Uzorkovanje i izračunavanje statističkih podataka

Dadić, Danijel

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Stručni studij

UZORKOVANJA I IZRAČUNAVANJE STATISTIČKIH PODATAKA

Završni rad

Danijel Dadić

Osijek, 2020.

SADRŽAJ

1. UVOD

Bitna stavka današnje industrije je posjedovanje informacije na svakoj razini poslovanja, kako na samoj operativnoj razini tako i na višim izvršnim razinama. Prikupljanje podataka o obilježjima svih jedinica statističkog skupa često je preskupo ili zahtijeva previše vremena, a ponekad nije ni moguće. Pojam koji se tu javlja je uzorkovanje. Uzorkovanje je metoda prikupljanja informacija o cijeloj populaciji bez ispitivanja svakog pojedinog člana populacije. Populacija ili statistički skup je skup stvari, osoba, pojava ili drugih objekata, čija svojstva promatramo ili istražujemo statističkom metodom. Uzorkovanje je pojam koji je proizašao iz statistike, a statistika kao znanost označava disciplinu koja proučava metode prikupljanja, sređivanja, analize i tumačenja podataka [1]. U tu svrhu u okviru ovoga završnoga rada razvijen je programski blok koji vrši uzorkovanje te izračun statističkih podataka. Program je razvijen u TIA Portal programskom okruženju. Programski blok bio bi primjenjiv u raznim postrojenjima. Primjerice ako je riječ o prehrambenoj industriji, bitnu ulogu imaju klimatski faktori. Takva postrojenja za mjerno osjetilo koriste senzore temperature. Glavna uloga temperaturnih senzora u takvim postrojenjima je regulacija protoka zraka. Ovisno o zahtjevima proizvodnje mjerno osjetilo šalje mjerni signal glavnom računalu. Glavno računalo potom regulira protok zraka te hlađenje odnosno zagrijavanje zraka. U takvom ili sličnom postrojenju bi bilo moguće koristiti izrađeni programski blok, povezivanjem senzora temperature s PLC-om, te postavljanjem nadgledane veličine kao ulaznu veličinu funkcijskoga bloka.

U drugom poglavlju su iznešeni temeljni pojmovi teorije informacije, koji su bitni za ovaj rad, treće poglavlje opisuje princip rada programskih blokova TIA portal-a, četvrto poglavlje opisu rad razvijenog programskog bloka, Peto poglavlje prikazuje testiranje programskoga bloka, zajedno s pripadajućom vizualizacijom.

1.1 Zadatak završnoga rada

Potrebno je razviti programski blok za uzorkovanje i izračunavanje osnovnih statističkih podatataka korištenjem programskoga paketa TIA portal te dodatka Simatic Step 7. Osnovni podatci koji se izračunavaju su minimalna i maksimalna vrijednost, standardna devijacija, aritmetička sredina, te broj uzoraka. Uz programski blok, također je potrebno razviti vizualizaciju. Vizualizacija je napravljena u programskome dodatku WinCC.

2. TIPOVI I PRETVORBA SIGNALA

Varijable sustava kao vremenske funkcije često se u inženjerskoj literaturi nazivaju signalima. Signalom se općenito smatra fenomen koji nosi neku informaciju. U sustavu relevantne varijable nose informaciju o procesima u sustavu. Varijable u fizikalnom sustavu su fizikalne veličine kao što su napon, struja, pritisak, ubrzanje, itd [2]. U teoriji informacija signal je poruka koja se može kodirati, tj. niz stanja u komunikacijskom kanalu koji predstavlja poruku. U komunikacijskom sustavu odašiljač kodira poruku u signal, koji se, nadalje, prenosi do prijemnika. Matematički se predstavlja kao funkcija jedne ili više nezavisnih varijabli. Obradba signala je djelovanje s ciljem da se ulaz (npr. sirovina u tvornici ili radio signal) pretvori u neki drugi oblik (npr. gotovi proizvod ili zvuk iz zvučnika) koji je svrishodan. Obradba signala je analiza, interpretacija i rukovanje signalom.

Prema podjeli, signali mogu biti analogni i digitalni. U elektrotehnici analogni signal predstavlja vrijednost električnog napona ili električne struje. Izgled signala poznat je u svakom vremenskom trenutku te se lako grafički prikazuje. Modeliranje sustava izvršava se raznim analognim tehnikama. Primjerice, analizom signala odziva pri različitim pobudama dobivamo jasnu sliku o djelovanju samoga sustava.

Digitalni signal u elektronici je razina napona ili struje, čija se vrijednost može mijenjati samo u određenom ograničenom broju stanja ili koraka. Amplituda signala može imati samo neki ograničeni broj vrijednosti, za razliku od analognoga signala. Primjeri digitalog signala su stanja prekidača („uključen“ ili „isključen“), binarni kod u računalima („0“ ili „1“), Morzeov kod i drugi [3].

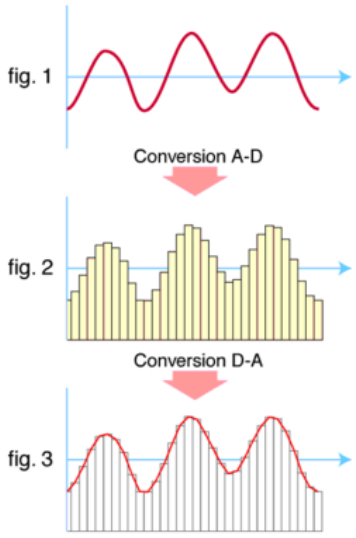
2.1. Kontinuirani i diskretni sustavi

Sve što se događa oko nas, ovisno je o vremenu, u funkciji je vremena. To se odnosi i na sve signale sustava vođenja, bez obzira radi li se o ulaznim ili izlaznim signalima. U ovom poglavlju obrađuje se razlika između vremenski promjenjivih signala koje nazivamo kontinuirani signali i vremenski promjenjivih signala koje nazivamo diskretni signali. U digitalnim sustavima vođenja bar jedan od signala mora biti diskretan, pa je za digitalne sustave vođenja pojam diskretizacija po vremenu posebno važan [4].

Digitalni sustavi rade s uzorcima, odnosno diskretnim vrijednostima signala. Ovo je posljedica same prirode digitalnih sustava, koji za razliku od analognih sustava imaju neka ograničenja. Digitalni sustavi imaju ograničenu memoriju, što znači da mogu spremati signale

ograničene dužine. Način spremanja signala temelji se na spremanju uzoraka, koji su predstavljeni u binarnom zapisu.

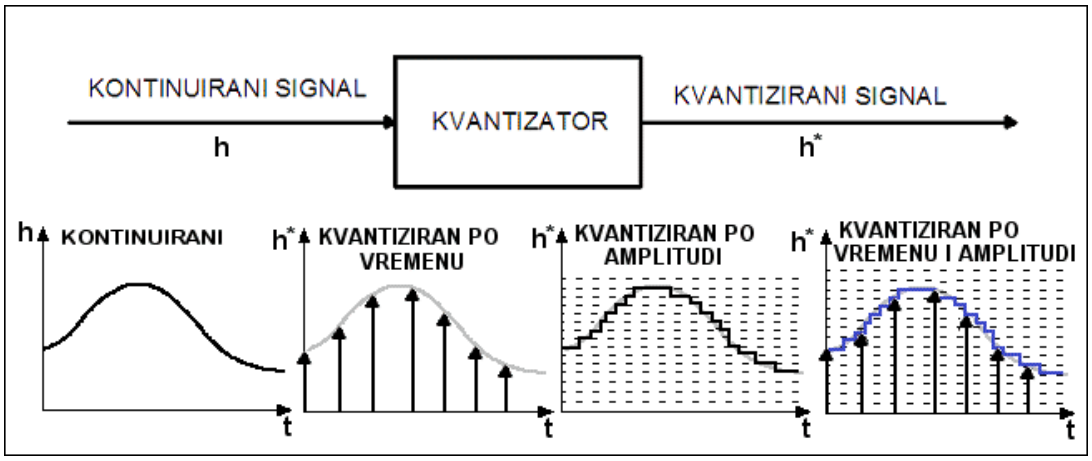
Analogno-digitalna pretvorba gotovo je redovni postupak elektroničke obradbe informacija. Područja koja su po prirodi analogna (zvuk, slika, mjerni podatci u sustavu i sl.) također često zahtjevaju analogno-digitalnu pretvorbu. Analogni signal je pri obradbi znatno podložniji negativnim utjecajima i smetnjama. Stoga je poželjno informaciju imati u digitalnom obliku. Pretvorba se vrši analogno-digitalnim pretvornikom.



Slika 2.1. A/D i D/A pretvorba

https://hr.wikipedia.org/wiki/Analogni_signal

Postupak pretvorbe analognoga u digitalni signal još se naziva diskretizacija. Pojam koji se javlja pri diskretizaciji je kvantizacija. Kvantizacija se može obaviti po vremenu, po amplitudi ili po vremenu i amplitudi. Uzorkovanje je postupak pretvorbe kontinuiranog signala u slijed brojčanih vrijednosti koje predstavljaju vrijednost signala u točno određenim vremenskim trenucima.

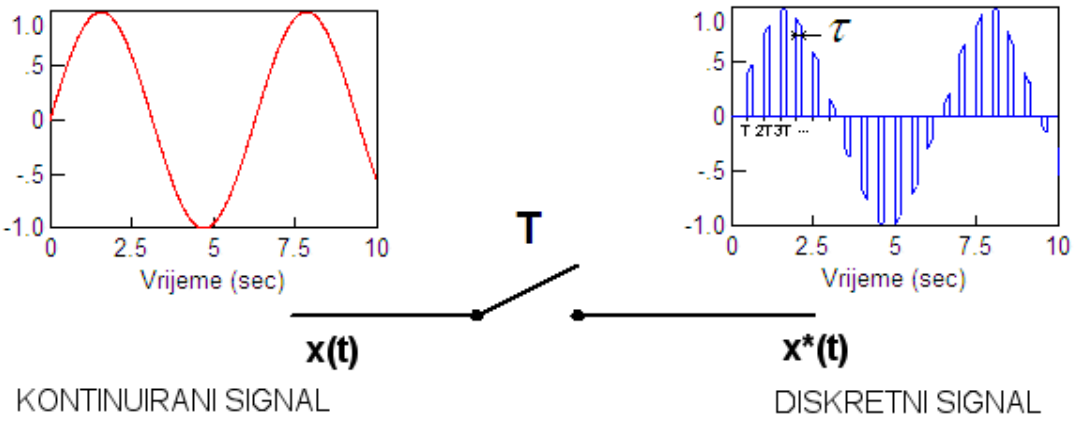


Slika 2.2. Metode kvantizacije

http://laris.fesb.hr/digitalno_vodjenje/text_2.htm

2.2. Uzorkovanje

Sklop u kojem se uzorkovanje provodi, također se često naziva *„sampler“*. Slika 2.3. prikazuje shemu sklopa za uzorkovanje, slovo **T** simbolizira njegov ciklički rad u jednakim periodima. Sklop ostaje uključen konačni period vremena τ.



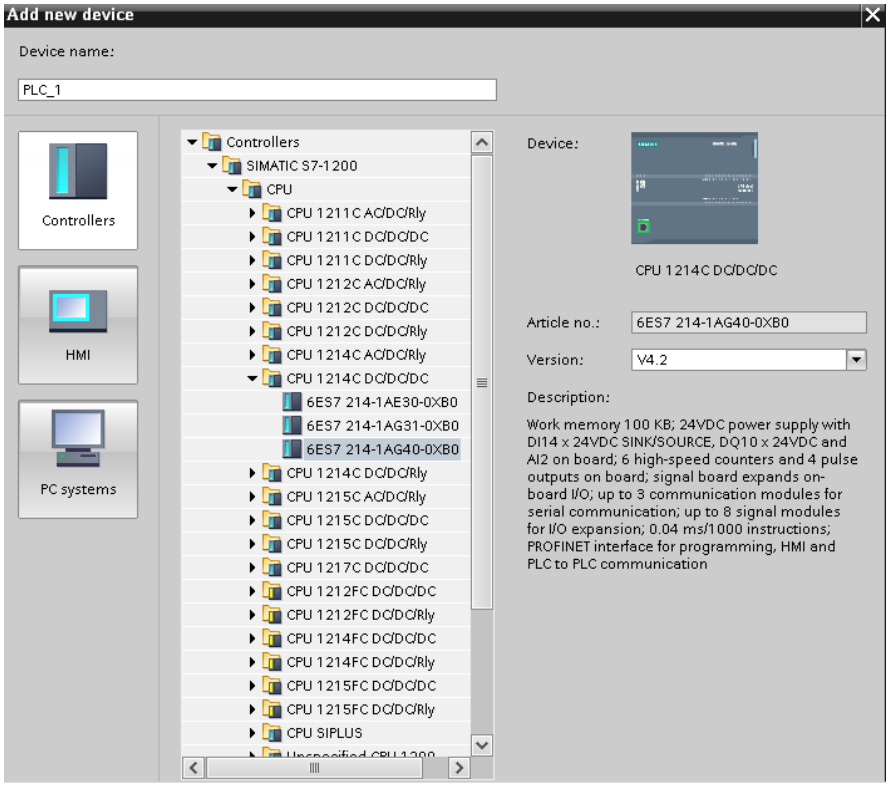
Slika 2.3. Shematski prikaz sklopa za uzorkovanje

http://laris.fesb.hr/digitalno_vodjenje/text_2-3.htm

3. PROGRAMSKI BLOKOVI TIA PORTALA

Totally integrated Automation Portal (TIA Portal) objedinjuje SIMATIC Totally Integrated Automation (TIA) proizvode u jedinstvenu programsku aplikaciju. Svi TIA proizvodi rade zajedno unutar istog programskog okruženja i pružaju podršku korisniku u svim segmentima potrebnim za stvaranje rješenja automatizacije. TIA Portal se koristi za konfiguraciju i programiranje PLC uređaja, ali i za vizualizaciju procesa u jedinstvenoj razvojnoj okolini. Svi podaci se spremaju u zajedničku projektnu datoteku. Koristi se zajedničko korisničko sučelje preko kojeg je moguće pristupiti svim programskim i vizualizacijskim funkcijama u svakom trenutku [7].

Prvi korak projekta je dodavanje PLC uređaja. Nakon odabira tipa procesora potrebno je dodati I/O modul. Prilikom dodavanja pojedinih modula potrebno je voditi računa o njihovom rasporedu te o verziji dodanih modula. Kod dodavanja sklopovlja, odnosno modula potrebno je odabrati ispravan modul i njegovu verziju. Ukoliko je dodan pogrešan modul i/ili verzija pojaviti će se greška. Nakon otvaranja novoga projekta, klikanjem na „Add new device“ dodaje se (PLC) „uređaj“.

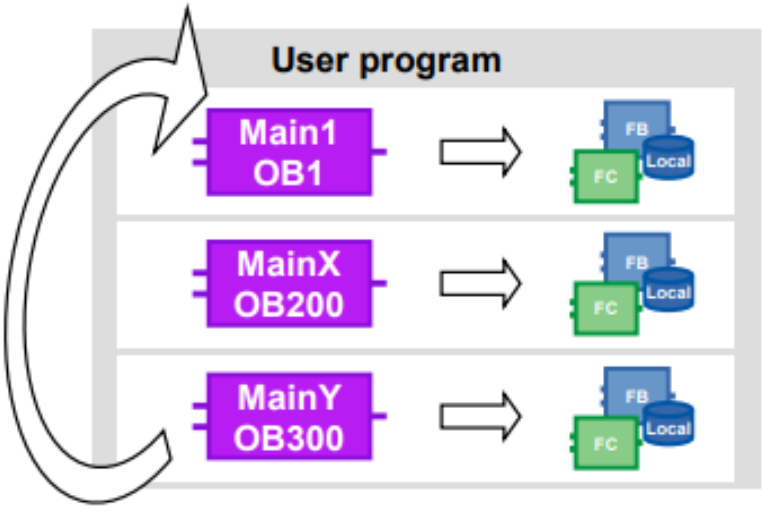


Slika 3.1. Odabir PLC modula TIA

Kontrolni modul (CM) predstavlja najmanji dio sustava upravljanja. Kontrolni modul se prema ANSI/ISA-88 standardu naziva komponenta. Definicija „CM-a“ prema ANSI/ISA-S88 glasi: „Kontrolni modul je u pravilu jedan ili grupa senzora, aktuatora, ostalih kontrolnih modula, i povezane procesne opreme, koji sa stajališta upravljanja djeluje kao zaseban uređaj“ [7].

3.1. Organizacijski blokovi (OB)

Cijela logika upravljačkih programa uobičajeno je smještena u organizacijski blok „OB1“. To je ujedno i glavni blok cijeloga upravljačkog programa iz razloga što se svi ostali programski blokovi upravo pozivaju iz bloka „OB1“ [8]. Organizacijski blokovi su sučelje između operacijskog sustava i upravljačkog programa, operacijski sustav ih poziva, a njihova zadaća je upravljanje kontrolera, cikličko izvođenje programa, upravljanje prekidima, te pronalazak pogrešaka. Ovisno o PLC-u dostupno je nekoliko različitih organizacijski blokova. Mogućnost korištenja većeg broja „Main OB-a“ također postoji, ali je nepoželjno. Komunikacija između pojedinih organizacijskih blokova je nepoželjna. Međutim, ukoliko se koristi veći broj organizacijskih blokova bitno je da oni ne koriste iste podatke, što omogućava njihov neometan rad.

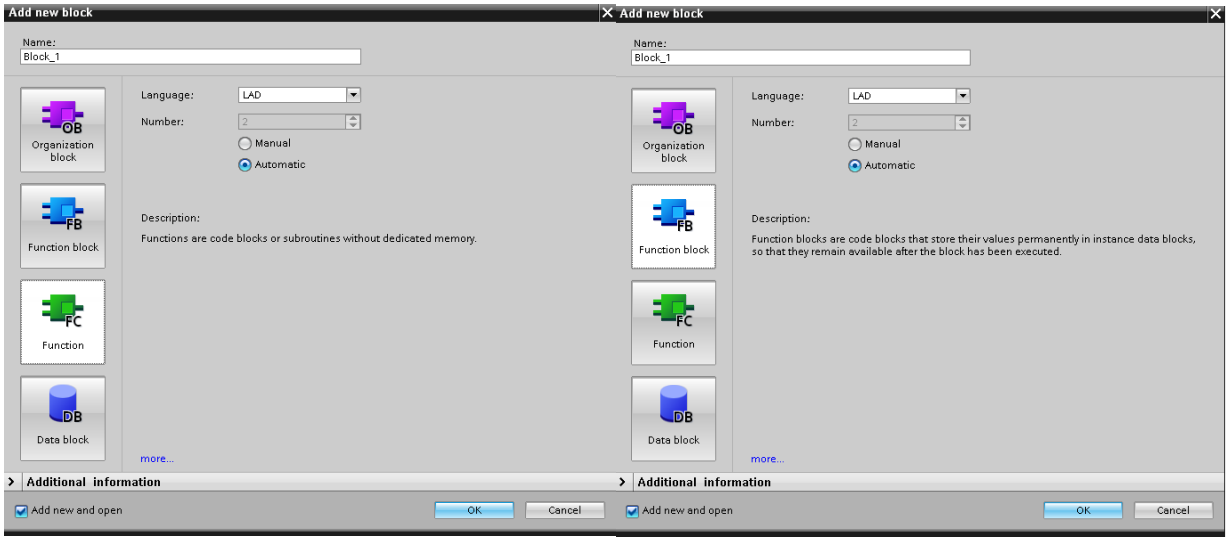


Slika 3.2. Korištenje više „OB-a“

https://cache.industry.siemens.com/dl/files/040/90885040/att_970576/v1/81318674_Programming_guideline_DOC_v16_en.pdf

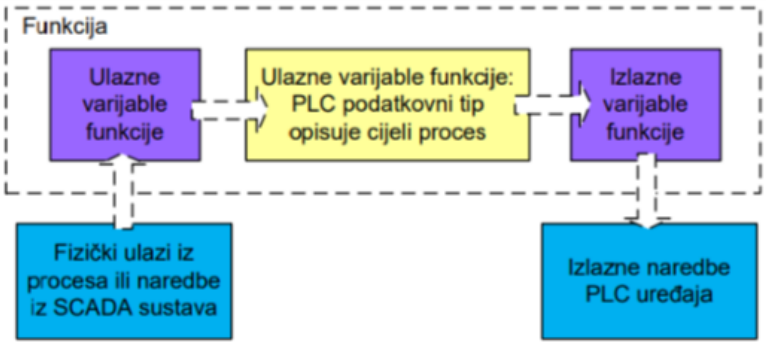
3.2. Funkcije (FC)

Kontrolni moduli TIA portala predstavljaju funkcije ili funkcijske blokove. Funkcije ili funkcijski blokovi predstavljaju pojedine dijelove upravljačkog programa koji se zasebno konfiguriraju te pozivaju u glavnom organizacijskom bloku „OB1“. Razlika funkcijskih blokova u odnosu na funkcije je korišteni tip podatkovnog bloka. Prilikom stvaranja novog funkcijskog bloka potrebno je i kreirati podatkovni blok za taj funkcijski blok. Funkcije za razliku od funkcijskih blokova nemaju pripadajuće vlastite podatkovne blokove, nego koriste globalne podatkovne blokove. Jedna od dobrih strana korištenja globalnih podatkovnih blokova je mogućnost dijeljenja podatkovnog prostora jednoga bloka sa većim brojem funkcija. Funkcije i funkcijski blokovi dodaju se klikanjem na „Add new block“, unutar „Program blocks“ na lijevoj strani alatne trake.



Slika 3.3. Dodavanje nove funkcije / funkcijskog bloka

Funkcije omogućuju izradu univerzalnih blokova koji se pozivaju u „OB1“ i tamo im se pridružuju fizičke adrese. Sva logika je sadržana u bloku koji kada pozovemo u glavnom programu ima vidljive izvode za fizičke ulaze i izlaze. Kao što je već spomenuto, nemaju vlastitu memoriju, nego se podaci spremaju preko PLC podatkovnog tipa koji sadrži varijable cijelog procesa. Ulazne varijable se upisuju pod „*Input*“, izlazne pod „*Output*“, ulazno-izlazne pod „*InOut*“, dok se privremene upisuju pod „*Temp*“. Odnos između tipova varijabli najbolje prikazuje slika 3.4.



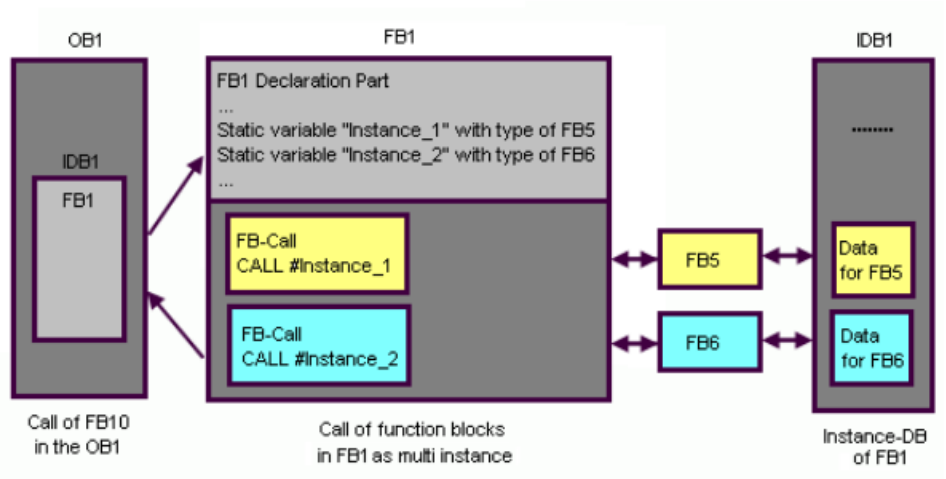
Slika 3.4. Blokovska shema toka podataka u funkciji

Slika preuzeta iz: „Izrada programskih komponenti u TIA Portal programskom okruženju“, Danijel Maršić, Goran Malčić i Ivica Vlašić, TVZ Zagreb

Sadržaj varijabli deklariranih kao *„Input“* treba se kopirati u odgovarajuće varijable u *„InOut“*. Isto tako sadržaj odgovarajućih varijabli iz *„InOut“* treba se kopirati u varijable u *„Output“* funkcije.

3.3. Funkcijski blokovi (FB)

Funkcijski blokovi za razliku od funkcija imaju pripadajuću memoriju, odnosno *„instance data block“* (IDB). Funkcijski blokovi ciklički spremaju podatke, te ih permanentno zadržavaju. Koriste se za stvaranje podprograma i stuktura. Mogu biti pozivani s više lokacija glavnoga programa, a između svakoga poziva zadržavaju svoje vrijednosti. Poželjno je koristiti odvojene *„instance“* ako ih se poziva s više lokacija u glavnome programu, upravo za tu svrhu je omogućeno korištenje *„multi-instance“* unutar funkcijskog bloka. *„Multi-instance“* doprinosi univerzalnosti funkcijskih blokova, jer se prilikom svakog poziva, ovisno o lokaciji na kojoj je pozvan, obrađuju podaci za pripadajuću lokaciju, te ovisno o ulaznim podacima na izlaze se šalju izlazni podatci. Ulazne varijable FB se spremaju kao *„Input“*, izlazne kao *„Output“*, a ulazno-izlazne kao *„InOut“*. Pod *„Static“* se upisuju varijable koje se obrađuju isključivo unutar funkcijskog bloka.

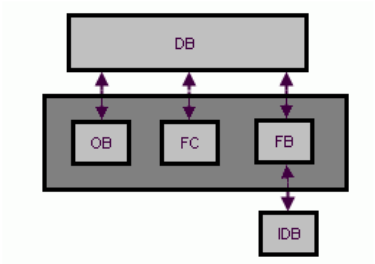


Slika 3.5. Primjer pozivanja „multi-instance“ „FB1“ u „OB1“.

https://support.industry.siemens.com/cs/document/15360455/what-is-the-difference-between-an-instance-data-block-and-a-global-data-block-and-how-does-a-call-call-influence-the-db-register-?dti=0&lc=en-WW

3.4. Podatkovni blokovi (DB), (IDB)

Razliku između globalnih podatkovnih blokova i „*instance*“ podatkovnih blokova čini mogućnost čitanja i pisanja podataka pojedinih blokova. Globalne podatkovne blokove mogu čitati i pisati svi drugi blokovi (fukcijski blokovi, funkcije, organizacijski blokovi). „*Instance*“ podatkovni blok je dodijeljen pojedinom funkcijskom bloku, njegov sadržaj može čitati i pisati samo taj jedan funkcijski blok. Prilikom dodavanje funkcijskog bloka „FB“ nužno je kreiranje i „*instance*“ podatakovni blok „IDB“ pripadajućem funkcijskom bloku [9].



Slika 3.6. Mogućnost pristupa podatkovnim blokovima

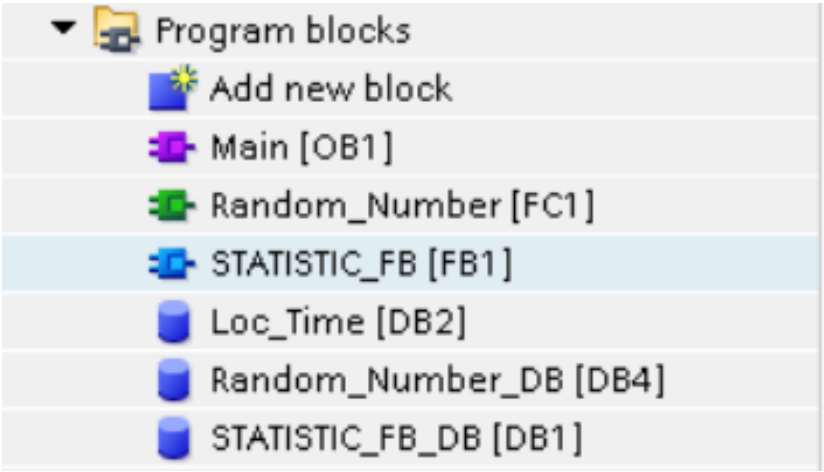
https://support.industry.siemens.com/cs/document/15360455/what-is-the-difference-between-an-instance-data-block-and-a-global-data-block-and-how-does-a-call-call-influence-the-db-register-?dti=0&lc=en-WW

4. PROGRAMSKI BLOK ZA UZORKOVANJE I IZRAČUN STATISTIČKIH PODATAKA

Zadatak završnoga rada bio je izraditi programski blok za uzorkovanje i statističku obradu podataka. Programski blok izrađen je korištenjem programskoga paketa TIA portal v15.1, a pripadajuća vizualizaciju s WinCC dodatkom. Zamišljen je kao sklop za uzorkovanje koji ovisno o želji korisnika u zadanim intervalima pohranjuje nasumičnu vrijednost u polje elemenata. Ulazna vrijednost izvedena je funkcijom, a može biti zamišljena kao vanjska veličina kao npr. temperatura sobe.

4.1. Rad programskog bloka za uzorkovanje

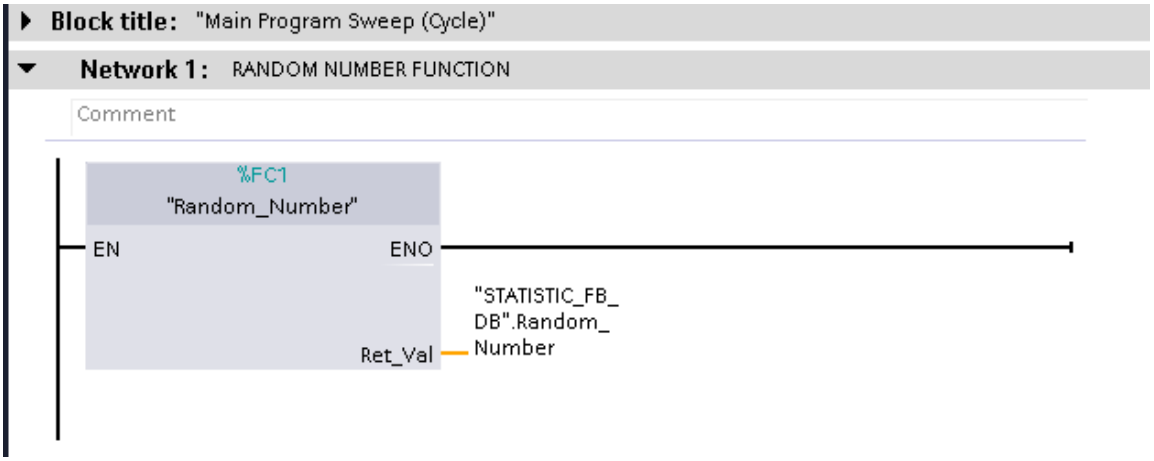
Razvijeni programski blok temelji se na primjeni funkcije, funkcijskoga bloka, organizacijskoga bloka, te podatkovnih blokova. Glavni organizacijski blok „OB1“ poziva funkciju „FC1“, te funkcijski blok „FB1“. Funkcija „FC1“ služi generiranju nasumičnoga broja. Funkcijski blok „FB1“ statistički obrađuje te ispisuje vrijednosti koje su dobivene pozivanjem „FC1“.



Slika 4.1.. Korišteni programski blokovi

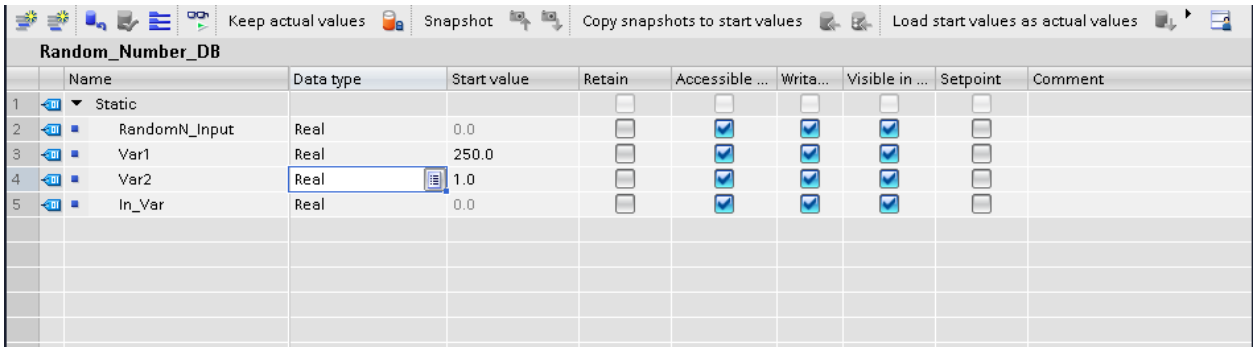
4.2. Funkcija „FC1“

Slika 4.2. prikazuje poziv funkcije „FC1“ u glavnom organizacijskom bloku „OB1“. Vrijednost koju funkcija „FC1“ vraća, sprema se u varijablu Random_Number, čija je adresa pohranjena u podatkovnom bloku funkcijskog bloka „FB1“.

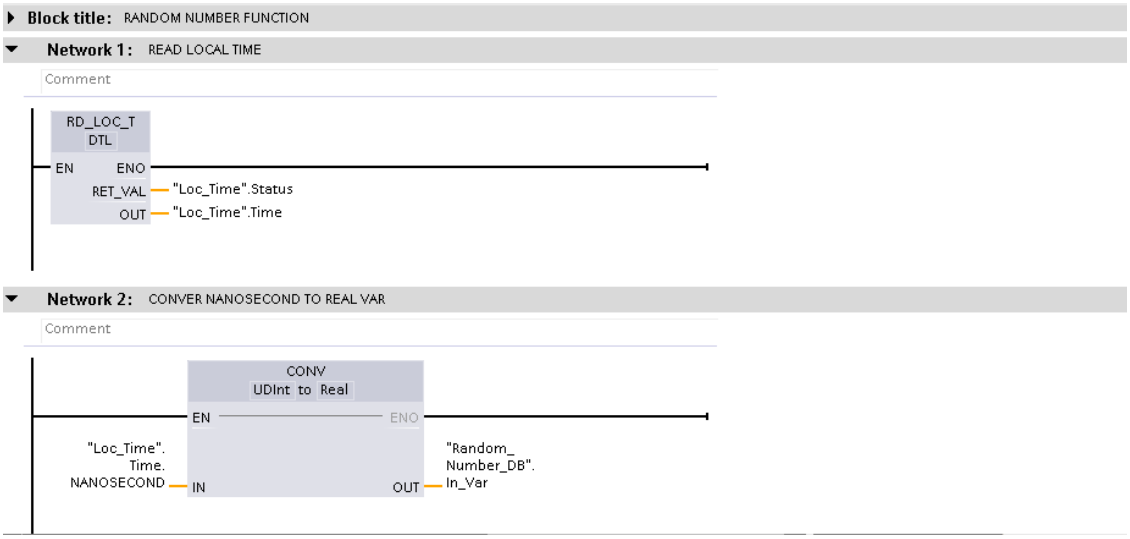


Slika 4.2. Poziv „FC1“ u „OB1“

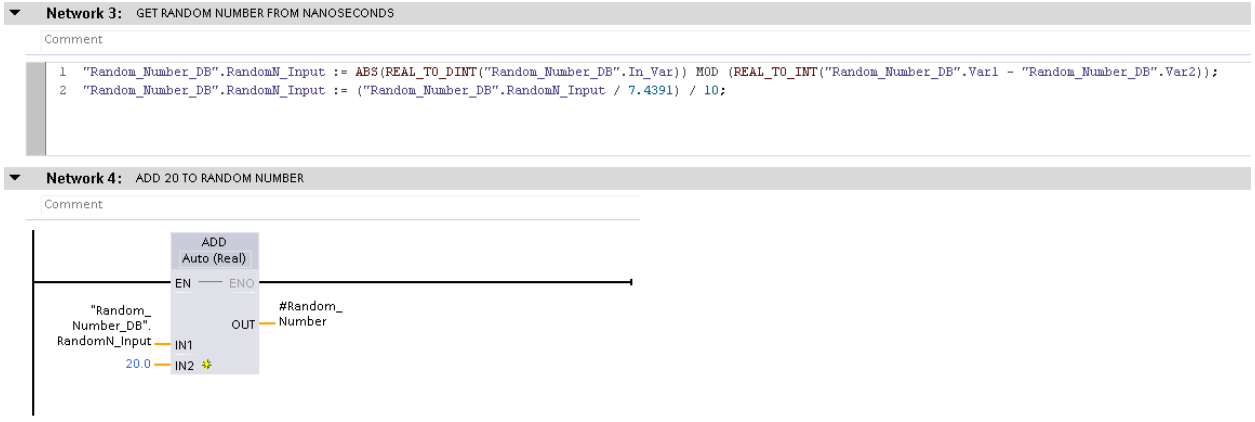
Slika 4.3. prikazuje globalni podatkovni blok sa *„Static“* varijablama koje su tipe *„Real“* i služe generiranju nasumičnog broja. Slika 4.4. i slika 4.5. prikazuju pojedine mreže koje se nalaze unutar funkcije „FC1“. Mreža, odnosno *„Network 1“* i *„Network 2“* služe konverziji varijable Time.NANOSECOND koja je spremljena u globalnom podatkovnom bloku „Loc_Time“ iz UDInt (Unsigned double integer) u „Real“ tip podatka, odnosno realni broj. *„Network 3“* radi matematičku obradu nad varijablom Time.NANOSECOND sa ciljem davanja nasumičnog broja. Dobiveni nasumični broj iznosi približno između 0 i 3, te se u konačnici taj broj zboji sa 20 u *„Network-u 4“*.



Slika 4.3. Globalni podatkovni blok



Slika 4.4. „FC1“ *„Network 1“* i *„Network 2“*



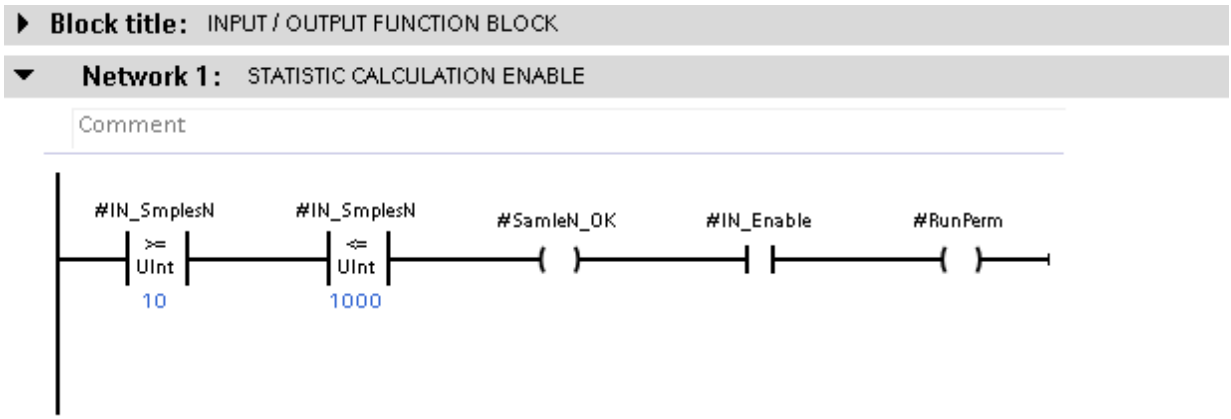
Slika 4.5. „FC1“ *„Network 3“* i *„Network 4“*

Kako je nemoguće izvesti generator nasumičnog broja, dobiveni broj se naziva „*pseudo random number*“. Izveden je korištenjem vrijednosti nanosekunde realnoga vremena. Budući da vrijedi: $1\,[s] = 1\,x\,10^9\,[ns]$, u praksi nije moguće spremati vrijednost nanosekunde u periodima 1000 ms i svaki puta dobiti istu vrijednost nanosekunde, jer se ta vrijednost toliko brzo mijenja da većina današnjih procesora ne uspijeva u istom djeliću sekunde pohraniti tu vrijednost. Dobiveni pseudo nasumični broj predstavlja temelj za statističku obradu opisanu u ovom radu.

4.3. Funkcijski blok „FB1“

„Network 1“ i *„Network 2“* prikazani su na slikama 4.6 i 4.7. Kao i u prehodnim primjerima napisani su „Ladder“ programskim jezikom. Slika 4.8. prikazuje *„Input“* i *„Output“* varijable funkcijskog bloka „FB1“. IN_Enable, In_Value, IN_SmplesN su *„Input“* varijable, a OUT_Max, OUT_Min, OUT_Avr, OUT_Std_dev, Smp_N, Status su *„Output“* varijable.

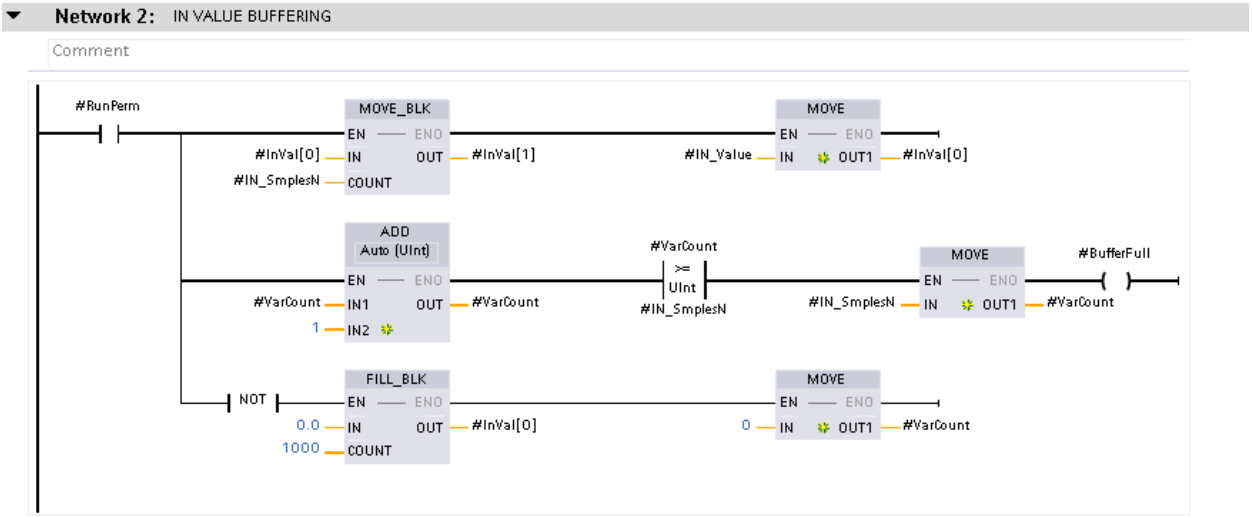
„Network 1“ sastoji se od 2 komparatora, te 2 „Coila“ i jednog *„Normaly closed contact“*. Programski blok ima „data buffer“ koji je „Array“ tipa „Real“ od maksimalno 1000 elemenata, IN_SmplesN mora biti postavljen na vrijednost između 10 i 1000 (elemenata), inače se javlja pogreška. Kada je IN_SmplesN pravilno podešen, varijable SamleN_OK i RunPerm koje su tipa „Bool“ prelaze u „1“.



Slika 4.6. *„Network 1“* „FB1“

Kad je RunPerm u stanju „1“ *„Network 2“* počinje s izvršavanjem, te se varijabla „IN_Value“ koja je predhodno dobivena iz funkcije „FC1“ ciklički pohranjuje u „Array“ nazvan

InVal[0...1000]. Pohrana varijabli izvršava se po „Shift“ principu, tj. svaki novi element koji se pohranjuje u polje dolazi na prvo mjesto polja odnosno dobiva indeks 0, dok se indeksi predhodno pohranjenih elemenata uvećavaju za 1. Time novi elementi „guraju“ stare. Varijabla BufferFull koja se nalazi u *„Network 2“* predstavlja svojevrsni indikator da je „data buffer“ popunjen, time započinje statistička obrada nad podacima u *„Network 3“*.



Slika 4.7. *„Network 2“* „FB1“

STATISTIC_FB

Name	Data type	Default value	Retain	Accessible ...	Writa...	Visible in ...	Setpoint	Comment
Input				☐	☐	☐	☐	
IN_Enable	Bool	false	Non-retain	☑	☑	☑	☐	
IN_Value	Real	0.0	Non-ret...	☑	☑	☑	☐	
IN_SmplesN	UInt	20	Non-retain	☑	☑	☑	☐	Number of samples (10..1000)
Output				☐	☐	☐	☐	
OUT_Max	Real	0.0	Non-retain	☑	☑	☑	☐	
OUT_Min	Real	0.0	Non-retain	☑	☑	☑	☐	
OUT_Avr	Real	0.0	Non-retain	☑	☑	☑	☐	
OUT_Std_dev	Real	0.0	Non-retain	☑	☑	☑	☐	
Smp_N	Int	0	Non-retain	☑	☑	☑	☐	
Status	Int	0	Non-retain	☑	☑	☑	☐	

Slika 4.8. *Input* i *Output* varijable „FB1“

„Network 3“ izveden je “Structure Control Language“ (SCL) programskim jezikom, te je za razliku od „Ladder“ jezika baziran na tekstu. „SCL“ predstavlja idealno rješenje za matematičku obradu nad podacima jer ima mogućnost korištenja petlji što bitno pojednostavljuje aritmetičko-logičku obradu „Array“ podataka. SCL kod prikazan je na slikama 4.9., 4.10. i 4.11. SCL kod također je postavljen u dodatku. *„Static“* varijable koje su korištene uglavnom za međuobradu podataka prikazane su na slici 4.12.

```
#Loop_Lenght := #IN_SmplesN - 1;
//*****************   DB LOGIC / OUTPUT   ******************************//
IF #BufferFull THEN
    #Sum := 0.0;
    #Sum_SD_Num := 0.0;
    #OUT_Min := #InVal[0];
    #OUT_Max := 0.0;

    FOR #j := 0 TO #Loop_Lenght DO
        IF #InVal[#j] > #OUT_Max THEN
            #OUT_Max := #InVal[#j];
        END_IF;                                          //Max Number
        IF #InVal[#j] < #OUT_Min THEN
            #OUT_Min := #InVal[#j];
        END_IF;                                          //Min Number
        #Sum := #Sum + #InVal[#j];
    END_FOR;

    #OUT_Avr := #Sum / #IN_SmplesN;                  //Avr Number

    FOR #j := 0 TO #Loop_Lenght DO
        #SD_Num := #InVal[#j] - #OUT_Avr;
        #SD_Num_Pow := #SD_Num * #SD_Num;
        #Sum_SD_Num := #Sum_SD_Num + #SD_Num_Pow;
    END_FOR;
```

4.9. *„Network 3“* „FB1“ linije(1-25)

```

    #Variance := #Sum_SD_Num / #Loop_Lenght;
    #OUT_Std_dev := SQRT(#Variance);              //Standard Deviation

ELSIF #BufferFull = FALSE THEN
    #OUT_Min := 0.0;
    #OUT_Max := 0.0;
    #OUT_Std_dev := 0.0;
    #OUT_Avr := 0.0;
END_IF;


//STATUS
IF #SamleN_OK THEN

    IF #BufferFull THEN
        #Status := 2;

```

4.10. *„Network 3“* „FB1“ linije(26-43)

```
    ELSIF #BufferFull = FALSE THEN
        #Status := 1;

        IF #IN_Enable = FALSE THEN
            #Status := 0;
        END_IF;

    END_IF;              //Status 0 || Status 1  || Status 2



ELSIF #SamleN_OK = FALSE THEN              //Status 3
    #Status := 3;
END_IF;


#Smp_N := #VarCount;     //Sample Number Counter
```

4.11. *„Network 3“* „FB1“ linije(44-60)

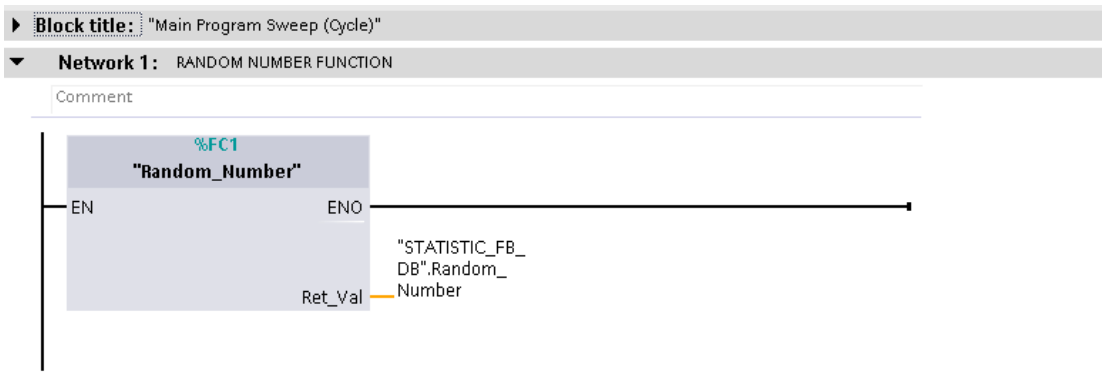
STATISTIC_FB

Name	Data type	Default value	Retain	Accessible ...	Writa...	Visible in ...	Setpoint	Comment
Static								
InVal	Array[0..1001] of R...		Non-retain	☑	☑	☑	☐	
SmpleN_OK	Bool	false	Non-retain	☑	☑	☑	☐	
RunPerm	Bool	false	Non-retain	☑	☑	☑	☐	
VarCount	UInt	0	Non-retain	☑	☑	☑	☐	
BufferFull	Bool	false	Non-retain	☑	☑	☑	☐	
Random_Number	Real	0.0	Non-retain	☑	☑	☑	☐	
j	Int	0	Non-retain	☑	☑	☑	☐	
Sum_SD_Num	Real	0.0	Non-retain	☑	☑	☑	☐	
SD_Num_Pow	Real	0.0	Non-retain	☑	☑	☑	☐	
SD_Num	Real	0.0	Non-retain	☑	☑	☑	☐	
Variance	Real	0.0	Non-retain	☑	☑	☑	☐	
Loop_Lenght	Int	0	Non-retain	☑	☑	☑	☐	
Sum	Real	0.0	Non-retain	☑	☑	☑	☐	
Time	Time	T#500MS	Non-retain	☑	☑	☑	☐	
Avr_Temp	Real	0.0	Non-retain	☑	☑	☑	☐	

Slika 4.12 . Static varijable „FB1“

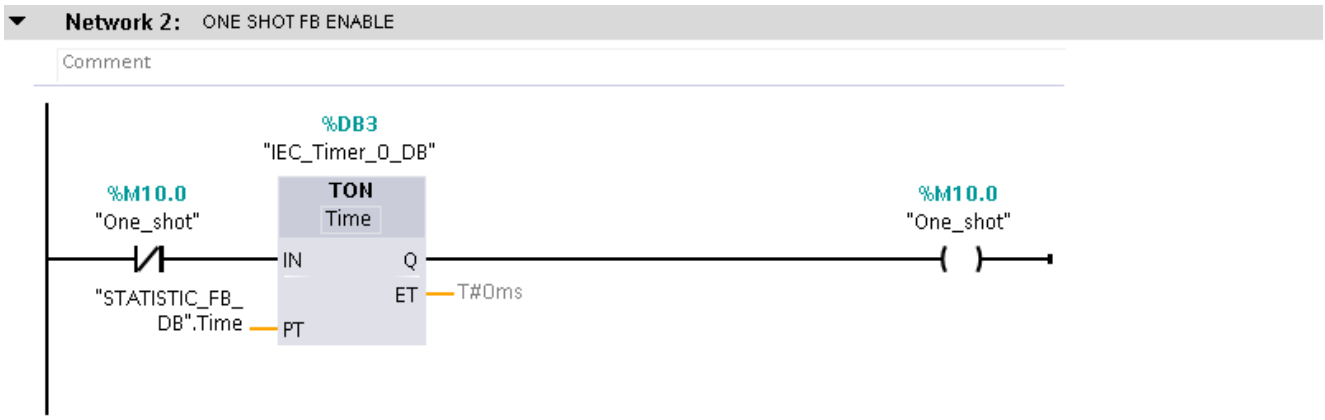
4.4. Organizacijski blok „OB1“

„OB1“ dijeli se na 3 *„Network-a“*. *„Network 1“* prikazan na slici 4.13. poziva funkciju „FC1“, podsjetimo se ona služi za generiranje „nasumičnog“ broja.



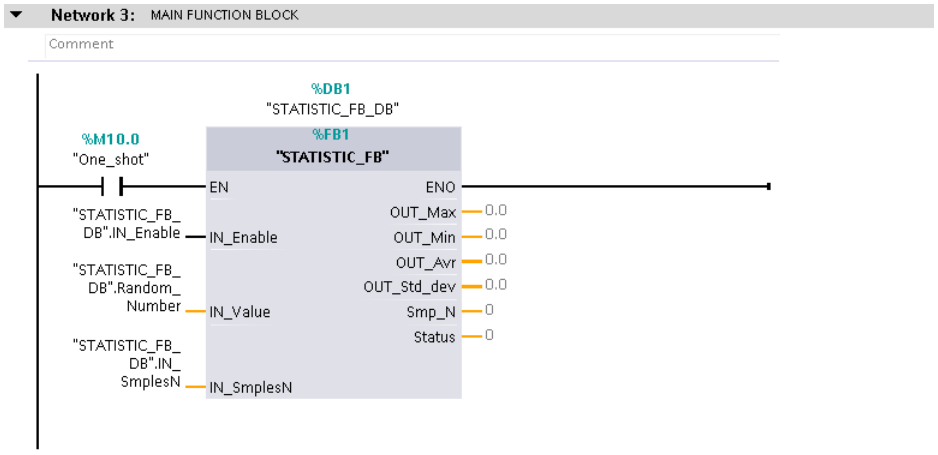
Slika 4.13. *„Network 1“* poziv „FC1“

„Network 2“ prikazan na slici 4.14. predstavlja „one shot“, odnosno programski konfiguriran sklop koji ovisno o želji korisnika daje impuls u željenim intervalima. Drugim riječima, „one shot“ poziva funkcijski blok „FB1“ u zadanim intervalima. Varijabla PT koja je na ulazu u „timer“ definira vrijeme intervala impulsa „one shot-a“. Memorijska lokacija %M10.0 postavljena je kao *„Enable“* u *„Network 3“* odnosno služi pozivanju „FB1“.



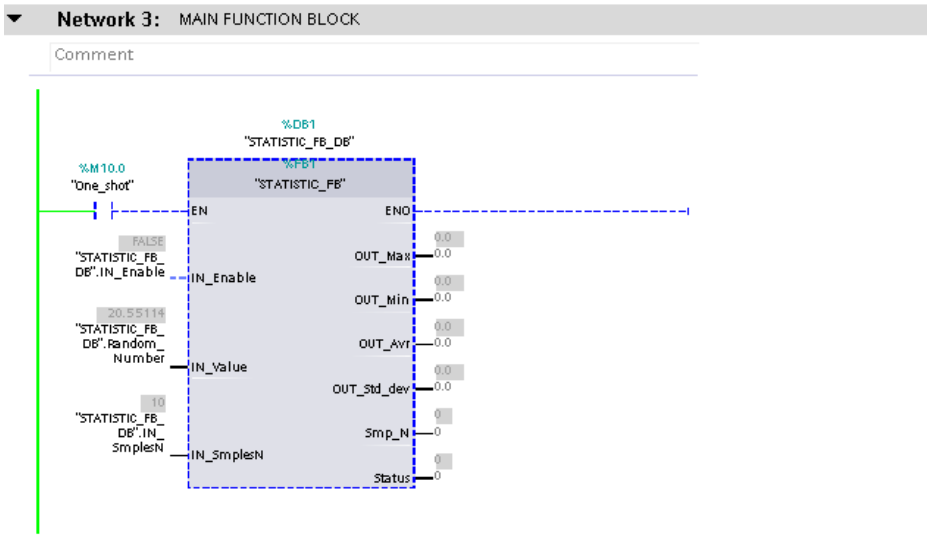
Slika 4.14. *„Network 2“* „One shot“

„Network 3“ prikazan na slici 4.15. je ujedno najvažniji dio upravljačkog programa. On objedinjuje sve ostale dijelove programa u funkcionalnu cjelinu. Kada su svi preduvjeti za uzorkovanje i pohranu podataka ispunjeni, a to su definicija *„input“* varijable IN_smplesN (između 10 i 1000) te definicija vremena intervala poziva „FB1“. Desnim klikom na *„input“* varijablu IN_Enable pokreće se sklop za uzorkovanje. Kada je broj pohranjenih uzoraka dosegao definiranu vrijednost. Sve *„output“* vrijednosti mijenjaju svoju vrijednost iz 0.0, u vrijednost analognu njihovom imenu.



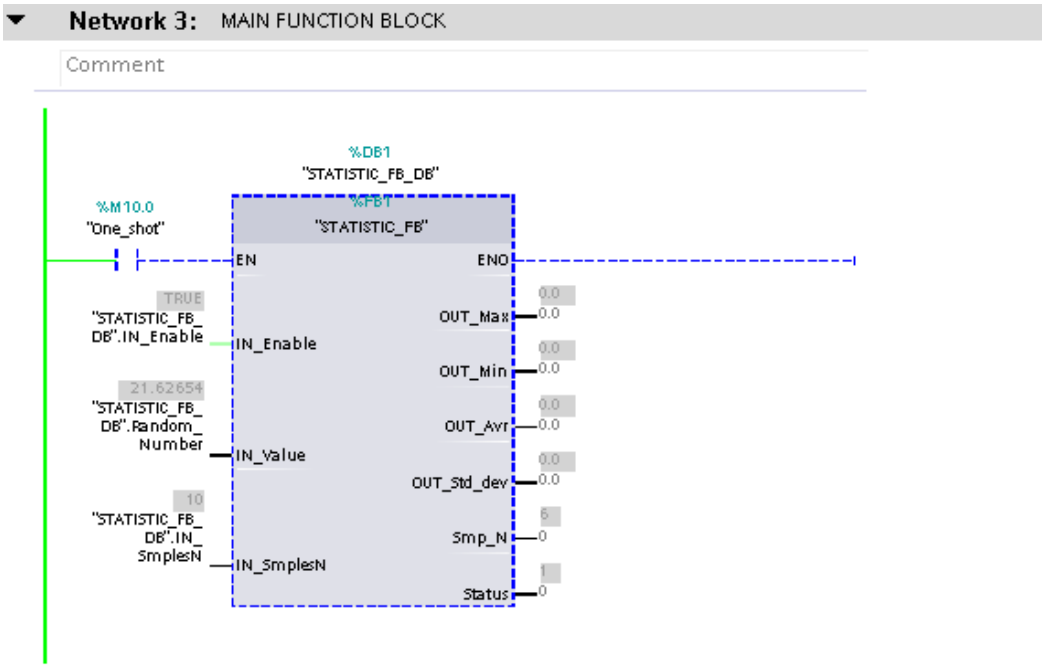
Slika 4.15. *„Network 3“* poziv „FB1“

Slika 4.16. prikazuje izgled *„Network 3“* prilikom pokretanja simulacije, prije pokretanja sklopa za uzorkovanje. Prisjetimo se, sklop se pokreće postavljanjem IN_Enable u stanje „TRUE“.

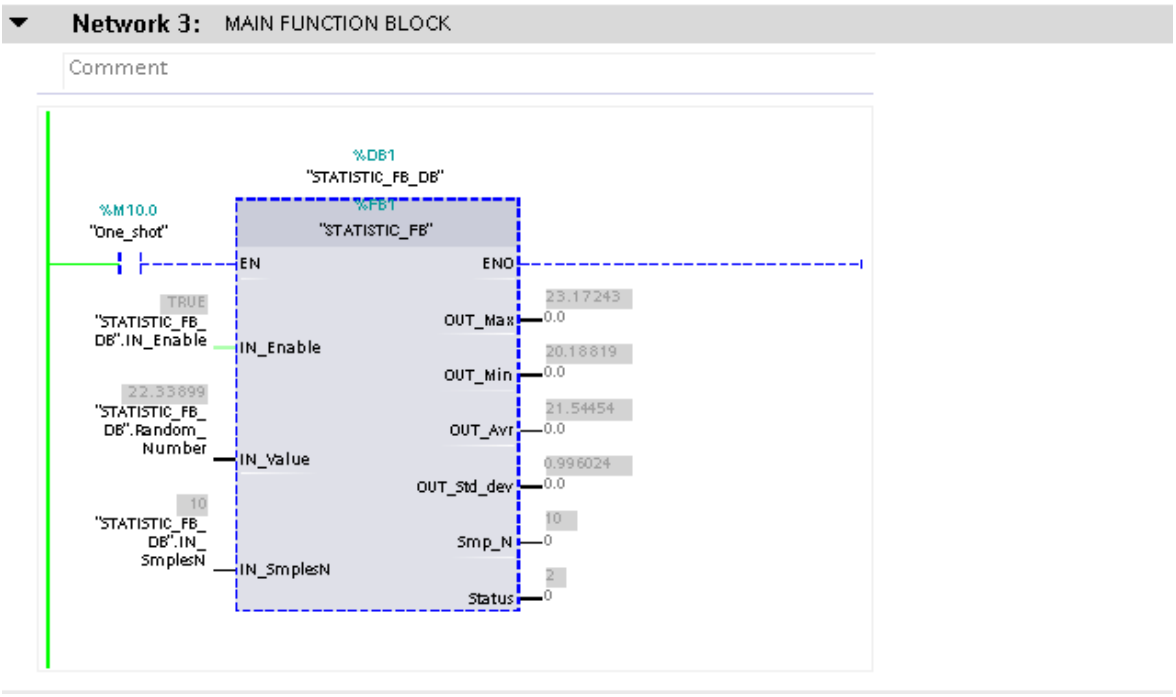


Slika 4.16. „FB1“ pri pokretanju (IN_Enable= FALSE)

Slika 4.17. prikazuje sklop za uzorkovanje prilikom *„buffering-a“*. Sklop je pokrenut te se polje elemenata ciklički popunjava. IN_SmplesN je postavljen na 10, a trenutna vrijednost Smp_N iznosi 6, detaljnije u nastavku.



Slika 4.17. „FB1“ *„Buffering“* (IN_Enable = TRUE)



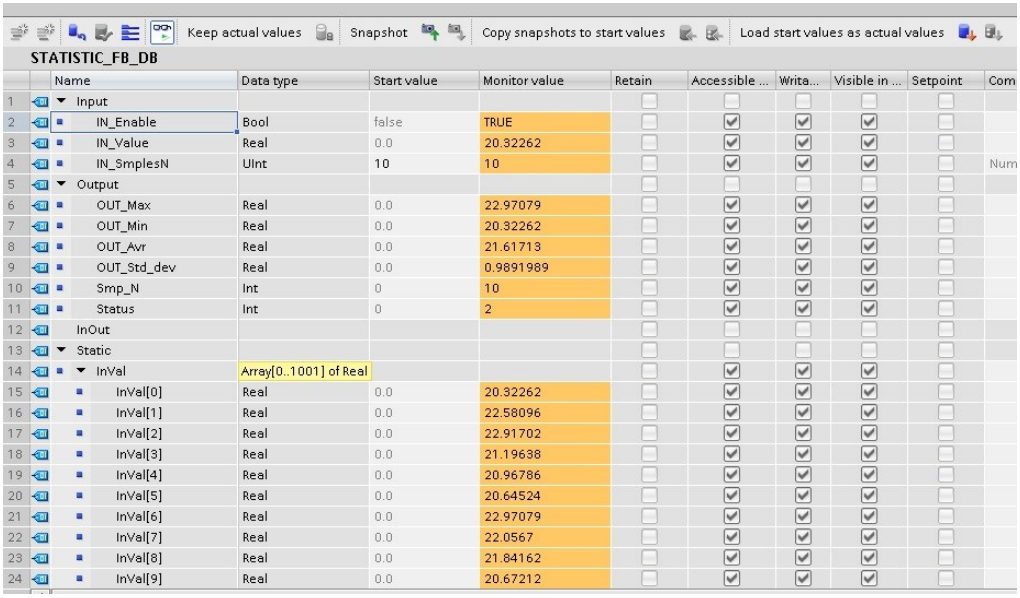
Slika 4.18. „FB1“ *„BufferFull“* (*„Output“* != 0.0)

Slika 4.18. prikazuje rad „FB1“ kada je *„data buffer“* popunjen. Na izlazima se vide vrijednosti statistički izračunatih veličina: OUT_Max, OUT_Min, OUT_Avr, OUT_Std_dev. OUT_Max predstavlja maksimalnu vrijednost, OUT_Min je minimalna vrijednost, OUT_Avr je aritmetička sredina, a OUT_Std_dev standardna devijacija. Izračun se vrši u *„Network 3“* „FB1“. (SCL kod je također dodan u prilog). Postoje dvije dodatne varijable, a to su: Smp_N i Status. Smp_N je vrijednost koja govori koliko *„data buffer“* u tom trenutku ima unešenih vrijednosti. Varijabla Status definira trenutni režim rada programskog bloka. „FB1“ konfiguriran je da ima 4 statusa rada, status „Error“ označava da je *„input“* vrijednost IN_SmplesN izvan definiranih granica između 10 i 1000 elemenata. Tablica 1. prikazuje moguće statuse programskoga bloka za uzorkovanje, s njihovim opisom.

Tablica 1. Statusi programskoga bloka „FB1“

Status	Opis
0	„FB1“ OFF
1	„FB1“ ON, Running/Buffering
2	„FB1“ ON, BufferFull
3	„FB1“ ON, Error

Osim nadgledanja podataka unutar „OB1“ moguće je i pratiti promjene pripadnog podatkovnog bloka „FB1“ (STATISTIC_FB_DB). Prisjetimo se *„instance data block-a“* (IDB). Slika 4.19. prikazuje *„input“*, *„output“* i *„static“* varijable funkcijskoga bloka „FB1“. Promjene unutar podatkovnih blokova nadgledaju se klikanjem na ikonu naočala u alatnoj traci, odnosno „MonitoringOn/Off“.



	Name	Data type	Start value	Monitor value	Retain	Accessible ...	Writa...	Visible in ...	Setpoint	Com
1	Input									
2	IN_Enable	Bool	false	TRUE						
3	IN_Value	Real	0.0	20.32262						
4	IN_SmplesN	UInt	10	10						Num
5	Output									
6	OUT_Max	Real	0.0	22.97079						
7	OUT_Min	Real	0.0	20.32262						
8	OUT_Avr	Real	0.0	21.61713						
9	OUT_Std_dev	Real	0.0	0.9891989						
10	Smp_N	Int	0	10						
11	Status	Int	0	2						
12	InOut									
13	Static									
14	InVal	Array[0..1001] of Real								
15	InVal[0]	Real	0.0	20.32262						
16	InVal[1]	Real	0.0	22.58096						
17	InVal[2]	Real	0.0	22.91702						
18	InVal[3]	Real	0.0	21.19638						
19	InVal[4]	Real	0.0	20.96786						
20	InVal[5]	Real	0.0	20.64524						
21	InVal[6]	Real	0.0	22.97079						
22	InVal[7]	Real	0.0	22.0567						
23	InVal[8]	Real	0.0	21.84162						
24	InVal[9]	Real	0.0	20.67212						

Slika 4.19. Monitoring „IDB“ „FB1“

Formule korištene za izračunavanje standardne devijacije i aritmetičke sredine unutar SCL koda prikazane su u nastavku.

Standardna devijacija:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2} \tag{3-1}$$

N - broj uzoraka

x_i - vrijednost pojedinog uzorka

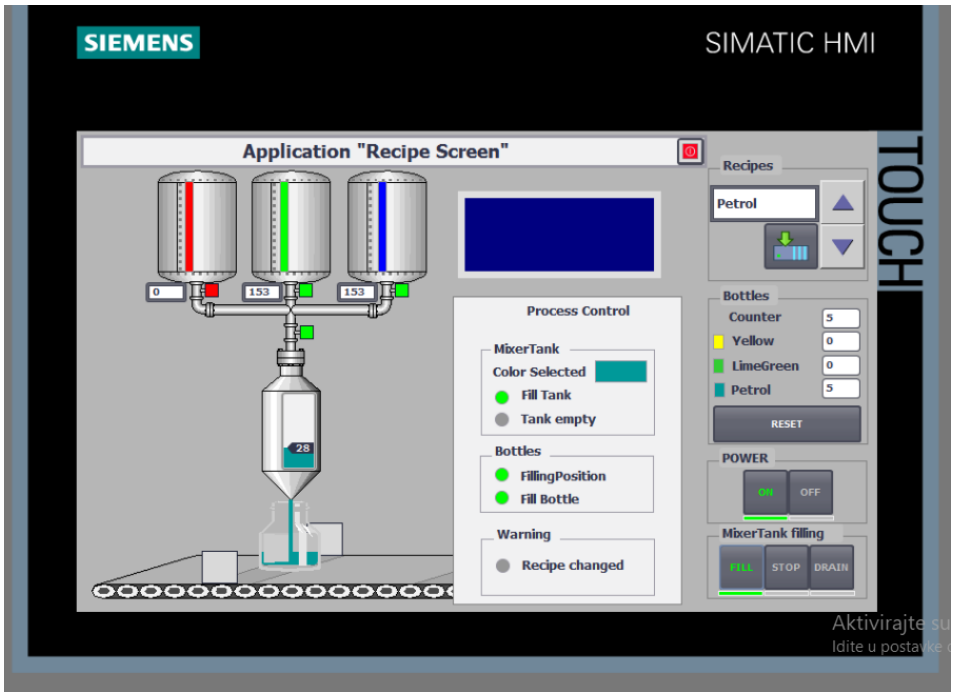
$\bar{x}$ - aritmetička sredina

Aritmetička sredina:

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N} \tag{3-2}$$

5. SCADA I VIZUALIZACIJA

SCADA (Supervisory Control And Data Acuisition) je tehnologija koja omogućuje prikupljanje odnosno praćenje svih signala i veličina iz jednog ili više udaljenih postrojenja u kojemu se odvijaju neki procesi, te omogućuje slanje upravljačkih signala u postrojenje čime se dobiva stalni nadzor i upravljanje nad procesom. Programska podrška TIA Portal koristi za stvaranje slika vizualizacije procesa na ekranu HMI (Human Machine Interface) uređaja koji služi za upravljanje i nadgledanje strojeva i postrojenja. Predhodno definirani objekti i elementi dostupni unutar programa olakšavaju stvaranje tih ekrana. Grafički objekti i elementi su svi oni elementi koji mogu biti korištenpoi za vizualizaciju projekta u HMI sustavu. To uključuje tekst, tipke, dijagrame ili grafike za vizualizaciju dijelova procesa (senzori, aktuatori, uređaji). Grafički objekti mogu biti statično vizualizirani ili korišteni kao dinamički objekti uz pomoć oznaka [7]. WinCC unutatar TIA Portala, WinCC flexible i WinCC SCADA sustav jesu HMI inženjerski alati za konfiguriranje HMI uređaja. HMI uređaji, kao što je npr. PC panel, prikazuju aplikaciju koja omogućava operatorsko upravljanje i nadzor procesa. Brojni dodatni paketi proširuju osnovne funkcionalnosti softvera za inženjering i vizualizaciju [5].

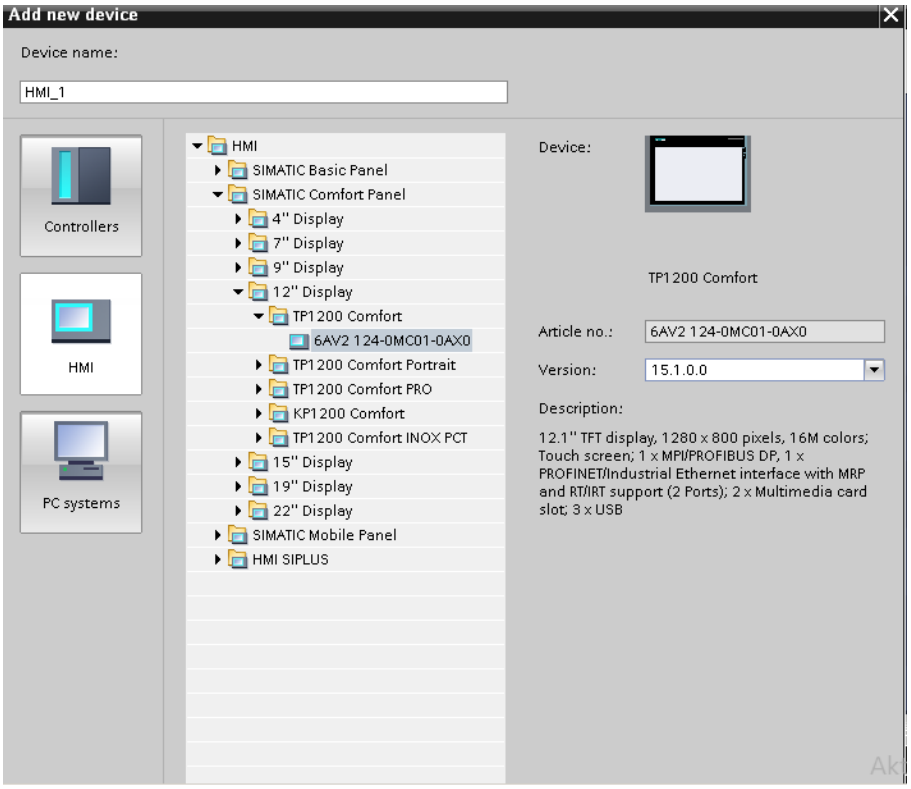


Slika 5.1. Primjer HMI panela procesa upravljanja spremnicima

https://www.fiverr.com/mit_automac/do-plc-and-hmi-programming-of-siemens-and-allen-bradley

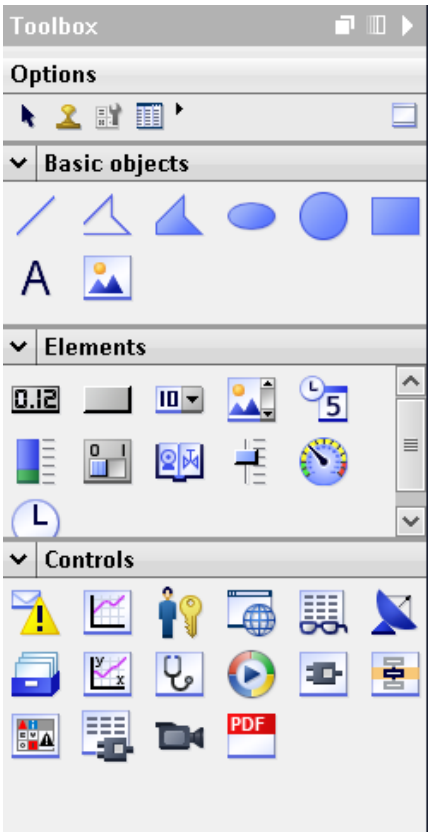
5.1. Vizualizacija u WinCC

Prvi korak pri kreiranju HMI sučelja je dodavanje uređaja, odabit njegovog tipa i odabit veličine ekrana HMI uređaja. Klikanjem na „Add new device“ na lijevoj strani alatne trake dodaje se novi „uređaj“. Uređaj korišten za vizualizaciju podataka u ovome radu je TP1200 Comfort (6AV2 124-0MC01-0AX0).



Slika 5.2. Dodavanje novog HMI sučelja

Osnovni grafički objekti nalaze se u kartici „Toolbox“ pod nazivom „Basic objects“. Osnovni grafički objekti su jednostavni objekti koji mogu poslužiti za izradu odnosno dizajniranje jednostavnih rješenja vizualizacije procesa, ali se od njih također mogu stvoriti kompleksni objekti potrebni za vizualizaciju složenih procesa. Osim uređivanja svojstava osnovnih objekata moguće im je postaviti animacije što isto vrijedi i za naprednije grafičke objektne elemente. Također se može raditi i s predefiniranim elementima koji se nalaze u kartici „Toolbox“ pod nazivom „Elements“ [7]. Izgled „Toolbox-a“ WinCC dodatka prikazan je na slici 5.3.



Slika 5.3. „Toolbox“ HMI sučelja

Funkcije SCADA sustava su:

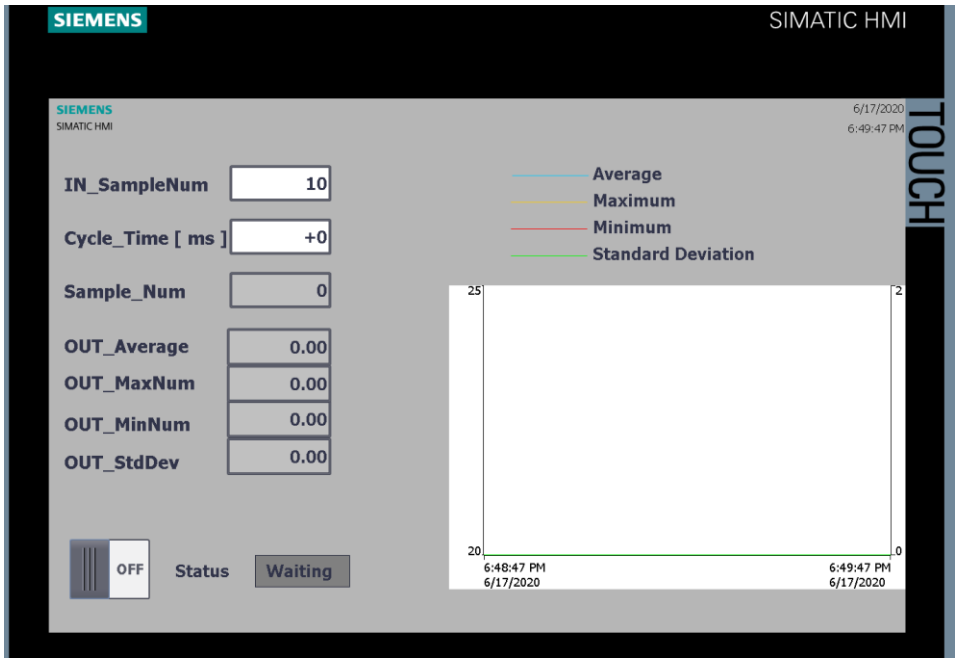
- Prikupljanje podataka i prikaz
- Alarmi i događaji
- Baza povjesnih podataka
- Obrada izmjenjenih podataka
- Evidentiranje i izvještavanje
- Sučelje čovjek-stroj (HMI)
- Rukovanje operatorskim naredbama

Trending predstavlja dodatnu funkciju. Trendovi osiguravaju praćenje ponašanja procesa i nadziranje mogućih odstupanja. Nadzirani procesni podaci (uzorkovani ili događajno upravljani) pohranjuju se u povjesnu bazu podataka [6].

5.2. Simulacija u WinCC

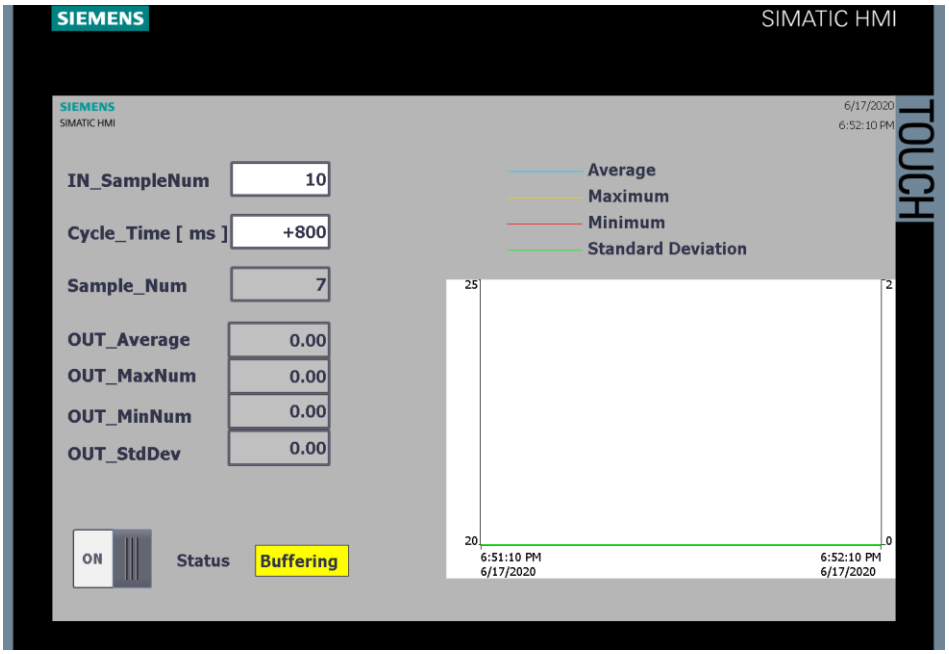
Kada je HMI pravilno konfiguriran i postavljen, moguće je u potpunosti koristiti HMI panel kao interface, jednom kada je program pokrenut uopće više nije potrebno ulaziti u korisnički program, budući da HMI panel ima mogućnost nadgledanja i modifikacije upravljačkih tagova, sve potrebne promjene mogu se raditi na panelu.

Napravljeni panel ima dvije ulazne veličine koje korisnik definira. To su IN_SampleNum, te Cycle_Time. Prisjetimo se da je u korisničkom programu također bilo potrebno definirati broj elemenata, odnosno proširivost „data buffera“, te duljinu intervala uzorkovanja, Cycle_Time izražen je u milisekundama, prošitivosti 1-999999 [ms]. Ostale varijable su izlazne varijable koje se računaju i pišu na izlaze onog trenutka kada Sample_Num ima istu vrijednost kao IN_SampleNum, odnosno kada je „data buffer“ popunjen. HMI panel ima status „Waiting“ sve dok se ne pokrene duplim klikom na sklopku u donjem lijevom kutu panela.



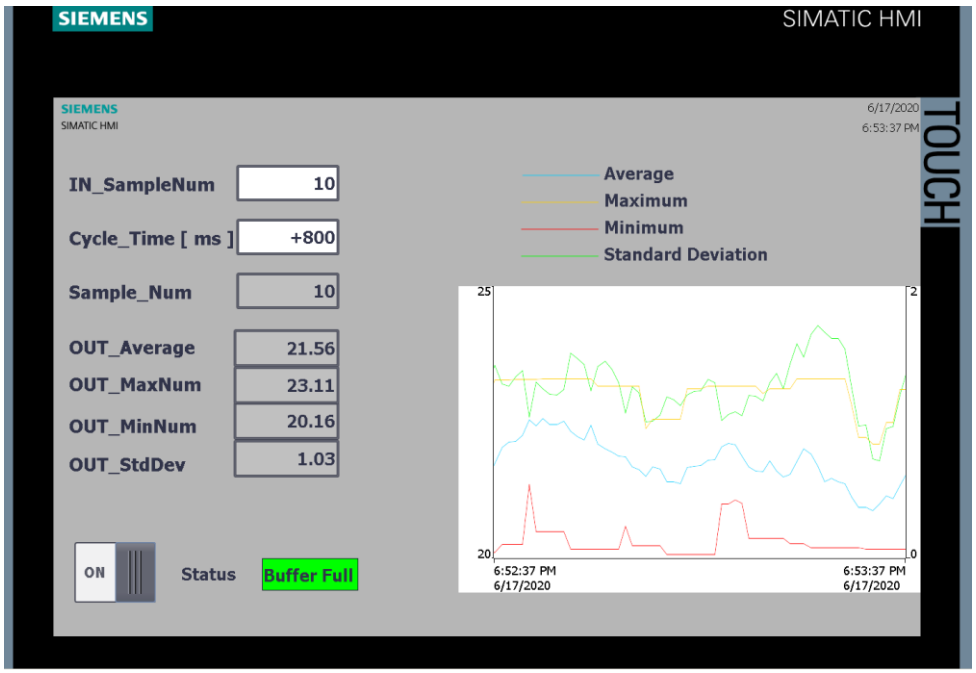
Slika 5.4. HMI Panel „Waiting“ (vizualizacija)

Pokretanjem programa status prelazi u „Buffering“, na slici 5.5. prikazan je HMI panel sa ulaznim vrijednostima: IN_SampleNum = 10, te Cycle_Time = 800 [ms].



Slika 5.5. HMI panel „Buffering“ (vizualizacija)

Kada je „data buffer“ popunjen, na izlaze se ispisuju vrijednosti izlaznih skala za veličine. Skala „Average“, „Maximum“, te „Minimum“ prikazana je na lijevoj strani trendinga, a za „Standard Deviation“ na desnoj strani trendinga.



Slika 5.6. HMI panel „Buffer Full“ (vizualizacija)

Kada vrijednost ulazne veličine IN_SampleNum nije u granicama 10-1000, HMI panel je u stanju Error.

Slika 5.7. HMI panel „Error“ (vizualizacija)

6.ZAKLJUČAK

Ovaj rad predstavlja programski konfigurirani sklop za uzorkovanje ulaznih veličina. izveden je na način da statistički obrađuje proizvoljnu vrijednost u željenim intervalima. Dodatna vizualizacija pomaže pri interpretaciji dobivenih izračuna. Konfigurirani sklop ima bezbroj mogućnosti za unaprjeđenje. Uz male preinake programski blok može računati i mnoge druge statističke podatke. Jednostavnim spajanjem senzora ili neke druge komponente, te izmjenom memorijskih lokacija, programski sklop bi mogao izračunavati razne fizikalne veličine kao što su: tlak, protok, vlaga, napon, struja i mnoge druge. Ovisno o potrebi te dinamici mijenjanja ulazne veličine postavlja se vrijeme uzorkovanja. Dinamičniji sustavi zahtjevaju kraće periode uzorkovanja, dok oni sustavi koji su manje promjenjivi imaju manju potrebu za kratkim vremenima uzorkovanja.

.

7. LITERATURA

[1] http://www.unizd.hr/portals/4/nastavni_mat/2_godina/statistika/statistika_01.pdf

[2] Prof. Dr. Sc. Babić Hrvoje „Signali i sustavi“ Sveučilište u Zagrebu, FER. Zagreb, 1996

URL: http://sis.zesoi.fer.hr/predavanja/pdf/sis_2001_skripta.pdf

[3] https://sh.wikipedia.org/wiki/Digitalni_signal

[4] http://laris.fesb.hr/digitalno_vodjenje/text_2.htm

[5] Berger Hans „SIMATIC automatizacijski sustavi“ 1. Hrvatsko izdanje, 2013.

URL: http://www.graphis.hr/news/simatic/Simatic_perje_web.pdf

[6] Mr. Sc. Malčić Goran„Sustavi nadzora i upravljanja indusrijskih postrojenja“ XIII. Tečaj 2012

URL: http://seminar.tvz.hr/materijali/materijali13/13E07.pdf

[7] „Izrada programskih komponenti u TIA Portal programskom okruženju“ – Maršić Danijel, Malčić Goran, Vlašić Ivica– MIPRO 2014

[8] „Polytechnic & Design“ - Matika Dario, Blažević David, Vol 2. No1, 2014.

[9]https://support.industry.siemens.com/cs/document/15360455/what-is-the-difference-between-an-instance-data-block-and-a-global-data-block-and-how-does-a-call-call-influence-the-db-register-?dti=0&lc=en-WW

SAŽETAK

Ovaj rad opisuje izrađeneni programski blok za uzorkovanje i izračun statističkih podataka. Programski blok ovisno o ulaznoj veličini, i ulaznim varijablama, na svoje izlaze postavlja vrijednosti koje su rezultat matematičke obrade nad ulaznim veličinama. Korištenjen je TIA Portal za korisnički program, te dodatak WinCC za izradu vizualizacije. Veličine koje programski blok izračunava su: maksimalna vrijednost, minimalna vrijednost, aritmetička sredina, standardna devijacija te broj uzoraka.

Ključne riječi: uzorkovanje, TIA Portal, simatic, WinCC, statistika, vizualizacija

ABSTRACT

This thesis describes a program block for sampling and statistical analysis. Depending on the inputs, the program block writes the outputs after mathematical processing. The user application is created in TIA Portal, and the visualisation is implemented by WinCC. The statistical analysis of the application includes: maximal value, minimal value, mean value, standard deviation and sample number.

Keywords: sampling, TIA Portal, simatic, WinCC, statistics, inputs, outputs, visualisation

DODATAK

SCL kod funkcijskog bloka „FB1“, *„Network 3“*

```
#Loop_Lenght := #IN_SmplesN - 1;
//*****************   DB LOGIC / OUTPUT   ******************************//
IF #BufferFull THEN
    #Sum := 0.0;
    #Sum_SD_Num := 0.0;
    #OUT_Min := #InVal[0];
    #OUT_Max := 0.0;

    FOR #j := 0 TO #Loop_Lenght DO
        IF #InVal[#j] > #OUT_Max THEN
            #OUT_Max := #InVal[#j];
        END_IF;                                     //Max Number
        IF #InVal[#j] < #OUT_Min THEN
            #OUT_Min := #InVal[#j];
        END_IF;                                     //Min Number
        #Sum := #Sum + #InVal[#j];
    END_FOR;

    #OUT_Avr := #Sum / #IN_SmplesN;                 //Avr Number

    FOR #j := 0 TO #Loop_Lenght DO
        #SD_Num := #InVal[#j] - #OUT_Avr;
        #SD_Num_Pow := #SD_Num * #SD_Num;
        #Sum_SD_Num := #Sum_SD_Num + #SD_Num_Pow;
    END_FOR;


    #Variance := #Sum_SD_Num / #Loop_Lenght;
    #OUT_Std_dev := SQRT(#Variance);               //Standard Deviation

ELSIF #BufferFull = FALSE THEN
    #OUT_Min := 0.0;
    #OUT_Max := 0.0;
    #OUT_Std_dev := 0.0;
    #OUT_Avr := 0.0;
END_IF;


//STATUS
IF #SamleN_OK THEN

    IF #BufferFull THEN
        #Status := 2;


    ELSIF #BufferFull = FALSE THEN
        #Status := 1;

        IF #IN_Enable = FALSE THEN
            #Status := 0;
        END_IF;

    END_IF;               //Status 0 || Status 1  || Status 2


ELSIF #SamleN_OK = FALSE THEN              //Status 3
    #Status := 3;
END_IF;


#Smp_N := #VarCount;     //Sample Number Counter
```

ŽIVOTOPIS

Danijel Dadić rođen je 23.10.1991. u Slavonskom Brodu. Osnovnu školu završava u Županji. Tehničku školu u Županji pohađa u periodu 2006-2010, te stječe srednju stručnu spremu Elektrotehničar. 2015. godine upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija, smjer elektroenergetika. Na 2. godini stručnoga studija odabire drugi smjer, Automatiku.