Upravljački algoritmi u realnom vremenu
Projekat 3 - *Pump Station*

Kandidati: Nenad Radović, Djordje Ristić, Petar Popov, Filip Stevanović
Profesor: Zeljko Kanović

Jun 29., 2023.

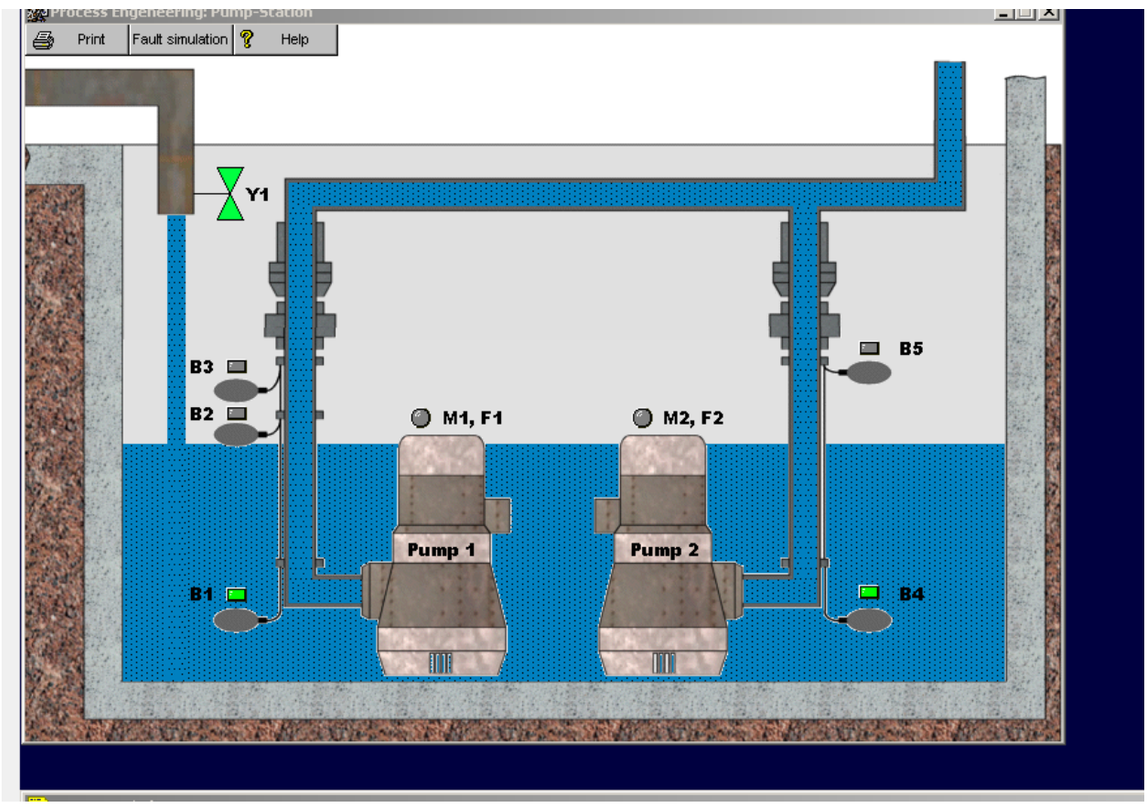
Sadržaj

1 Opis sistema

1.1 Elementi sistema

Sistem koji pokušavamo regulisati sastoji se od:

1. **Regulacionog ventila** Y_1, kojim se dovodi tečnost u rezervoar.
2. **Pumpe** P_1 **i** P_2, kojima se ispumpava dovedena tečnost iz rezervoara.
3. **Senzori** B_1**,** B_2**,** B_3**,** B_4 **i** B_5, koji su binarni senzori i daju naznaku o dostignutom nivou tečnosti.



Slika 1: *Izgled sistema*

1.2 Opis rada sistema

1.2.1 Automatski režim rada

Pretpostavimo da je u rezervoaru (prije uključivanja sistema) postojao odredjen nivo tečnosti i da je taj nivo iznad senzora B_1 i B_4 (dakle, ti senzori će davati signal aktivacije, a pošto su **normalno otvoreni**, davaće logičku jedinicu - baš kao na slici iznad). Neka je, takodje, ventil Y_1 koji pušta tečnost u rezervoar **otvoren**, u početnom trenutku.

Ako aktiviramo automatski režim rada, sistem **neće reagovati** (pri pretpostavljenim početnim uslovima) sve dok ne dodje do aktivacije senzora B_2, koji je takodje **normalno otvoren**. Kada izlaz pomenutog senzora postane logička jedinica, u tom trenutku automatski režim aktivira jednu od pumpi (podrazumijevano je da aktivira pumpu P_1).

Po uslovu zadatka, ako je nivo tečnosti u bilo kojem vremenskom trenutku aktivirao senzor B_2 ali nije B_3, potrebno je da pumpe **rade naizmjenično** sve dok se tečnost ne spusti iznad senzora B_1 ili B_4. To je ostvareno tajmiranjem vremena rada pumpi, gdje svaka ima pravo da radi maksimalno 30 sekundi, osim ako se ne desi ikakva promjena u sistemu.

Kada jedna pumpa nije sama sposobna da ispumpava tečnost iz rezervoara, dolaziće do rasta nivoa tečnosti - samim time i do aktivacije senzora B_3. Ako senzor B_3 da naznaku da je do njega došla tečnost, tada se **bezuslovno aktiviraju obje pumpe** sve dok nivo tečnosti ne padne ispod senzora B_1 ili B_4.

Ekstreman slučaj u sistemu predstavlja prejak dotok i prebrzo punjenje rezervoara, te obje pumpe nisu dovoljno moćne da ispumpavaju tečnost iz sistema. U tom slučaju, tečnost

će rasti sve do senzora B_5, a kada je on aktiviran, daće naznaku upravljačkom sistemu da zatvori ventil Y_1, te će obje pumpe nastaviti sa radom sve dok nivo tečnosti ne padne ispod senzora B_1 ili B_4.

1.2.2 Ručni režim rada

Ručni režim rada ostvaruje se preklopom prekidača *Hand/Auto* sa *Auto* na *Hand*. Tada je isključivo moguće mijenjati stanja pumpi pomoću dugmadi *Pumps ON* i *Pumps OFF*.

Ako se stisne dugme *Pumps ON*, ono daje naznaku aktivacije obje pumpe, te one kreću sa ispumpavanjem tečnost iz rezervoara. Obratnu funkciju vrši prekidač *Pumps OFF*.

Potrebno je paziti na rad pumpi "na suvo", te je u tu svrhu dodat "stražar" pod nazivom *Hand Error*, koji neće dozvoliti paljenje pumpi ako je nivo tečnost ispod detekcije senzora B_1 ili B_4.

2 *LabView* kod za regulaciju sistema

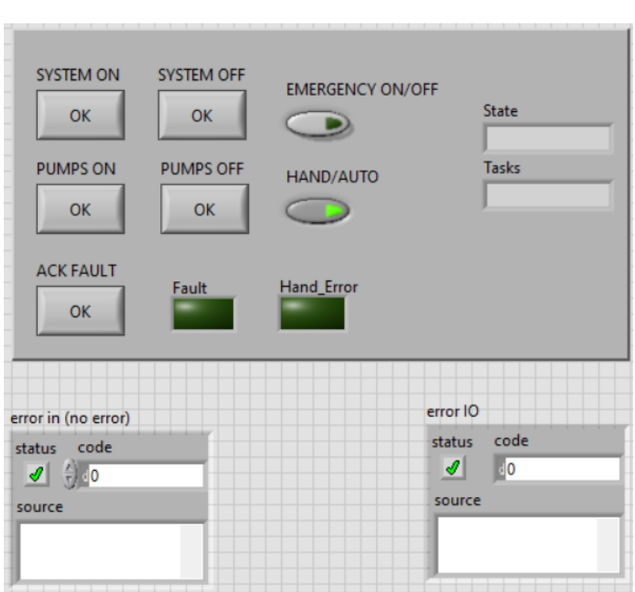
LabView programska podrška za regulaciju opisanog sistema koncipirana je iz nekoliko dijelova:

1. **Dio aplikacije na računaru** predstavlja nadzorno-upravljački sistem kojim se daju komande za regulisanje sistema, medju kojima su prethodno navedene funkcionalnosti - **prebacivanje iz automatskog u ručni režim i obratno**, **uključivanje i isključivanje pumpi**, **uključivanje i isključivanje regulacionog sistema u automatskom režimu**, kao i **ukazivanje na i otklanjanje greške**.

2. **Dio aplikacije na *cRIO* modulu** predstavlja modul koji prima komande sa računara, adekvatno ih interpretira te daje parametre za regulaciju sistema (prebacivanje iz jednog stanja u drugo i sl.).

2.1 Dio aplikacije na računaru

Dio aplikacije na računaru zadužen je za davanje komandi koje kontrolišu sistem. *VI* na kojem se nalazi upravljački *front panel* nalazi se u fajlu *SCADA.vi*. Pored pomenutog *.vi* fajla nalazi se još jedan "pomoćni" fajl, nazvan *MIDDLE.vi*, čija funkcionalnost će biti objašnjena kasnije.



Slika 2: *Upravljački Front Panel*

Upravljačka tabla se tako sastoji od nekoliko elemenata:

1. ***Hand/Auto* prekidač**, kojim se bira željeni režim rada - **ručni** i **automatski**.

2. ***System ON / System OFF* dugmad**, koji isključivo imaju funkcionalnost u automatskom režimu rada i čijim se pritiscima daje naznaka da li postoji automatska regulacija ili ne.

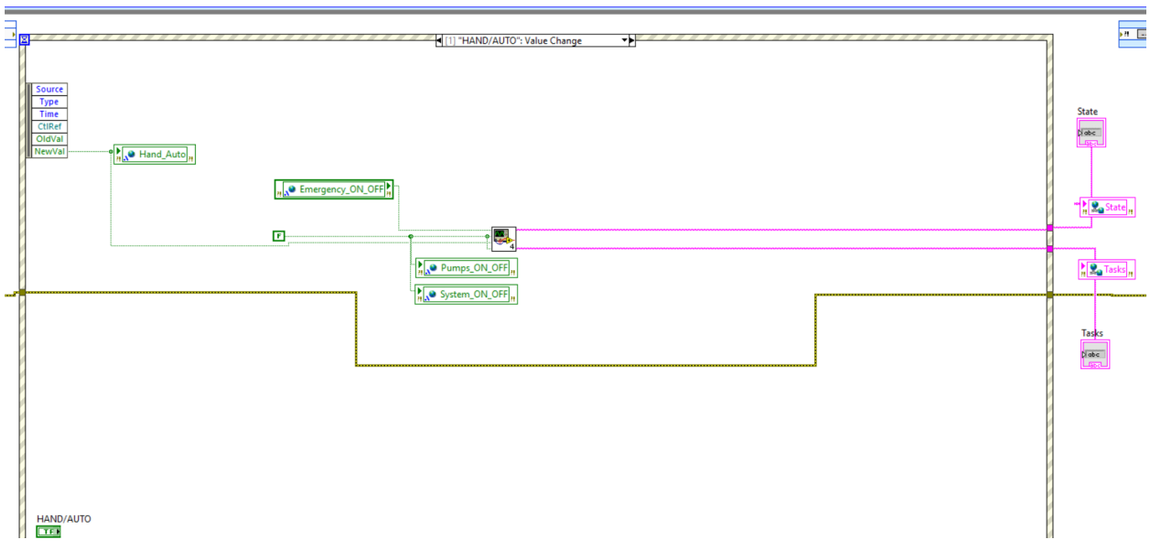
3. ***Pumps ON / Pumps OFF* dugmad**, koji isključivo imaju funkcionalnost u ručnom režimu rada i čijim se pritiscima daje naznaka da li je omogućen rad pumpi ili ne.

4. ***Emergency OFF* dugme**, čijom aktivacijom ili deaktivacijom dajemo naznaku da je došlo do greške u sistemu ili da smo "svjesni" te doznake, respektivno.

5. ***Acknowledge Fault* dugme**, na čiji pritisak se "oslobadjamo", odnosno otklanjamo grešku u sistemu.

Kod koji pomaže pri realizaciji funkcionalnosti sistema "trči" kroz *Timed Loop* strukturu, koja je formatirana na izvršavanje na svakih 100 ms. Razlog tome jeste enkapsulacija vremena koje je potrebno da senzor da odziv na prisustvo tečnosti, kao i vremena koje je potrebno da signal dodje do upravljačke konzole.

U *Timed Loop* strukturi nalazi se *Event Case* struktura, koja je postavljena da na pritisak proizvoljnog dugmeta reaguje na zahtjev, obradi ga i pošalje adekvatnu komandu *cRIO* modulu, ako je to moguće.



Slika 3: *Izgled Event strukture na promjenu Hand/Auto prekidača*

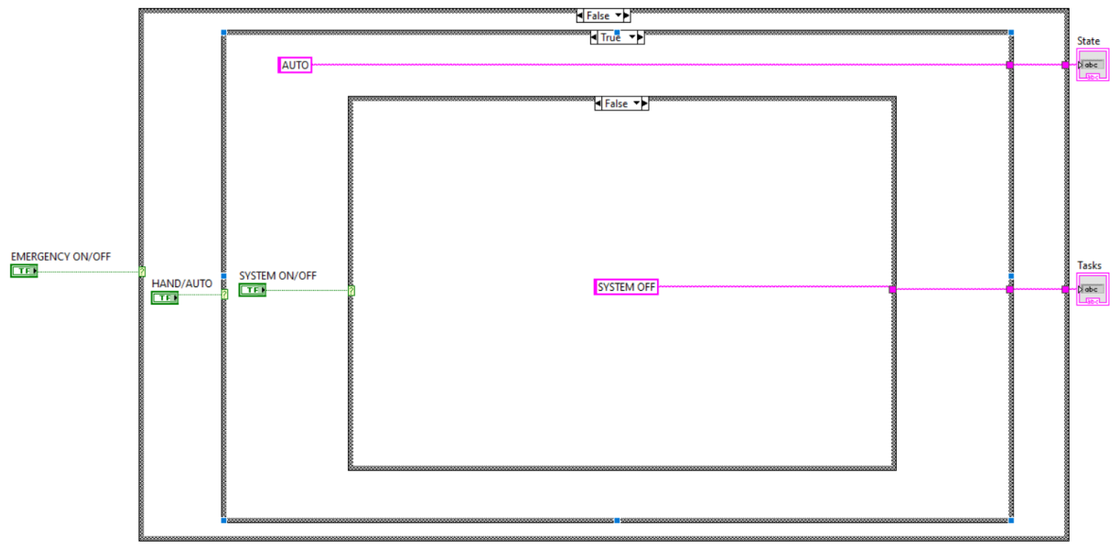
Promjenljive koje vidimo na slici 3. služe za čuvanje trenutnog stanja sistema, odnosno trenutnog stanja prekidača i dugmadi (recimo, pritisak na *Pumps ON* izazvaće promjenu promjenljive *Pumps_ON_OFF* na *True* vrijednost, dok će pritisak na dugme *Pumps OFF* ovu promjenljivu promijeniti na *False* vrijednost - slično i za ostale promjenljive), te se oni prosledjuju pri pozivanju funkcionalnosti *MIDDLE.vi*.

MIDDLE.vi ima ulogu odlučivanja stanja rada sistema - proizvodi adekvatnu komandu na pritisak jednog od dugmadi, ali ne vrši odlučivanje šta se dalje dešava u sistemu, kako se reaguje na senzore i sl. Suštinski, uloga ovog dijela sistema najviše se ogleda pri prelazu iz režima u režim.

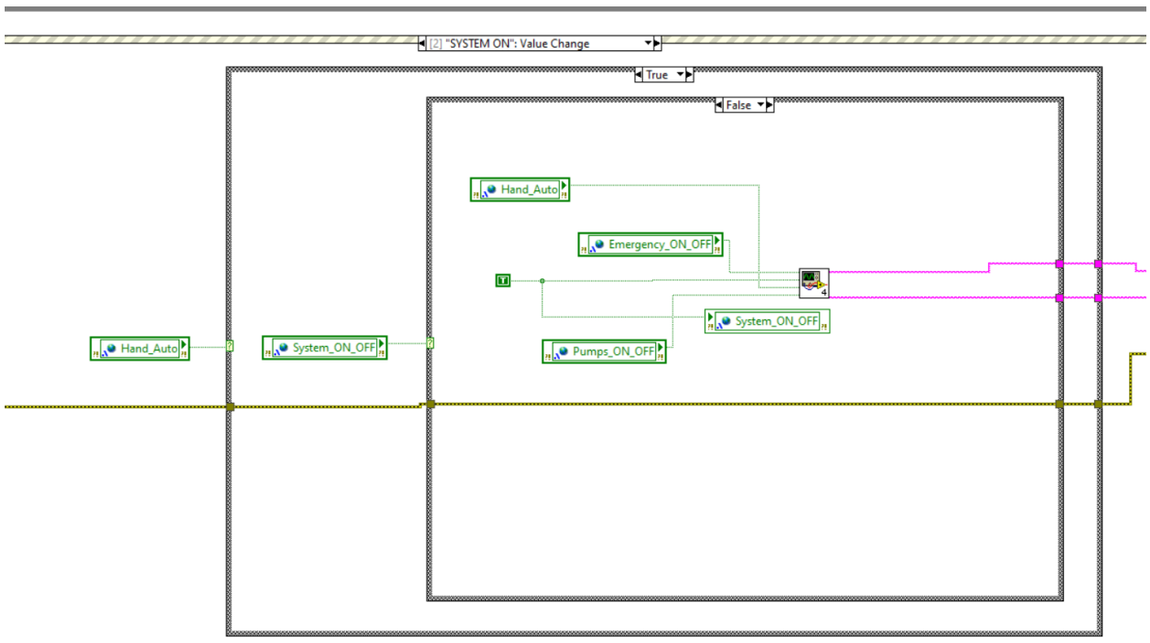
Komande koje se dobijaju iz *MIDDLE.vi* funkcionalnosti prosledjuju se u mrežne promjenljive *State* (koja označava da li bi trebalo da reaguje *Hand*, *Auto* ili *Emergency* dio sistema) i *Tasks* (koja označava koje "zadatke" dobija pomenuti dio sistema).

Problem koji se može desiti jeste ponovno pritiskanje prethodno pritisnutog dugmeta, gdje sistem ne bi trebalo da reaguje jer zapravo nema nove komande. Pomenuti problem se upravo rješava promjenljivima, čije prethodno stanje se provjerava te ako novo nije jednako sa njim, prelazi se u obradu komande.

Pored toga, u obavezi smo zaustaviti i pogrešno, odnosno nedozvoljeno davanje komandi. Pod time se misli da se komande ograniče na svoj režim - ne možemo davati komande vezane za automatskom režimu ako smo u ručnom režimu, kao i da se komande ne priznaju ako smo u *Emergency* slučaju.



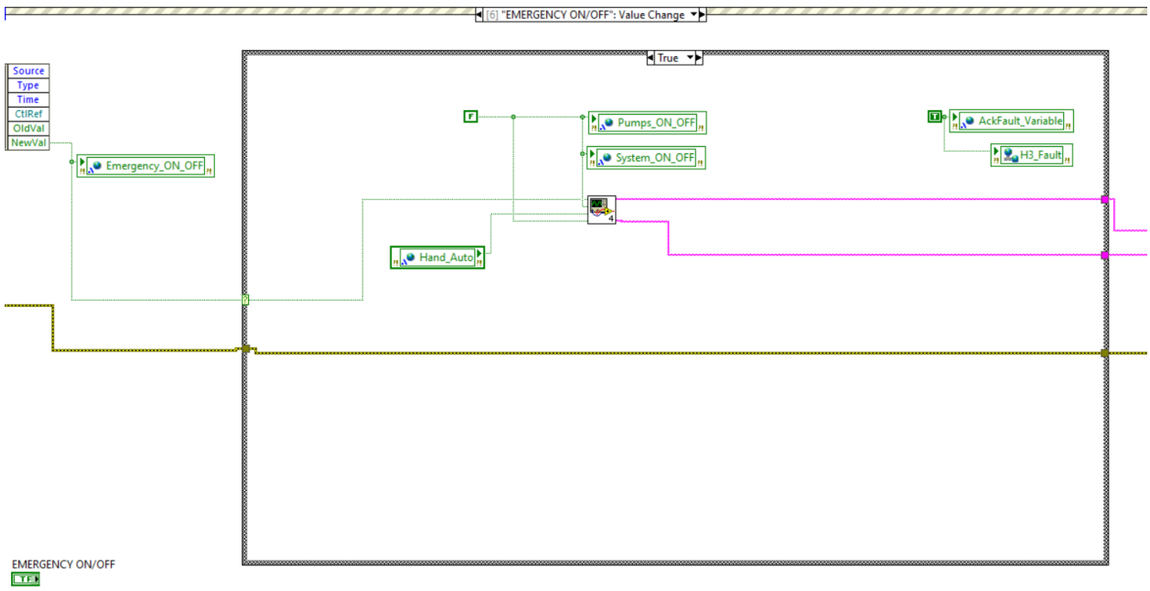
Slika 4: *Izgled MIDDLE.vi*



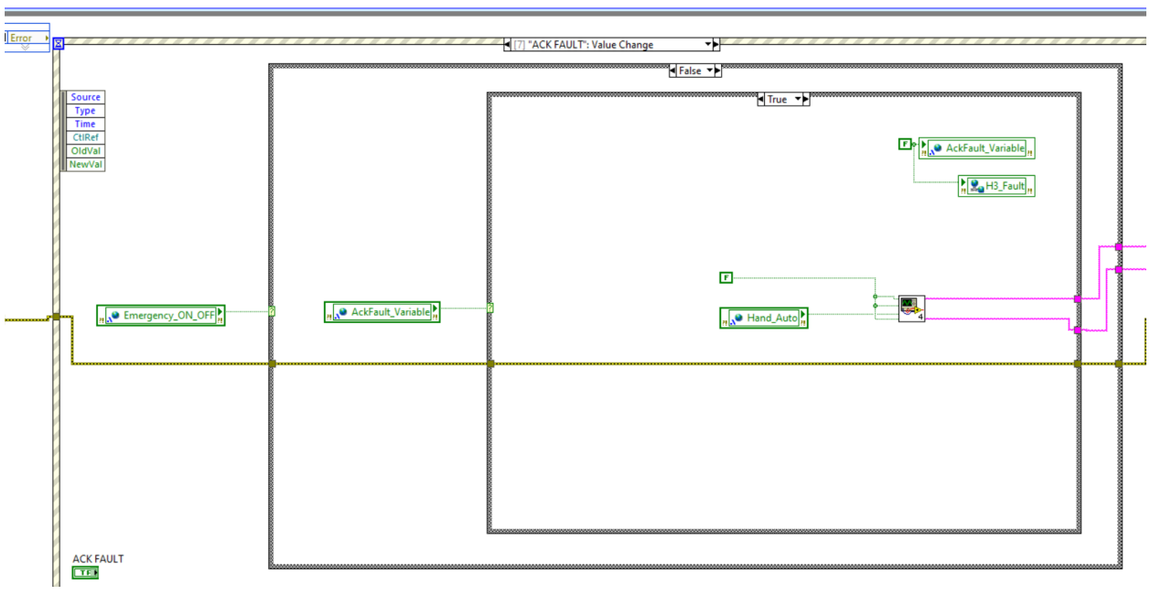
Slika 5: *Prikaz slučaja System ON, gdje se provjeravaju uslovi za ispunjenje komande*

Posebno interesantan slučaj jeste kada je sistem u *Emergency* stanju, te je potrebno pored pritiska na dugme koje "nas oslobadja" od takvog stanja stisnuti i dugme *Acknowledge Fault* koje briše grešku iz sistema.

Uslov da se predje u dalje otklanjanje greške jeste da se prvo vrati *Emergency ON/OFF* dugme na *False* vrijednost, te zatim pritisne *Acknowledge Fault* dugme. Nakon toga se sistem vraća u stanje koje je prethodilo grešci, ali sa isključenim motorima. Može se smatrati da se sistem "restartuje".



Slika 6: *Prikaz slučaja Emergency, kada dodje do prekida u radu*



Slika 7: *Prikaz slučaja kada se pritisne Acknowledge Fault dugme*

Ostali slučajevi su slični po konstrukciji, u smislu razmatranja promjenljivih za svoj rad.

2.2 Dio aplikacije na *cRIO* kontroleru

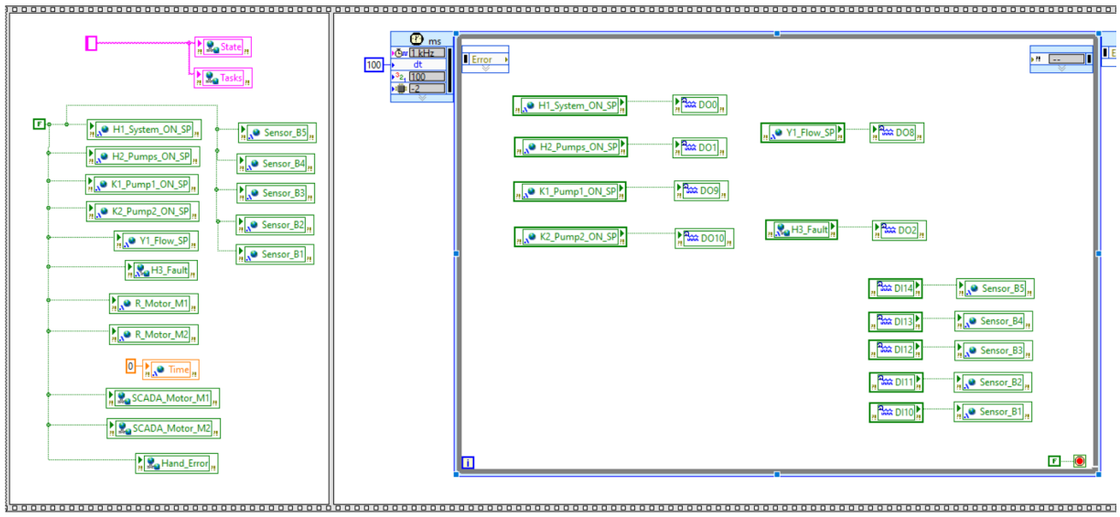
Postoje dva fajla koja se nalaze na *cRIO* kontroleru, a to su *cRIO.vi* i *HAND_AUTO.vi*.

2.2.1 *cRIO.vi*

U *cRIO.vi* fajlu vršimo inicijalizaciju jednoprocesnih i mrežnih promjenljivih na njihove podrazumijevane vrijednosti, te konstantno prosledjujemo stanje ulaza i izlaza sa *cRIO* uredjaja, kao i na *cRIO* uredjaj.

Flat Sequence struktura jeste tu da natjera izvršavanje inicijalizacije promjenljivih prije samog kupljenja njihovih vrijednosti sa ulaza, kao i postavljanja njihovih vrijednosti na izlaze *cRIO* uredjaja.

Razlog postojanja *Timed Loop* strukture isti je kao i u prethodnoj sekciji - enkapsulacija vremena kupljenja i postavljanja signala.



Slika 8: *Izgled cRIO.vi*

2.2.2 *HAND_AUTO.vi*

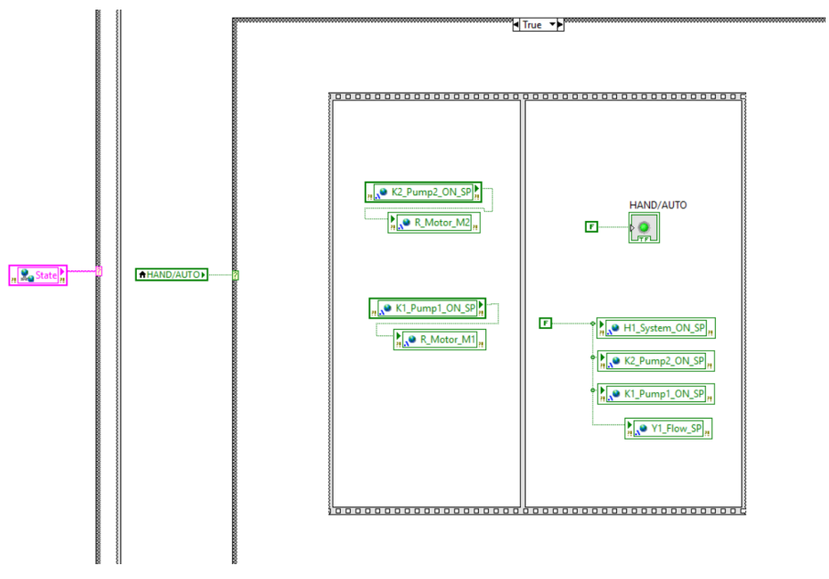
Najinteresantniji dio gdje se odvija čitava logika automatskog i ručnog sistema nalazi se u *HAND_AUTO.vi* fajlu.

U samom *VI* izvršavaju se dvije *Timed Loop* strukture, gdje jedna vodi glavnu logiku, dok druga služi za tajmiranje, o kojem će biti više riječi u nastavku teksta.

U prvoj petlji smještena je spoljašnja *Case* struktura, koja razlikuje svoje slučajeve pomoću mrežne promjenljive *State*, čiji je tip *String* i čiju vrijednost mijenjamo direktno u *SCADA.vi*, kao što smo već vidjeli. Pomenuli smo da stanja koja može uzeti pomenuta promjenljiva jesu ***HAND***, ***AUTO*** i ***EMERGENCY***.

Razmotrimo stanje ***HAND***. Od značaja dolazi indikator *HAND\AUTO*, koja govori da li se dešava prelaz sistema iz *HAND* režima u *AUTO* režim. Ovo je posebno značajno jer se dešava sledeće:

- Sistemski, prelaz iz stanja *AUTO* u stanje *HAND* je isti kao prelaz iz *Pumps ON* internog stanja u *Pumps OFF* interno stanje - u oba slučaja dešava se gašenje pumpi.
- Potrebno je razlikovati ta dva slučaja gašenja pumpi jer je potrebno u slučaju prelaska iz *AUTO* u *HAND* zapamtiti stanje i aktivnost pumpi (zahtjev pri izradi sistema - pumpe "pamte" kakve su bile u *AUTO* režimu).
- Uvodi se pomoćna varijabla *HAND\AUTO*, koja će razlikovati prethodno opisana dva slučaja.



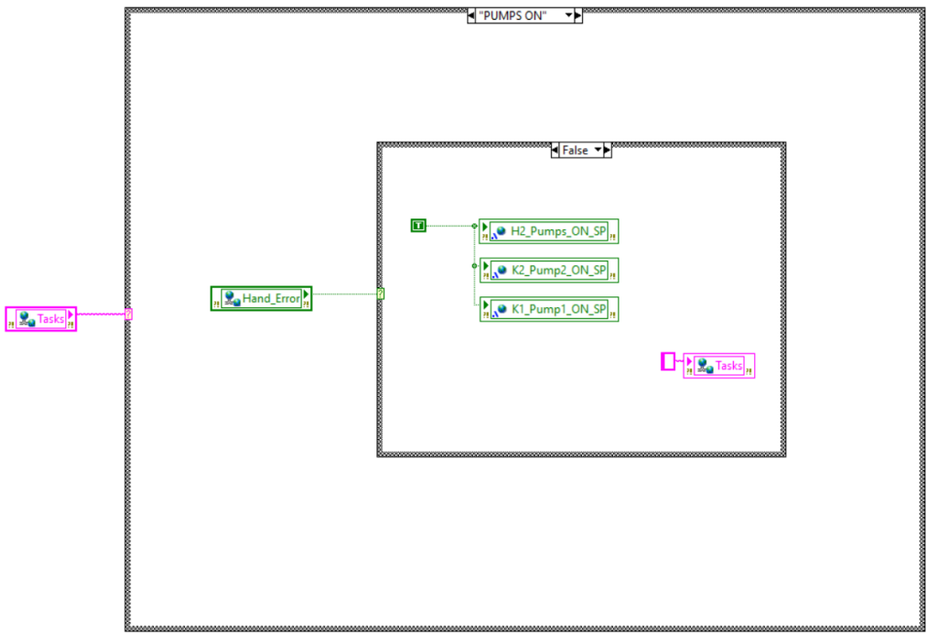
Slika 9: *Prelaz iz AUTO u HAND*

HAND\AUTO bilježi prethodno stanje (uzeto je stanje *True* za automatski, kao i *False* stanje za ručni režim), te se tako provjerava da li je došlo do prelaza. Kada je to slučaj, indikator se stavlja na vrijednost koja mu odgovara u zavisnosti od režima rada, te se vrši pamćenje prethodnog stanja pumpi (za konkretan slučaj sa slike) te gašenje istih.

Ručni režim rada je praktično jednostavan - postoje dvije komande i stanje "čekanja". Ako postoji "zadatak" izdat od upravljačkog dijela sistema, on će se nalaziti u mrežnoj promjenljivoj *Tasks*.

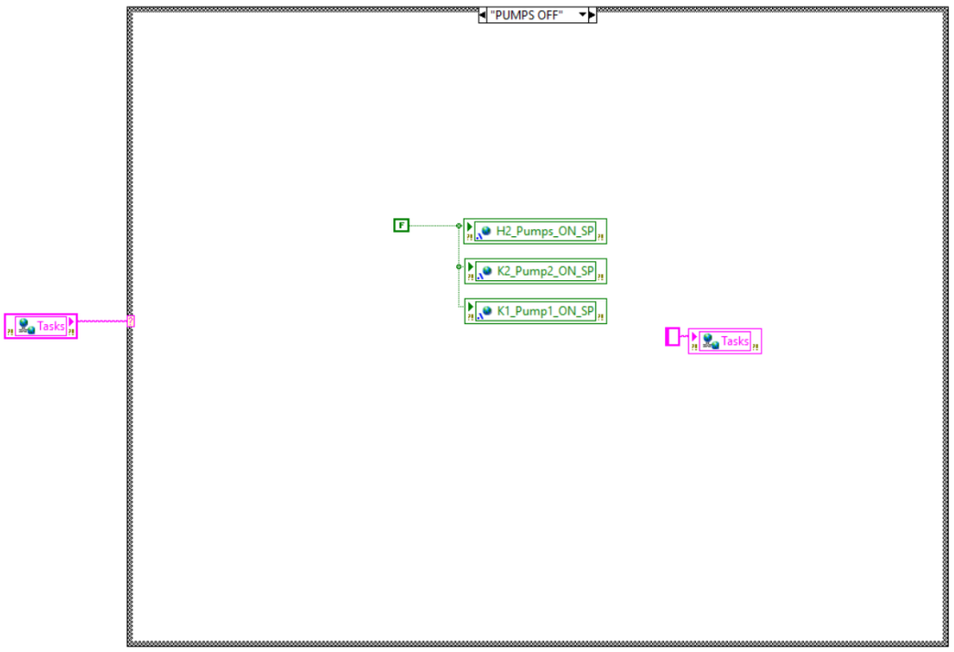
Za ručni režim, promjenljiva *Tasks* može uzeti vrijednosti *Pumps ON* i *Pumps OFF*. U *Pumps ON* režimu, provjeravamo da li postoji prethodno pomenuta greška, odnosno pad nivoa tečnosti ispod detekcije senzora B_1 ili B_4. To će odredjivati dozvolu rada pumpi.

Naravno, kada odradimo zadatak, potrebno je "očistiti" *Tasks* promjenljivu. To radimo tako što postavljamo praznu riječ u nju.



Slika 10: *Pumps ON interno stanje*

Sličan je rad i *Pumps OFF* internog stanja.

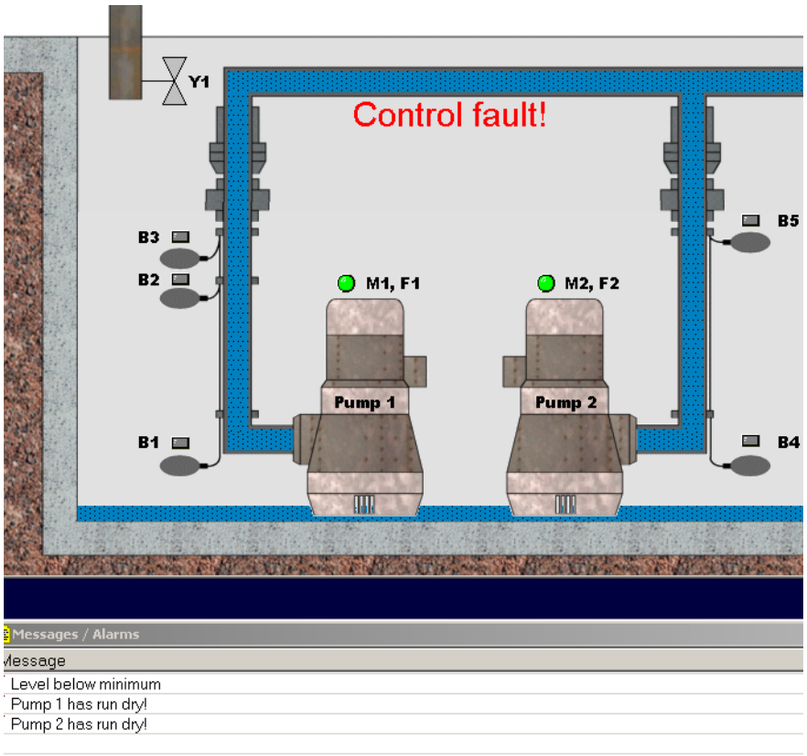


Slika 11: *Pumps OFF interno stanje*

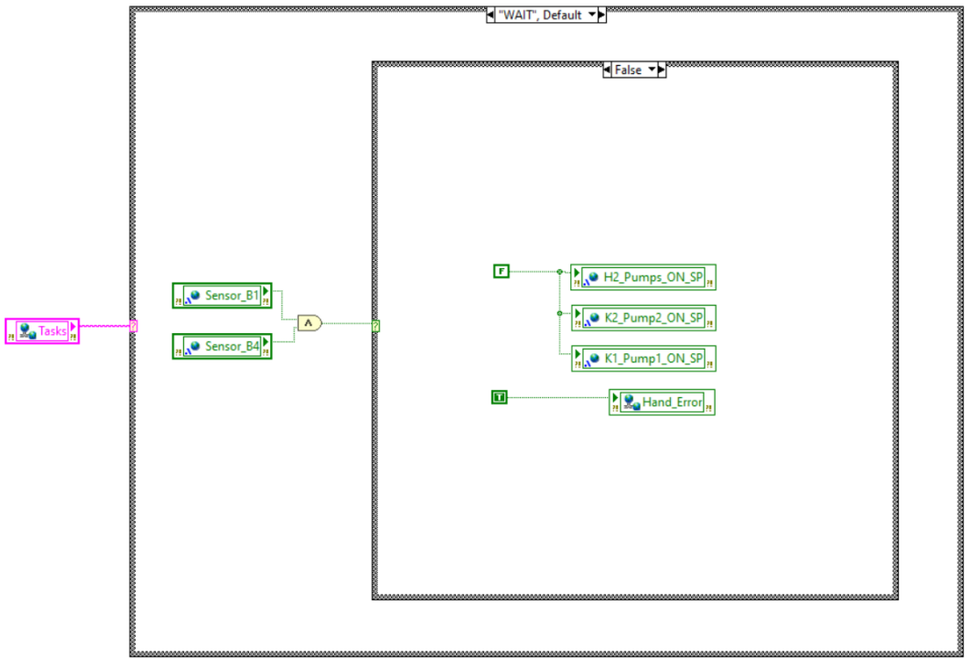
Ako promjenljiva sadrži praznu riječ, to je oznaka da se režim postavi u stanje čekanja.

Konkretno za *HAND* režim, čekanje predstavlja posmatranje već pomenutih senzora B_1 i B_4. Ako nivo tečnosti padne ispod njihove detekcije, dolazi do prestanka rada pumpi - zaštita od rada na suvo. Indikator *Hand Error* se postavlja na *True* vrijednost, te se pumpe ne mogu ponovo upaliti sve dok se ne otkloni greška.

Ova greška nije greška za čitav sistem. Naime, potrebno je samo preći u automatski režim rada, u kojem je moguće aktivirati ventil Y_1 te pustiti tečnost da ponovo napuni rezervoar.

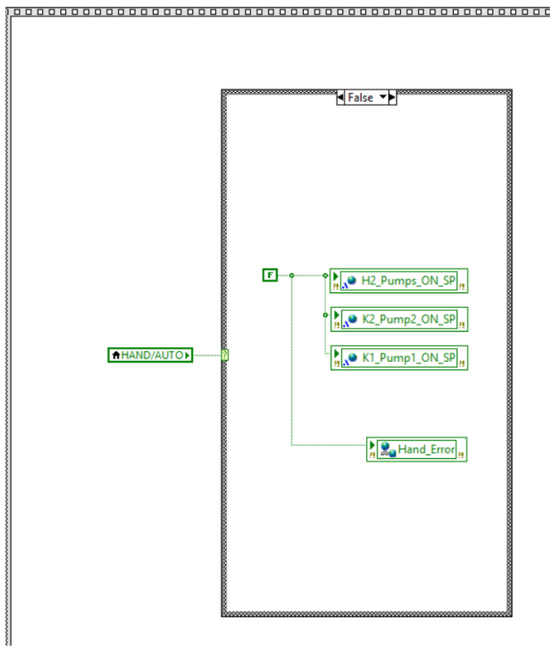


Slika 12: *Detekcija rada pumpi na suvo*



Slika 13: *Interno stanje čekanja - zaštita od rada na suvo*

Sada ćemo razmotriti stanje ***AUTO***. Detekcija prelaska je potrebna i suštinski je ista kao i u *HAND* režimu.



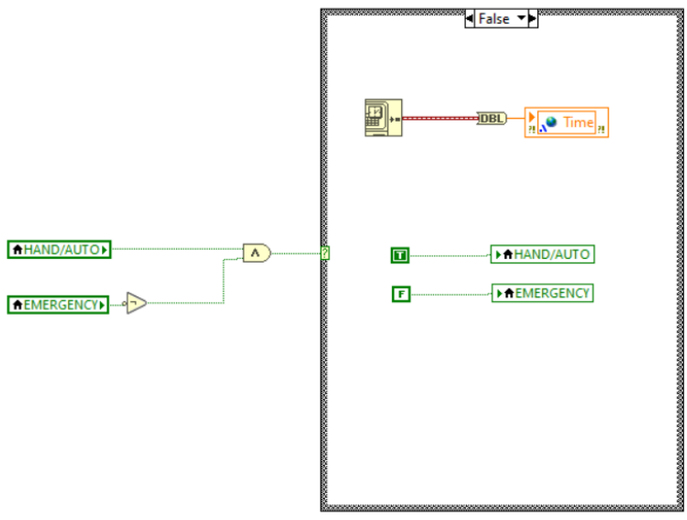
Slika 14: *Prelaz iz Hand režima u Auto režim*

Primjetimo da se otklanja *Hand_Error* greška jer smo sada u mogućnosti ponovno napuniti rezervoar. Takodje, važno je uvidjeti da ne prebacujemo $HAND \backslash AUTO$ promjenljivu na *True* vrijednost sada - razlog tome će ubrzo biti objašnjen.

Kada smo u automatskom režimu, moguće je alternirati izmedju stanja *System ON*, *System OFF* i *Wait*. Razmotrimo stanje *System OFF*.

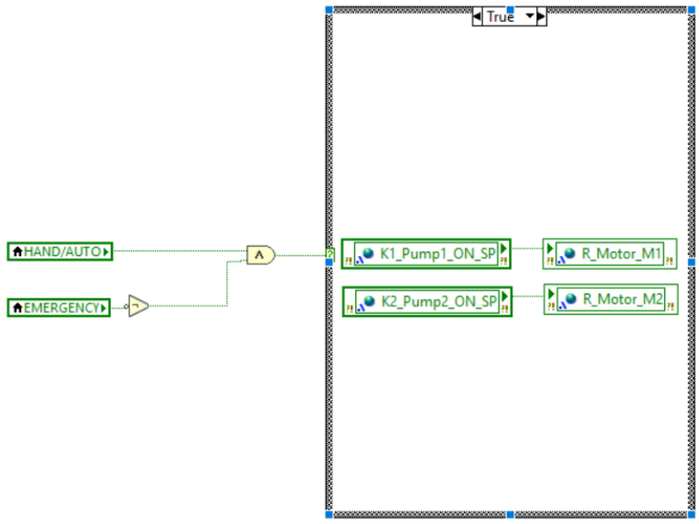
U pomenuto interno stanje se ulazi svaki put kada dodje do prebacivanja u *Auto* režim, ili kada se prebacuje iz internog stanja *System ON*, te je ono dužno da ugasi pumpe. U tu svrhu uvodimo novu *Case* strukturu, čija istinitost zavisi od logičkog "I" dvije promjenljive - $HAND \backslash AUTO$ i negirane *EMERGENCY* promjenljive.

Naime, ako je $HAND \backslash AUTO$ promjenljiva bila na *False* vrijednosti, dešava se prelaz sa *Hand* režima u *Auto* režim, te je potrebno prebaciti pomenutu promjenljivu na *True* vrijednost. Takodje, otkanja se greška (ako je postojala).



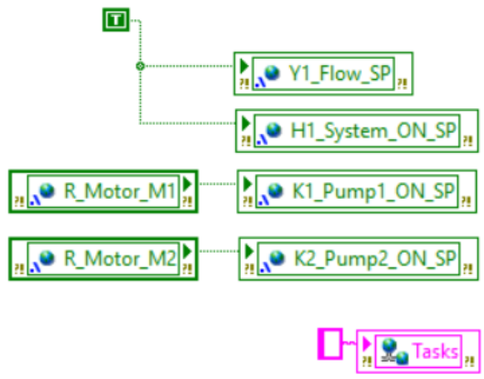
Slika 15: *System OFF kada dodje do prelaska izmedju režima*

Medjutim, ako se desilo da je $HAND \backslash AUTO$ promjenljiva bila na *True* vrijednosti, kao i da nije postojala greška u sistemu, to znači da smo došli iz *System ON* internog stanja, pa je potrebno zapamtiti koja od pumpi je radila.



Slika 16: *System OFF kada dodje do prelaska izmedju internih stanja*

System ON slučaj je vrlo jednostavan - dolazi do paljenja pumpi (odnosno prepisivanja zapamćenog stanja).



Slika 17: *System ON interno stanje*

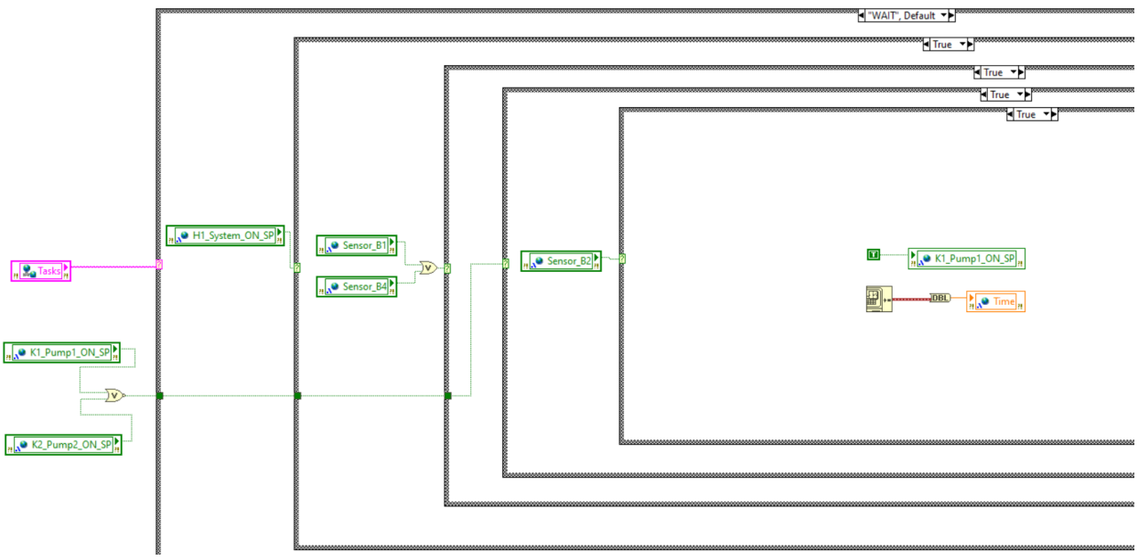
Slučaj gdje se čitava automatska regulacija odvija jeste stanje čekanja - *Wait* interno stanje. Ono prati logiku dešavanja opisanu u pasusu 1.2.1, dok se njegova realizacija sastoji od mnoštvo *Case* struktura koje zavise od provjere aktivnosti senzora.

Kreće se od provjere da li je sistem za regulaciju uključen, što se nalazi u *H1_System_On_SP* promjenljivoj. Pristup ide "od dole ka gore" - prvenstveno se provjerava aktivnost donjih senzora. Kao što je i opisano, ako jedan od senzora B_1 ili B_4 ne detektuje prisustvo tečnosti, daje se signal gašenja pumpama, ali se otvara ventil Y_1.

Zanimljivo za razmatrati jeste slučaj kada se nivo vode nalazi izmedju detekcije senzora B_1 (B_4) i B_2. Mi tada nismo sigurni da li tečnost ni u jednom momentu nije došla do senzora B_2, ili je došla, ali zbog rada ispumpavanja spustila. Medjutim, ako je tečnost već posjetila senzor B_2, bar jedna od pumpi će biti upaljena, te ćemo tako razlikovati prethodno pomenuta dva slučaja. U tu svrhu koristimo logičko "NILI" kolo, čija tabela izgleda kao

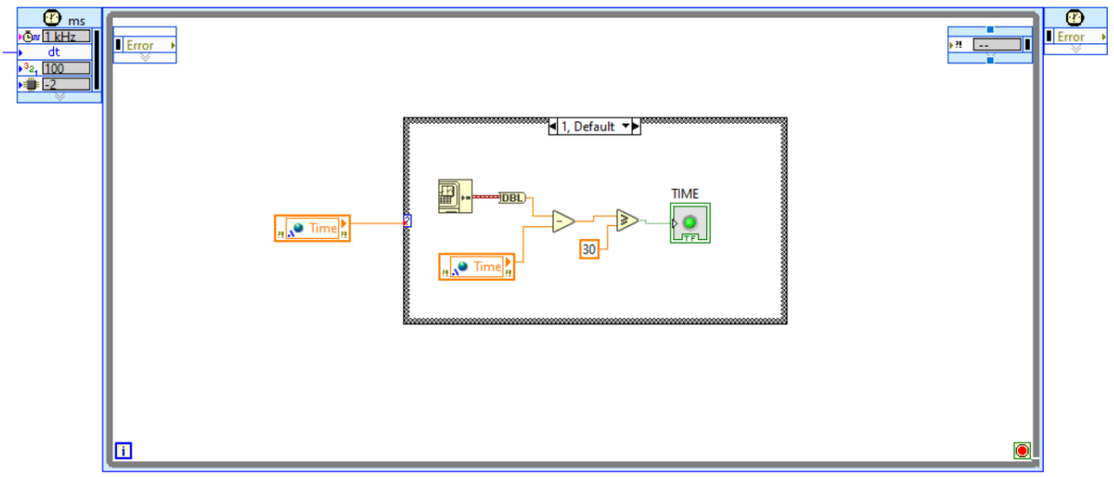
Pumpa 1 - aktivnost	Pumpa 2 - aktivnost	Rezultat operacije
Pumpa nije upaljena (0)	Pumpa nije upaljena (0)	Nismo nikako došli do senzora B_1 (1)
Pumpa je upaljena (1)	Pumpa nije upaljena (0)	Već smo bili do senzora B_1 (0)
Pumpa nije upaljena (0)	Pumpa je upaljena (1)	Već smo bili do senzora B_1 (0)
Pumpa je upaljena (1)	Pumpa je upaljena (1)	Već smo bili do senzora B_1 (0)

Slika 18: *Logička reprezentacija opisanog slučaja*



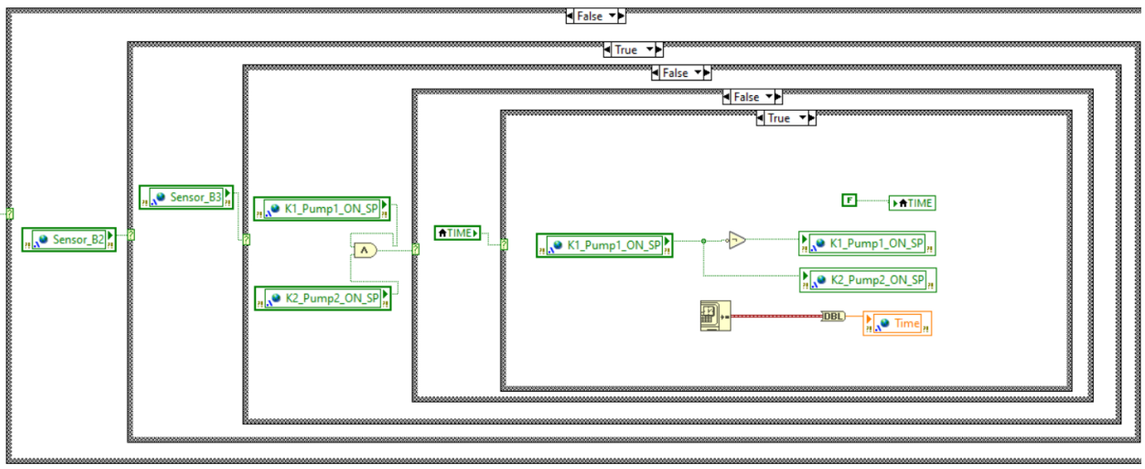
Slika 19: *Slučaj kada tečnost nijednom nije bila na senzoru B_2*

Pričali smo o tome da je potrebno da pumpe rade naizmjenično. Takva naizmjeničnost izvedena je zapisivanjem vremena početka rada jedne pumpe u promjenljivu *Time*, te provjerom da li je prošlo dovoljno vremena da se pumpe "zamijene". Provjera se vrši oduzimanjem od trenutnog vremena i ako je ta razlika veća od zadatih sekundi smjene, dolazi do aktivacije indikatora *TIME*. Tajmer "trči" u zasebnoj *Timed Loop* strukturu, kako bi paralelno radio sa ostatkom automatskog sistema.



Slika 20: *Provjera vremena*

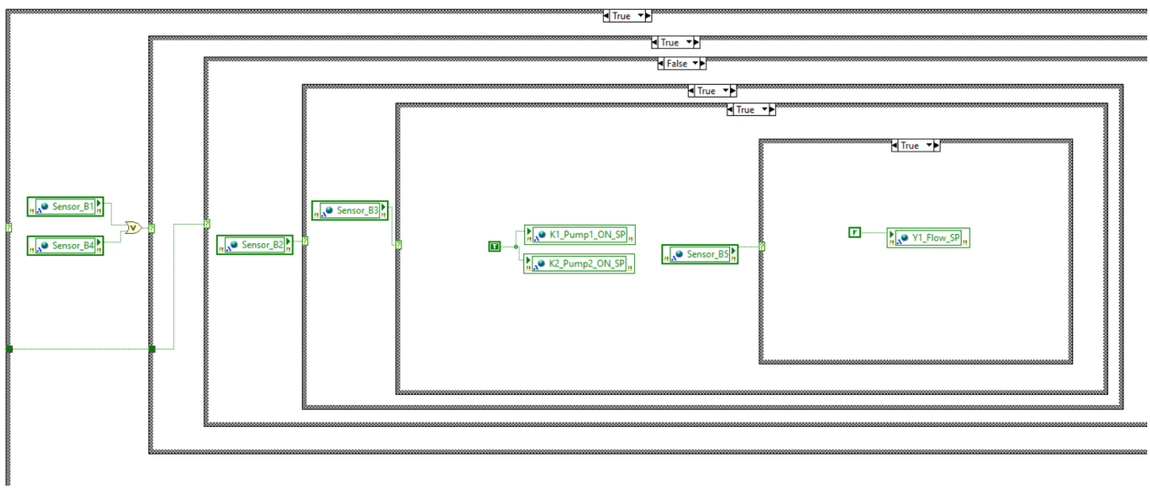
Naravno, potrebno je da pri naizmjeničnom radu pazimo da li je došlo do aktivacije senzora B_3. Ako nije, provjeravamo da li je indikator *TIME* aktiviran, te u skladu sa time mijenjamo rad pumpi.



Slika 21: *Naizmjeničan rad pumpi ostvaruje se invertovanjem stanja*

Važno je napomenuti da se inverzija rada pumpi radi i kada se tečnost nalazi izmedju detekcije senzora B_2 i B_3. Kod i logika za pomenuto su isti.

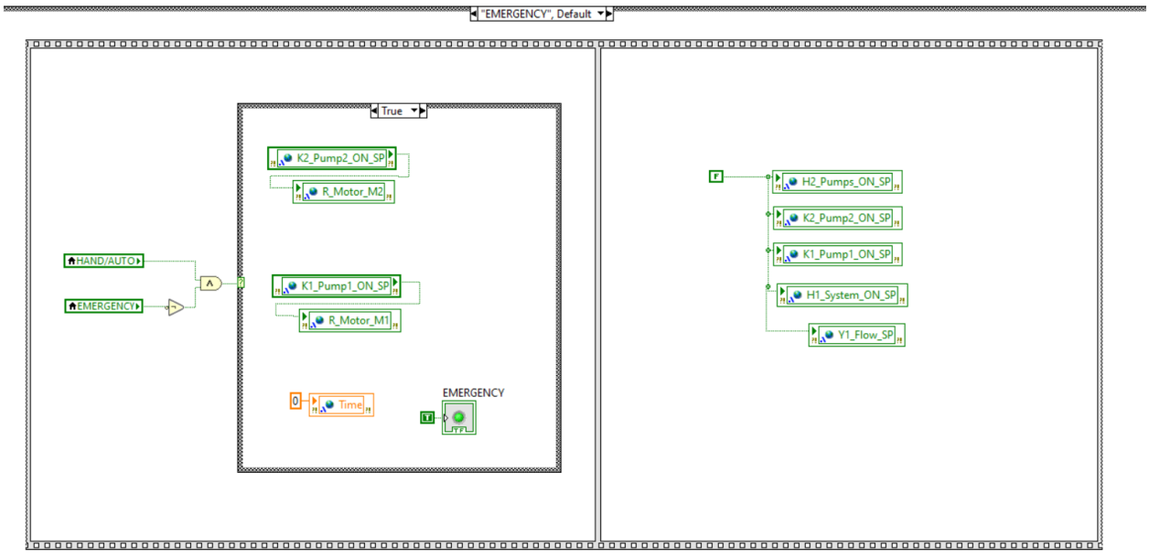
Ako je pak došlo do aktivacije senzora B_3, obje pumpe kreću sa radom, te se provjerava da li je tečnost dostigla i nivo B_5 - ako jeste, zatvara se regulacioni ventil i pušta se totalno pražnjenje rezervoara.



Slika 22: *Reagovanje na aktivaciju senzora B_3 i B_5*

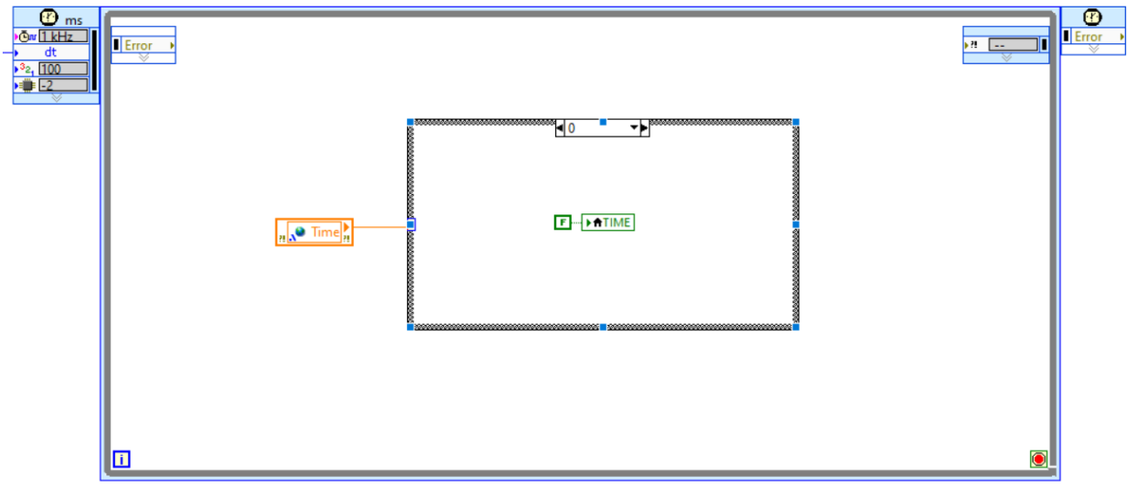
Sistem automatske regulacije radi kontinualno, provjeravajući stanje senzora i kotirajući se po tome.

Poslednje stanje koje je potrebno razmotriti jeste *Emergency* stanje. Kao što je i očekivano, stanje je dužno da ugasi sve pumpe i zapamti prethodna dešavanja u sistemu. Takodje, bitno je da se ne vrši konstantno pamćenje stanja jer će "prava memorija" sistema biti izbrisana, te u tu svrhu se uvodi slična logička provjera kao kod prelaza u automatskom režimu.



Slika 23: *Emergency slučaj*

Takodje, potrebno je postaviti vrijednost promjenljive *Time* na 0, što će u drugoj petlji izazvati gašenje tajmera koji ne bi trebalo bespotrebno da broji dok se sistem nalazi u *Emergency* stanju.



Slika 24: *Gašenje tajmera*