MDA Explained – The Model-Driven Architecture: Practice and Promise“, 2003m., 192 p.

20. http://ac.els-cdn.com/S1571066106001435/1-s2.0-S1571066106001435-main.pdf?_tid=6b1528ae-821d-11e4-80b9-00000aacb35f&acdnat=1418402539_cee8b2d67fc5170fce77810cabe971c2 2015.04.05

21. „Process modeling using UML“ G. ENGELS, A. F¨ORSTER, R. HECKEL, S. TH¨ONE, 2008 m., 31p

22. http://publib.boulder.ibm.com/infocenter/rsdvhelp/v6r0m1/index.jsp?topic=%2Fcom.ibm.xtools.transformations.doc%2Ftopics%2Fccpptransf.html 2016.02.15

23. http://www.dthomas.co.uk/dtalm/products/technologies/why-use-uml.htm 2016.03.02

24. „Batch Control Part 1: Models and Terminology“, American National Standard, 98p

25. „OMG Unified Modeling Language Infrastructure“ , OMG, 2011 m., 230p.

8 Priedai

8.1 Matlab programos kodas

8.1.1 Pagrindinė programa

```
clear all;
close all;

addpath('C:\Users\Aldis\Desktop\magistrinis\magistrinis\matlabas\foto')
addpath('C:\Users\Aldis\Desktop\magistrinis\magistrinis\matlabas\funkcijos')
addpath('C:\Users\Aldis\Desktop\magistrinis\magistrinis\matlabas\templates')

warning('off', 'Images:initSize:adjustingMag');

nuotrauka='schema_pilna20160412.png';

punkt_linija=30;
punk_kof=0.3;
peak_sk=150;
riba=0.1;
    raides= 1;
    objektai=1;
    tekstas=1;

    naikinti_linijas=1;
    naikinti_tekstal=1;
    naikinti_teksta=1;
    linijos=1;
    grupavimas=1;
    failas=1;

visas = imread(nuotrauka);
Pilkas_pav = rgb2gray(visas);
dyd=size(Pilkas_pav);


if objektai==1
 radimo_riba=0.65;
 template=imread('sklende_be.png');
 Pilkas_T = rgb2gray(template);
 [Pilkas_pav, sk_kiek,
sk_kord_x1,sk_kord_x2,sk_kord_y1,sk_kord_y2,sk_kof]=func_template(Pilkas_pav,Pilkas_T,'sklendes',radimo_riba,4,0);

 radimo_riba=0.65;
 template=imread('siurblys_be.png');
 Pilkas_T = rgb2gray(template);
 [Pilkas_pav, var_kiek,
var_kord_x1,var_kord_x2,var_kord_y1,var_kord_y2,var_kof]=func_template(Pilkas_pav,Pilkas_T,'variklis',radimo_riba,
4,1);


 imwrite(Pilkas_pav,'tesses.png')

end

if raides==1
    system('start tesseract.exe tesse.png text makebox');
    pause(5);
end

if tekstas==1
    [teksto_x1,teksto_x2,teksto_y1,teksto_y2,raides,teksto_grupe] = teksto_tvarkymas( Pilkas_pav,dyd,4);
    teksto_sk=length(teksto_x1);
    teksto_unik=unique(teksto_grupe);
    ilgis=length(teksto_unik);
    for i = 1:ilgis
        teksto_kiekis(i) = sum(teksto_grupe==teksto_unik(i));
    end
    figure, imshow(Pilkas_pav), hold on;
    title(['tik tie kur po 1 raudoni kiti melyni']);
    for i=1:ilgis
        if teksto_kiekis(i)==1
            kelintas=teksto_unik(i);
       %    figure, imshow(Pilkas_pav), hold on;
       %    title(['liniju grupe ',num2str(i),' grupes numeris ',num2str(kelintas)]);
            for j=1:teksto_sk
                if teksto_grupe(j) == kelintas
                   for j=0:(teksto_x2(i)-teksto_x1(i))
                       x=teksto_x1(i)+j;
                        if x> 0
                            for z=0:(teksto_y1(i)-teksto_y2(i))
                               y=teksto_y2(i)+z;
                                  if y>0
```

```
                                    if naikinti_teksta1 > 0
                                        Pilkas_pav(y,x)=255;
                                    end
                                end
                            end
                        end
                    end
                    viscircles([teksto_x1(j) teksto_y1(j) ],10);
                end
            end
        else
            kelintas=teksto_unik(i);
            for i=1:teksto_sk
                if teksto_grupe(i) == kelintas
                   for j=0:(teksto_x2(i)-teksto_x1(i))
                       x=teksto_x1(i)+j;
                        if x> 0
                            for z=0:(teksto_y1(i)-teksto_y2(i))
                               y=teksto_y2(i)+z;
                                  if y>0
                                    if naikinti_teksta > 0
                                        Pilkas_pav(y,x)=255;
                                    end
                                end
                            end
                        end
                    end
                   viscircles([teksto_x1(i) teksto_y1(i) ],10,'EdgeColor','b');
                end
            end
        end
    end
end

if objektai==1
     radimo_riba=0.65;
     template=imread('funkcija_be.png');
     Pilkas_T = rgb2gray(template);
     [Pilkas_pav, funkc_kiek,
funkc_kord_x1,funkc_kord_x2,funkc_kord_y1,funkc_kord_y2,funkc_kof]=func_template(Pilkas_pav,Pilkas_T,'funkcija',ra
dimo_riba,2,0);
     imwrite(Pilkas_pav,'tesses.png')
end

    nuotrauka='Pilkas.png';
    Pilkas_pav = imread('tesses.png');

    level = graythresh(Pilkas_pav);
    BW = im2bw(Pilkas_pav, level);
    invBW=~BW;
    BW=bwmorph(invBW,'thin',inf);

%    BW = edge(BW,'Prewitt');

     figure; imshow(BW); title('liniju naikinimas'), hold on;
if linijos==1
%    Pilkas_pav = imread('Pilkas.png'); Pilkas_pav = rgb2gray(Pilkas_pav); imwrite(Pilkas_pav,'Pilkas.png');

    if naikinti_linijas==1
        BW=liniju_naikinimas(Pilkas_pav,BW,riba,peak_sk,punkt_linija);
    end
    [linijos_x1,linijos_x2,linijos_y1,linijos_y2,linijos_dif_x,linijos_dif_y,linijos_grupe] = punktyru_radimas(
Pilkas_pav,BW,riba,peak_sk,punkt_linija,punk_kof);
     if 1<0
        ilgis=length(linijos_x1);
        for i = 1:ilgis
            vienodi=find(linijos_x1(i)==linijos_x1);
            vienodi_sk=length(vienodi);
            vienodi_kelintas=find(vienodi==i);
            if vienodi_sk>1 && vienodi_kelintas<vienodi_sk
               for j = vienodi_kelintas+1:vienodi_sk
                    if linijos_x2(vienodi(vienodi_kelintas))==linijos_x2(vienodi(j)) &&
linijos_y1(vienodi(vienodi_kelintas))==linijos_y1(vienodi(j)) &&
linijos_y2(vienodi(vienodi_kelintas))==linijos_y2(vienodi(j))
                        naikinti_lin(vienodi(j))=vienodi(j);
                        vienodi;
                    end
               end
            end
        end

        naikinti_lin=unique(naikinti_lin);
        if naikinti_lin(1)==0
            naikinti_lin(1) = [] ;
        end
        linijos_x1(naikinti_lin) = [] ;
        linijos_x2(naikinti_lin) = [] ;
        linijos_y1(naikinti_lin) = [] ;
        linijos_y2(naikinti_lin) = [] ;
```

```
        linijos_dif_x(naikinti_lin) = [] ;
        linijos_dif_y(naikinti_lin) = [] ;
        linijos_grupe(naikinti_lin) = [] ;
    end

    linijos_sk=length(linijos_x1);

    linijos_unik=unique(linijos_grupe);
    ilgis=length(linijos_unik);
    kiekis_ne1=0;

    for i = 1:ilgis
        linijos_kiekis(i) = sum(linijos_grupe==linijos_unik(i));
        if linijos_kiekis(i)>1
           kiekis_ne1=kiekis_ne1+1;
        end
    end

    figure, imshow(Pilkas_pav), hold on;
    for i=1:ilgis
        if linijos_kiekis(i)>1
            kelintas=linijos_unik(i);
%            figure, imshow(Pilkas_pav), hold on;
%            title(['liniju grupe ',num2str(i),' grupes numeris ',num2str(kelintas)]);
%            title(['panaikinus kurie po 1']);
            for j=1:linijos_sk
                if linijos_grupe(j) == kelintas
%                    plot([linijos_x1(j), linijos_x2(j)],[linijos_y1(j),
linijos_y2(j)],'LineWidth',2,'Color','green');
                end
            end
        end
    end
end

sujungimo_atstumas=30;

if grupavimas>0
    sujungiami=0;
    zz=1;
    for z=1:var_kiek
        for i=1:ilgis
                if linijos_kiekis(i)>0
                    kelintas=linijos_unik(i);

                        x1=var_kord_x1(z);
                        x2=var_kord_x2(z);
                        y1=var_kord_y1(z);
                        y2=var_kord_y2(z);
                        for j=1:linijos_sk
                            if linijos_grupe(j) == kelintas
        %                        atstumas(x0,x1,x2,y0,y1,y2)
                                x00=linijos_x1(j);
                                x01=linijos_x2(j);
                                y00=linijos_y1(j);
                                y01=linijos_y2(j);

                                a1=atstumas(x00,x1,x2,y00,y1,y2);
                                a2=atstumas(x01,x1,x2,y01,y1,y2);
                                if a1<sujungimo_atstumas || a2<sujungimo_atstumas
                                    sujungiami(zz)=kelintas;
                                    zz=zz+1;
                                    break;
                                end
                            end
                        end
                end
        end
        var_grupe(z)=sujungiami(1);
        if length(sujungiami)>1
            for za=2:length(sujungiami)
                for zb=1:linijos_sk
                    if sujungiami(za)==linijos_grupe(zb)
                        linijos_grupe(zb)=sujungiami(1);
                    end
                end
                for zb=1:length(var_grupe)
                    if sujungiami(za)==var_grupe(zb)
                        var_grupe(zb)=sujungiami(1);
                    end
                end
            end
        end
        zz=1;
    %    sujungiami
        clearvars sujungiami;
        sujungiami=0;
    end
    bbb=-1;
```

```
sujungiami=0;
zz=1;
for z=1:funkc_kiek
    for i=1:ilgis
            if linijos_kiekis(i)>0
                kelintas=linijos_unik(i);

                    x1=funkc_kord_x1(z);
                    x2=funkc_kord_x2(z);
                    y1=funkc_kord_y1(z);
                    y2=funkc_kord_y2(z);
                    for j=1:linijos_sk
                        if linijos_grupe(j) == kelintas
    %                        atstumas(x0,x1,x2,y0,y1,y2)
                            x00=linijos_x1(j);
                            x01=linijos_x2(j);
                            y00=linijos_y1(j);
                            y01=linijos_y2(j);

                            a1=atstumas(x00,x1,x2,y00,y1,y2);
                            a2=atstumas(x01,x1,x2,y01,y1,y2);
                            if a1<sujungimo_atstumas || a2<sujungimo_atstumas
                                sujungiami(zz)=kelintas;
                                zz=zz+1;
                                break;
                            end
                        end
                    end
            end
    end

    funkc_grupe(z)=sujungiami(1);
    if length(sujungiami)>1
        for za=2:length(sujungiami)
            for zb=1:linijos_sk
                if sujungiami(za)==linijos_grupe(zb)
                    linijos_grupe(zb)=sujungiami(1);
                end
            end
            for zb=1:var_kiek
                if sujungiami(za)==var_grupe(zb)
                    var_grupe(zb)=sujungiami(1);
                end
            end
           for zb=1:length(funkc_grupe)
                if sujungiami(za)==funkc_grupe(zb)
                    funkc_grupe(zb)=sujungiami(1);
                end
            end
        end
    end
    zz=1;
%    sujungiami
    clearvars sujungiami;
    sujungiami=0;
end

sujungiami=0;
zz=1;
for z=1:sk_kiek
    for i=1:ilgis
            if linijos_kiekis(i)>0
                kelintas=linijos_unik(i);

                    x1=sk_kord_x1(z);
                    x2=sk_kord_x2(z);
                    y1=sk_kord_y1(z);
                    y2=sk_kord_y2(z);
                    for j=1:linijos_sk
                        if linijos_grupe(j) == kelintas
    %                        atstumas(x0,x1,x2,y0,y1,y2)
                            x00=linijos_x1(j);
                            x01=linijos_x2(j);
                            y00=linijos_y1(j);
                            y01=linijos_y2(j);

                            a1=atstumas(x00,x1,x2,y00,y1,y2);
                            a2=atstumas(x01,x1,x2,y01,y1,y2);
                            if a1<sujungimo_atstumas || a2<sujungimo_atstumas
                                sujungiami(zz)=kelintas;
                                zz=zz+1;
                                break;
                            end
                        end
                    end
            end
    end
    sk_grupe(z)=sujungiami(1);
    if length(sujungiami)>1
        for za=2:length(sujungiami)
```

```
                for zb=1:linijos_sk
                    if sujungiami(za)==linijos_grupe(zb)
                        linijos_grupe(zb)=sujungiami(1);
                    end
                end
                for zb=1:var_kiek
                    if sujungiami(za)==var_grupe(zb)
                        var_grupe(zb)=sujungiami(1);
                    end
                end
                for zb=1:funkc_kiek
                    if sujungiami(za)==funkc_grupe(zb)
                        funkc_grupe(zb)=sujungiami(1);
                    end
                end
                for zb=1:length(sk_grupe)
                    if sujungiami(za)==sk_grupe(zb)
                        sk_grupe(zb)=sujungiami(1);
                    end
                end
            end
        end
        zz=1;
    %    sujungiami
        clearvars sujungiami;
        sujungiami=0;
    end

if failas>0
    ai_skaicius=1;
    reg_skaicius=1;
    for i=1:funkc_kiek
        tikrinam=funkc_tipas(i,length(funkc_tipas(1,:)));
       if tikrinam == 'C' || tikrinam == 'S'
            reg_vardas(reg_skaicius,:)=funkc_vardas(i,:);
            reg_tipas(reg_skaicius,:)=funkc_tipas(i,:);
            reg_grupe(reg_skaicius)=funkc_grupe(i);
            reg_kiek=reg_skaicius;
            reg_skaicius=reg_skaicius+1;
       else
            ai_vardas(ai_skaicius,:)=funkc_vardas(i,:);
            ai_tipas(ai_skaicius,:)=funkc_tipas(i,:);
            ai_grupe(ai_skaicius)=funkc_grupe(i);
            ai_kiek=ai_skaicius;
            ai_skaicius=ai_skaicius+1;
       end
    end

    ai_id= bitsll(ai_grupe, 32);
    var_id= bitsll(var_grupe, 32);
    sk_id= bitsll(sk_grupe, 32);
    reg_id= bitsll(reg_grupe, 32);

    for i=1:ai_kiek
        ai_id(i)=ai_id(i)+bitsll(i, 0);
    end
    for i=1:var_kiek
        var_id(i)=var_id(i)+bitsll(i, 8);
    end
    for i=1:sk_kiek
        sk_id(i)=sk_id(i)+bitsll(i, 16);
    end
    for i=1:reg_kiek
        reg_id(i)=reg_id(i)+bitsll(i, 24);
    end

    filename='eksportas.xls';
    sheet='objektai';
    eilute=1;
    ID=1;

    xlRange=strcat('A',num2str(eilute));
    A = {'Diagram','ID', 'Name', 'Type'};
    xlswrite(filename,A,sheet,xlRange);
    clearvars A;
    eilute=eilute+1;

    xlRange=strcat('A',num2str(eilute));
    A = {'',ID,'Objektu diagrama', 'ERDiagram'};
    xlswrite(filename,A,sheet,xlRange);
    clearvars A;
    eilute=eilute+2;
    ID=ID+1;

   for i=1:ai_kiek
        A = {'Object','ID', 'Model ID', 'Name','Class'};
        xlRange=strcat('A',num2str(eilute));
        xlswrite(filename,A,sheet,xlRange);
        clearvars A;
```

```
    eilute=eilute+1;

    A = {'',ID, ID+1, ai_vardas(i,:),'Analogas'};
    xlRange=strcat('A',num2str(eilute));
    xlswrite(filename,A,sheet,xlRange);
    clearvars A;
    eilute=eilute+1;
    ID=ID+2;

   A = {'','Column','ID', 'Model ID', 'Name','Type','Default Value'};
   xlRange=strcat('A',num2str(eilute));
    xlswrite(filename,A,sheet,xlRange);
    clearvars A;
    eilute=eilute+1;
   A = {'','',ID, ID, 'ID' ,'int',ai_id(i)};
   xlRange=strcat('A',num2str(eilute));
    xlswrite(filename,A,sheet,xlRange);
    clearvars A;
    eilute=eilute+1;
    ID=ID+1;
   A = {'','',ID, ID, 'vardas' ,'sting[10]',ai_vardas(i,:)};
   xlRange=strcat('A',num2str(eilute));
    xlswrite(filename,A,sheet,xlRange);
    clearvars A;
    eilute=eilute+1;
    ID=ID+1;
   A = {'','',ID, ID, 'tipas' ,'string[10]',ai_tipas(i,:)};
   xlRange=strcat('A',num2str(eilute));
    xlswrite(filename,A,sheet,xlRange);
    clearvars A;
    eilute=eilute+1;
    ID=ID+1;
   A = {'','',ID, ID, 'Range_min' ,'float','0.0'}; % reikia ziureti specifikacija
   xlRange=strcat('A',num2str(eilute));
    xlswrite(filename,A,sheet,xlRange);
    clearvars A;
    eilute=eilute+1;
    ID=ID+1;
   A = {'','',ID, ID, 'Range_max' ,'float','100.0'}; % reikia ziureti specifikacija
   xlRange=strcat('A',num2str(eilute));
    xlswrite(filename,A,sheet,xlRange);
    clearvars A;
    eilute=eilute+1;
    ID=ID+1;
   A = {'','',ID, ID, 'LL_Limit' ,'float','0.0'}; % reikia ziureti salygose
   xlRange=strcat('A',num2str(eilute));
    xlswrite(filename,A,sheet,xlRange);
    clearvars A;
    eilute=eilute+1;
    ID=ID+1;
   A = {'','',ID, ID, 'L_Limit' ,'float','0.0'}; % reikia ziureti salygose
   xlRange=strcat('A',num2str(eilute));
    xlswrite(filename,A,sheet,xlRange);
    clearvars A;
    eilute=eilute+1;
    ID=ID+1;
   A = {'','',ID, ID, 'H_Limit' ,'float','0.0'}; % reikia ziureti salygose
   xlRange=strcat('A',num2str(eilute));
    xlswrite(filename,A,sheet,xlRange);
    clearvars A;
    eilute=eilute+1;
    ID=ID+1;
   A = {'','',ID, ID, 'HH_Limit' ,'float','0.0'}; % reikia ziureti salygose
   xlRange=strcat('A',num2str(eilute));
    xlswrite(filename,A,sheet,xlRange);
    clearvars A;
    eilute=eilute+2;
    ID=ID+1;
end

for i=1:var_kiek
    A = {'Object','ID', 'Model ID', 'Name','Class'};
    xlRange=strcat('A',num2str(eilute));
    xlswrite(filename,A,sheet,xlRange);
    clearvars A;
    eilute=eilute+1;

    A = {'',ID, ID+1,var_vardas(i,:),'Variklis'};
    xlRange=strcat('A',num2str(eilute));
    xlswrite(filename,A,sheet,xlRange);
    clearvars A;
    eilute=eilute+1;
    ID=ID+2;

   A = {'','Column','ID', 'Model ID', 'Name','Type','Default Value'};
   xlRange=strcat('A',num2str(eilute));
    xlswrite(filename,A,sheet,xlRange);
    clearvars A;
    eilute=eilute+1;
   A = {'','',ID, ID, 'ID' ,'int',var_id(i)};
```

```
    xlRange=strcat('A',num2str(eilute));
     xlswrite(filename,A,sheet,xlRange);
     clearvars A;
     eilute=eilute+1;
     ID=ID+1;
    A = {'','',ID, ID, 'vardas' ,'sting[10]',var_vardas(i,:)};
    xlRange=strcat('A',num2str(eilute));
     xlswrite(filename,A,sheet,xlRange);
     clearvars A;
     eilute=eilute+2;
     ID=ID+1;
end


for i=1:sk_kiek
     A = {'Object','ID', 'Model ID', 'Name','Class'};
     xlRange=strcat('A',num2str(eilute));
     xlswrite(filename,A,sheet,xlRange);
     clearvars A;
     eilute=eilute+1;

     A = {'',ID, ID+1, sk_vardas(i,:),'Sklende'};
     xlRange=strcat('A',num2str(eilute));
     xlswrite(filename,A,sheet,xlRange);
     clearvars A;
     eilute=eilute+1;
     ID=ID+2;

    A = {'','Column','ID', 'Model ID', 'Name','Type','Default Value'};
    xlRange=strcat('A',num2str(eilute));
     xlswrite(filename,A,sheet,xlRange);
     clearvars A;
     eilute=eilute+1;
    A = {'','',ID, ID, 'ID' ,'int',sk_id(i)};
    xlRange=strcat('A',num2str(eilute));
     xlswrite(filename,A,sheet,xlRange);
     clearvars A;
     eilute=eilute+1;
     ID=ID+1;
    A = {'','',ID, ID, 'vardas' ,'sting[10]',sk_vardas(i,:)};
    xlRange=strcat('A',num2str(eilute));
     xlswrite(filename,A,sheet,xlRange);
     clearvars A;
     eilute=eilute+2;
     ID=ID+1;

end


 for j=1:reg_kiek
    A = {'Object','ID', 'Model ID', 'Name','Class'};
     xlRange=strcat('A',num2str(eilute));
     xlswrite(filename,A,sheet,xlRange);
     clearvars A;
     eilute=eilute+1;

     A = {'',ID, ID+1, reg_vardas(j,:),'Reguliatorius'};
     xlRange=strcat('A',num2str(eilute));
     xlswrite(filename,A,sheet,xlRange);
     clearvars A;
     eilute=eilute+1;
     ID=ID+2;

    A = {'','Column','ID', 'Model ID', 'Name','Type','Default Value'};
    xlRange=strcat('A',num2str(eilute));
     xlswrite(filename,A,sheet,xlRange);
     clearvars A;
     eilute=eilute+1;
    A = {'','',ID, ID, 'ID' ,'int',reg_id(j)};
    xlRange=strcat('A',num2str(eilute));
     xlswrite(filename,A,sheet,xlRange);
     clearvars A;
     eilute=eilute+1;
     ID=ID+1;
    A = {'','',ID, ID, 'vardas' ,'sting[10]',reg_vardas(j,:)};
    xlRange=strcat('A',num2str(eilute));
     xlswrite(filename,A,sheet,xlRange);
     clearvars A;
     eilute=eilute+1;
     ID=ID+1;
    A = {'','',ID, ID, 'tipas' ,'string[10]',reg_tipas(j,:)};
    xlRange=strcat('A',num2str(eilute));
     xlswrite(filename,A,sheet,xlRange);
     clearvars A;
     eilute=eilute+1;
     ID=ID+1;
    A = {'','',ID, ID, 'SP' ,'float','0.0'}; % reikia ziureti specifikacija
    xlRange=strcat('A',num2str(eilute));
     xlswrite(filename,A,sheet,xlRange);
```

```
 clearvars A;
 eilute=eilute+1;
 ID=ID+1;
A = {'','',ID, ID, 'Y_min' ,'float','100.0'}; % reikia ziureti specifikacija
xlRange=strcat('A',num2str(eilute));
 xlswrite(filename,A,sheet,xlRange);
 clearvars A;
 eilute=eilute+1;
 ID=ID+1;
A = {'','',ID, ID, 'Y_max' ,'float','0.0'}; % reikia ziureti salygose
xlRange=strcat('A',num2str(eilute));
 xlswrite(filename,A,sheet,xlRange);
 clearvars A;
 eilute=eilute+1;
 ID=ID+1;
A = {'','',ID, ID, 'Ti' ,'float','0.0'}; % reikia ziureti salygose
xlRange=strcat('A',num2str(eilute));
 xlswrite(filename,A,sheet,xlRange);
 clearvars A;
 eilute=eilute+1;
 ID=ID+1;
A = {'','',ID, ID, 'Gain' ,'float','0.0'}; % reikia ziureti salygose
xlRange=strcat('A',num2str(eilute));
 xlswrite(filename,A,sheet,xlRange);
 clearvars A;
 eilute=eilute+2;
 ID=ID+1;

     kelintas=reg_grupe(j);

     for i=1:var_kiek
         if var_grupe(i)==kelintas
             A = {'Association ','ID', 'Model ID', 'Name','From','To'};
            xlRange=strcat('A',num2str(eilute));
             xlswrite(filename,A,sheet,xlRange);
             clearvars A;
             eilute=eilute+1;
             jung_i=var_id(i);
             jung_is=reg_id(j);
             A = {'',ID, ID+1, '',jung_is,jung_i};
             xlRange=strcat('A',num2str(eilute));
             xlswrite(filename,A,sheet,xlRange);
             clearvars A;
             eilute=eilute+2;
             ID=ID+2;
              A = {'Association ','ID', 'Model ID', 'Name','From','To'};
            xlRange=strcat('A',num2str(eilute));
             xlswrite(filename,A,sheet,xlRange);
             clearvars A;
             eilute=eilute+1;
             A = {'',ID, ID+1, '',jung_i,jung_is};
             xlRange=strcat('A',num2str(eilute));
             xlswrite(filename,A,sheet,xlRange);
             clearvars A;
             eilute=eilute+2;
             ID=ID+2;
         end
     end

     for i=1:sk_kiek
         if sk_grupe(i)==kelintas
             A = {'Association ','ID', 'Model ID', 'Name','From','To'};
            xlRange=strcat('A',num2str(eilute));
             xlswrite(filename,A,sheet,xlRange);
             clearvars A;
             eilute=eilute+1;
             jung_i=sk_id(i);
             jung_is=reg_id(j);
             A = {'',ID, ID+1, '',jung_is,jung_i};
             xlRange=strcat('A',num2str(eilute));
             xlswrite(filename,A,sheet,xlRange);
             clearvars A;
             eilute=eilute+2;
             ID=ID+2;
              A = {'Association ','ID', 'Model ID', 'Name','From','To'};
            xlRange=strcat('A',num2str(eilute));
             xlswrite(filename,A,sheet,xlRange);
             clearvars A;
             eilute=eilute+1;
             A = {'',ID, ID+1, '',jung_i,jung_is};
             xlRange=strcat('A',num2str(eilute));
             xlswrite(filename,A,sheet,xlRange);
             clearvars A;
             eilute=eilute+2;
             ID=ID+2;
         end
     end

     for i=1:ai_kiek
         if ai_grupe(i)==kelintas
```

```
                    A = {'Association ','ID', 'Model ID', 'Name','From','To'};
                   xlRange=strcat('A',num2str(eilute));
                    xlswrite(filename,A,sheet,xlRange);
                    clearvars A;
                    eilute=eilute+1;
                    jung_is=ai_id(i);
                    jung_i=reg_id(j);
                    A = {'',ID, ID+1, '',jung_is,jung_i};
                    xlRange=strcat('A',num2str(eilute));
                    xlswrite(filename,A,sheet,xlRange);
                    clearvars A;
                    eilute=eilute+2;
                    ID=ID+2;
                     A = {'Association ','ID', 'Model ID', 'Name','From','To'};
                   xlRange=strcat('A',num2str(eilute));
                    xlswrite(filename,A,sheet,xlRange);
                    clearvars A;
                    eilute=eilute+1;
                    A = {'',ID, ID+1, '',jung_i,jung_is};
                    xlRange=strcat('A',num2str(eilute));
                    xlswrite(filename,A,sheet,xlRange);
                    clearvars A;
                    eilute=eilute+2;
                    ID=ID+2;
                end
            end
    end
end
```

8.1.2 Šablonų radimo funkcija

```
function [Likes_pav, kiek, objektas_x1,objektas_x2, objektas_y1,objektas_y2,radimas ] = func_template( Pradinis,
Pilkas_T, pavadinimas,riba, pasukimu_sk, flip)

spalva=255;
 priedas=2;

 Likes_pav=Pradinis;

  [template_y template_x] = size(Pilkas_T);

  template_x=template_x+priedas*2;
  template_y=template_y+priedas*2;
  kiek=100;
  skaicius=0;
  buves_sk=0;

  for sk=1:pasukimu_sk
  %    figure;imshow(Pilkas_T); title('Template');
      cc=normxcorr2(Pilkas_T,Likes_pav);

 %    figure; imshow(cc); title('CC matric');

       for k=skaicius+1:kiek
          rast=max(cc(:));
          if rast>riba
             [a b]=find(cc==rast);
             skaicius=k;
             kampas_x(k)=a(1);
             kampas_y(k)=b(1);
             pasuktas(k)=mod(sk,2);
             radimas(k)=rast;
             for j=0:template_x
                 x=b(1)-floor(template_x*0.5)+j;
                    if x> 0
                        for z=0:template_y
                           y=a(1)-floor(template_y*0.5)+z;
                              if y>0
                                cc(y+priedas,x+priedas)=0;
                              end
                        end
                    end
              end
          end
       end
  %    figure; imshow(cc); title('CC ats');


 %     figure; imshow(Likes_pav); title(pavadinimas);
      for k=buves_sk+1:skaicius
 %       viscircles([kampas_y(k) kampas_x(k) ],template_x/2);
        for j=0:template_x
                 x=kampas_y(k)-j+priedas*0;
                    if x> 0
                        for z=0:template_y
                           y=kampas_x(k)-z+priedas*0;
                              if y>0
                                Likes_pav(y+priedas,x+priedas)=spalva;
```

```
                            end
                        end
                    end
                end
        end;
        buves_sk=skaicius;
        Pilkas_T=rot90(Pilkas_T);
        tt=template_x;
        template_x=template_y;
        template_y=tt;
    end
    if flip==1
        Pilkas_T = fliplr(Pilkas_T);
        for sk=1:pasukimu_sk
      %      figure;imshow(Pilkas_T); title('Template');
            cc=normxcorr2(Pilkas_T,Likes_pav);

   %         figure; imshow(cc); title('CC matric');

            for k=skaicius+1:kiek
               rast=max(cc(:));
               if rast>riba
                  [a b]=find(cc==rast);
                  skaicius=k;
                  kampas_x(k)=a;
                  kampas_y(k)=b;
                  pasuktas(k)=mod(sk,2);
                  radimas(k)=rast;
                  for j=0:template_x
                      x=b-floor(template_x*0.5)+j;
                      if x> 0
                          for z=0:template_y
                              y=a-floor(template_y*0.5)+z;
                              if y>0
                                cc(y+priedas,x+priedas)=0;
                              end
                          end
                      end
                  end
               end
            end
    %        figure; imshow(cc); title('CC ats');

  %          figure; imshow(Likes_pav); title(pavadinimas);
            for k=buves_sk+1:skaicius
  %            viscircles([kampas_y(k) kampas_x(k) ],template_x/2);
              for j=0:template_x
                       x=kampas_y(k)-j+priedas*0;
                       if x> 0
                           for z=0:template_y
                              y=kampas_x(k)-z+priedas*0;
                                 if y>0
                                   Likes_pav(y+priedas,x+priedas)=spalva;
                                 end
                           end
                       end
                  end
            end;
            buves_sk=skaicius;
            Pilkas_T=rot90(Pilkas_T);
            tt=template_x;
            template_x=template_y;
            template_y=tt;
        end
    end
     figure; imshow(Pradinis); title(pavadinimas);
            for k=1:skaicius
              objektas_x1(k)=kampas_y(k)-template_x*pasuktas(k)-template_y*(1-pasuktas(k))+priedas;
              objektas_y1(k)=kampas_x(k)-template_y*pasuktas(k)-template_x*(1-pasuktas(k))+priedas;
              objektas_x2(k)=kampas_y(k);
              objektas_y2(k)=kampas_x(k);
   %           viscircles([objektas_x1(k) objektas_y1(k) ],template_x/4);
   %           viscircles([objektas_x2(k) objektas_y2(k) ],template_x/4);
              c=(template_x/2+template_y/2)/2;
              a=(objektas_x2(k)-objektas_x1(k))/2+objektas_x1(k);
              b=(objektas_y2(k)-objektas_y1(k))/2+objektas_y1(k);
              viscircles([a b],template_x/4);
            end
%   figure; imshow(Likes_pav); title('likutis');
    kiek=skaicius;
end
```

8.1.3 Punktyrų funkcija

```
function [linijos_x1,linijos_x2,linijos_y1,linijos_y2,linijos_dif_x,linijos_dif_y,kaimynas] = punktyru_radimas(
Pilkas_pav,BW,riba,peak_sk,punkt_linija,punkt_kof)

zz=1000;
min_punktyras=punkt_kof*punkt_linija;
```

```
punkt_tarpas=min_punktyras;
atstumas_tarp=punkt_linija*0.8;

figure, imshow(Pilkas_pav), hold on; title(['punkt linija ',num2str(punkt_linija),' punkt kof
',num2str(punkt_kof)]);

kofas=1;

linijos_nr=1;
while zz> 0
 %   figure, imshow(BW), hold on; title(['punkt linija radimas']);
    [H,T,R] = hough(BW);
    maks=ceil(riba*max(H(:)));
    P  = houghpeaks(H,peak_sk,'threshold',maks,'NHoodSize',[kofas kofas]);
    lines = houghlines(BW,T,R,P,'FillGap',2,'MinLength',3);
    zz=length(lines);
    max_len = 0;
     for k = 1:length(lines)
       xy = [lines(k).point1; lines(k).point2];

       x1=xy(1,1);
       y1=xy(1,2);
       x2=xy(2,1);
       y2=xy(2,2);
       dif_x=x1-x2;
       dif_y=y1-y2;
       mm=max(abs(dif_x),abs(dif_y));
       if mm<punkt_linija && mm> min_punktyras
           if abs(dif_x) >0
               linijos_kampas=dif_y/dif_x;
           else
               linijos_kampas=dif_x/dif_y;
           end
           if abs(linijos_kampas) <= 0
               plot(xy(:,1),xy(:,2),'LineWidth',5,'Color','green');
               linijos_dif_y(linijos_nr)=dif_x;
               linijos_dif_x(linijos_nr)=dif_y;
               linijos_x1(linijos_nr)=x1;
               linijos_x2(linijos_nr)=x2;
               linijos_y1(linijos_nr)=y1;
               linijos_y2(linijos_nr)=y2;
               linijos_nr=linijos_nr+1;
           end

       end
           for j=1:mm
                yy=y1+round(-dif_y*j/mm);
                xx=x1+round(-dif_x*j/mm);

                BW(yy,xx)=0;
                BW(yy+1,xx+1)=0;
                BW(yy+2,xx+2)=0;
                BW(yy+3,xx+3)=0;

                if xx>3 & yy>3
                    BW(yy-1,xx-1)=0;
                    BW(yy-2,xx-2)=0;
                    BW(yy-3,xx-3)=0;
                elseif xx>2 & yy>2
                    BW(yy-1,xx-1)=0;
                    BW(yy-2,xx-2)=0;
                elseif xx>1 & yy>1
                    BW(yy-1,xx-1)=0;
                end
           end
     end
end
%figure; imshow(BW); title('po punktyru');
 linijos_nr=linijos_nr-1;
[B,indeks]=sort(linijos_x1);
linijos_x1=pakeist(linijos_x1,indeks,linijos_nr);
linijos_x2=pakeist(linijos_x2,indeks,linijos_nr);
linijos_y1=pakeist(linijos_y1,indeks,linijos_nr);
linijos_y2=pakeist(linijos_y2,indeks,linijos_nr);
linijos_dif_x=pakeist(linijos_dif_x,indeks,linijos_nr);
linijos_dif_y=pakeist(linijos_dif_y,indeks,linijos_nr);
kaimynas=zeros(linijos_nr,1);
pakeistas=zeros(linijos_nr,1);
for j=1:linijos_nr
    if kaimynas(j)==0
        kaimynas(j)=j;
    end;
    Kaimyno_nr=zeros(linijos_nr,1);
    for linija=1:linijos_nr
        ok=0;
        if linijos_x1(j)+ atstumas_tarp>= linijos_x1(linija) && linijos_x1(j)- atstumas_tarp <= linijos_x1(linija)
            ok=1;
        elseif linijos_x2(j)+ atstumas_tarp>= linijos_x1(linija) && linijos_x2(j)- atstumas_tarp <=
linijos_x1(linija)
```

```
            ok=1;
        elseif linijos_x1(j)+ atstumas_tarp>= linijos_x2(linija) && linijos_x1(j)- atstumas_tarp <=
linijos_x2(linija)
            ok=1;
        elseif linijos_x2(j)+ atstumas_tarp>= linijos_x2(linija) && linijos_x2(j)- atstumas_tarp <=
linijos_x2(linija)
            ok=1;
        end
        if ok==1
            if linijos_y1(j)+ atstumas_tarp>= linijos_y1(linija) && linijos_y1(j)- atstumas_tarp <=
linijos_y1(linija)
                ok=1;
            elseif linijos_y2(j)+ atstumas_tarp>= linijos_y1(linija) && linijos_y2(j)- atstumas_tarp <=
linijos_y1(linija)
                ok=1;
            elseif linijos_y1(j)+ atstumas_tarp>= linijos_y2(linija) && linijos_y1(j)- atstumas_tarp <=
linijos_y2(linija)
                ok=1;
            elseif linijos_y2(j)+ atstumas_tarp>= linijos_y2(linija) && linijos_y2(j)- atstumas_tarp <=
linijos_y2(linija)
                ok=1;
            else
                ok=0;
            end
        end
        if ok==1
            Kaimyno_nr(linija)=linija;
        end
    end

    zzz=Kaimyno_nr;
    zzz(~zzz)=inf;
    mm = min(zzz);
    for aa=1:linijos_nr
       if Kaimyno_nr(aa) > 0 && kaimynas(aa) > 0 && kaimynas(aa) < mm
           mm=kaimynas(aa);
       end
    end
    pakeistas_nr=1;
    for aa=1:linijos_nr
       if Kaimyno_nr(aa) > 0
           if kaimynas(aa)> mm || kaimynas(aa)==0
              pakeistas(pakeistas_nr)=kaimynas(aa);
              kaimynas(aa)=mm;
              pakeistas_nr=pakeistas_nr+1;
           end
       end
    end

    unik=unique(pakeistas);
    ilgis=length(unik);

    for aa=1:linijos_nr
       for ii=1:ilgis
           if kaimynas(aa) == unik(ii) && kaimynas(aa)>0
               kaimynas(aa)=mm;
           end
       end
    end
end
```

8.2 UML importavimo failas

Diagram	**ID**	**Name**	**Type**			
	1	Objektu diagrama	ERDiagram			
Object	**ID**	**Model ID**	**Name**	**Class**		
	2	3	P01	Analogas		
	Column	**ID**	**Model ID**	**Name**	**Type**	**Default**
		4	4	ID	int	1
		5	5	vardas	sting[10]	P01
		6	6	tipas	string[10]	PI
		7	7	min	float	0
		8	8	max	float	100
		9	9	LL Limit	float	0
		10	10	L Limit	float	0
		11	11	H Limit	float	0
		12	12	HH Limit	float	0
Object	**ID**	**Model ID**	**Name**	**Class**		
	13	14	T02	Analogas		
	Column	**ID**	**Model ID**	**Name**	**Type**	**Default**
		15	15	ID	int	2
		16	16	vardas	sting[10]	T02
		17	17	tipas	string[10]	TI
		18	18	min	float	0
		19	19	max	float	100
		20	20	LL Limit	float	0
		21	21	L Limit	float	0
		22	22	H Limit	float	0
		23	23	HH Limit	float	0

Object	ID	Model ID	Name	Class		
	24	25	P07	Analogas		
	Column	ID	Model ID	Name	Type	Default
		26	26	ID	int	3.52E+11
		27	27	vardas	sting[10]	P07
		28	28	tipas	string[10]	PI
		29	29	min	float	0
		30	30	max	float	100
		31	31	LL Limit	float	0
		32	32	L Limit	float	0
		33	33	H Limit	float	0
		34	34	HH Limit	float	0
Object	ID	Model ID	Name	Class		
	35	36	P03	Analogas		
	Column	ID	Model ID	Name	Type	Default
		37	37	ID	int	4
		38	38	vardas	sting[10]	P03
		39	39	tipas	string[10]	PI
		40	40	min	float	0
		41	41	max	float	100
		42	42	LL Limit	float	0
		43	43	L Limit	float	0
		44	44	H Limit	float	0
		45	45	HH Limit	float	0
Object	ID	Model ID	Name	Class		
	46	47	T04	Analogas		
	Column	ID	Model ID	Name	Type	Default
		48	48	ID	int	4.29E+09
		49	49	vardas	sting[10]	T04
		50	50	tipas	string[10]	TIH
		51	51	min	float	0
		52	52	max	float	100
		53	53	LL Limit	float	0
		54	54	L Limit	float	0
		55	55	H Limit	float	0
		56	56	HH Limit	float	0
Object	ID	Model ID	Name	Class		
	57	58	T06	Analogas		
	Column	ID	Model ID	Name	Type	Default
		59	59	ID	int	4.29E+09
		60	60	vardas	sting[10]	T06
		61	61	tipas	string[10]	TI
		62	62	min	float	0
		63	63	max	float	100
		64	64	LL Limit	float	0
		65	65	L Limit	float	0
		66	66	H Limit	float	0
		67	67	HH Limit	float	0
Object	ID	Model ID	Name	Class		
	68	69	P05	Analogas		
	Column	ID	Model ID	Name	Type	Default
		70	70	ID	int	1.59E+11
		71	71	vardas	sting[10]	P05
		72	72	tipas	string[10]	PI
		73	73	min	float	0
		74	74	max	float	100
		75	75	LL Limit	float	0
		76	76	L Limit	float	0
		77	77	H Limit	float	0
		78	78	HH Limit	float	0
Object	ID	Model ID	Name	Class		
	79	80	P08	Analogas		
	Column	ID	Model ID	Name	Type	Default
		81	81	ID	int	3.52E+11
		82	82	vardas	sting[10]	P08
		83	83	tipas	string[10]	PIL
		84	84	min	float	0
		85	85	max	float	100
		86	86	LL Limit	float	0
		87	87	L Limit	float	0
		88	88	H Limit	float	0
		89	89	HH Limit	float	0
Object	ID	Model ID	Name	Class		
	90	91	S-1	Variklis		
	Column	ID	Model ID	Name	Type	Default
		92	92	ID	int	1
		93	93	vardas	sting[10]	S-1
Object	ID	Model ID	Name	Class		
	94	95	S-2	Variklis		
	Column	ID	Model ID	Name	Type	Default
		96	96	ID	int	2
		97	97	vardas	sting[10]	S-2
Object	ID	Model ID	Name	Class		
	98	99	S-3	Variklis		
	Column	ID	Model ID	Name	Type	Default
		100	100	ID	int	3.52E+11
		101	101	vardas	sting[10]	S-3
Object	ID	Model ID	Name	Class		
	102	103	TR-1	Sklende		
	Column	ID	Model ID	Name	Type	Default
		104	104	ID	int	4.3E+09
		105	105	vardas	sting[10]	TR-1

Object	**ID**	**Model ID**	**Name**	**Class**		
	106	107	REG3	Reguliatorius		
	Column	ID	Model ID	Name	Type	Default
		108	108	ID	int	3.52E+11
		109	109	vardas	sting[10]	REG3
		110	110	tipas	string[10]	SIC
		111	111	SP	float	0
		112	112	Y min	float	100
		113	113	Y max	float	0
		114	114	Ti	float	0
		115	115	Gain	float	0
Association	**ID**	**Model ID**	**Name**	**From**	**To**	
	116	117		3.52E+11	3.52E+11	
Association	**ID**	**Model ID**	**Name**	**From**	**To**	
	118	119		3.52E+11	3.52E+11	
Association	**ID**	**Model ID**	**Name**	**From**	**To**	
	120	121		3.52E+11	3.52E+11	
Association	**ID**	**Model ID**	**Name**	**From**	**To**	
	122	123		3.52E+11	3.52E+11	
Association	**ID**	**Model ID**	**Name**	**From**	**To**	
	124	125		3.52E+11	3.52E+11	
Association	**ID**	**Model ID**	**Name**	**From**	**To**	
	126	127		3.52E+11	3.52E+11	
Association	**ID**	**Model ID**	**Name**	**From**	**To**	
	128	129		3.52E+11	3.52E+11	
Association	**ID**	**Model ID**	**Name**	**From**	**To**	
	130	131		3.52E+11	3.52E+11	
Object	**ID**	**Model ID**	**Name**	**Class**		
	132	133	REG1	Reguliatorius		
	Column	**ID**	**Model ID**	**Name**	**Type**	**Default**
		134	134	ID	int	4.33E+09
		135	135	vardas	sting[10]	REG1
		136	136	tipas	string[10]	YC
		137	137	SP	float	0
		138	138	Y min	float	100
		139	139	Y max	float	0
		140	140	Ti	float	0
		141	141	Gain	float	0
Association	**ID**	**Model ID**	**Name**	**From**	**To**	
	142	143		4.33E+09	4.3E+09	
Association	**ID**	**Model ID**	**Name**	**From**	**To**	
	144	145		4.3E+09	4.33E+09	
Association	**ID**	**Model ID**	**Name**	**From**	**To**	
	146	147		4.29E+09	4.33E+09	
Association	**ID**	**Model ID**	**Name**	**From**	**To**	
	148	149		4.33E+09	4.29E+09	
Association	**ID**	**Model ID**	**Name**	**From**	**To**	
	150	151		4.29E+09	4.33E+09	
Association	**ID**	**Model ID**	**Name**	**From**	**To**	
	152	153		4.33E+09	4.29E+09	
Object	**ID**	**Model ID**	**Name**	**Class**		
	154	155	REG2	Reguliatorius		
	Column	**ID**	**Model ID**	**Name**	**Type**	**Default**
		156	156	ID	int	1.59E+11
		157	157	vardas	sting[10]	REG2
		158	158	tipas	string[10]	YS
		159	159	SP	float	0
		160	160	Y min	float	100
		161	161	Y max	float	0
		162	162	Ti	float	0
		163	163	Gain	float	0
Association	**ID**	**Model ID**	**Name**	**From**	**To**	
	164	165		1.59E+11	1.59E+11	
Association	**ID**	**Model ID**	**Name**	**From**	**To**	
	166	167		1.59E+11	1.59E+11	
Association	**ID**	**Model ID**	**Name**	**From**	**To**	
	168	169		1.59E+11	1.59E+11	
Association	**ID**	**Model ID**	**Name**	**From**	**To**	
	170	171		1.59E+11	1.59E+11	

8.3 UML sudarytų programų kodai

8.3.1 Analogas.h

```
#ifndef _ANALOGAS_H
#define _ANALOGAS_H
```

```
class Analogas
{
public:
                float           Value;
                uint8_t         status;
                uint8_t cfg;
                uint16_t        ai;

                int ID;
                std::string vardas;
                std::string tipas;
                float           Range_Min;
                float           Range_Max;
                float           LL_Limit;
                float           L_Limit;
                float           H_Limit;
                float           HH_Limit;
                /* Atkomentuoti
                Analogas(int nID = 0, std::string nVardas = "", std::string nTipas = "", float nRange_Min = 0.0, float
nRange_Max = 100.0, float nLL_Limit = 0.0, float nL_Limit = 0.0, float nH_Limit = 100.0, float nHH_Limit = 100.0)
                                : ID(nID), vardas(nVardas), tipas(nTipas), Range_Min(nRange_Min), Range_Max(nRange_Max),
LL_Limit(nLL_Limit), L_Limit(nL_Limit), H_Limit(nH_Limit), HH_Limit(nHH_Limit)
                {
                }
                */
                void A_funkcija();

};

Analogas P01(1,"P01","PI",0.0, 16.0, 0.0, 0.0, 0.0, 0.0);
Analogas T02(2,"T02","TI",0.0,100.0, 0.0, 0.0, 0.0, 0.0);
Analogas P03(4,"P03","PI",0.0, 16.0, 0.0, 0.0, 0.0, 0.0);
Analogas T04(4294967301,"T04","TI",0.0,100.0, 0.0, 0.0,80.0,90.0);
Analogas P05(158913789959,"P05","PI",0.0, 16.0, 0.0, 0.0, 0.0, 0.0);
Analogas T06(4294967302,"T06","TI",0.0,100.0,45.0,50.0, 0.0, 0.0);
Analogas P07(352187318275,"P07","PI",0.0, 16.0, 0.0, 0.0, 0.0, 0.0);
Analogas P08(352187318280,"P08","PI",0.0, 16.0, 5.0, 6.0, 0.0, 0.0);
```

8.3.2 Sklende.h

```
#ifndef _SKLENDE_H
#define _SKLENDE_H

class Sklende
{
public:
                uint16_t        status;
                uint16_t        iejimai;

                int ID;
                std::string vardas;
                int             FB_cntr;
                int             Eigos_T;
                /* Atkomentuoti
                Sklende(int nID = 0, std::string nVardas = "", int nEigos_T = 60)
                                : ID(nID), vardas(nVardas), Eigos_T(nEigos_T))
                {
                }
                */
                void S_funkcija();

};

Sklende TR1(4295032832,"TR1",180.0);
```

8.3.3 Variklis.h

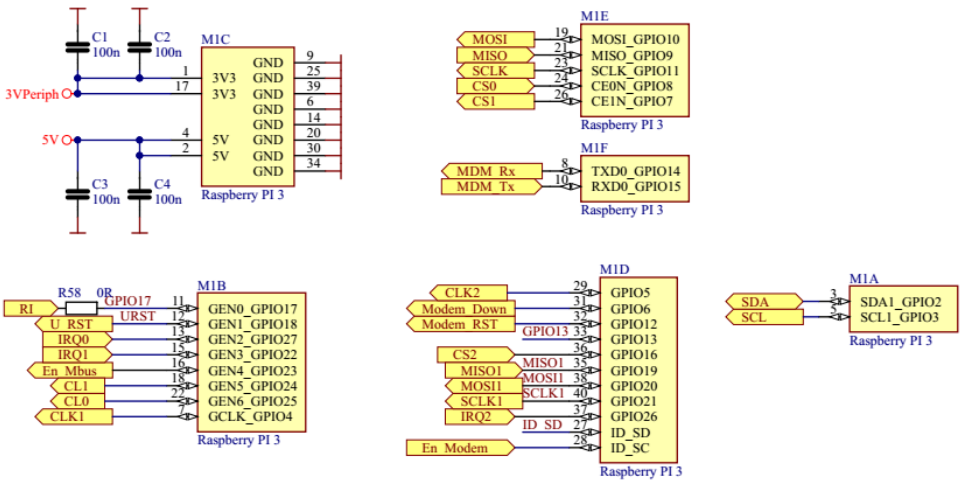
```
#ifndef _VARIKLIS_H
#define _VARIKLIS_H

class Variklis
{
public:
                uint16_t        status;
                uint16_t        iejimai;
                uint16_t        Run_cycles;
                float           Run_hours;
                int ID;
                std::string vardas;
                int             FB_cntr;
                int             Eigos_T;
                /* Atkomentuoti
                Variklis(int nID = 0, std::string nVardas = ""
                                : ID(nID), vardas(nVardas))
                {
```

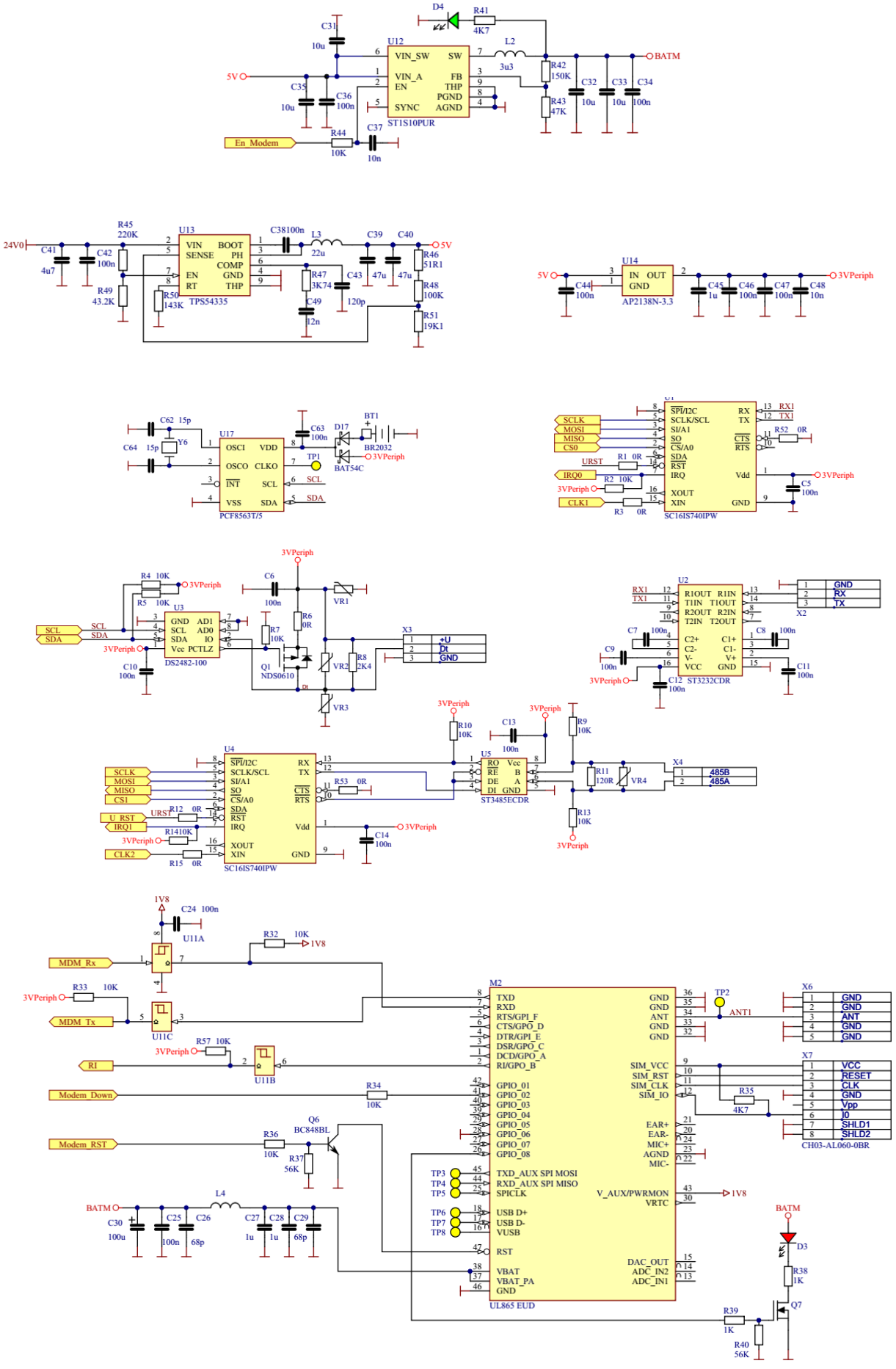
```
                }
                */
                void V_funkcija();

};

Variklis CS1(352187318528,"CS1");
Variklis CS2(352187318784,"CS2");
Variklis PS1(158913790720,"PS1");
```

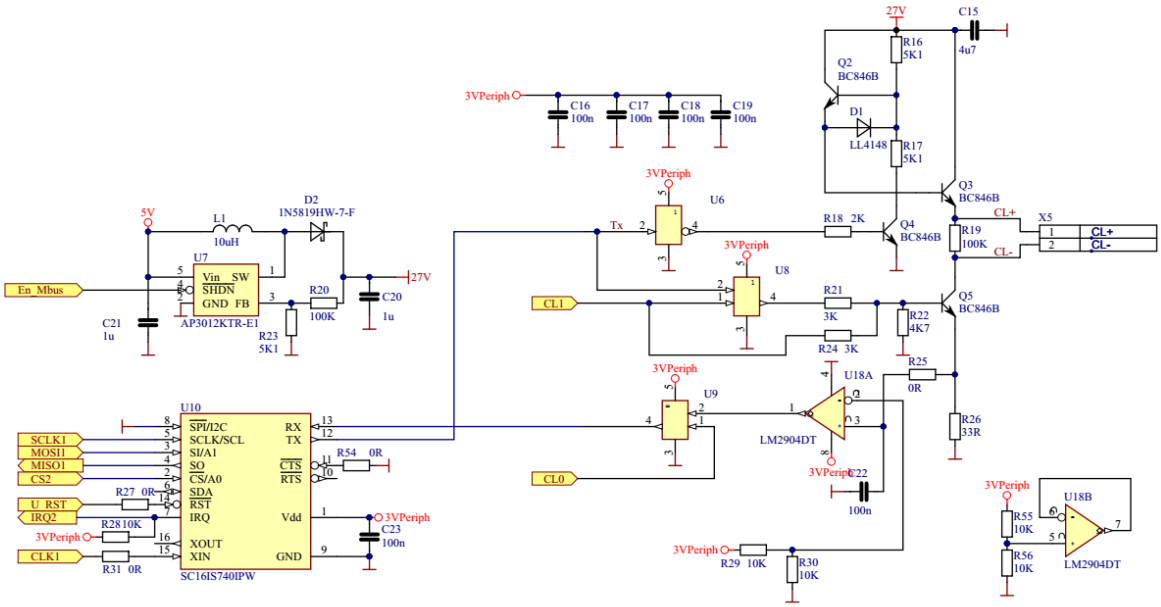
8.3.4 Reguliatorius.h

```
#ifndef _REGULIATORIUS_H
#define _REGULIATORIUS_H

class Reguliatorius
{
public:
                uint8_t         status;
                float           PV;
                float           PVb;
                float           Y;
                float           DeadB;
                float           S3p_Hyst;
                float           ActPosT;
                float           VlV_pos;
                float           SampleT;
                float           Reg_ErrWgt;

                int ID;
                std::string vardas;
                std::string tipas;
                float           SP;
                float           Y_min;
                float           Y_max;
                float           Ti;
                float           Gain;

                /* Atkomentuoti
                Reguliatorius(int nID = 0, std::string nVardas = "", std::string nTipas = "", float nSP = 100.0, float
nY_min = 100.0, float nY_max = 100.0, float nTi = 100.0, float nGain = 100.0)
                                : ID(nID), vardas(nVardas), tipas(nTipas), SP(nSP), Y_min(nY_min), Y_max(nY_max), Ti(nTi),
Gain(nGain))
                {
                }
                */
                void R_funkcija();

};

Reguliatorius REG1(4328521728,"REG1","SIC",65.0,10.0,90.0,40,0.2);
Reguliatorius REG2(158964121600,"REG2","YS",6.5,0.0,100.0,30,0.5);
Reguliatorius REG3(352204095488,"REG3","SIC",9.0,10.0,90.0,60,0.5);
```

8.4 Papildomo modulio schema




D4
R41
4K7
C31
10u
U12
VIN_SW
SW
VIN_A
FB
EN
THP
PGND
SYNC
AGND
ST1S10PUR
L2
3u3
5V
BATM
R42
150K
R43
47K
C32
10u
C33
10u
C34
100n
C35
10u
C36
100n
C37
10n
R44
10K
En_Modem
R45
220K
U13
VIN
BOOT
SENSE
PH
COMP
EN
GND
RT
THP
TPS54335
C38100n
L3
22u
C39
C40
24V0
C41
4u7
C42
100n
R46
51R1
R47
3K74
C43
47u
47u
R48
100K
R49
43.2K
R50
143K
C49
12n
120p
R51
19K1
U14
IN
OUT
GND
AP2138N-3.3
C44
100n
C45
1u
C46
100n
C47
100n
C48
10n
3VPeriph
C62
15p
C64
15p
Y6
U17
OSCI
VDD
OSCO
CLKO
INT
SCL
VSS
SDA
PCF8563T/5
C63
100n
D17
BT1
BR2032
BAT54C
TP1
SCL
SDA
SCLK
MOSI
MISO
CS0
URST
IRQ0
CLK1
SPI/I2C
SCLK/SCL
SI/A1
SO
CS/A0
SDA
RST
IRQ
XOUT
XIN
RX
TX
CTS
RTS
Vdd
GND
SC16IS740IPW
R1 0R
R2 10K
R3 0R
R52 0R
RX1
TX1
C5
100n
R4 10K
R5 10K
U3
GND
AD1
SCL
AD0
SDA
IO
Vcc
PCTLZ
DS2482-100
C10
100n
R6
0R
R7
10K
Q1
NDS0610
VR1
VR2
VR3
R8
2K4
X3
+U
Dt
GND
U2
R1OUT
R1IN
T1IN
T1OUT
R2OUT
R2IN
T2IN
T2OUT
C2+
C1+
C2-
C1-
V-
V+
VCC
ST3232CDR
C7
100n
C8
100n
C9
100n
C12
100n
C11
100n
X2
GND
RX
TX
U4
U5
RO
Vcc
RE
B
DE
A
DI
GND
ST3485ECDR
R9
10K
R10
10K
C13
100n
R11
120R
VR4
X4
485B
485A
R13
10K
R53 0R
U_RST
IRQ1
CLK2
R12 0R
R1410K
R15 0R
C14
100n
1V8
C24 100n
U11A
U11C
U11B
R32 10K
R33 10K
R57 10K
MDM_Rx
MDM_Tx
RI
Modem_Down
Modem_RST
R34
10K
R36
10K
Q6
BC848BL
R37
56K
M2
TXD
RXD
RTS/GPI_F
CTS/GPO_D
DTR/GPI_E
DSR/GPO_C
DCD/GPO_A
RI/GPO_B
GPIO_01
GPIO_02
GPIO_03
GPIO_04
GPIO_05
GPIO_06
GPIO_07
GPIO_08
TXD_AUX SPI MOSI
RXD_AUX SPI MISO
SPICLK
USB D+
USB D-
VUSB
RST
VBAT
VBAT_PA
GND
GND
GND
ANT
GND
GND
SIM_VCC
SIM_RST
SIM_CLK
SIM_IO
EAR+
EAR-
MIC+
AGND
MIC-
V_AUX/PWRMON
VRTC
DAC_OUT
ADC_IN2
ADC_IN1
UL865 EUD
TP2
ANT1
X6
GND
GND
ANT
GND
GND
X7
VCC
RESET
CLK
GND
Vpp
I0
SHLD1
SHLD2
CH03-AL060-0BR
R35
4K7
TP3
TP4
TP5
TP6
TP7
TP8
L4
C30
100u
C25
100n
C26
68p
C27
1u
C28
1u
C29
68p
D3
R38
1K
Q7
R39
1K
R40
56K


27V
C15
4u7
R16
5K1
Q2
BC846B
D1
LL4148
R17
5K1
Q3
BC846B
3VPeriph
C16
100n
C17
100n
C18
100n
C19
100n
U6
Tx
R18 2K
Q4
BC846B
CL+
X5
R19
100K
CL-
3VPeriph
U8
CL1
R21
3K
R24 3K
Q5
BC846B
R22
4K7
R25
0R
R26
33R
3VPeriph
U9
CL0
U18A
LM2904DT
3VPeriph
C22
100n
3VPeriph
R29 10K
R30
10K
3VPeriph
U18B
R55
10K
R56
10K
LM2904DT
5V
L1
10uH
D2
1N5819HW-7-F
U7
Vin SW
SHDN
GND FB
AP3012KTR-E1
En_Mbus
C21
1u
R23
5K1
R20
100K
C20
1u
27V
U10
SPI/I2C
SCLK/SCL
SI/A1
SO
CS/A0
SDA
RST
IRQ
XOUT
XIN
RX
TX
CTS
RTS
Vdd
GND
SC16IS740IPW
SCLK1
MOSI1
MISO1
CS2
U_RST
IRQ2
CLK1
R27 0R
R28 10K
R31 0R
R54 0R
3VPeriph
C23
100n