TALLINNA TEHNIKAÜLIKOOL
INSENERITEADUSKOND
Virumaa kolledž

P&ID digitaliseerimine ja veebipõhise HMI koodi loomine

Digitization of P&ID and generating code for web based HMI

ARUKAD SÜSTEEMID JA RAKENDUSINFOTEHNOLOOGIA ÕPPEKAVA
LÕPUTÖÖ

Üliõpilane: Roomet Ruunik

Üliõpilaskood: 212612EDTR

Juhendaja: Sergei Pavlov, lektor

Kaasjuhendaja: Sergei Ponomar,
Doktorant-nooremteadur

Kohtla-Järve, 2024

AUTORIDEKLARATSIOON

Olen koostanud lõputöö iseseisvalt.

Lõputöö alusel ei ole varem kutse- või teaduskraadi või inseneridiplomit taotletud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

LIHTLITSENTS LÕPUTÖÖ ÜLDSUSELE KÄTTESAADAVAKS TEGEMISEKS JA REPRODUTSEERIMISEKS[1]

Mina Roomet Ruunik (sünnikuupäev: 15.09.1994)

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „P&ID digitaliseerimine ja veebipõhise HMI koodi loomine", mille juhendaja on Sergei Pavlov:
   1.1. reprodutseerimiseks säilitamise ja elektroonilise avaldamise eesmärgil, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta kolmandate isikute intellektuaalomandi ega isikuandmete kaitse seadusest ja teistest õigusaktidest tulenevaid õigusi.

[1] *Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautori(d) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.*

SISUKORD

LÜHENDITE JA TÄHISTE LOETELU

BPMN	Business Process Modeling Notation
CVAT	Computer Vision Annotation Tool
CTPN	Connectionist Text Proposal Network
DFS	Depth-first search
GPU	Graphics processing unit
GUI	Graphical user interface
HMI	Human Machine Interface
HTML	HyperText Markup Language
JSON	JavaScript Object Notation
LMI	Locally mounted instrument
LSD	Line Segment Detector
OCR	Optical Character Recognition
P&ID	Piping and Instrumentation Diagram
SVG	Scalable Vector Graphics
YOLO	You Only Look Once

SISSEJUHATUS

Tänapäeva tööstusautomaatikas ja tootmisprotsesside juhtimises on suur rõhk süsteemide digitaliseerimisel ja efektiivsuse tõstmisel. Üheks olulise tähtsusega komponendiks selles valdkonnas on inimese ja masina vahelised liidesed (HMI), mis võimaldavad seadmete operaatoritel protsesse jälgida ja neid juhtida. Traditsiooniliselt on HMI liidesed loodud käsitsi, tuginedes protsesside ja instrumentide (P&ID) joonistele, mis on aga aeganõudev ja töömahukas. Käesoleva lõputöö eesmärgiks on arendada terviklik lahendus, mis koosneb kolmest põhiosast: P&ID jooniste digitaliseerimine, kasutajale mõeldud graafiline liides jooniste modifitseerimiseks ning veebipõhise HMI jaoks vajaliku koodi genereerimine. Peamine rõhk on pandud veebipõhise HMI koodi loomisele.

Tööstusseadmete P&ID joonised sisaldavad kriitilist teavet süsteemide kohta, mis on oluline nii projekteerimisel kui igapäevases töös. HMI liideste loomine nende jooniste põhjal nõuab tihedat koostööd inseneride ja tarkvaraarendajate vahel, mis võib pikendada projekti valmimise aega. Lisaks on tööstusettevõtetes üha kasvav nõudlus paindlike ja veebipõhiste lahenduste järele, mis pakuvad olenemata asukohast tootmisandmetele reaalajas ligipääsu. Käesoleva töö eesmärk on töötada välja lahendus, mis automatiseerib HMI liideste loomise digitaliseeritud P&ID jooniseid kasutades. Rakendus suudab joonistel tuvastada seadmed ja ühendused ning tõlgendada need automaatselt HTML koodiks, mida saab kasutada veebipõhiste kasutajaliideste arendamisel.

Lõputöö uurimisobjektiks on P&ID joonised ning nendega seotud HMI liideste genereerimise meetodid. Hetkel on olemas erinevaid rakendusi, mis pakuvad tuge HMI liideste loomiseks, näiteks Siemens WinCC ja AVEVA InTouch. Need süsteemid põhinevad peamiselt käsitsi projekteerimisel ega võimalda automatiseeritud protsessi P&ID jooniste põhjal. Samuti ei ole eeltoodud programmidega võimalik eksportida HMI lahendusi veebipõhise koodina. Eksisteerib rakendusi, mis automatiseerivad teatud HMI liideste loomise aspekte, kuid täielikult integreeritud lahendusi, mis suudaksid P&ID jooniseid digitaliseerida ning veebiliidesteks teisendada, ei ole tänaseni loodud või avalikustatud.

Käesoleva lõputöö hüpoteesiks on: „P&ID jooniste automaatne digitaliseerimine on efektiivsem kui nende käsitsi tegemine, andes kasutajale märkimisväärse ajalise võidu. Samuti on võimalik luua automaatselt veebipõhine HMI kood, mis põhineb digitaliseeritud P&ID joonistel."

Rakenduse arendamine on väga mahukas töö, sest hõlmab erinevaid etappe, alustades andmete digitaliseerimisest kuni veebipõhise HMI koodi väljatöötamiseni. Kuigi

käesoleva töö raames on rakendus arendatud minimaalses vormis ja mõeldud töötama väikese andmekoguga, on selle eesmärk pakkuda selget tööpõhimõtet ja panna alus automatiseeritud HMI liideste loomiseks. Töö autor on võtnud arvesse, et testimise ja arendamise käigus võib esineda vigu, kuid töös keskendutakse peamiselt rakenduse tööprotsessi ja funktsionaalsuse tõestamisele.

Antud bakalaureusetöö keskendub P&ID jooniste digitaliseerimise ja HMI koodi genereerimise võimaluste uurimisele ja lahenduse arendamisele.

1. P&ID DIGITALISEERIMINE

Selles peatükis käsitletakse P&ID diagrammide digitaliseerimist, rakendades sümbolite ja ühenduste automaatseks tuvastamiseks erinevaid tehnoloogiad. Autor teeb ülevaate eelnevalt uuritud töödest ja nende tulemustest ning töötab seejärel välja oma rakenduse.

Antud projektis kasutatav süsteem alustab sümbolite tuvastamisega YOLOv8 mudeli abil, mis pakub objekti täpset tuvastamist piltidel. Seejärel rakendatakse OpenCV Line Segment Detector (LSD) meetodit, et tuvastada pildil olevad jooned. Viimane ja kõige olulisem etapp on sümbolite ning joonte omavahelise ühendatuse kontroll. See analüüs põhineb sümbolite ja joonte vahelise kauguse kontrollimisel, et määrata, kas kaks või enam sümbolit on joone kaudu omavahel seotud.

1.1 P&ID digitaliseerimise uurimuste analüüs

P&ID digitaliseerimine ei ole uus uurimisvaldkond ning seda on viimastel aastakümnetel aina enam käsitletud ning praktikas rakendatud. Sel teemal on avaldatud arvukalt teadustöid, kus on kirjeldatud erinevaid P&ID digitaliseerimise meetodeid ning nende efektiivsust põhjalikult analüüsitud. Mitmed uurimused on käsitlenud süvaõppe, pilditöötluse, tehisintellekti ja graafipõhiste meetodite rakendamist, et parandada P&ID-de digitaliseerimise kvaliteeti, täpsust ja efektiivsust. Need tööd on keskendunud eelkõige sümbolite, tekstielementide, pidev- ja katkendjoonte tuvastamisele ning tuvastatud elementide omavaheliste seoste loomisele.

Sümbolite tuvastamisel on kasutatud mitmeid süvaõppe mudeleid, näiteks AlexNet [1] ja YOLOv5 [2]. Uuringud on tõestanud kõrgeid tuvastustäpsuseid, mis ulatuvad keerukamate sümbolite puhul kuni 97% [3]. Samuti on saavutatud häid tulemusi pidevjoonte äratundmisel, kus täpsus võib küündida üle 90% [1,3,4]. Katkendjoonte tuvastamine osutub seevastu keerukamaks, sest müra, madal resolutsioon ja skeemi keerukus mõjutavad täpsusnäitajaid negatiivselt [3]. Tekstielementide tuvastamiseks on kasutatud erinevaid optilise märgituvastuse (OCR) lahendusi, sealhulgas Tesseract ja CTPN, mille edukus sõltub oluliselt nii pildi kvaliteedist kui ka kirjastiilist ja -paigutusest [1,5,2].

Mitmed tööd on keskendunud ka elementide omavaheliste ühenduste leidmisele, kasutades selleks graafipõhiseid otsingumeetodeid. Antud meetodi juures modelleeritakse skeem graafidena ja rakendatakse graafipõhiseid otsingumeetodeid, näiteks sügavuti otsingut (DFS) [5]. See võimaldab teisendada P&ID-s sisalduvad andmed masinloetavaks andmestruktuuriks. Samas on rõhutatud, et sageli on eelkõige keerukamate skeemide puhul tuvastuse kvaliteedi ja usaldusväärsuse tagamiseks

vajalik inimese järelkontroll ja täiendav valideerimine [2].

Uuringutest nähtub ka protsessi efektiivsuse oluline tõus automatiseeritud tööriistade kasutamise korral. Näiteks vähendas Sung-O Kangi, Eul-Bum Lee ja Hum-Kyung Baeki lahendus 400 P&ID lehe käsitsi digitaliseerimiseks kuluvat aega 3200 töötunnilt ligikaudu 200 tunnini [4]. Seega võimaldab taoline tehnoloogia suurendada nii ajalist kui ka rahalist efektiivsust, mis omakorda võib aidata kaasa tööstuslike projektide tõhusamale haldamisele.

Tabel 1.1 koondab erinevate uuringute tulemused sümbolite, tekstide ja joonte tuvastamise täpsuse kohta. Kõrgeim sümbolite tuvastuse täpsus (97%) saavutati arvutinägemise ja süvaõppe meetodite kombineerimisel, samas kui tekstide tuvastamise täpsus jäi kohati alla 90% [3].

Tabel 1.1 P&ID digitaliseerimise täpsuse tulemuste võrdlus

	[1]	[2]	[3]	[4]	[5]
Sümbolid	91,6%	80%	97%	93%	~95%
Tähemärgid	83,1%	-	79,21%	87%	-
Jooned	90,6%	-	91,125%	93%	-

Kokkuvõtvalt on senised uurimused näidanud, et P&ID digitaliseerimisel saavutatakse kõrged täpsusnäitajad sümbolite, tekstide ning joonte tuvastamisel. Samas on jätkuvalt arenguruumi spetsiifilisemates valdkondades, näiteks katkendjoonte tuvastamises, keerukate tekstielementide korrektsuses ja sümbolite varieeruvuses erinevate tööstusharude vahel. Edasised uuringud ja andmekogumite standardiseerimine võimaldavad tõenäoliselt parandada tuvastusmudeleid ning vähendada vajadust inimjärelevalve järele, kindlustades seeläbi P&ID skeemide digitaliseerimise protsessi stabiilsuse, täpsuse ja tõhususe.

1.2 Sümbolite tuvastamine

Järgnevalt rakendab autor sarnaseid meetodeid alustades sümbolite tuvastamisest, et luua ka ise P&ID digitaliseerimise rakendus.

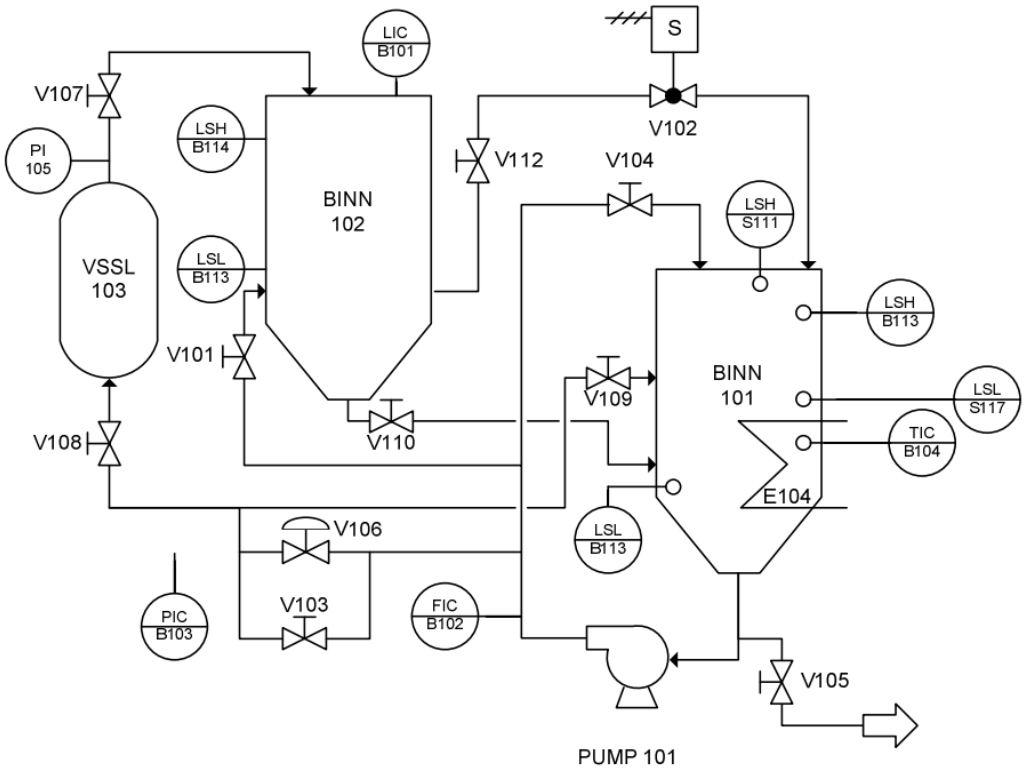
Sümbolite tuvastamine tehnilistes skeemides nõuab täpseid ja usaldusväärseid meetodeid objektide eristamiseks ja määratlemiseks. Selle ülesande lahendamiseks on oluline valida võimekas objekti tuvastamise algoritm, mis suudab kiiresti ja tõhusalt tuvastada diagrammil esinevaid sümboleid. Käesolevas töös on sümbolite

tuvastamiseks valitud YOLOv8 mudel, mis on tuntud oma kõrge täpsuse ja kiiruse poolest ning on varasemalt teadustöödes tõestanud oma võimekust tuvastada vajalikke sümboleid P&ID skeemide digitaliseerimises.

Ultralyticsi arendatava YOLO seeria uusim ja kõrgeimal tasemel mudel YOLOv8 on märkimisväärne edasiminek objektituvastuse, pildiklassifikatsiooni ja segmenteerimise valdkonnas. Selle edu võib seostada mudeli muljetavaldava täpsuse ja kompaktsusega. YOLOv8 muudab eriti märkimisväärseks selle võime treenida ühe GPU abil, mis teeb mudeli kasutamise sobilikuks laiale arendajate ringile. [6]

Lisaks sisaldab YOLOv8 sisseehitatud andmete suurendamise tehnoloogiaid, mis aitavad treenimisprotsessi veelgi tõhustada. Selle mudeli paindlikkus ja võime kasutada eeltreenitud mudelit vähendab oluliselt treeningandmete vajadust, muutes selle antud projekti kontekstis suurepäraseks lahenduseks.

P&ID diagrammides võib esineda sadu erinevaid sümboleid, kuid nende kõigi sildistamine oleks käesoleva töö raames liialt ajakulukas ja ebaotstarbekas. Seetõttu on lõputöö eesmärgi saavutamiseks valitud konkreetne P&ID skeem, mille pealt on tehtud tuvastatavate sümbolite valim ja millel viiakse läbi rakenduse töövõime testid. (vt Joonis 1.1).

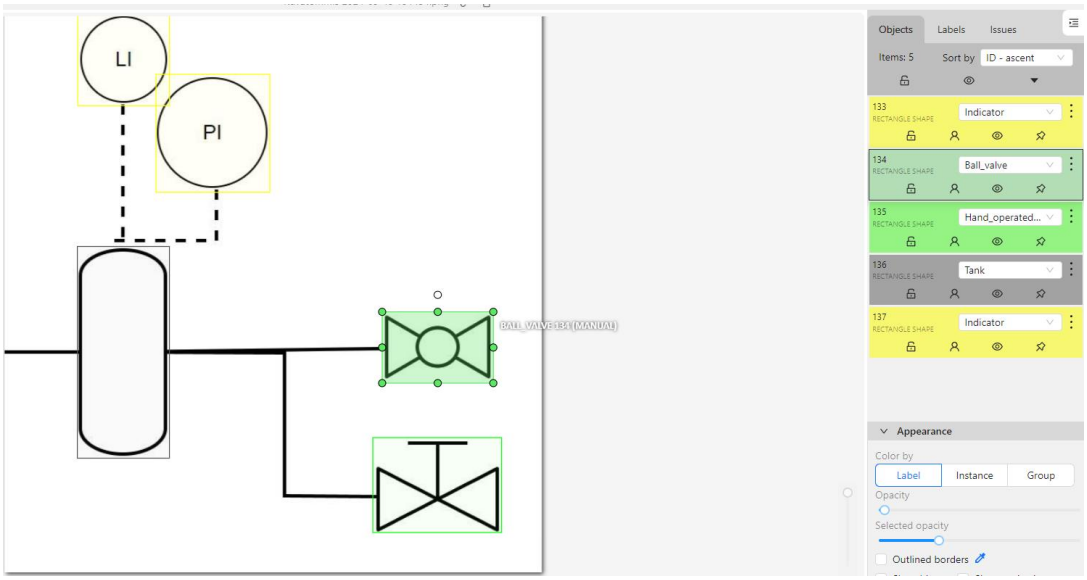


Joonis 1.1 P&ID joonis tuvastatavate sümbolitega [7]

1.2.1 Mudeli treenimine

Sümbolite tuvastamise mudeli väljatöötamine eeldab treeningandmete kogumit, mille põhjal on võimalik teha järeldusi sümbolite identifitseerimiseks ja klassifitseerimiseks. Antud lõputöö raames kasutati vabavara Draw.io, et luua 81 pilti protsesside ja seadmete diagrammidest ning vastavatest sümbolitest.

Loodud piltidel viidi läbi sümbolite käsitsi sildistamine, et tagada masinõppemudeli suutlikkus objektide korrektseks tuvastamiseks ja klassifitseerimiseks (vt Joonis 1.2). Protsessi käigus on kasutatud tasuta veebipõhist interaktiivset tööriista CVAT (Computer Vision Annotation Tool), mis võimaldab videote ja piltide täpset märgistamist arvutinägemise ülesannete jaoks [8].



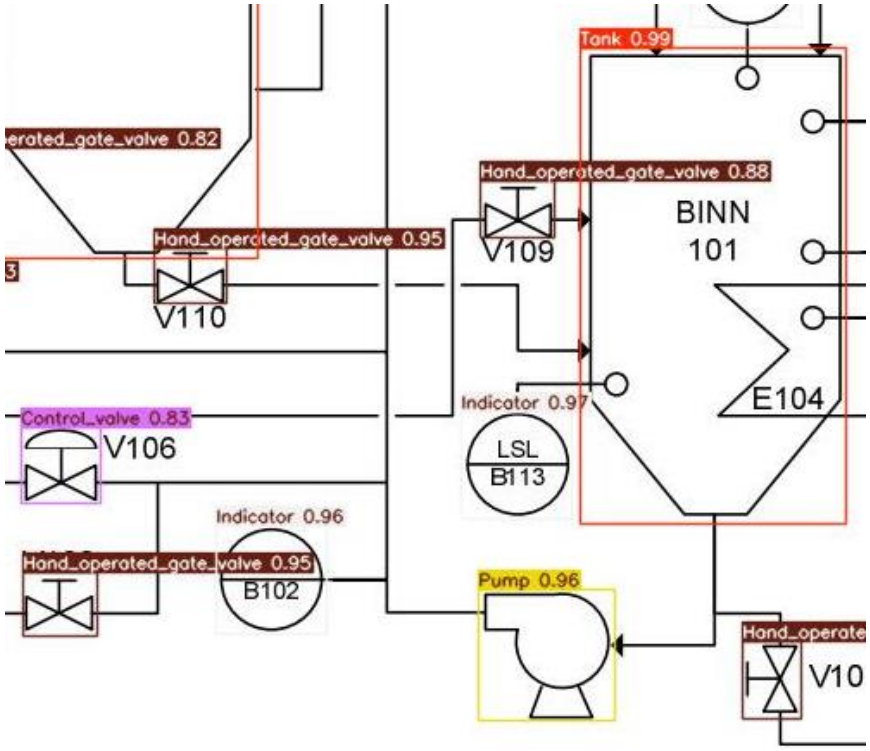
Joonis 1.2 Sümbolite märgistamine CVAT vabavara abil

P&ID joonistel kasutatavad sümbolid võivad olla omavahel väga sarnased, näiteks andurite sümbolid kujutavad endast sageli ringi, mille sees olevad tähed on erinevuste tuvastamise aluseks. Seetõttu klassifitseeritakse kõik ringikujulised sümbolid esmalt üldistatud „indikaatori" märgise alla. Esmase töötlemise järel toimub sümbolite eristamine vastavalt nende sisemistele tunnustele.

Esialgse mudeli testimisel, mida treeniti 81 pildi põhjal, selgus, et treeningandmete maht oli ebapiisav. Tulemused näitasid, et mudel suutis tuvastada vaid üksikuid sümboleid ning sedagi madala usaldusväärsusprotsendiga. Probleemi lahendamiseks rakendati YOLO algoritmi sisseehitatud andmete laiendamise (*data augmentation*) meetodit, mille käigus suurendati treeningpiltide hulka. Lisanduvad pildid saadi erinevate modifikatsioonide abil, näiteks horisontaalne pööramine, suuruse muutmine

ja piltide lõikamine.

Tehislikult suurendatud treeningandmestik osutus piisavaks, et luua mudel, mis suudab diagrammidel kõik sümbolid usaldusväärselt tuvastada. Lõplik mudel andis võrreldes algsega märkimisväärselt paremaid tulemusi, kinnitades andmete suurendamise efektiivsust sümbolite tuvastamisel (vt Joonis 1.3).



Joonis 1.3 Tuvastatud sümbolid

1.3 Teksti tuvastamine

Andurite sümbolite sarnasuse tõttu, kus eristamine põhineb ainult sümbolite sees olevatel tähtedel, on nende täpseks eristamiseks vajalik täiendav järeltöötlus, mille käigus tuvastatakse sümbolite sees olev tekst. Kõik eelnevalt tuvastatud sümbolid salvestatakse eraldi piltidena ja seejärel läbivad need optilise märgituvastuse (OCR) protsessi.

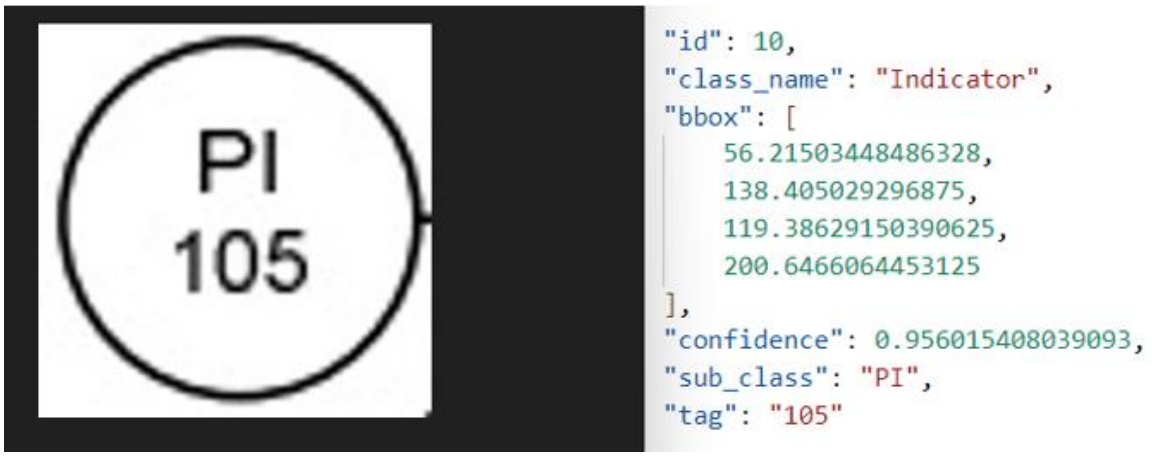
Töös kasutatava OCR raamistiku valikul võeti aluseks artikkel „Effectiveness of Modern Text Recognition Solutions and Tools for Common Data Sources“, kus autorid võrdlesid kahte tuntud ja avatud lähtekoodiga raamistikku: EasyOCR ning TesserOCR. Nende kahe raamistiku võrdluses on EasyOCR andnud paremaid tulemusi mürarohkete, moonutatud ja madalama kvaliteediga piltide puhul. Kuigi EasyOCR on oma töötlemiskiiruse poolest aeglasem, osutub see tõhusaks keeruliste ja raskesti

äratuntavate piltide puhul, kus täpsus on kiirusest olulisem. [9]

Indikaatorite sümbolite joonistelt eraldamise ja nende tekstide tuvastamise käigus esines märkimisväärsel hulgal juhtumeid, kus OCR ei suutnud sümbolite sees olevaid tähti korrektselt tuvastada. Täiendava analüüsi käigus selgus, et probleem ilmnes peamiselt väiksemõõduliste sümbolite puhul (50x50 pikslit või väiksem). Probleemi lahendamiseks otsustati suurendada kõigi salvestatud indikaatorite sümbolite mõõtmeid 240x240 pikslini. Suurendamise järel töötas OCR täpselt ning suutis kõigi sümbolite tekstid korrektselt tuvastada.

Peale sümboli ja teksti tuvastamist salvestatakse kõik sümbolid JSON faili koos järgnevate andmetega: (vt Joonis 1.4)

- sümboli ID
- sümboli kategooria
- sümboli koordinaadid joonisel
- usaldusväärsusprotsent
- kategooria alamklass
- sümboli silt



Joonis 1.4 Pilt eraldatud sümbolist ja salvestatud andmetest JSON-failis

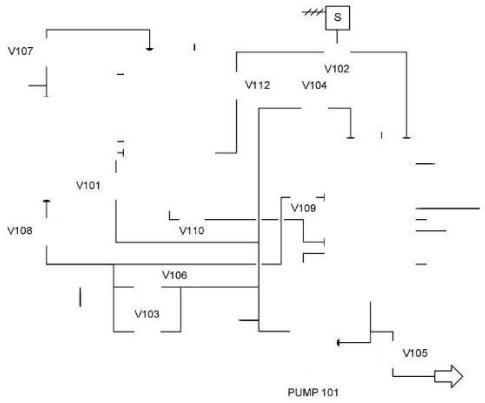
1.4 Ühenduste tuvastamine

Ühenduste tuvastamise protsess jaguneb kaheks põhietapiks. Esimeses etapis tuvastatakse pildil esinevad jooned ja salvestatakse nende koordinaadid edasiseks analüüsiks. Teises etapis analüüsitakse tuvastatud joonte põhjal sümbolite omavahelist seotust, et määrata kindlaks, millised sümbolid on joonte kaudu ühendatud.

1.4.1 Joonte markeerimine

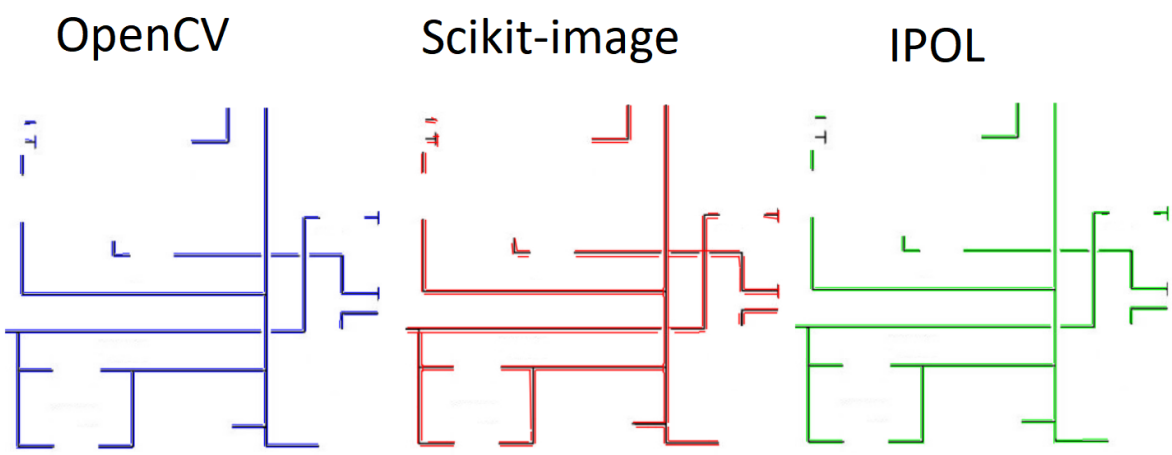
Pärast tutvumist artikliga "Object Detection in Design Diagrams with Machine Learning" võttis töö koostaja ühendust artikli autoritega, et saada täiendavat teavet nende kasutatud meetodite kohta [10]. Kari Rainio poolt edastatud lisamaterjalides väljatoodud lahendused inspireerisid ühenduste tuvastamiseks kasutama OpenCV Line Segment Detector'it (LSD). Selle meetodi rakendamine võimaldab tuvastada kõik pildil olevad jooned ning salvestada need JSON faili, kus saab andmeid hiljem ühenduse leidmiseks järeltöödelda.

LSD tuvastab joonisel kõik jooned, sealhulgas need, mis kuuluvad sümbolite kontuuridele. See võib aga põhjustada ekslikke tuvastusi ja raskendada ühendavate joonte korrektset identifitseerimist. Seetõttu on oluline enne joonte tuvastamise protsessi eemaldada kõik varasemalt tuvastatud sümbolid. Selleks asendatakse sümbolite koordinaatidele vastavad piirkonnad joonisel valge taustaga, mis võimaldab OpenCV-l keskenduda ainult joonte tuvastamisele ja vähendab sümbolitest tingitud müra (vt Joonis 1.5).



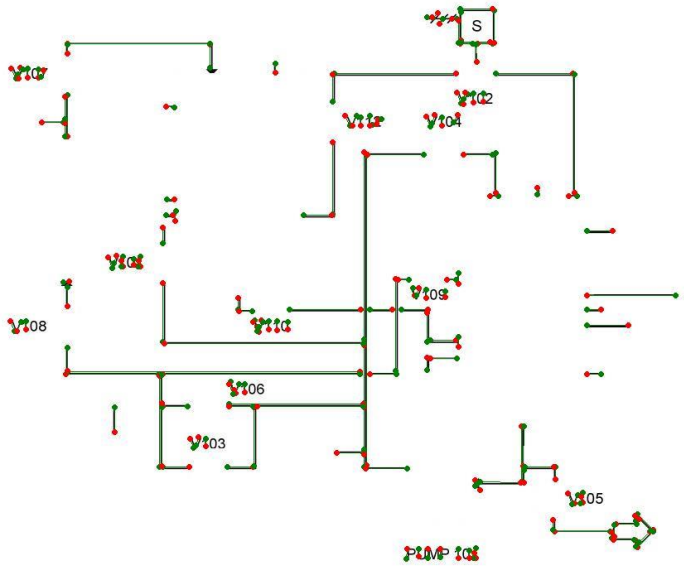
Joonis 1.5 Diagramm peale sümbolite eemaldamist

Töö käigus on lisaks eelmainitud OpenCV Line Segment Detectorile katsetatud ka Scikit-image [11] ja Pythoni jaoks modifitseeritud IPOL LSD tuvastusalgoritme [12]. Eesmärgiks oli võrrelda erinevate rakenduste suutlikkust testjoonisel. Kuigi kõik algoritmid andsid märkimisväärseid tulemusi, eristus OpenCV oma täpsusega. IPOL LSD detektor jättis kohati mõned jooned tuvastamata ning Scikit-image andis osaliselt ebatäpseid tulemusi (vt Joonis 1.6). Seevastu OpenCV LSD suutis identifitseerida kõik jooned täpselt ega jätnud ühtegi joont märkimata. Seega pakkus OpenCV antud testkeskkonnas parima lahenduse ning see osutus töö raames kõige sobivamaks valikuks.



Joonis 1.6 OpenCV, Scikit-image ja IPOL LSD tulemuste võrdlus

Peale sümbolite eemaldamist saab rakendada OpenCV joontetuvastuse algoritmi, mille tulemusena tuvastatakse kõik jooned ning nende algus- ja lõpp-punktid salvestatakse JSON-faili edasiseks analüüsiks. Visuaalseks kontrolliks on loodud pilt (vt Joonis 1.7), kus kõikide joonte algus- ja lõpp-punktid on tähistatud vastavalt roheliste ja punaste täppidega. Lisaks on automaatselt tuvastatud jooned ka joonisel olevatele tekstielementidele, mis aga ei sega soovitud eesmärgi saavutamist. Tuvastatud jooned ja nende täpsed koordinaadid on selgelt eristatavad ning analüüsi jaoks kasutatavad, tagades vajaliku täpsuse joonte eraldamiseks ülejäänud visuaalsest sisust.



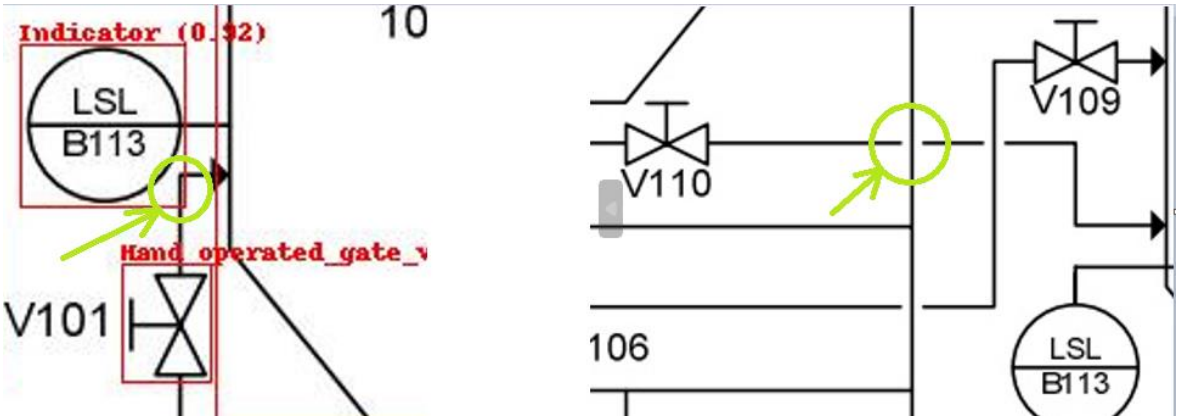
Joonis 1.7 OpenCV abil tuvastatud jooned

1.4.2 Ühenduste tuvastamise metoodika

Jooniste digitaliseerimise protsessis osutub ühenduste tuvastamine üheks keerukamaks ülesandeks. Parim viis seoste tuvastamiseks leitud joonte ja sümbolite vahel on kasutada joonte otspunktide ja sümbolite piirdekastide lähedusanalüüsi. Selleks on välja töötatud metoodika, mis määrab kindlaks, kas joone algus- või lõpp-punkt asub mõne sümboli serva lähedal, võimaldades seeläbi tuvastada sümbolite omavahelised ühendused. Lähedust hinnatakse väikese pikslilise tolerantsiga, mis aitab tagada täpsuse ning samal ajal säilitada paindlikkuse.

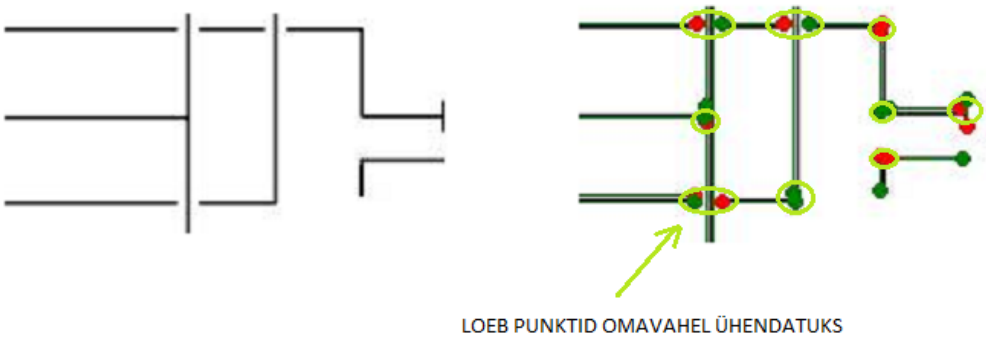
Kui süsteem tuvastab, et joon on sümboliga ühendatud, käivitub rekursiivne joonte ja sümbolite ühenduste otsingumeetod. See protsess võimaldab tuvastada kõik võimalikud ühendused sümbolite ja joonte vahel ning luua võrgustruktuuri, kus sümbolid võivad olla ühendatud ühe või mitme joone kaudu.

Probleemiks on aga ühendusjooned, mis kulgevad teistele sümbolitele liiga lähedalt või ristuvad omavahel (vt Joonis 1.8). Sellised ristumised ja lähedalt möödumised muudavad ühenduste täpse eristamise ja tuvastamise äärmiselt keeruliseks.



Joonis 1.8 Ühenduste tuvastamise murekohad märgitud roheliste ringidega

Omavahel ühenduvatest või ristuvatest joontest tingitud probleemide lahendamiseks on aga välja töötatud eraldi lahendus ehk täiendatud ühenduste tuvastamise meetodit funktsiooniga, mis kontrollib ka ühendusjoonte algus- ja lõpp-punktide omavahelist kaugust. Kui ühendusjoone algus- ja lõpp-punktide vaheline kaugus on väiksem kui 5 pikslit, loetakse jooned omavahel ühendatuks (vt Joonis 1.9). Arvestades, et 5 pikslit on jooniste kontekstis väga väike vahemaa, vähendab see meetod oluliselt vigade esinemist valeühenduste loomisel. Lisaks lahendab see lähenemine ka probleemi, kus mitu omavahel ühendatud joont ühes punktis ristuvad.



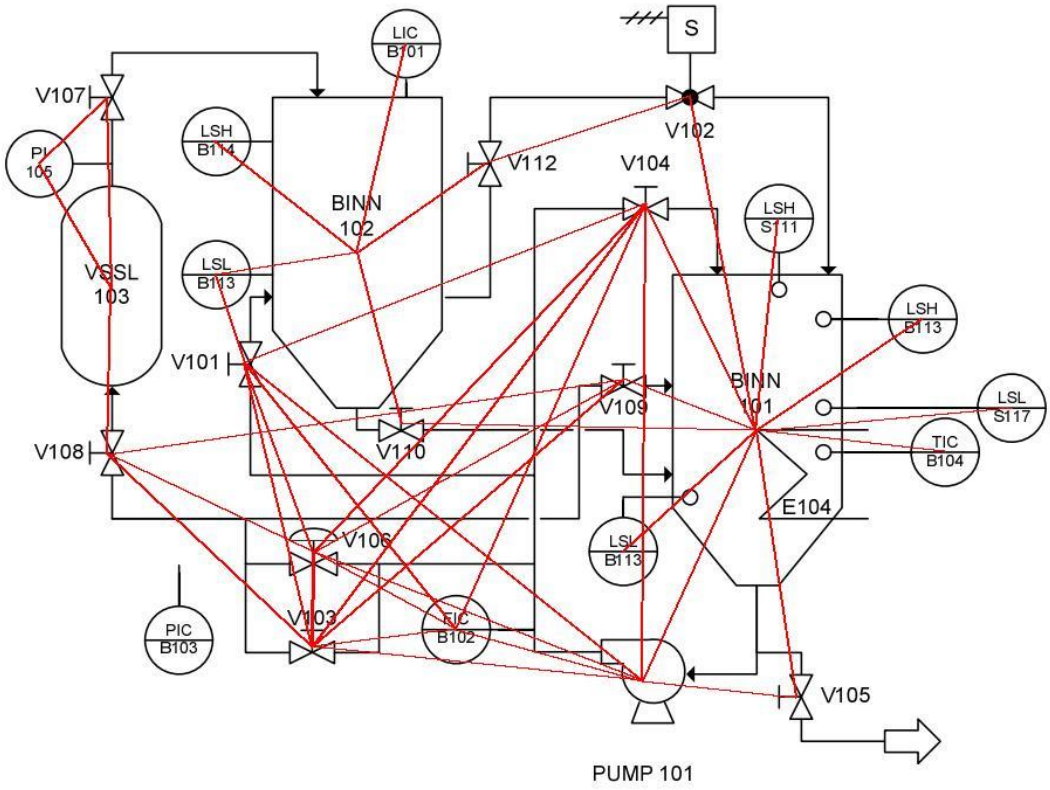
Joonis 1.9 Ristuvate ühendusjoonte sidumine

Teatud juhtudel võib esineda olukordi, kus ühendusjooned jäävad sümbolitele liiga lähedale (vt Joonis 1.8). Sellisel juhul lõigatakse sümbol koos ühendusjoonega välja, mistõttu võib kahe sümboli vaheline ühendus jääda loomata. Kuigi tegemist on veaga, on selliste erandolukordadega meetodi arendamise käigus arvestatud.

1.5 P&ID digitaliseerimise tulemus

Töös käsitletav P&ID digitaliseerimine on oluline komponent, kuid see ei ole töö peamine fookus. Suurem rõhk on asetatud veebipõhise HMI koodi genereerimisele, mis võimaldab kasutajatel hallata ja visualiseerida süsteemi tööd kasutajaliidese kaudu. P&ID digitaliseerimine on siiski kriitiline etapp, mis toetab HMI süsteemide täpset ja tõhusat loomist, pakkudes diagrammide töötlemise kiirendamist toetavaid automatiseeritud lahendusi.

Lõputöö raames loodud rakenduse praktiline tulemus on visuaalselt esitatud joonisel 1.10. Testimiseks kasutatud P&ID digitaliseerimise tulemuste põhjal on näha, et saavutatud täpsus on üldjoontes rahuldav ning see pakub olulist ajavõitu võrreldes käsitsi tehtava digitaliseerimisega. Visualiseeritud pildil võib märgata, et kuigi tulemus on piisavalt täpne, esineb siiski mõningaid väiksemaid vigu. Puudu on kaks ühendust ning loodud üks üleliigne ja ekslik ühendus. Sellised vead on aga hõlpsasti parandatavad ja nende käsitsi korrigeerimine võtab oluliselt vähem aega kui traditsiooniline ehk täielikult manuaalne diagrammi loomise protsess.



Joonis 1.10 P&ID digitaliseerimisel loodud sümbolite ühendusvõrgustik

P&ID diagrammide digitaliseerimine on tehnoloogiline väljakutse, mille keerukust rõhutab ka Microsofti tarkvarainsener Oscar Fimbres. Tema hinnangul võtab P&ID lehtede digitaliseerimise lahenduse arendamine aega umbes 3–6 kuud ning ka siis võivad esineda vead. Diagrammide keerukus tuleneb sümbolite ja joonte tihedatest ristumistest ja ühendustest, mis muudab nende täpse tõlgendamise raskeks.[2]

Kokkuvõttes on P&ID digitaliseerimise protsessi peamine pluss ajaline võit ja vigade vähendamine. Kuigi täielikku automatiseerimist ei ole võimalik veel saavutada, parandavad käesolevas töös välja töötatud meetodid oluliselt digitaliseerimise efektiivsust. See loob head eeldused, et edasine HMI koodi genereerimine toimub digitaliseeritud P&ID diagrammidele tuginedes kiiremini ja täpsemini.

Testskeemi analüüsi tulemused näitasid, et õigesti tuvastati 95% sümbolitest, 92,86% joontest ja 100% tähemärkidest. Tulemused viitavad meetodi efektiivsusele, kuigi tuleb arvestada, et antud projekti raames kasutati piiratud hulka erinevaid sümboleid ning skeemide keerukusaste oli madal. Sellest hoolimata olid tulemused piisavad, et kinnitada hüpoteesi ja tõestada välja pakutud meetodite sobivust antud kontekstis.

2. GRAAFILINE KASUTAJALIIDES

Teine samm on graafilise kasutajaliidese loomine (GUI) laetud skeemi redigeerimiseks. See on vajalik lõppväljundi täpsuse, kohandamise ja kasutatavuse tagamiseks. P&ID teisendamisel JSON-formaati võib automaatne pildituvastusprotsess põhjustada vigu, tõlgendades teatud komponente või nendega seonduvaid andmeid valesti. P&ID diagrammid võivad olla keerulised, sisaldades erineva kvaliteediga sümboleid, mitmetähenduslikke silte või kattuvaid elemente, mida tuvastusalgoritm ei pruugi täielikult ära tunda või õigesti kategoriseerida.

GUI võimaldab kasutajatel automaatselt genereeritud JSON-andmeid kontrollida ja täpsustada, tagades, et komponendid vastavad algsele P&ID eesmärgile. See võimaldab kasutajal lisada puuduvaid ühendusi ja parandada vigu, näiteks valesti tuvastatud klappe või andureid. Samuti annab see kasutajale paindlikkuse kohandada skeemi elemente vastavalt konkreetsetele projektinõuetele või eelistustele enne lõpliku HMI-koodi loomist. Alati ei ole kõik süsteemi toimimise detailid üksikasjalikult skeemil kuvatud ning mõned neist võivad olla selgitatud juhendis. Kuivõrd tegemist ei ole tehisarul põhineva rakendusega, mis suudaks iseseisvalt juhenditest infomatsiooni koguda, peab kasutaja seda ise tegema.

Lisaks pakub GUI kasutajasõbralikku viisi skeemi muutmiseks või täiustamiseks. Kasutajad võivad parema visualiseerimise huvides soovida muuta paigutust, lisada täiendavaid funktsioone või diagrammi osi ümber korraldada. See samm annab lõppkokkuvõttes suurema täpsuse reaalse süsteemi ja loodud HMI vahel, tagades täpse ning kasutajasõbralik lõppväljundi.

2.1 Tarkvarateekide võrdlus

GUI loomisel on suurimaks otsustuskohaks sobiva tarkvarateegi valik. Teegi valikul lähtutakse seatud nõuetest ja võimalustest, mis on antud töö puhu järgmised:

- vabavaraline lahendus;
- sümbolite redigeerimise võimalused;
- täisnurksete ortogonaalsete ühenduste loomine;
- kliendipoolne teek, mis töötab ka võrguühenduseta;
- ühilduvus erinevate taustsüsteemi tehnoloogiatega.

Saadaval on väga palju erinevaid diagrammide loomiseks mõeldud tarkvarateeke. Jordi Cabot on oma artiklis toonud välja 23 tarkvarateeki, mis aitavad luua rakendusi diagrammide joonistamiseks [13]. Antud töö jaoks valiti välja 4 teeki, mis võiksid selle kontekstis kõige paremini sobituda. Järgnevalt on võrreldud nende eeliseid ja puudusi ning tehtud valik, millist teeki töös kasutada.

2.1.1 JointJS

JointJS on modulaarse arhitektuuriga avatud lähtekoodiga diagrammide teek, mille kommertslaiendus JointJS+ lisab täiendavaid funktsioone ja kasutajaliidese komponente. JointJS on kirjutatud JavaScriptis, toetab kaasaegseid veebistandardeid ning sobib interaktiivsete veebirakenduste arendamiseks.

Eelised:

- Kohandamine: võimaldab laialdast diagrammi elementide kohandamist, mis on oluline P&ID komponentide täpselt kuvamiseks [14].
- Raamistike integratsioon: hea ühilduvus populaarsete *front-end* raamistikega nagu React, Angular ja Vue.js [14].
- Ulatuslik dokumentatsioon ja tugi: pakub põhjalikku dokumentatsiooni ja toetavat kogukonda [14].

Puudused:

- Järsk õppimiskõver: teegi keerukus võib tuua kaasa märkimisväärse õppimisperioodi. [15]
- JointJS+ maksumus: täiustatud funktsioonide kasutamiseks tuleb osta kommertslitsents. [16]

Litsents:

- JointJS: Mozilla Public License Version 2.0 (MPL 2.0) [17].
- JointJS+: Kommertslitsents [17].

2.1.2 mxGraph

MxGraph on JavaScripti teek, mis annab võimaluse luua interaktiivseid diagramme brauseris ilma *plugin*'aid kasutamata. Antud teegi põhjal on loodud populaarne diagrammide joonestamise rakendus Diagrams.net (endine Draw.io).

Eelised:

- Laiaulatuslik funktsionaalsus: pakub laia valikut sisseehitatud funktsioone nagu lohistamine, suurendamine ja automaatsed paigutusalgoritmid [17].
- Brauseri ühilduvus: toetab kõiki populaarsemaid brausereid ja on täielikult kliendipoolne [17].
- Laiendatavus: võimaldab kohandamist ja laiendamist projekti spetsiifiliste vajaduste jaoks, näiteks kohandatud kujundid [17].

- Hind: litsentseeritud Apache License 2.0 alusel, tasuta kasutamiseks ka kommertsprojektides [18].

Puudused:

- Järsk õppimiskõver: API keerukus võib nõuda märkimisväärset aega õppimiseks.
- Aktiivse hoolduse puudumine: ametlik arendamine on lõpetatud. Uuenduste puudumine võib tulevikus tekitada probleeme ühilduvuse osas [18].
- Piiratud kogukonna tugi: projekti arhiveerimise tõttu on aktiivne kogukonna kaasamine vähenenud [19].

Litsents:

- Apache License 2.0: lubab tasuta kasutamist, muutmist ja levitamist [18].

2.1.3 Diagram-js

Diagram-js on diagrammide tööriistakomplekt, mida kasutab bpmn.io BPMN (Business Process Modeling Notation) diagrammide renderdamiseks. See on kavandatud paindlikkust ja modulaarset arhitektuuri silmas pidades. [20]

Eelised [20]:

- Paindlikkus ja modulaarne arhitektuur: võimaldab arendajatel lisada või asendada komponente vastavalt vajadusele [20].
- Aktiivne arendus: aktiivne avatud lähtekoodiga kogukond, rakendust haldab Camunda [20].
- *Plugin*'ate süsteem: toetab *plugin*'aid ja laiendusi funktsionaalsuse suurendamiseks [20].

Puudused:

- Nõuab kohandatud arendust: ei paku valmis tuge P&ID-spetsiifilistele elementidele.
- Kohandamise õppimiskõver: põhikasutus on lihtne, kuid kohandamine nõuab Diagram-jsi sisemise arhitektuuri mõistmist. [21]

Litsents:

- MIT litsents: lubab vaba kasutamist nii kommerts- kui mittekommertsprojektides [20].

2.1.4 Cytoscape.js

Cytoscape.js on avatud lähtekoodiga teek, mis on kirjutatud JavaScriptis. See on loodud keerukate võrkude ja graafide visualiseerimiseks ja analüüsimiseks veebirakendustes.

Eelised:

- Kõrge jõudlus: optimeeritud suurte graafide jaoks, suudab renderdada tuhandeid sõlmi ja servi [22].
- Laialdane paigutuste komplekt: pakub erinevaid automaatseid paigutusalgoritme [22].
- Põhjalik API ja dokumentatsioon: pakub põhjalikku API-d koos detailse dokumentatsiooni ja näidetega [22].
- Aktiivne kogukond ja tugi: regulaarne uuendamine [22].

Puudused:

- Pole suunatud P&ID-le: eelkõige loodud võrgugraafide jaoks, puudub sisseehitatud tugi P&ID sümbolitele [22].
- Keerukas kohandamine: enda standarditele vastavaks muutmine võib olla keeruline [22].

Litsents:

- MIT litsents: lubab paindlikku kasutamist nii kommerts- kui mittekommertsprojektides [22].

2.1.5 Töös kasutatava teegi valik

P&ID skeemide redigeerimiseks mõeldud kasutajaliidese arendamisel on saadaval mitmeid erinevaid teeke, mis kõik on eesmärgi saavutamiseks sobilikud. Valik sõltub enamasti arendaja isiklikest eelistustest, projekti konkreetsetest nõuetest ja varasemast kogemusest.

MxGraph valiti selle projekti jaoks mitmel olulisel põhjusel, mis vastasid nii projekti vajadustele kui ka arendaja eelistustele:

- Paindlikkus ja kohandatavus: mxGraph pakub ulatuslikke võimalusi diagrammi elementide kohandamiseks. See võimaldab luua kohandatud sümboleid, stiile ja käitumismustreid, mis on kriitilise tähtsusega P&ID skeemide täpseks esitamiseks.

- Kliendipoolne renderdamine: mxGraph töötab täielikult kliendipoolselt, mis lihtsustab rakenduse juurutamist ja parandab jõudlust. See välistab vajaduse serveripoolse töötlemise järele [17].
- Rikkalik funktsionaalsus: teek pakub mitmesuguseid sisseehitatud funktsioone nagu lohistamine, suurendamine, ühenduste loomine ja automaatsed paigutusalgoritmid. Need funktsioonid kiirendavad arendusprotsessi ja parandavad kasutajakogemust.
- Dokumentatsioon ja kogukonna ressursid: kuigi mxGraphi aktiivne arendus on peatatud, on olemas ulatuslik dokumentatsioon ja suur hulk näiteid. See võimaldab arendajatel teeki efektiivselt kasutada ja leida lahendusi võimalikele probleemidele.
- Litsentsimine ja kulud: mxGraph on avatud lähtekoodiga ja litsentseeritud Apache License 2.0 alusel, mis lubab tasuta kasutamist ka kommertsprojektides. See on oluline eelis eelarvepiirangutega projektide jaoks.

Lõppkokkuvõttes ei ole ühte ainuõiget valikut, vaid oluline on valida tööriist, mis vastab kõige paremini projekti spetsiifilistele vajadustele ja arendaja tugevustele. MxGraph võimaldas saavutada projekti eesmärgid efektiivselt ja tõhusalt, pakkudes samal ajal platvormi, mis sobis arendaja isiklike eelistustega.

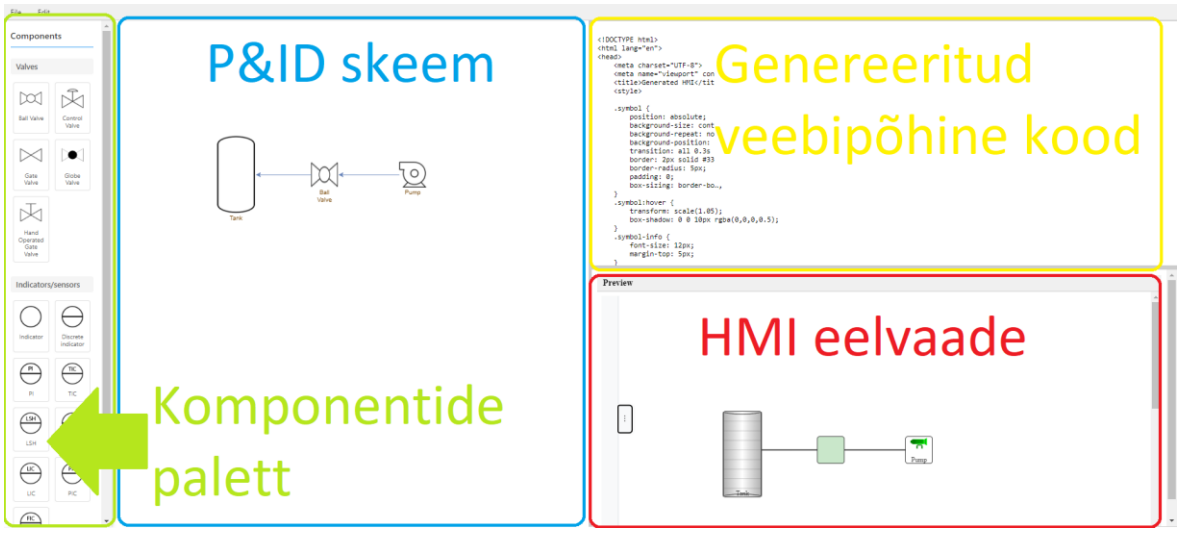
2.2 GUI loomine

Visuaalsel kasutajaliidesel kuvatav P&ID skeem esindab andmete viimast esitlust enne nende muutmist HMI veebipõhiseks koodiks. Arvestades, et HMI eesmärk on pakkuda võimalikult kasutajasõbralikku kogemust, võivad tekkida olukorrad, kus P&ID skeemi otsene konverteerimine HMI-ks ei pruugi olla optimaalne lahendus. Kuigi veebipõhises koodis on võimalik hiljem teha vajalikke muudatusi, on käesoleva töö eesmärk automatiseerida veebipõhise koodi genereerimist, mis kiirendab tööprotsessi ja vähendab käsitsi kodeerimise vajadust. Seetõttu on oluline, et kasutajal oleks võimalus teha vajalikud muudatused otse GUI kaudu.

GUI on struktureeritud neljaks põhikomponendiks (vt Joonis 2.1):

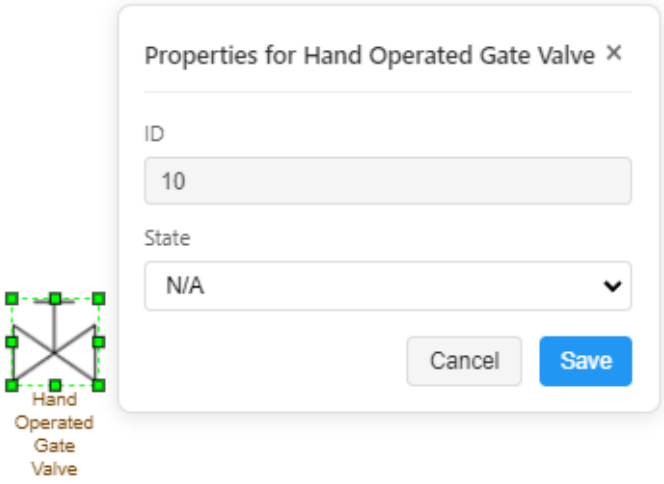
- Komponentide palett: sisaldab erinevaid P&ID sümboleid ja komponente, mida kasutaja saab skeemile lisada.
- P&ID skeem: tööala, kus kasutaja koostab ja redigeerib skeemi, paigutades komponente ja luues ühendusi.

- Genereeritud veebipõhine kood: näitab loodud koodi, mis vastab praegusele skeemile.
- HMI eelvaade: pakub reaalajas visuaalset eelvaadet sellest, kuidas HMI kasutajaliides pärast koodi rakendamist välja näeb.



Joonis 2.1 GUI ülevaade

Kuna teatud komponendid võivad omada erinevaid atribuute või väärtusi, on redigeerimise lihtsustamiseks loodud omaduste aken. See võimaldab kasutajal määrata komponentide spetsiifilisi omadusi otse GUI-s, ilma vajaduseta hiljem HTML-koodi käsitsi muuta. Näiteks saab kasutaja seadistada mahutite maksimaalsed ja minimaalsed väärtused või määrata klappide vaikeseaded (vt Joonis 2.2).



Joonis 2.2 Komponendi omaduste aken

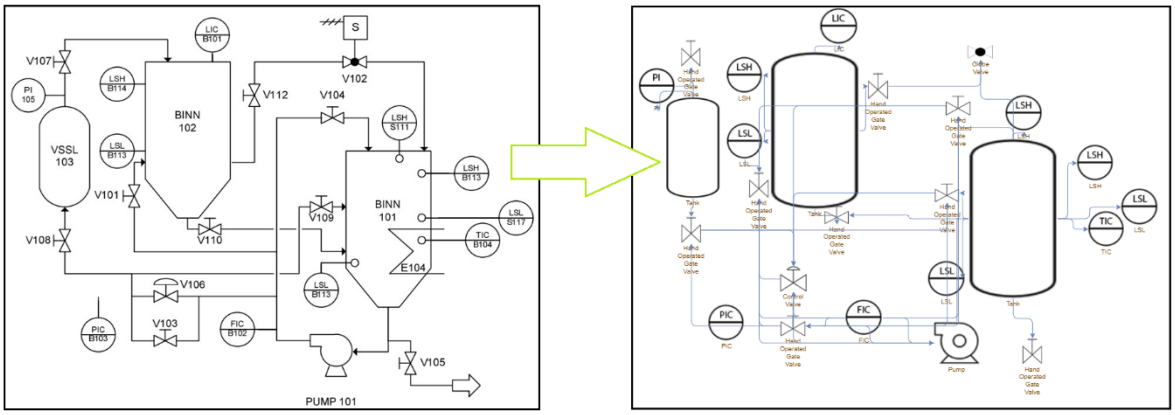
2.3 Digitaalse skeemi loomine

Kasutajaliidese üks ülesannetest on esimeses peatükis loodud pildituvastusmudeli abil kogutud andmete visualiseerimine. Need andmed hõlmavad sümbolite kuju, asukohta, suurust ning omavahelisi ühendusi. Rakenduse eesmärk on genereerida nende andmete põhjal digitaalne P&ID, mis on võimalikult sarnane originaalskeemile.

Rakendusse on eelnevalt käsitsi sisestatud andmekogu, mis sisaldab iga sümboliga seotud pildimaterjali. Sisendandmete alusel määrab rakendus iga sümboli täpse asukoha ja mõõtmed ning esitab need sobivalt eelmääratletud visuaalsete standardite järgi.

Sümbolite vahelised ühendusjooned genereeritakse mxGraph'i sisseehitatud funktsioonide abil, mis tagavad ühendusjoonte korrapärase ja loogilise paigutuse. Ühendusjooned kujutatakse täisnurksete joontena, et säilitada korrapärasus ja lihtsustada diagrammi tõlgendamist.

Töös kasutatav testdiagramm laaditakse rakendusse .jpg-vormingus failina, mille põhjal genereeritakse digitaalne skeem. Kasutajal on võimalus loodud diagrammi vastavalt vajadusele redigeerida (vt Joonis 2.3). Nagu varasemates peatükkides kirjeldatud, on pildituvastusmudel eksperimentaalne ja selle treenitusaste piiratud. Seetõttu võib digitaalses skeemis esineda vigu, näiteks puududa ühendused või sümbolid. Siiski on nende parandamine kasutajapoolse sekkumisega enamasti vähemahukas.



Joonis 2.3 Pildituvastusmudeli abil P&ID automaatne digitaliseerimine

Antud näite põhjal, kus mudel suutis korrektselt tuvastada 95% sümbolitest ja 92,86% joontest, kuid ei määranud õigesti ühenduste liikumissuundi, on korrektse diagrammi koostamiseks vajalik kõigi ühenduste käsitsi uuesti lisamine. Lisaks sellele peab kasutaja täiendavalt lisama ühe puuduva sümboli. Vaatamata nendele piirangutele on

diagrammide digitaliseerimise protsess toonud kaasa märkimisväärse ajavõidu võrreldes traditsioonilise manuaalse meetodiga.

Kangi, Lee ja Baeki uurimuses [4] rõhutati, et automatiseeritud digitaliseerimise rakendamine võimaldas saavutada ajavõidu kuni 93,75%. Kuigi käesolevas projektis nii suurt efektiivsuse kasvu ei saavutatud, näitavad tulemused, et rakenduse edasine arendamine võiks anda sarnaseid tulemusi. Selline potentsiaalne ajavõit suurendaks lahenduse väärtust ja kasutatavust tööstuspraktikas.

3. VEEBIPÕHISE HMI KOODI LOOMINE

See peatükk sisaldab HMI veebipõhise koodi genereerimise analüüsi. Koodigeneratsiooni kasutamine parandab koodi kvaliteeti, võimaldab vigade süsteemset parandamist ja suurendab arenduse tootlikkust.

Rakenduse antud osa on loodud Pythoni mikroveebiraamistiku Flask abil, mis on tuntud oma lihtsuse ja paindlikkuse poolest. See vastutab veebiserveri seadistamise, lõpp-punkti päringute käsitlemise ning sisendandmete põhjal HTML-sisu genereerimise eest.

Antud peatükis kasutatavad tehnoloogiad:

- Flask: Pythoni mikroveebiraamistik
- HTML ja CSS: HMI komponentide struktureerimiseks ja kujundamiseks
- JavaScript: võimaldab interaktiivsust ja dünaamilisi värskendusi HMI-s
- SVG: skaleeritav vektorgraafika, mida kasutatakse dünaamiliste komponentide renderdamiseks

3.1 Sisendandmed

Sisendandmed on HMI loomise aluseks. Antud töös on andmed JSON-vormingus ning koosnevad kahest põhikomponendist: sümbolid ja ühendused. Need struktureeritud andmed sisaldavad üksikasjalikku teavet tööstussüsteemi komponentide ja nende omavaheliste ühenduste kohta.

3.1.1 Sümbolite struktuur ja atribuudid

Sümbolid esindavad skeemi erinevaid elemente nagu näiteks klapid, pumbad ja andurid. Igal sümbolil on mitmeid olulisi atribuute (vt Joonis 3.1):

- ID: unikaalne identifikaator, mis võimaldab sümbolit tuvastada ja luua seoseid teiste elementidega.
- Klassi nimi: tüüp või kategooria, näiteks "valve", "pump" või "pressure indicator controller". See määrab, kuidas sümbolit visualiseeritakse ja millised interaktsioonid on võimalikud.
- Piirav kast (bbox): koordinaatide massiiv [left, top, right, bottom], mis määrab sümboli positsiooni ja mõõtmed.
- Omadused: tüüpspetsiifilised atribuudid, näiteks klapi olek ("*state*") või anduri näidud ("*pressure*", "*temperature*").

```
{
  "id": "8",
  "class_name": "Hand Operated Gate Valve",
  "bbox": [
    598,
    360,
    648,
    410
  ],
  "properties": {
    "state": "N/A"
  }
},
```


Joonis 3.1 Sümbolite sisendandmed

3.1.2 Ühendused

Ühendused esindavad sümbolite vahelisi seoseid, näiteks füüsilisi ühendusi (torud) või loogilisi seoseid. Ühenduste atribuudid hõlmavad (vt Joonis 3.2):

- ID: unikaalne identifikaator.
- Allikas ja sihtkoht: sümbolite ID-d, mis määratlevad ühenduse algus- ja lõpp-punkti.
- Punktid: koordinaatide massiiv, mis määrab ühenduse visuaalse tee.


```
{
  "id": "13",
  "source": "12",
  "target": "7",
  "points": [
    {
      "x": 282,
      "y": 406.5
    },
    {
      "x": 347,
      "y": 406.5
    }
  ]
}
```


Joonis 3.2 Füüsiliste ühenduste sisendandmed

3.2 Koodi genereerimise loogika

HMI koodi genereerimise loogika keskendub sisendandmete korrapärasele muutmisele, mis looks interaktiivse ja kasutajasõbraliku kasutajaliidese. See protsess hõlmab mitmeid samme.

Antud projektis sisaldab iga genereeritud HMI kood vaikimisi standard HTML-struktuuri,

mis omakorda sisaldab:

- DOCTYPE deklaratsiooni - <!DOCTYPE html>, mis täpsustab, et kasutusel on HTML5.
- HTML silt - <html> element, mis on HTML-dokumendi algus.
- HTML päis – <head> element, mis sisaldab metaandmeid.

  Päisesse on lisatud CSS stiilid, et määratleda erinevate elementide välimus.
- HTML keha - <body> element, mis sisaldab veebilehe sisu.

  Keha osas on lisatud HMI-s kuvatud komponentide omadused ja JavaScript funktsioonid.

3.2.1 Komponentide tuvastamine

Esimese sammuna tuvastatakse skeemi alusel erinevad komponendid, mida tuleb HMI-s esitada. See võib hõlmata klappe, indikaatoreid, mahuteid jne. Iga komponendi jaoks määratakse sisendandmete põhjal selle omadused nagu asukoht, suurus, värv ja kõik dünaamilised atribuudid, mis võivad kasutaja interaktsiooni või andmesisestuse põhjal muutuda (vt Joonis 3.3).

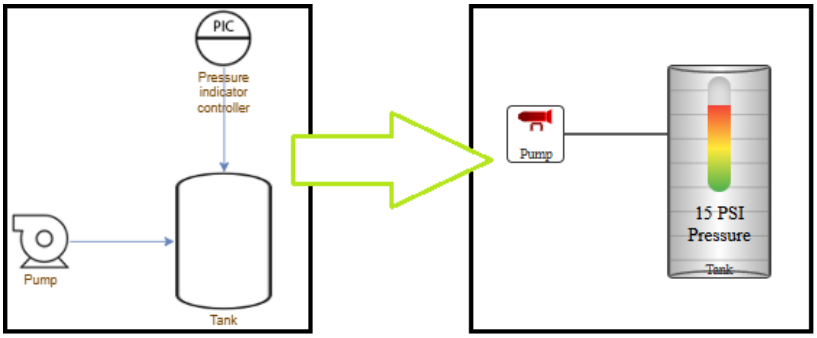
```
<div id="symbol-25"
     class="symbol"
     data-class-name="Control Valve"
     data-state="open"
     data-properties='{"id": "25", "type": "Control Valve", "state": "open"}'
     style="
left: 299.6236877441406px;
top: 479.18212890625px;
width: 45.8900146484375px;
height: 44.123779296875px;
background-image: url('images/symbols/control_valve.png');

</div>
```

Joonis 3.3 Komponentide määramine

3.2.2 Ühenduste ja suhete kaardistamine

Kui sümbolid on laetud, kontrollitakse omavahelisi ühendusi. Sümbolid võivad olla omavahel ühenduses füüsiliste ühendustena (näiteks torud), kuid tegu võib olla ka loogiliste ühendustega andurite ja kontrolleritega. Selleks, et HMI näeks välja lihtsam ja selgem, on töö autor otsustanud, et kontrollerite visuaal peaks asuma selle

komponendi sees, millega ta seotud on (vt Joonis 3.4).



Joonis 3.4 Komponentide seosed

Eesmärgi saavutamiseks luuakse pärimise suhted. Süsteem tuvastab ülemklassi sümbolid (nt paak) ja nendega seotud alamklassi sümbolid (nt paagiga ühendatud andurid). See suhe on oluline positsioneerimise ja interaktsiooni loogika jaoks.

Süsteem arvutab iga sümboli jaoks vajalikud positsioonid ja mõõtmed, et vältida kattumist ja tagada kooskõlaline paigutus. Ülemklassi sümbolid kohandavad oma mõõtmeid, et mahutada endasse alamklassi sümboleid, säilitades seeläbi visuaalse hierarhia.

3.2.3 Visuaalide loomine

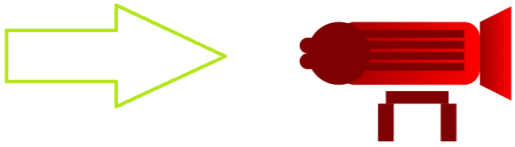
Iga sümbol muudetakse HTML elemendiks koos vajalike andmetega (vt Joonis 3.3). Sümbolitele parema visuaalse välimuse andmiseks on töös kasutatud kahte erinevat võimalust.

Esimene variant on lisada sümbolile pildiformaadis taust, kasutades HTML-koodi nagu näiteks „background-image: "url('images/symbols/ball_valve.png')“. Sellisel juhul peab olema igale sümbolile eelnevalt joonistatud pilt ning koodis lisatud sellele õige asukoha aadress. Sel otstarbel on otsene ja laialdaselt toetatud PNG-formaadi kasutamine, kuid sellel on ka teatud puudused. PNG eelisteks on kasutamise lihtsus, kvaliteetne visuaalne esitlus ja ühilduvus kõikide brauseritega. Samas ei ole PNG-failid skaleeritavad ehk nende suurendamine vähendab pildi kvaliteeti ning valmis pildi muutmine (näiteks värvi või mõõtmete osas) nõuab uue pildi loomist läbi pilditöötlustarkvara. Lisaks võib kõrge resolutsiooniga PNG-formaadis failide suur maht aeglustada veebilehe laadimist.

Teine võimalus on kasutada SVG (Scalable Vector Graphics) graafikat (vt Joonis 3.5). SVG puhul on võimalik joonistada tekstipõhiselt erinevaid kujundeid, muuta nende suurust, värve jne. SVG eelisteks on selle skaleeritavus, mis tagab, et sümboli kvaliteet säilib olenemata suurusest ja seda on võimalik muuta otse koodis. Vektorgraafika failid

on tihti väiksema mahuga, mis parendab veebilehe kiirust. Kujundite joonistamiseks on võimalik kasutada vabavara, näiteks Inkscape'i [23], kus saab hõlpsasti luua ja salvestada tekstipõhiseid graafilisi elemente. Käesolevas töös on autor esmalt loonud Inkspace'i kasutades komponentidest visuaalsed kujutised ning seejärel OpenAI mudeli GPT-4o abil kohandanud selle vektorkoodi. SVG kasutamine võib algajate jaoks osutuda PNG-formaadist keerukamaks ning väga detailsete või fotolaadsete sümbolite puhul on selle kasutusvõimalused piiratud.

```
<svg width="220.0" height="160.0" viewBox="0 0 75 55">
    <defs>
        <linearGradient id="pumpGradient-3" x1="0" x2="1" y1="0" y2="0">
            <stop offset="0%" class="pump-gradient-off" />
            <stop offset="100%" class="pump-gradient-off-end" />
        </linearGradient>
    </defs>

    <rect x="15" y="10" width="45" height="20" rx="5" ry="5"
        fill="url(#pumpGradient-3)" />

    <rect x="20" y="12.5" width="35" height="2.5" class="pump-dark-off" />
    <rect x="20" y="16" width="35" height="2.5" class="pump-dark-off" />
    <rect x="20" y="19.5" width="35" height="2.5" class="pump-dark-off" />
    <rect x="20" y="23" width="35" height="2.5" class="pump-dark-off" />

    <rect x="5" y="15" width="10" height="10"
        fill="url(#pumpGradient-3)" />
    <circle cx="5" cy="17.5" r="2.5" class="pump-dark-off" />
    <circle cx="5" cy="22.5" r="2.5" class="pump-dark-off" />

    <circle cx="15" cy="20" r="10"
        fill="url(#pumpGradient-3)"
        stroke-width="1"
        class="pump-dark-off" />

    <polygon points="60,10 70,5 70,35 60,30"
        fill="url(#pumpGradient-3)" />

    <rect x="30" y="32" width="20" height="4" class="pump-dark-off" />

    <rect x="27.5" y="36" width="5" height="12" class="pump-dark-off" />
    <rect x="47.5" y="36" width="5" height="12" class="pump-dark-off" />
</svg>
```

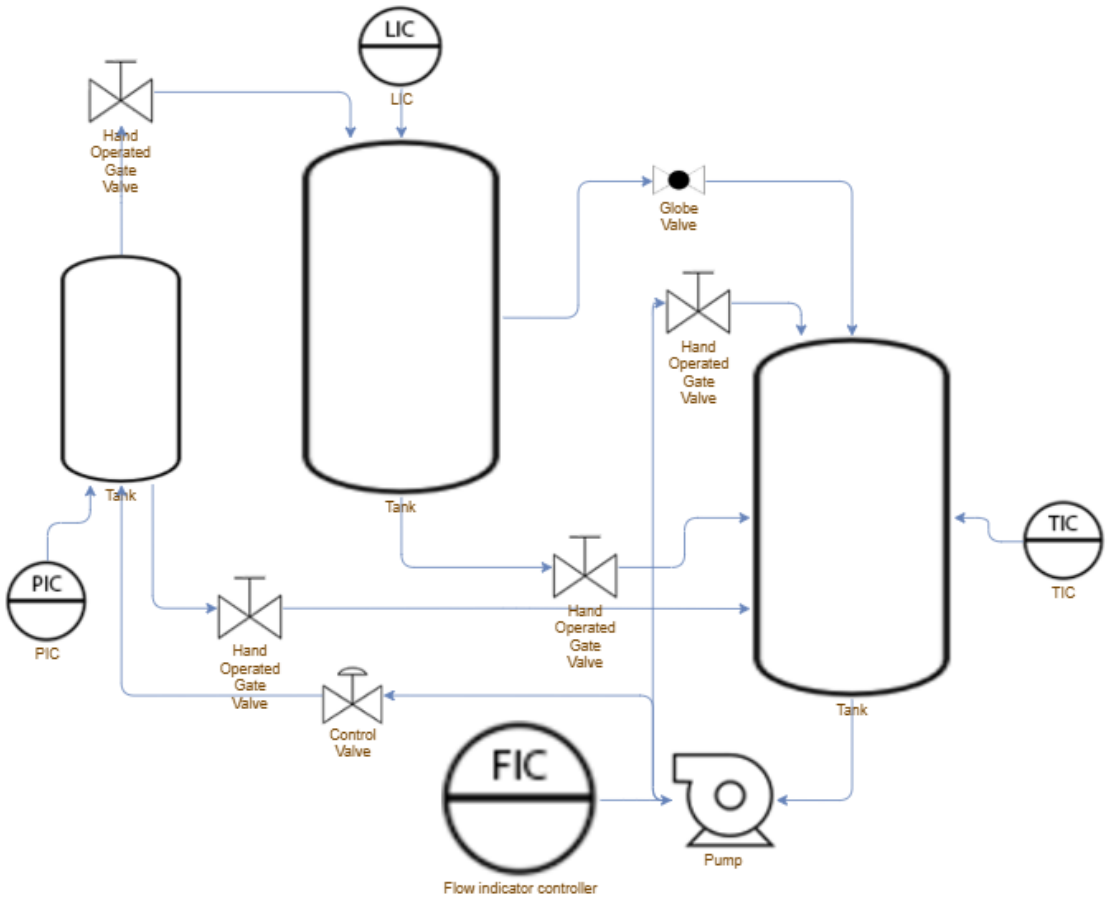


Joonis 3.5 Pumba sümbol SVG graafikaga

3.3 Koodi genereerimise tulemus

Käesoleva töö eesmärk on luua terviklik inimese ja masina liides (HMI). Selle saavutamiseks testitakse töös kasutatud piiratud hulka sümboleid, et hinnata nende põhjal automaatselt genereeritud HMI funktsionaalsust. Arvestades, et tegemist on eksperimentaalse projektiga ning rakenduses on kasutatud minimaalset valikut P&ID (Process and Instrumentation Diagram) komponente, on eesmärgiks luua HMI, mis põhineb rakendusse sisestatud komponentidel.

Testimise eesmärgil kasutatakse töö alguses kirjeldatud näidis P&ID-d, mis esmalt digitaliseeritakse ja seejärel kohandatakse kasutajaliidese (GUI) kaudu vastavalt kasutaja vajadustele ja eelistustele (vt Joonis 3.6). See protsess võimaldab hinnata loodud HMI vastavust praktilistele nõuetele ning süsteemi võimekust kohanduda erinevate kasutusstsenaariumitega.

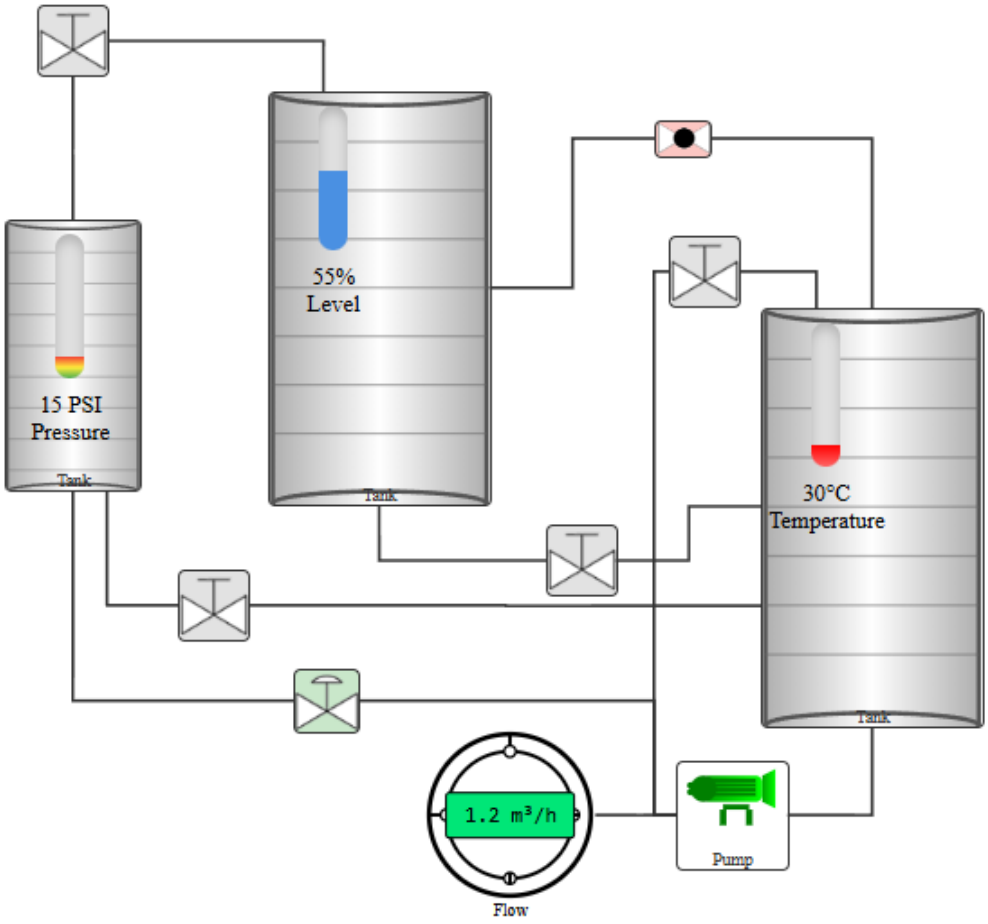


Joonis 3.6 P&ID skeem, mis sisaldab rakenduses olevaid komponente

Rakenduse poolt genereeritud tulemus (vt Joonis 3.7) vastab autori ootustele. Loodud HMI sisaldab kõiki skeemil esitatud elemente ning säilitab algse komponentide paigutuse. Täiendavalt on mahuti ikoon suurendatud, et mahutada selle sees olev rõhuandur. Klapile on määratud vaikimisi olek "suletud", mistõttu on selle tausta värv punane. Lisaks kasutatakse klapi tähistamiseks sümbolit, mis on lisatud vastava .png-failina.

Pump on esitatud rohelisena, et näidata selle tööolekut. Manuaalselt opereeritavad klapid on samuti HMI-le lisatud, et operaator saaks parema ülevaate alternatiivsetest ühendustest, mida on võimalik HMI-st sõltumatult kasutada. Need klapid on kujutatud halli taustavärviga, kuna HMI-l puudub teave nende reaalse oleku kohta.

Vooluanduri sümbol on parema nähtavuse tagamiseks suurendatud ning HMI-s on see kujutatud selge ja terava SVG-kujutisena. Anduritele on demonstratsiooni eesmärgil lisatud näitlikud andmed, mis aitavad visualiseerida nende võimalikku käitumist.



Joonis 3.7 Rakenduse poolt automaatselt genereeritud veebipõhine HMI

3.4 HMI genereerimise analüüs

HMI veebipõhise koodi genereerimise arendamine näitas, et automaatne protsess võib pakkuda olulisi eeliseid traditsiooniliste meetodite ees. Loodud lahendus tõestas võimet luua toimiv ja visuaalselt selge HMI kood, mis vastas algsele P&ID skeemile.

Peamised tugevused ja saavutused:

1. Protsessi kiirus ja efektiivsus: automaatne koodigeneratsioon vähendas märkimisväärselt aega, mis kuluks samaväärse tulemuse saavutamiseks käsitsi kodeerides.
2. Visuaalne selgus: kasutatud SVG-graafika ja sümbolite dünaamiline positsioneerimine tagasid selge ja lihtsasti kasutatava HMI loomise.
3. Modulaarsus: rakenduse ülesehitus võimaldab hõlpsasti kohandada ja lisada uusi komponente, muutes lahenduse paindlikuks ka edasiseks arendamiseks.

Piirangud ja parendusvõimalused:

1. Komplekssete P&ID skeemide käsitlemine: kuigi lihtsamate skeemide väljatöötamine oli edukas, võivad rohkemate sümbolite ja ühendustega keerukamad skeemid nõuda täiendavat käsitsi sekkumist.
2. Ühenduste suunad: loodud süsteem ei suuda hetkel tuvastada täpselt kõigi ühenduste suundi, mis võib põhjustada vigu lõplikus HMI koodis.
3. Funktsionaalne integreeritus: praegune lahendus keskendub visuaalsele täpsusele, kuid HMI funktsionaalsuse edasiarendamiseks tuleb teha veel täiendavat tööd.

Antud peatüki tulemused näitavad, et P&ID skeemidest automaatselt genereeritud HMI kood suudab oluliselt suurendada protsesside automatiseerimise tõhusust. Samas annab töö aluse edasistele täiustustele, et rakendus oleks kasutatav laiemates ja keerukamates tööstuspraktikates.

KOKKUVÕTE

Käesolev lõputöö keskendub P&ID jooniste digitaliseerimise ja veebipõhise HMI koodi genereerimise automatiseerimisele, eesmärgiga parandada projekteerimis- ja tootmisprotsesside efektiivsust ning vähendada käsitsi tehtava töö mahtu. Töö eel püstitas autor hüpoteesi, et P&ID automaatne digitaliseerimine on efektiivsem kui käsitsi tehtavad meetodid ning see võimaldab kasutajal säästa aega. Lisaks sooviti tõestada, et digitaliseeritud P&ID-st on võimalik automaatselt luua veebipõhine HMI kood.

Töö käigus saavutatud eesmärgid:

1. P&ID jooniste digitaliseerimine: rakendati YOLOv8 objekti tuvastamise mudelit ja OpenCV joonetuvastust, et tuvastada sümbolid ja nende ühendused tehnilistel diagrammidel. Loodud lahendus oli suure täpsusega, kuigi tuvastamisel esines teatud piiranguid keeruliste ühenduste ja väikeste sümbolite puhul. Süsteem vähendas digitaliseerimise ajakulu võrreldes käsitsi meetoditega märkimisväärselt.
2. Graafiline kasutajaliides: arendati paindlik GUI, mis võimaldab kasutajal parandada ja kohandada automaatselt tuvastatud P&ID elemente enne lõpliku HMI koodi genereerimist. Kasutajaliides osutus tõhusaks vahendiks digitaliseerimisprotsessi vigade parandamisel ja töövoo kiirendamisel.
3. Veebipõhine HMI koodi genereerimine: loodud süsteem lõi automaatselt HTML-põhise HMI, mis visualiseerib ja võimaldab hallata tööstusprotsesse. Lõpptulemuse täpsus ja komponentide selge visuaalne esitlus vastasid projekti ootustele. Eriti tõhusaks osutus SVG-graafika kasutamine, mis tagas kõrgekvaliteedilise ja dünaamilise visuaali.

Järeldused ja hinnang tehtud tööle:

Töö tulemused kinnitasid hüpoteesi, et P&ID automaatne digitaliseerimine on efektiivsem kui käsitsi tehtavad meetodid. Automaatse digitaliseerimise ja HMI koodi genereerimise protsess säästsid kasutaja aega ning pakkusid samal ajal täpset ja kasutajasõbralikku väljundit.

Kuigi töö tulemused on julgustavad, vajab süsteem keerukamate skeemide puhul kasutamiseks edasist täiustamist. Keerukamate ühenduste ja standardiseerimata sümbolite tuvastamine osutus mõningates aspektides keeruliseks ning seadis omad piirangud.

Autorina osalesin töö kõigis etappides ning vastutasin lahenduse väljatöötamise ja testimise eest, alustades masinõppemudeli valikust ja treenimisest kuni GUI arendamise

ja HMI koodi genereerimise süsteemi loomiseni.

Võimalused edasiseks arenduseks:

1. Tuvastustäpsuse parandamine: sümbolite ja ühenduste tuvastamise algoritme saaks rohkemate treeningandmete lisamise abil täiustada.
2. Rakenduse skaleerimine: lahenduse arendamine keerukamate skeemide ja suuremate andmemahtude töötlemiseks.

Kokkuvõttes saavutas autor töö käigus püstitatud eesmärgid ning lõi praktilise ja tulevikus skaleeritava lahenduse. Tulemused demonstreerivad lahenduse potentsiaali muuta tööstusautomaatika projekteerimisprotsessid efektiivsemaks ja paindlikumaks. Töö loob tugeva aluse edasiseks arendustegevuseks ja lahenduse praktikas rakendamiseks.

SUMMARY

In modern-day automation and manufacturing management, there is an growing awareness on digitalization and improving efficiency. Human-machine interfaces (HMI) are vital factors on this area, permitting operators to supervise and control processes efficiently. Historically, HMI interfaces are created manually from Process and Instrumentation Diagrams (P&ID), making the procedure labor-intensive and time-consuming. This thesis seeks to develop an solution comprising three essential elements: the digitalization of P&ID diagrams, a graphical user interface (GUI) for editing the diagrams, and automatic generation of code for web-oriented HMI. The main emphasis is on the automated creation of web-based HMI code.

The research object of the thesis is P&ID diagrams and the methods of HMI interface generation based on them. Current tools, such as Siemens WinCC and AVEVA InTouch, support manual design processes and lack functionality for automated web-based code generation directly from P&ID diagrams. Existing solutions automate some aspects of HMI interface development but do not offer fully integrated systems for converting P&ID diagrams into web interfaces.

The hypothesis of this thesis is that automatic digitalization of P&ID diagrams is more efficient than manual methods, providing significant time savings for the user and it is possible to automatically create web-based HMI code from P&ID.

The application is developed in a minimal form to handle small datasets. It is created just to demonstrate a workflow and to lay the foundation for automated HMI interface creation. Despite potential challenges in testing and development, the focus of this work was on validating the applications functionality and workflow.

Results accomplished in the course of the thesis:

- P&ID diagram digitalization: The YOLOv8 object detection model and OpenCV line detection were implemented to identify symbols and connections in diagrams. The solution demonstrated high accuracy but faced challenges with complex connections and small symbols. When compared to human labor, the application shortened the P&ID digitization time.
- Graphical user interface (GUI): A flexible GUI was developed, allowing users to adjust and refine automatically detected P&ID elements before generating HMI code.
- Web-based HMI code generation: The system successfully generated web-based HMI code, enabling visualization and control of industrial processes.

These findings confirm that automated P&ID digitization can save a considerable amount

of time and generally performs better than manual methods. Still, it needs refinement, especially for handling more complex layouts and less common symbols. The next steps should involve expanding its capabilities to accommodate larger datasets and more intricate diagrams, as well as improving its accuracy through additional training data. In conclusion, this thesis successfully met its goals by delivering a practical solution. The findings show that this technology can genuinely boost efficiency in industrial automation design.

KASUTATUD KIRJANDUS

1. Features Recognition from Piping and Instrumentation Diagrams in Image Format Using a Deep Learning Network. [Online] https://www.mdpi.com/1996-1073/12/23/4425 (30.11.2024)
2. Fimbres, O. Engineering Document (P&ID) Digitization. [Online] https://devblogs.microsoft.com/ise/engineering-document-pid-digitization/ (10.10.2024)
3. Digitize-PID: Automatic Digitization of Piping and Instrumentation Diagrams [Online] https://arxiv.org/abs/2109.03794 (30.11.2024)
4. Kang, L., Lee E., Baek H. A Digitization and Conversion Tool for Imaged Drawings to Intelligent Piping and Instrumentation Diagrams (P&ID) [Online] https://www.mdpi.com/1996-1073/12/13/2593 (30.11.2024)
5. Automatic Digitization of Engineering Diagrams using Deep Learning and Graph Search [Online] https://openaccess.thecvf.com/content_CVPRW_2020/papers/w8/Mani_Automatic_Digitization_of_Engineering_Diagrams_Using_Deep_Learning_and_Graph_CVPRW_2020_paper.pdf (30.11.2024)
6. Ken, A. M. YOLOV8, Is It Better than Its Predecessor? [Online] https://medium.com/@georgekenjiputra/yolo-v8-is-it-better-than-v7-4f978ceaec52 (10.10.2024)
7. Helmich, J. Festo Process automation MPS PA Compact Workstation Manual: Adiro Automatisierungstechnik GmbH, 2008.
8. Computer Vision Annotation Tool. [Online] https://docs.cvat.ai/about/ (10.10.2024)
9. Effectiveness of Modern Text Recognition Solutions and Tools for Common Data Sources. [Online] https://ceur-ws.org/Vol-2870/paper15.pdf (10.10.2024)
10. Object Detection in Design Diagrams with Machine Learning. [Online] https://tuhat.helsinki.fi/ws/portalfiles/portal/128457068/Object_Detection_in_Design_Diagrams_with_ML.pdf (10.10.2024)
11. Scikit-image image processing in Python. [Online] https://scikit-image.org/ (10.10.2024)
12. Python bindings for the LSD line segment detector. [Online] https://github.com/ghing/python-lsd?tab=readme-ov-file (10.10.2024)
13. Cabot, J. 20+ JavaScript libraries to draw your own diagrams (2024 edition).

[Online] https://modeling-languages.com/javascript-drawing-libraries-diagrams/ (11.11.2024)

14. JointJS dokumentatsioon. [Online] https://docs.jointjs.com/ (11.11.2024)
15. Stackoverflow arutelud teemal JointJS. [Online] https://stackoverflow.com/questions/tagged/jointjs (11.11.2024)
16. JointJS veebileht. [Online] https://www.jointjs.com/license (11.11.2024)
17. MxGraphi Githubi veebileht. [Online] https://github.com/jgraph/mxgraph (11.11.2024)
18. MxGraphi Githubi veebileht. [Online] https://jgraph.github.io/mxgraph/ (11.11.2024)
19. MxGraph arutelud Stackoverflows. [Online] https://stackoverflow.com/questions/tagged/mxgraph (11.11.2024)
20. Diagram-js Githubi veebileht. [Online] https://github.com/bpmn-io/diagram-js (11.11.2024)
21. Arutelu BPMN ametlikus foorumis. [Online] https://forum.bpmn.io/t/tutorial-intro-on-creating-shape-business-objects-connections-in-diagram-js/6727 (11.11.2024)
22. Cytoscape.js veebileht. https://js.cytoscape.org [online] (11.11.2024)
23. Inkscape veebilhet. https://inkscape.org/ [online] (19.11.2024)