Primjena LabVIEW-a u prikupljanju, obradi i generiranju karakterističnih digitalnih signala

Poprocki, Boris

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ELEKTROTEHNIČKI FAKULTET

Stručni studij

PRIMJENA LABVIEW-A U PRIKUPLJANJU, OBRADI I GENERIRANJU KARAKTERISTIČNIH DIGITALNIH SIGNALA

Završni rad

Boris Poprocki

Osijek, 2016. godina

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1S: Obrazac za imenovanje Povjerenstva za obranu završnog rada na preddiplomskom stručnom studiju

Osijek, 09.09.2016.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu završnog rada na preddiplomskom stručnom studiju

Ime i prezime studenta:	Boris Poprocki
Studij, smjer:	Preddiplomski stručni studij Elektrotehnika, smjer Elektroenergetika
Mat. br. studenta, godina upisa:	A 4053, 03.09.2012.
OIB studenta:	93220043298
Mentor:	Krešimir Miklošević
Sumentor:	
Predsjednik Povjerenstva:	Zorislav Kraus
Član Povjerenstva:	Željko Špoljarić
Naslov završnog rada:	Primjena LabVIEW-a u prikupljanju, obradi i generiranju karakterističnih digitalnih signala
Znanstvena grana rada:	**Elektrostrojarstvo (zn. polje elektrotehnika)**
Zadatak završnog rada	Opisati i objasniti osnovne i specifične osobine programskog paketa LabVIEW 2014 firme National Instruments. Izdvojiti područja znanosti i tehnike gdje se najčešće koristi. Sažeti i pokazati koje sve mogućnosti nam omogućava kompletno LabVIEW okruženje. Osmisliti i konstruirati kompletno virtualno okruženje (virtualnog sustava) za prikupljanje, obradu i generiranje proizvoljnih digitalnih signala. Ukazati na prednosti i eventualne nedostatke korištenja objektno orijentiranih alata za programiranje. (NEMA SUMENTORA)
Prijedlog ocjene pismenog dijela ispita (završnog rada):	Vrlo dobar (4)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 2 Postignuti rezultati u odnosu na složenost zadatka: 2 Jasnoća pismenog izražavanja: 2 Razina samostalnosti: 2
Datum prijedloga ocjene mentora:	09.09.2016.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek, 13.09.2016.

Ime i prezime studenta:	Boris Poprocki
Studij:	Preddiplomski stručni studij Elektrotehnika, smjer Elektroenergetika
Mat. br. studenta, godina upisa:	A 4053, 03.09.2012.
Ephorus podudaranje [%]:	2

Ovom izjavom izjavljujem da je rad pod nazivom**: Primjena LabVIEW-a u prikupljanju, obradi i generiranju karakterističnih digitalnih signala**

izrađen pod vodstvom mentora Krešimir Miklošević

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

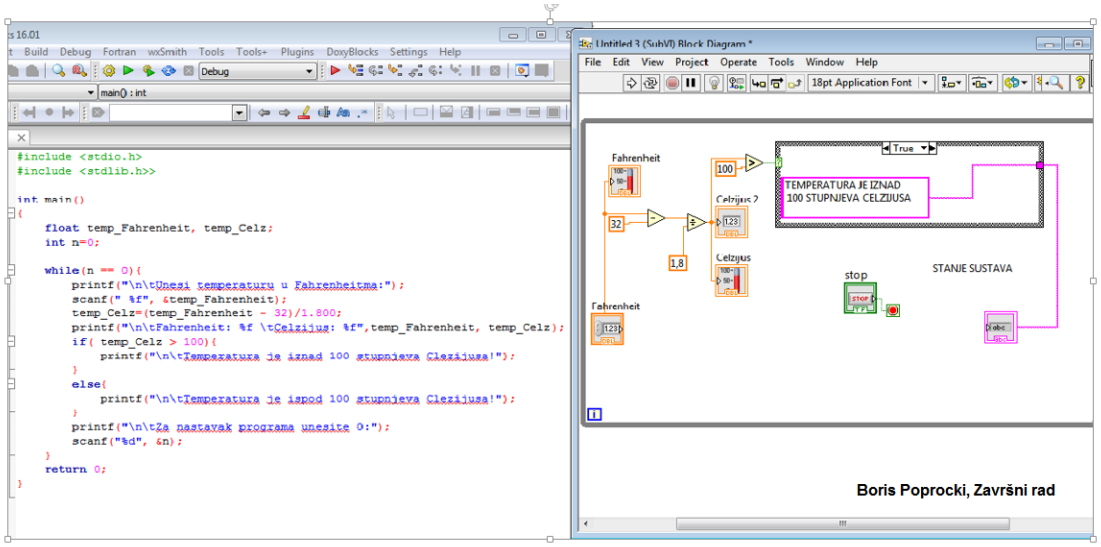
Kako bi se moglo raditi sa signalima iz vanjskog svijeta koji su analogni i kontinuirani u vremenu nužno je izvršiti pretvorbu takvih signala. Pretvorba analognih signala potrebna je prilikom računalne obrade podataka. Zahvaljujući domišljatosti čovjeka osmišljene su različiti alati i tehnike pretvorbe. Jedan od takvih alata je konstruiran od firme National Instruments pod nazivom NI LabVIEW 2014 koji se koristi u različite svrhe te posjeduje niz mogućnosti . Prikazao se izrazito pogodnim za prikupljanje, obradu kao i za generiranje digitalnog zapisa signala na temelju podataka iz vanjskog svijeta. Ova mogućnost uvelike je doprinijela komforu i kvaliteti ljudskog života automatiziranjem raznih procesa. NI LabVIEW 2014 doprinosi bržem i jednostavnijem načinu programiranja, većem zadovoljstvu korisnika koji kroz relativno kratko razdoblje ostvaruje veliki napredak.

U ovom radu biti će objašnjene prednosti kao i navedena područja znanosti i tehnike gdje je upotreba objektno orijentiranih (grafičkih) programskih jezika najveća. Ukratko opisan i dan teorijski uvod u digitalne signale.Opisane i objašnjenje osnovne i specifične osobine softverskog paketa firme National Instruments (NI) punog naziva LabView 2014 National Instruments. Na kraju rada biti će osmišljeno, kreirano i prikazano virtualno okruženje za prikupljanje i obradu signala te generiranje digitalnih signala.

U prvom poglavlju dana je kratka uvodna riječ , te ukratko opisan plan rada.U drugom poglavlju opisan je uvod u grafičko programiranje i razlika između proceduralnog i grafičkog programiranja. Treće poglavlje nam daje teorijski uvod u signale nužan za kreiranje VI-a u softveru LabVIEW , te je opisan proces pretvorbe analognog signala u digitalni. U četvrtom poglavlju su opisane i objašnjene osnovne i specifične osobine softverskog paketa LabView 2014, te kreirani i objašnjeni jednostavni VI. U petom poglavlju izrađen je VI-a za prikupljanje i obradu analognih signala te , generiranje digitalnih signala. Objašnjene funkcijske cjeline programa kao i elementi koji su korišteni u konstrukciji programa. Šesto poglavlje nam prikazuje rezultate izvođenja VI-a kreiranog u petom poglavlju. U sedmom poglavlju dana je kratka završna riječ te , kratak osvrt na cjelokupan završni rad.

2. UVOD U GRAFIČKO PROGRAMIRANJE

Osnovna ideja Labview-a je da ukloni programerske prepreke iz svakodnevnog rada inženjera (neprogramera).Za razliku od ostalih pristupa programiranja, u kojima je težište na akcijama koje se vrše na podatkovnim strukturama, kod objektno orijentiranih programskih jezika težište je na projektiranju aplikacije kao skupa objekata koji izmjenjuju poruke između sebe. Odnosno većina programskih jezika zasniva se na proceduralnom programiranju tj. davanju pisanih instrukcija od strane korisnika računalu u obliku naredbi za razliku od LabVIEW-a gdje se programira u grafičkom sučelju povezivanjem različitih grafičkih simbola. Svaki od grafičkih simbola posjeduje svoju funkciju koja je prethodno također programirana od strane proizvođača softvera[5].



Slika 2.1 Razlika između načina programiranja u programskom jeziku C (lijevo) i LabVIEW-a (desno)

Na slici 2.1 prikazan je program napisan u dva različita programska jezika kojima je rezultat izvođenja identičan. Vidljiva je razlika između programiranja u programskom jeziku C i grafičkog (blokovskog) programiranja u LabVIEW-u. Puno kompleksniji zadatak predstavlja proceduralno programiranje u odnosu na povezivanje i korištenje već gotovih objekata (elemenata). Proceduralno programiranje zahtjeva od programera pisanje svakog dijela funkcijskog dijela programa zasebno. Ukoliko koristimo objektno - orijentirani programski jezik nije nužno svaki funkcijski dio programa zasebno pisati nego koristimo već gotove

elemente (koji nam predstavljaju funkcijske cjeline) te ih međusobno povezujemo kako bi dobili željeni krajnji rezultat prema određenom logičkom slijedu.

Ovakav način programiranja omogućava programeru skraćivanje vremena pisanja programa i fokusiranje na projektni zadatak. Također postoje i određene granice OOP (objektno – orijentiranih programskih jezika). Brzina izvođenja programa puno je manja u odnosu na *low-level* proceduralne programske jezike. Nemogućnost programiranja hardvera (mikroprocesora, mikrokontrolera, elektronike). Naravno, objektno-orijentirani programski jezici se ni ne koriste za ovakve svrhe.Stoga na korisniku je i na temelju njegovo projektnog zadatka da se odluči koji će način programiranja kao i koji programski jezik koristiti u svom radu.

2.1 Područja primjene LabView-a

Tržišna utakmica, promjenjivi zahtjevi tržišta, povećanje cijene ljudskog rada, smanjenje ljudi unutar postrojenja te zakonski propisi o uvjetima sigurnosti i rada, nalažu brz razvoj naprednih tehnologija, među kojima su i visokoautomatizirani i visoko autonomni – inteligentni sustavi. LabView posjeduje mogućnost pretvaranja "sirovih" podataka sa senzora – napona u procesu potrebne veličine, te grafičko prikazivanje svih mjerenih veličina. Dobiveni podaci također se mogu analizirati, te na temelju analize zaključiti odstupaju li podatci ili ne odstupaju od očekivanih vrijednosti. Što govori da LabView posjeduje širok spektar svakodnevne primjene kako u znanstvenim disciplinama tako i u inženjerskim. Može ga se pronaći bilo gdje je nužna komunikacija računala sa vanjskim svijetom, pogotovo u svrhe mjerenja i obrade podataka[4].

Važno je spomenuti prednost softverskog paketa firme National Instruments (NI) LabView 2014 u odnosu na sve druge slične softvere (također objektno orijentirane) u posjedovanju već gotovog *NI Data Acquisition Systems* koji je zapravo grupa modula koji služe za prikupljanje signala i podataka te povezivanje mjernih instrumenata i LabView-a. Ova uvelike olakšava posao korisniku no zahtjeva upotrebu *NI DAQ*(kompatibilnih) uređaja[6].

Slika 2.2 Prikaz DAQ uređaja[6]

DAQ uređaji raspolažu s nekoliko različitih vrsta senzora i mogućnostima komunikacije USB, Ethernet, PCI, PCI Express i WiFi kao npr.:

- Termopar,RTD, termistor za mjerenje temperature
- Senzor za mjerenje osvijetljenosti
- Mikrofon za prikupljanje zvučnih signala
- Tenzometar i piezoelektrični prijenosnik za mjerenje sile i pritiska
- Potenciometar , LVDT , optički enkoder za određivanje položaja i pomaka
- Akcelerometar za određivanje ubrzanja

U ovom radu neće se koristiti *NI Data Acquisition Systems* zbog ne posjedovanja *DAQ* uređaja.

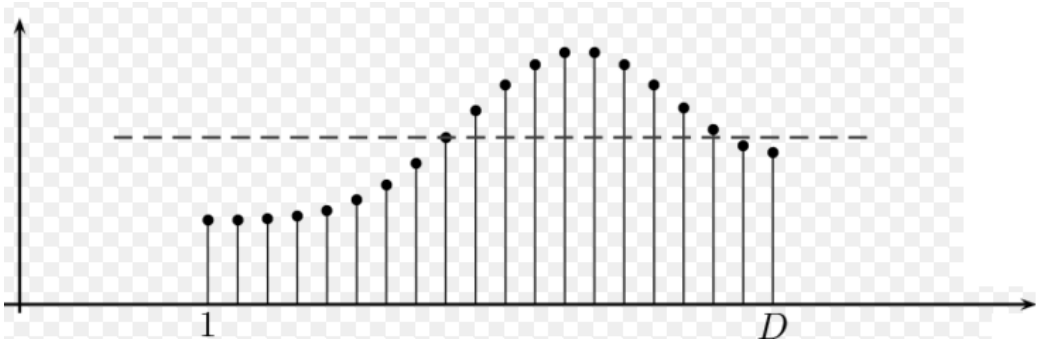
Kako bi interakcija vanjskog svijeta i računala bila uopće moguća potrebno je izvršiti prilagodbu signala vanjskog svijeta na signale razumljive elektroničkim sklopovima (računalu). Ovaka vrsta pretvorbe naziva se još i analogno – digitalna pretvorba. Kratak teorijski uvod u signale nužan za kreiranja VI-a u softveru LabVIEW biti će dan u sljedećem poglavlju.

3. DIGITALNI SIGNALI

3.1 Uvod u signale

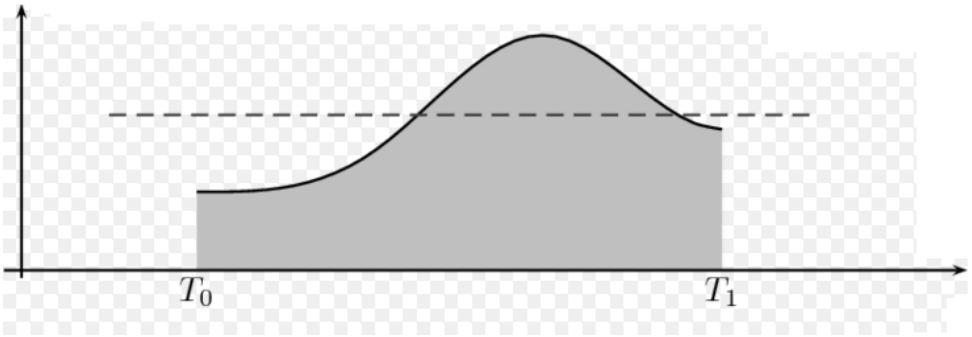
Analiza i projektiranje signala i sustava ima značajnu ulogu u svim područjima elektrotehnike kao i u mnogim drugim inženjerskim i znanstvenim područjima. Pod signalima podrazumijevamo sve ulaze i izlaze kao i unutrašnje funkcije koje sustavi analiziraju ili proizvode, kao što su napon, struja, tlak i slično. Definiciju signala teško je dati ali signal je moguće aproksimirati kao svaku vremenski promjenjivu fizičku pojavu ili pojavu koja sa sobom nosi ili prenosi neku informaciju. Vrlo često pod signalima podrazumijevamo i funkcije koje su vremenski neovisne, a ponekad i funkcije koje nisu usko vezane za fizičke pojave. Međutim, ono što je zajedničko za sve što se smatra pod širokim pojmom signala je informacija[2].

Signali koju su definirani samo u diskretnim vremenskim trenucima nazivaju se diskretnim signalima. Odnosno, ukoliko vršimo mjerenje nekog vremenski kontinuiranog signala u određenim vremenskim intervalima ili je signal po svojoj prirodi diskretan za njega se može reći da je diskretan signal. Primjer diskretnog signala se može vidjeti na slici 3.1.



Slika 3.1 Primjer diskretnog signala[2]

Druga vrsta signala su oni signali koji su dostupni i mjerljivi u svakom vremenskom trenutku. Ovi signali su definirani kontinuiranom, vremenski neprekinutom intervalu vremena. Ovaka vrsta signala zove se još i kontinuirani signali. Primjer kontinuiranog signala se može vidjeti na slici 3.2.



Slika 3.2 Primjer kontinuiranog signala[2]

3.2 Analogno-digitalna pretvorba

Pretvorba vremenski kontinuiranog (analognog) signala u digitalni signal skraćeno (A/D) i obrnuto (D/A) sastavni je dio suvremenih sustava za obradu, pohranu i prijenos informacija. Svi signali u stvarnom svijetu su analogni. Analogni signal vremenski je neprekidan i potrebno ga je pretvoriti u niz digitalnih vrijednosti. Analogno-digitalna pretvorba znači preuzeti analogne informacije te ih pretvoriti u digitalne kako bi ih se dalje moglo koristiti i obrađivati putem računala. Ukoliko u nekom postrojenju postoji zahtjev da se na temelju mjerenja nekog signala (informacija) podesi, regulira neki parametar našeg sustava potrebno je signale iz "stvarnog" svijeta dostaviti računalu (mikrokontoreleru, PLC uređaju) u njemu razumljivom obliku. Kako bi računalo moglo "shvatiti" podatak potrebno mu ga je dostaviti u digitalnom obliku. Rezultat A/D pretvorbe je električni signal u "vidu" pozitivnog ili negativnog izlaza na predajniku i prepoznavanje tog stanja (ne oblika) na prijemniku. To stanje se označava sa "1" ili "0". Digitalizacija signala poruke također omogućuje povećanu otpornost na smetnje. U odnosu na analogni signal unosi se svjestan gubitak vjerodostojnosti zbog raspona nivoa digitalizacije[2].

Proces pretvorbe analognog signala u digitalni vrši se u tri osnova koraka:

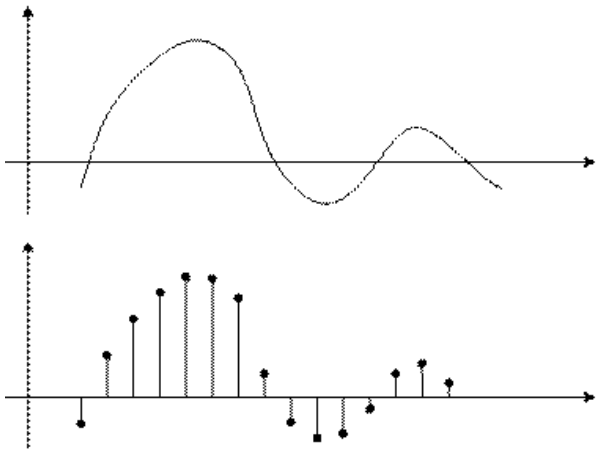
- uzorkovanje
- kvantiziranje
- kodiranje

Uzorkovanje predstavlja postupak pri kojemu se analogni signal iz izvora šalje na uzorkovanje (diskretiziranje signala po vremenskoj bazi), koje se vrši u pravilnim vremenskim razmacima. Frekvencija uzorkovanja f_s kojom se uzimaju uzorci mora biti najmanje dva puta viša od najviše frekvencije f_m koja se može pojaviti u analognom signalu koji se uzorkuje, odnosno vrijedi:

$$fs \geq 2fm \tag{3-1}$$

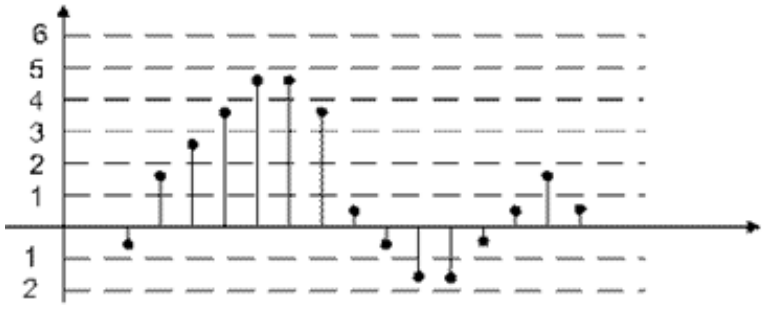
Odnosno period uzorkovanja mora biti:

$$Ts = \frac{1}{fs} \tag{3-2}$$



Slika 3.3 Uzorkovanje signala[2]

Nakon uzorkovanja signal će dalje biti poslanna kvantiziranje (diskretiziranje signala prema trenutnoj vrijednosti amplitude) zaokruživanje vrijednosti uzoraka na najbližu kvantizacijsku razinu. Spektar mogućih amplituda se podijeli na određeni broj kvantizacijskihintervala. Sredina kvantizacijskog intervala naziva se kvantizacijska razina ili stepenica. Ovisno o potrebi, koristi se linearno ili nelinearno kvantiziranje. Kvantizacijski intervali su jednake dužine kod linearnog kvantiziranja. Postoji i nelinerano kvantiziranje gdje intervali ne moraju biti jednake dužine[2].

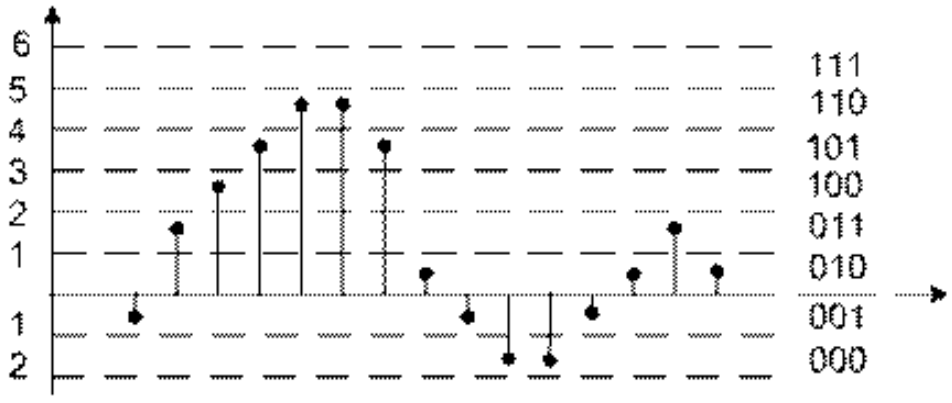


Slika 3.4 Linearno kvantiziranje gdje su kao što je vidljivo intervali jednakih dužina[2]

Svakoj razini kvantiziranjabiti će pridjeljena jedna kodna riječ, te se kvantiziranom uzorku, u ovom slučaju, dodjeljuje 3 bita koji ga opisuju jer ima 2^3tj. 8 kvantizacijskih razina. U ovisnosti o razlučivosti (broju bita koji se koristi) ovisi "kvaliteta" pretvorbe. Iako veća razlučivost neće uvijek značiti i optimalnije. Analogni signal predstavljen u digitalnom obliku može se uzorkovati sa (3-3) kvantizacijskih razina:

$$2^n \tag{3-3}$$

Time se dobije niz kodnih riječi koji odgovara nizu kvantiziranih uzoraka. Zaokruživanjem prilikom kvantiziranja na višu ili nižu razinu nastaje šum koji se zovekvantizacijski šum. Takav signal, kao niz bitova, postao je digitalni signal[2].



Slika 3.5 Kodiranje kvantizacijskih razina[2]

4. PROGRAMSKI PAKET NI LABVIEW 2014

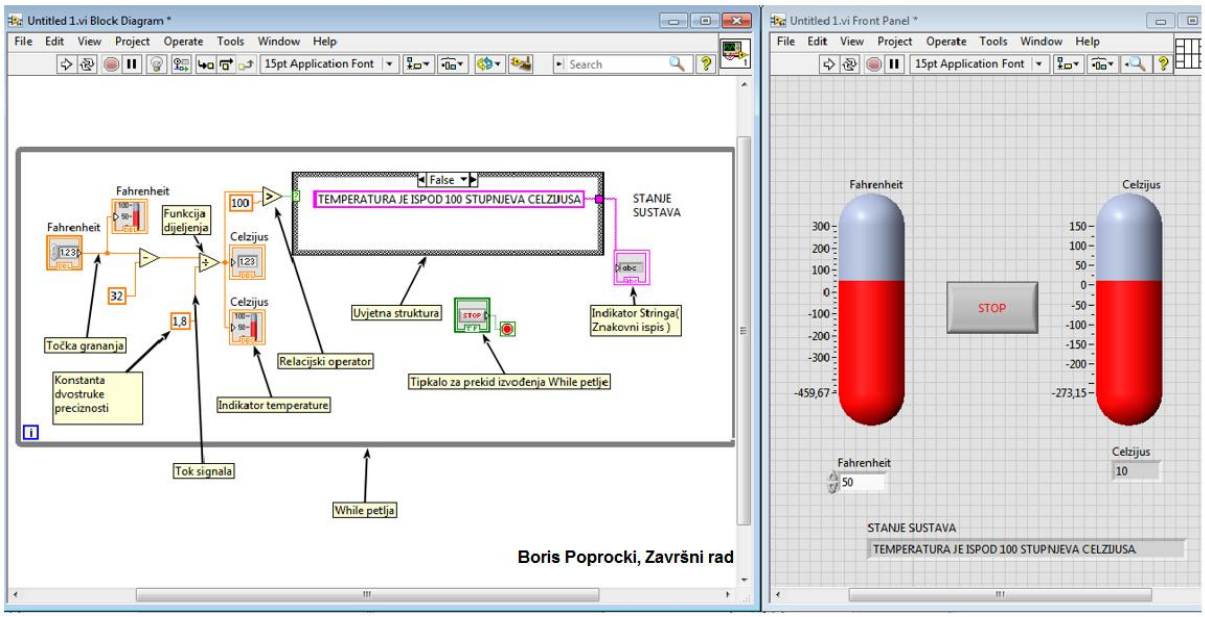
4.1. Uvod u programski paket LabVIEW

LabVIEW (LaboratoryVirtualInstrumentation Engineering Workbench) je platforma, mjerni softverski paket i razvojno okruženje za grafičko programiranje firme National Instruments (NI). Zasniva se na grafičkom programskom jeziku G. Podešen ja za komunikaciju s različitim priključcima i mjernim karticama. Omogućuje nam stvarno-vremensku komunikaciju, prikupljanje podataka, obradu, kontrolu instrumenata i industrijske automatike (mikrokontrolera i PLC uređaja).

LabVIEW je ušao u primjenu još 1986.godine, a njegovo usavršavanje i proširivanje odvija se i danas. Prvobitno je bio orijentiran za mjerenje i testiranje tehničkih sustava. S vremenom su modificirane i implementirane razne tehničke funkcije pa je dosegao razinu složenog softverskog paketa koji omogujuće raznoliku primjenu i upotrebu u inženjerskoj svakodnevici. LabVIEW se odlikuje velikim mogućnostima ali se razlikuje od većine programskih jezika s kojima se programer susretao u svojem radu[1].

4.2. Komponente i osnovni koncepti LabVIEW-a

LabVIEW je u potpunosti grafički program. Konfiguriranje se odvija u dva grafička sučelja, frontalnom panelu i u blok dijagramu. *Frontalni* (prednji) panel koristi se za podešavanje vrijednosti, unos podataka (eng. *input*) i za prikazivanje obrađenih podataka odnosno izlaza (eng. *output*). Blok dijagram se koristi za grafičko programiranje odnosno davanja niza naredbi računalu koje na temelju naših naredbi vrši operacije nad podacima koje smo mu proslijedili. LabVIEW je hijerarhijski programski jezik, što znači da se svaki *Visual instrumen*t (VI) sastoji od blok dijagrama i frontalnog panela. Svaki VI koji je ujedno potpuna funkcionalna cijelina može se pretvoriti u modul te koristiti unutar drugog VI-a.



Slika 4.1Izgled blok dijagram lijevo i desno prednje ploče

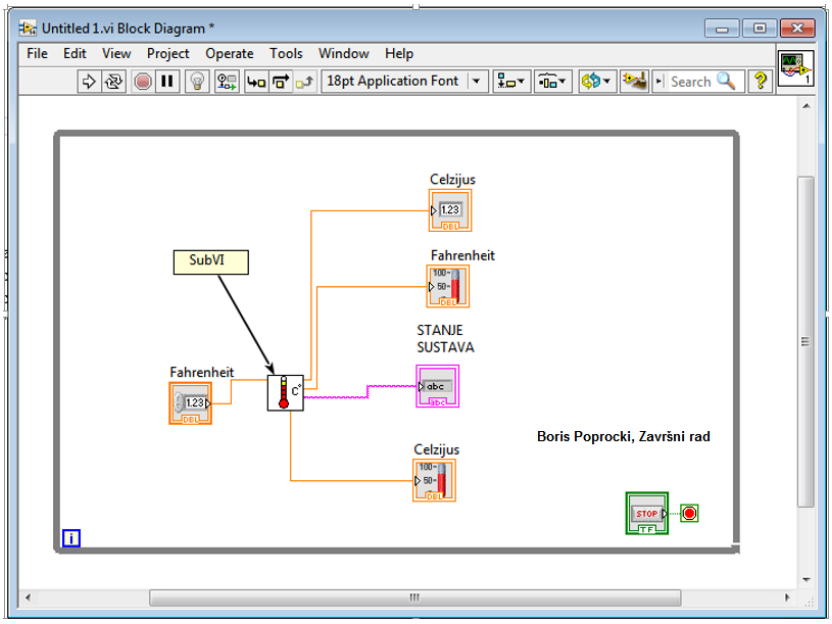
Slika 4.1 prikazuje jednostavniji VI sa oba spomenuta sučelja. Svaki VI sastoji se od sljedećih komponenata:

- prednje ploče preko koje korisnik komunicira s VI-om
- blok dijagrama, koji sadrži programski kod
- simbola ili priključaka, za povezivanje s ostalim VI-ovima

Programiranje svakog VI-a započinje na prednjoj ploči, također nam je uz pomoć prednje ploče tokom samog izvođenja omogućena interakcija odonosno upravljanje izvođenjem, promjenu zadanih vrijednosti, online prikaza podataka, prikaza raznih signala i stanja unutar nekog procesa, sustava. Ovime nam je omogućena velika fleksibilnost i utjecaj na proces u stvarnom vremenu te mogućnost kontroliranja i regulacije sustava. Što pronalazi značajnu svrhu u automatskim procesima te veliku primjenu u industriji.

Na slici 4.1 svaki od elemenata prikazan na prednjoj ploči (dva termometra, tipkalo za podešavanje temperature, tipkalo za prestanak rada i indikator stanja sustava) raspolaže sa svojim priključnicama u blok dijagramu. Podaci uneseni na prednjoj ploči teku od elementa do elementa kroz blok dijagram, pri čemu se nad njima vrši obrada pod utjecajem funkcija koje smo izabrali. Nakon obrade i prolaska kroz sve funkcije ukoliko nema pogrešaka rezultati obrade prikazuje se na elementima ponovno prikazanima na prednjoj ploči. Svaki element blok dijagrama međusobno je povezan(ožičen). Na slici 4.1 prvi element lijevo služi za unos jednog realnog broja (temperature izražene u Fahrenheitima °F). Podatak (signal) nailazi na točku grananja gdje odlazi na blok indikatora temperature koji prikazuje željenu vrijednost na prednjoj pločii dalje na obradu gdje se obrađuje pod utjecajem ostalih funkcija. Krajnji rezultat obrade se ponovno prikazuje na prednjoj ploči osim što je nakon obrade temperatura izražena u stupnjevima Celzijusima ° C. Također se dobiva i informacija o stanju sustava znakovnim ispisom uz pomoć bloka za indikator (ispis) Stringa[1]. Tipkalo za zaustavljanje programa služi za prekid izvođenja programa na način da pritiskom na tipkalo pošalje vrijednost (eng.*True*) logički 1 na priključak za uvjet izvođenja/prekidanja *while* petlje. Koristi se i relacijski operator uz pomoć kojeg se može odrediti istinitost nekog izraza. Na temelju istinitosti uvjeta korištenjem uvjetne strukture moguće je dalje manipulirati podatkom u ovisnosti o zahtjevima. Također je moguće od primjera sa slike 4.1 napraviti SubVI. SubVI nam omogućuje korištenje jednog VI-a unutar drugog VI-a. Mogućnost korištenja SubVI-a znatno pojednostavljuje kompliciranije i zahtjevnije programe na način da povećava preglednost programa te omogućuje programiranje u etapama.

[1] Znakovni niz



Slika 4.2 Blok dijagram sa slike 4.1 nakon kreiranja subVI-a

4.3. Alati za programiranje

Kao što je spomenuto ranije postoje dva sučelja za svaki korisnički program LabVIEW-a. To su blok dijagram i prednja ploča. Programiranje se odvija uz pomoć tri izbornika nazvanim paletama. Za rad u prednjoj ploči to je paleta upravljanja odnosno *Controls*, a u blok dijagramu paleta funkcija *Functions*. Treća paleta je zajednička i koristi se za programiranje u oba sučelja[1].

4.3.1Prednja ploča

Prednja ploča simulira panel fizičkog instrumenta. Sadrži alatnu traku (paletu s alatima) komandnih alata te statusne indikatore koji se koriste za pokretanje/zaustavljanje programa i otklanjanje grešaka *debugging.*

Slika 4.3 Alatna traka prednje ploče

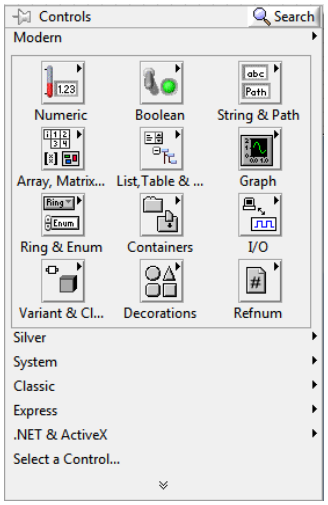
Slika 4.3 predstavlja alatnu traku prednje ploče.

Tri su gumba od posebnog značaja:

- - Gumb koji se koristi za pokretanje programa (izvođenje programa vrši se jednom)
- - Gumb koji se koristi za neprestano izvođenje programa (dok korisnik ne prekine)
- - Gumb za prekid izvođenja programa

4.3.2Paleta upravljanja

Paleta upravljanja (*eng. Controls*) sadrži elemente za posluživanje i prikazivanje te služi za njihovo smještanje na prednjoj ploči. Sadrži mnogo elemenata različitih oblika. Svaki element upravljačke palete ima neku vrstu slikovite informacije na osnovu koje korisnik zna trenutno stanje njegovog procesa/sustava. Ova mogućnost nam daje veliku prednost jer se ne mora posebno programirati stanje svakog elementa zasebno, što značajno smanjuje vrijeme programiranja.



Slika 3.4 Paleta upravljanja

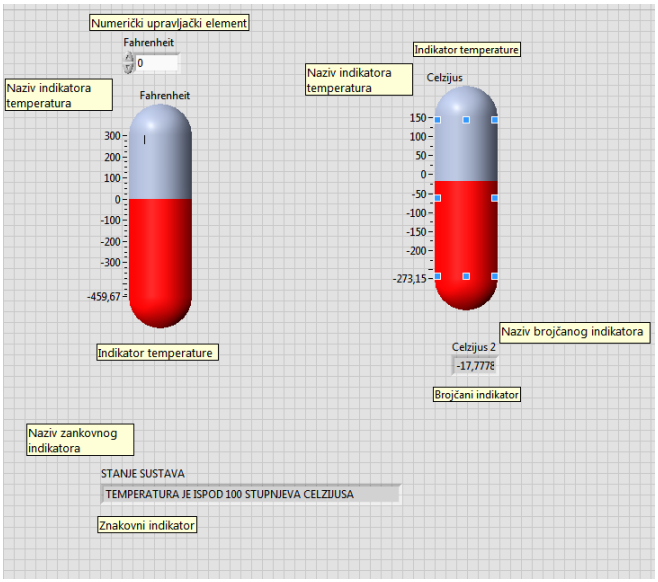
Svaka ikona "najvišeg nivoa" sadrži podpalete. Ovakvo rukovanje i prikaz elemenata upravljanja izrazito je koristan zbog preglednosti i funkcionalnosti. Što je jedan od faktora koji značajno doprinose brzom i jednostavnom upoznavanju sa softverom. Ako paleta upravljanja nije vidljiva, može se otvoriti opcijom Windows » Show Controls Palette iz Front panel izbornika[7].

4.3.3 Elementi upravljanja i indikatori

Indikatori također imaju mnoštvo različitih oblika. Neki od njih su kopija stvarnih indikatora (instrumenata i sl.) a neki su dizajnirani prikladnije za pozadinu računala. Koncept indikatora uključuje grafove *graphs* i dijagrame *charts* što je druga važna ušteda u vremenu jer se ti pokazivački elementi ne moraju dizajnirati posebno. Postoji više vrsta indikatora i upravljačkih elemenataovisno o vrsti i tipu podataka koji koristimo. Kontroleri (upravljački elementi) predstavljaju ulazne terminale, a indikatori (pokazivači) izlazne. Kontroleri simuliraju ulazne djelove uređaja i obezbjeđuju podatke za blok dijagram VI programa. Indikatori simuliraju izlazne djelove uređaja, i služe za prikaz rezultata iz blok dijagrama VI programa. Kontroleri i indikatori dostupni su u okviru panela upravljanja na upravljačkoj ploči[7].

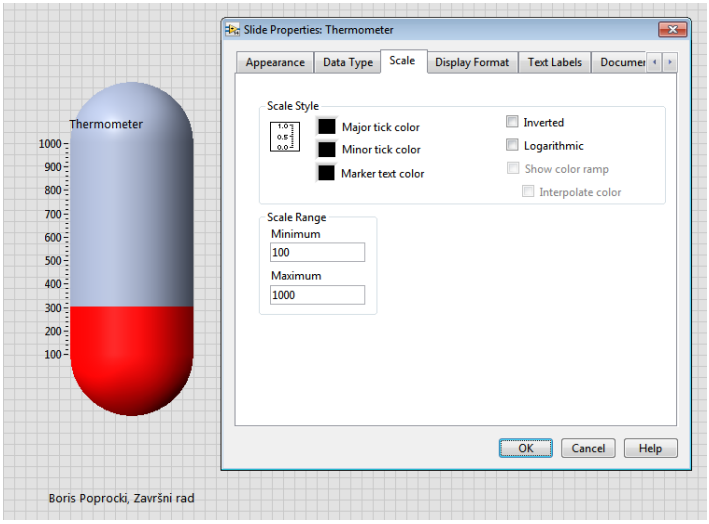
Najčešći indikatori i upravljački elementi koje koristimo su:

- Numerički upravljački elementi i indikatori
- Znakovni upravljački elementi i indikatori
- Booleovi upravljački elementi i indikatori



Slika 4.5 Upravljački elementi i indikatori (pokazivači) na prednjoj ploči

Svaki od indikator također posjeduje dodatne mogućnosti podešavanja. Za primjer sa slike 4.5 indikator predstavljen termometrom posjeduje razne dodatne mogućnosti. Jedna od dodatnih mogućnosti je podešavanja skale temperature slika 4.6.



Slika 4.6 Podešavanje skale indikatora (termometra)

Sa slike 4.6 vide se razne, dodatne mogućnosti podešavanja indikatora. U primjeru na slici 4.6podešava se skala termometra. Skala termometrapodešena je na prikaz minimalne vrijednosti temperature od 100 °C do maksimalne vrijednosti temperature 1000°C.

Također se može uočiti da je moguće podesiti skalu da bude logaritamska, što je izrazito korisno ukoliko su vrijednosti koje prikazujemo širokog raspona. Mogu se podesiti skoro svi upravljački elementii indikatori pomoću opcija iz njihovih pop-up izbornika.

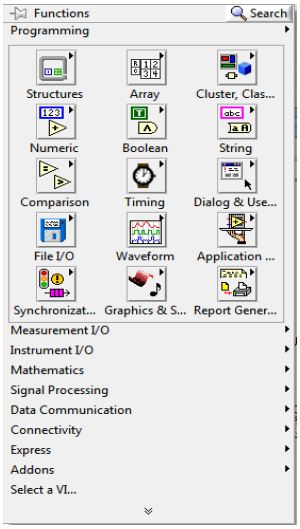
4.3.4 Blok dijagram

Blok dijagram predstavlja pozadinu prednje ploče koja prikazuje kako su upravljački elementi i indikatori povezani međusobno kao i skrivene module gde se odvija sva programska obrada podataka. Za blok dijagram možemo reći da je jezgra svakog VI-a. Izgleda na neki način poput elektroničkog shematskog dijagrama i konceptualno jepovezan na identičan način.Pri LabVIEW programiranju posebno je potrebno posvetiti pažnju vremenu izvođenja i slijedu izvođenja operacija. U konvencionalnom programskom jeziku to je ostvareno slijedom naredbi i korištenjem različitih programskih petlji (for, while, do while itd.).

LabVIEW programiran je da radi na isti način. Koncept korišten u LabVIEW-u zasnovan je na podatkovnom toku*dataflow-a*umjesto dosadašnjeg *codeflow-a*što znači da se radnja na nekom elementu izvršava kad su svi njegovi ulazi na raspolaganju. To znači paralelnost u izvršavanju radnje ili barem pseudoparalelizam. Uobičajeno izvršavanje programa je s lijeva na desno jer su ulazi najčešće na lijevom dijelu elementa a izlazi na desnom dijelu. Ovo nije zahtjev nego preporuka. Programske petlje i redosljed kojim se odvijaju naredbe odvija se pomoću struktura[7].

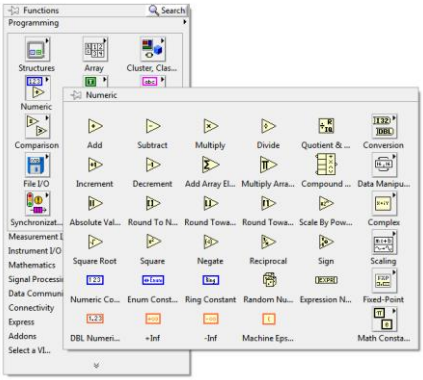
4.3.5 Paleta funkcija

Paleta funkcija sadrži grafičke palete koje se automatski otvaraju kada se pozove blok dijagram. Ova paleta se koristi da postavimo čvorove na blok dijagram VI-a. Svaka ikona“najvišeg nivoa” sadrži podpalete. Ako paleta funkcija nije vidljiva, opcijom Windows » Show Functions Palette iz blok dijagrama je aktiviramo.



Slika 4.7 Paleta funkcija

Paleta funkcija posjeduje različite korisnepodpalete koje se sadrže različite funkcionalne elemente koji se mogu koristiti u ovisnosti o potrebi. Neki od najčešće korištenih su numerička podpaleta koja sadrži elemente sa matematičkim funkcijama, slika 4.8. Podpaleta struktura(eng.*Structures*) gdje se nalaze sve vrste petlji. Također vrlo korisna paleta je paleta za obradu i generiranje signala (eng.*Signal Processing*). Elementi korišteni u primjeru pretvorbe temperature na blok dijagramu prikazani su i označeni na slici 4.1.

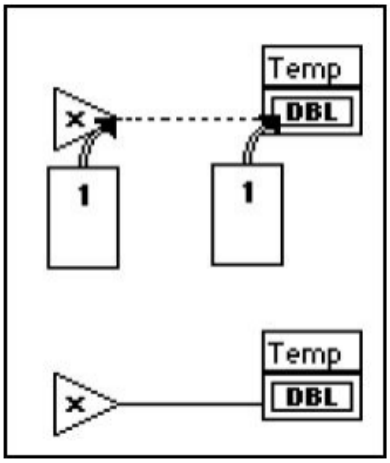


Slika 4.8 Podpaleta matematičkih funkcija glavne funkcijske palete

Slika 4.8 predstavlja podpaletu funkcijske palete i predstavlja matematičke funkcije koje se mogu koristiti pri programiranju. Ovako izvedba izrazito je jednostavna za rukovanje (*drag&drop*).

4.4. Povezivanje elemenata

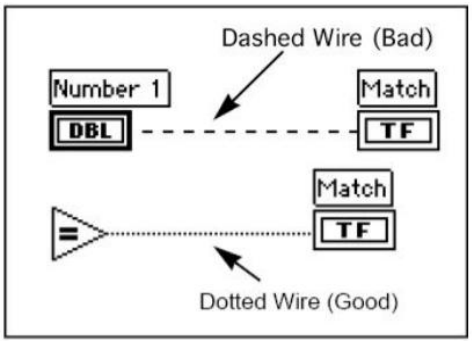
Da bi međusobno povezali dva elementa, aktiviramo alat za povezivanje na prvom priključku elementa, vučemo ga do drugog priključka elementai kliknemo na drugi priključak elementa. Kada je alat za povezivanje iznad priključka elementa, prostor oko elementa bljeska. Vodič možemo savijati pomjeranjem miša. Za mijenjanje pravca vodiča koristimo tipku space na tipkovnici.



Slika 4.9 Povezivanje elemenata blok dijagrama[3]

Isprekidani vodič predstavlja “loš“ vodič. Događa se kada elementi ne mogu biti povezani ili iz nekog razloga nisu povezani.

Najčešći razlog isprekidanog vodiča je nepodudaranje tipova podataka elemenata koji se međusobno povezuju. Ove vodiče briše se isto kao i vodiče kojiuspiješno povezuju dva elementa klikom miša na njih i pritiskom tipke delete. Opcijom Edit » Remove Bad Wires brišemo sve “loše vodiče”.



Slika 4.10 Uspješno i neuspješno povezivanje elemenata[3]

4.5. Tipovi podataka

Tip podatka ovisi o skupu vrijednosti koje varijabla[2] može poprimiti. Tipovi podataka koje koristimo unutar programskog jezika LabVIEW sličan je drugim programskim jezicima. Svaka vrsta podatka označena je različitom bojom pravokutnika. Slika 4.11 prikazuje tipove podataka koji se koriste za kontrolere i indikatore.

Slika 4.11Tipovi podataka korišteni u programskom jeziku LabVIEW[3]

U ovisnosti o tipu podatka varira i memorijski prostor potreban za smještanje varijable određenog tipa podatka. Ugrubo postoje dvije vrste tipova podatak jednostavni i složeni. Pod jednostavnim smatraju se sljedeći:

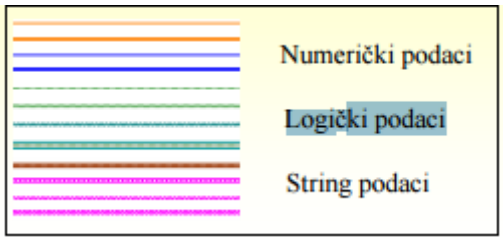
- cjelobrojni (*byte, short, int, long*) - koriste se za prikaz cijelih brojeva (pozitivnih i negativnih), u ovisnosti o rasponu vrijednosti koje može poprimit varijabla ovisi i broj bitova kojih se zauzima u memoriji računala
- brojevi s pokretnim zarezom (*float, double*) – poznati kao i realni brojevi
- znakovni (*string*) - niz znakova (svaki znak zauzima 16 bitova)
- logički (*boolean*) -ovaj tip vraćaju svi operatori usporedbe, može sadržavati samo dvije vrijednosti (ISTINA (1), LAŽ(0)), zauzima 1 bit

Pod složene tipove podataka spadaju sljedeće vrste podataka:

- nizovi (*array*)-skup podatak istog tipa
- skup različitih tipova podataka (*cluste*r)

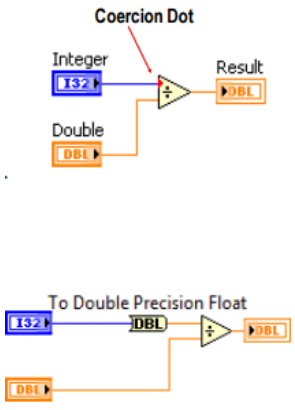
[2]imenovani podatak

Za kontrolere i indikatore vrijedi sve navedeno osim što kontroleri imaju deblji okvir. Vodiči su također označeni u ovisnosti i tipu terminala (elemenata) koje povezuju. Vodiči se razlikuju po boji (jedinstvena za svaki tip podatka) i debljini. Slika 4.12 prikazuje izgled vodiča u ovisnosti o tipu terminala koji povezuje.



Slika 4.12Izgled vodiča u ovisnosti o tipu terminala koji povezuje[8]

LabVIEW "pametan" je softver koji najčešće automatski i samostalno vrši konverziju numeričkih tipova podatak ukoliko se podatci ne poklapaju. Pri tome je važno naglasiti da LabView izabire reprezentaciju koja koristi više bitova. Konverziju podataka također je moguće izvršiti i samostalno od strane korisnika. Automatska konverzija podataka od strane softvera obilježena je korekcijskom točkom (*eng. CorectionDot*). Oba slučaja konverzije podataka prikazani su na slici 4.13[8].



Slika 4.13 Konverzija podataka[8]

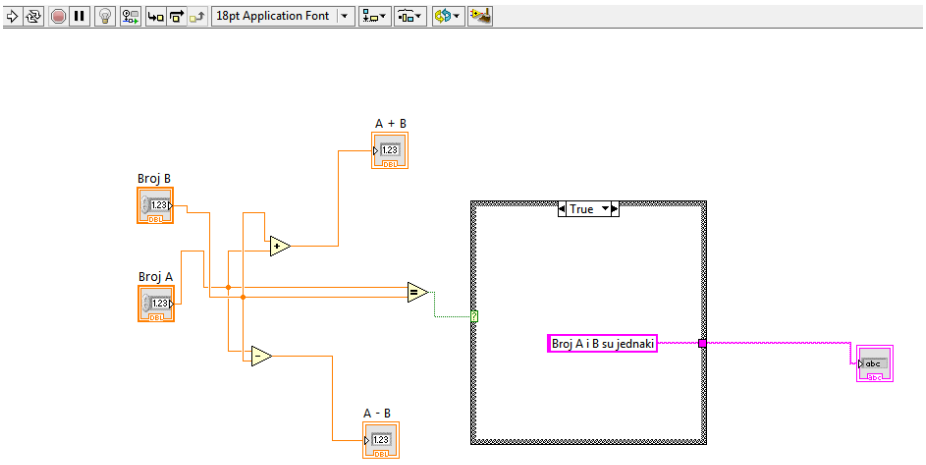
Na slici 4.13 gornji slučaj predstavlja konverziju podatka od strane softvera. Donji slučaj predstavlja konverziju podatka od strane korisnika.

4.6. Primjeri VI-a

Pod poglavlje 4.6 biti će iskorišteno za kreiranje nekoliko jednostavnih VI čiji koncept rada je potreban za shvaćanje programiranja u softveru LabView. Također će programi biti objašnjeni kao i predstavljeni blokovi koji su korišteni unutar svakog VI-a.

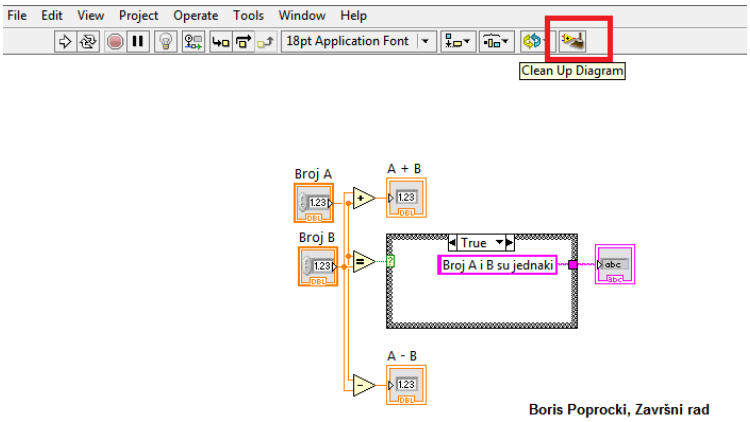
4.6.1 Primjer sa matematičkim funkcijama

U prvom primjeru u ovom pod poglavlju napraviti ćemo VI-a koji će zahtijevati od korisnika upis dva broja te vraćati razliku, zbroj i istinitost uvjeta jesu li brojevi jednaki.



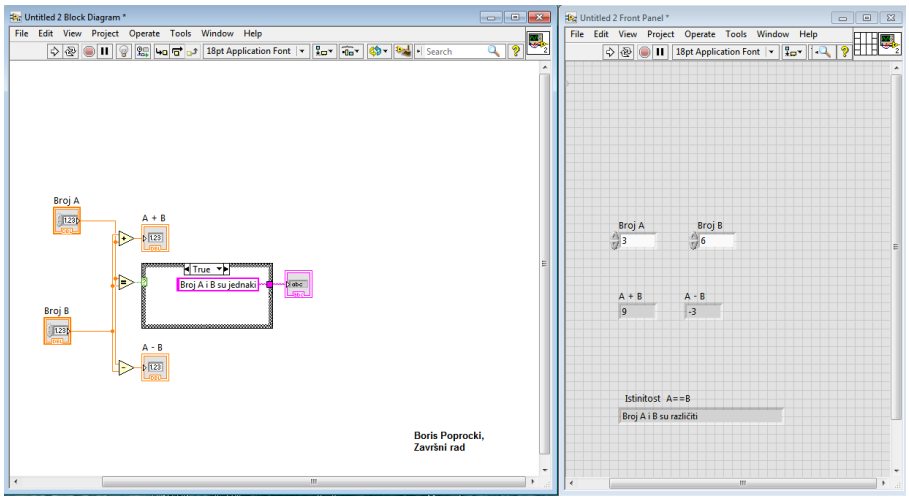
Slika 4.14 Izgled blok dijagrama prvog primjera

Na slici 4.14 može se vidjeti blok dijagram (funkcijska cjelina) prvog primjera programa. LabView posjeduje mogućnost pojednostavljivanja blok dijagrama nakon što je blok dijagrama kreiran od strane korisnika pritiskom na gumb (*eng. CleanUpDiagram*). Reakcija softvera na pritisak gumba za sređivanje i preglednost blok dijagrama može se vidjeti na slici 4.15.



Slika 4.15Sređeni blok dijagram sa slike 4.14

Pritiskom na gumb za izvođenje programa i unošenjem dva broja od strane korisnika vrši se izvršavanje programa. Rezultati izvršavanja programa mogu se vidjeti na slici 4.16

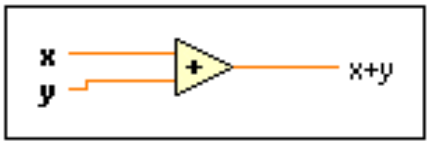


Slika 4.16 Blok dijagram (lijevo) i prednja ploča (desno) krajnji rezultat izvršavanja programa

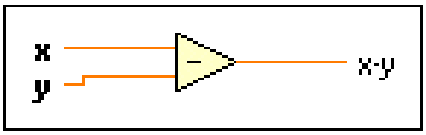
U primjeru prikazanom na slici 4.16korištena su dvakontrolera realnih brojeva A=3 slika 3.17, B=6 (upisani su cjelobrojni brojevi, mogu i cjelobrojni jer su oni podskup skupa realnih brojeva).

Slika 4.17 Kontroler realnih brojeva

Uz pomoć kontrolera korisnik unosi željene brojeve. Nakon što korisnik unese brojeve šalje ih dalje na elemente matematičkih funkcija za zbrajanje slika 4.18 i oduzimanje slika 4.19.

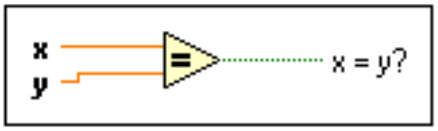


Slika 4.18 Element za zbrajanje



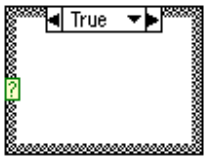
Slika 4.19 Element za oduzimanje

Izlaz iz elemenata matematičkih funkcija šalje se na indikatore realnih brojeva te se rezultat izvođenja prikazuje na prednjoj ploči. Kako bi usporedili jesu li dva unesena broja jednaka, koristi se komparatorska funkcija slika4.20čija je funkcionalnost izrađena od strane proizvođača softvera LabView-a.



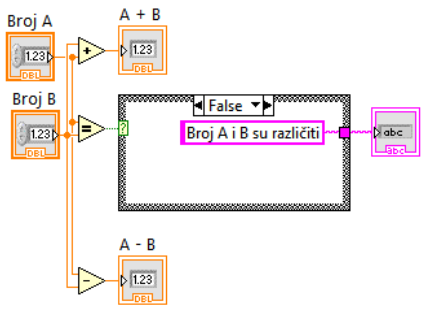
Slika 3.20 Element za uspoređivanje (komparatora)

Izlaz komparatorske funkcije je logički tip podatka (1 ili 0) poglavlje 4.6. Ukoliko je ovo laž izlaz će biti jednak nuli i obratno. Izlaz komparatora se prosljeđuje uvjetnoj (*engl. Case*) strukturi slika 4.21.



Slika 4.21 Uvjetna struktura

Uvjetna struktura na temelju ulaznog podatka koji može biti istina ili laž vrši izvođenje petlje. Ukoliko je ulazni podatak laž (0) kao na primjeru sa slike 4.16 vrši se dio petlje za laž slika 4.22.



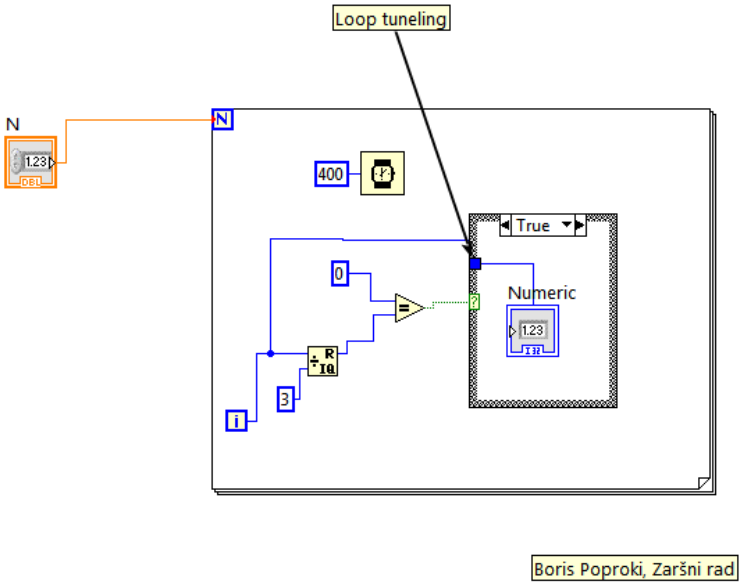
Slika 4.22 Uvjetna struktura ukoliko je podatak dostavljen na ulaz petlje (0)

Prikaz istinitosti korisniku na prednjoj ploči je predstavljen u obliku rečenice. Kako bi ovo bilo moguće nužno je koristiti znakovni (tip podatka) indikator (*string* indikator).

Slika 4.23String indikator

4.6.2 Primjer sa *for* petljom

U drugom primjeru biti će izrađen VI koji će ispisati prvih N brojeva djeljivih sa tri. Korisnik unosi broj N (N – broj do kojeg se ispisuju brojevi djeljivi sa tri počevši od nule). Kreiran i funkcionalni program prikazani je na slici 4.24.

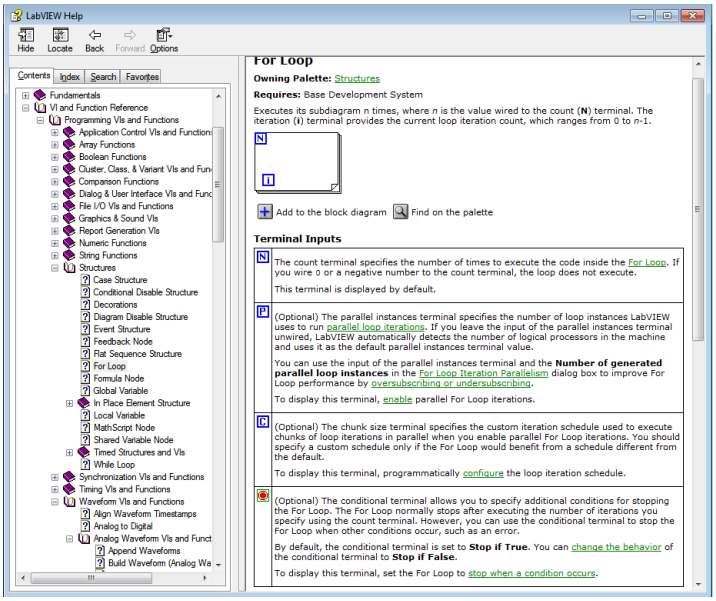


Slika 4.24 Blok dijagram drugog primjera

Primjer drugog programa zahtjeva od korisnika unos cijelog broja N. Ovo je omogućeno već prethodno objašnjenim kontrolerom cjelobrojnih brojeva. Novi element koji se koristi i zašto je uopće prikazan ovaj primjer je for petlja. For petlja je izuzetno korisna stvar u svakom programskom jeziku pa tako i u ovom. Uz pomoć for petlje omogućen je niz iteracija do broja N-1 u koracima po jedan. Svaku iteraciju varijabla *i* koja je dio strukture for petlja poprima vrijednost uvećanu za jedan u odnosu na prošlu iteraciju. Ovo se ponavlja sve dok varijabla *i* ne bude jednaka vrijednosti N-1. For petlja prikazana je na slici 4.25.

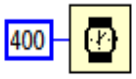
Slika 4.25 For petlja

NAPOMENA: Ukoliko funkcija elementa nije jasna ili je potrebno dodatno objašnjenje, funkcionalnost svakog od elemenata može se pronaći u help-u softvera koji je izrazito opširan i detaljan. Na slici 4.26 prikazan je help za for petlju.



Slika 4.26 Help LabView-a za for petlju

Ukoliko je izvođenje for petlje potrebno usporiti koriste se razni timeri. Timeri se kontroliraju numeričkim kontrolerima ili numeričkim konstantama. Numerička konstanta se koristiti kad korisnik nema potrebe utjecati na vrijednost tog podatka nego je vrijednost podatka određena od prije.



Slika 4.27 Numerička konstanta i timer

Numerička konstanta kojom se koristi timer izražena u ms i određuje svakih koliko ms će se dogoditi nova iteracija for petlje. Izlaz strukture for petlje (varijable *i*) koja se povećava svaku iteraciju za vrijednost +1, dovodi se na matematički element koji vraća ostatak dijeljenja. Ukoliko je ostatak dijeljena jednak nuli onda je broj koji se dijeli sa tri sigurno djeljiv sa tri.

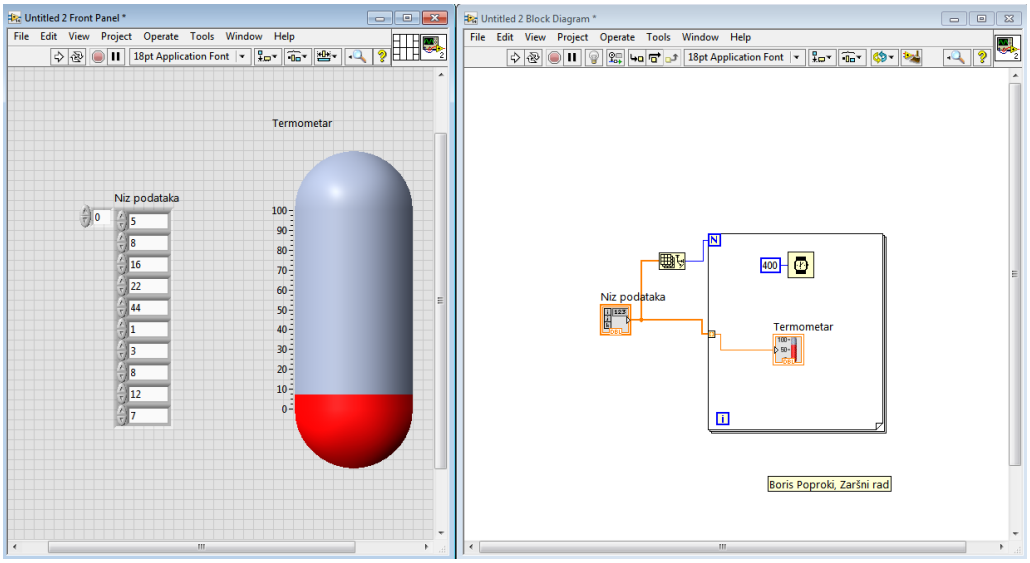
Slika 4.28 Matematički element koji vraća kvocijent i ostatak dijeljena dva broja

Ostatak dijeljenja se odvodi na komparatorski element. Komparator uspoređuje ostatak i numeričku konstantu koja je unaprijed definirana i iznosi nula. Ukoliko je ostatak nula, komparator vraća vrijednost jednaku 1 zato što je 0==0 (istinitost) te prosljeđuje na uvjetnu strukturu gdje se vrši slijed programa za istinu. Slijed programa u slučaju istine je ispis varijable i na prednju ploču. Kako bi ispis bio moguć potrebno je iskoristiti dodatnu mogućnost LabView-a (*eng. looptuneling*) koja nam omogućava prosljeđivanje podataka izvan neke strukture/petlje unutar strukture petlje, te obratno prosljeđivanje podataka koji su unutar strukture/petlje izvan strukture/petlje. Uvjetna struktura i komparatorska funkcija objašnjene su u prošlom primjeru.

4.6.3 Primjer sa poljima (*eng. array*)

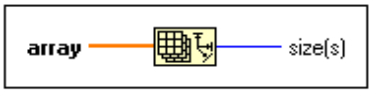
U trećem i zadnjem primjeru biti će prikazan rad sa poljima - nizovima podataka (*eng.array*). Ovakvi nizovi podataka predstavljaju zapravo jedan element (blok) koji se sačinjen od niza vrijednosti. Kako bi se moglo pristupiti svakoj od vrijednosti zasebno nužno je korištenje prethodno objašnjenih petlji (najčešće *for* petlje). Također vrijedi i obratno, od skupa nekih vrijednosti (podataka) moguće je napraviti niz podataka i predstaviti ih jednim blokom. Program treba na temelju niza podataka pohranjenih u polje podataka pristupiti svakoj od vrijednosti zasebno i prikazati ga u obliku temperature na indikatoru temperature (termometru).

Primjer tri prikazan je na slici 4.29.



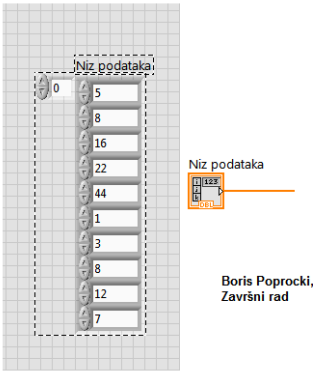
Slika 4.29 Blok dijagram desno i prednja ploča lijevo VI primjera tri

Niz cjelobrojnih podataka pohranjenih od strane korisnika prikazan je na prednjoj ploči. Ovom nizu podataka pristupa se *for* petljom. *For* petlja traži argument koliko puta će se iteracija izvršiti. Ovaj podatak for petlji prosljeđuje se uz pomoć gotove funkcije koja vraća broj elemenata niza slika 4.30. Unutar svake iteracija for petlji je dostavljena vrijednost koja odgovara tom mjestu retka. U primjeru je prikazan jednodimenzionalni niz (1x9) slika 4.31. Svaki niz podataka sastoji se od stupaca i redaka. Ukoliko su nizovi više dimenzionalni potrebno je koristiti ugniježđene[3] petlje. Svaka od vrijednosti u iteraciji dostavlja se indikatoru temperature te prikazuje na prednjoj ploči.



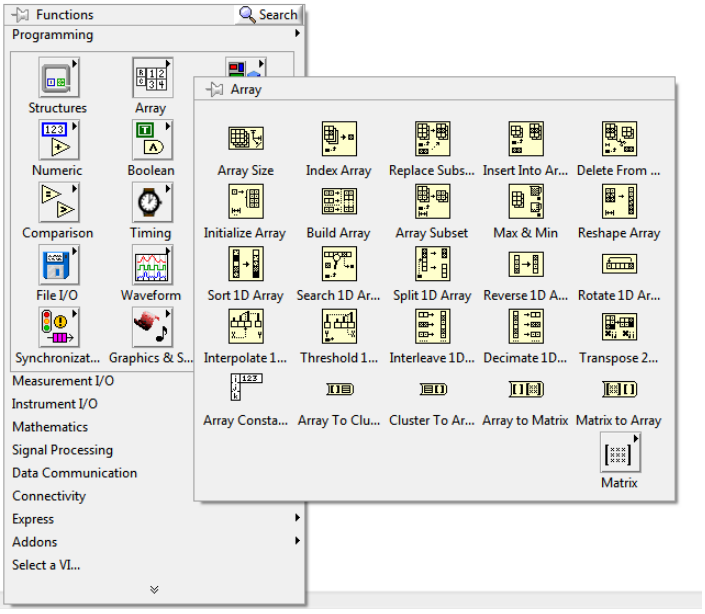
Slika 4.30 Funkcija za vraćanje broja elemenata niza

[3]petlja unutar petlje



Slika 4.31 Niz podataka na prednjoj ploči (lijevo), niz podataka unutar blok dijagrama (desno)

Jedna od posebnosti LabVIEW-a je što posjeduje već gotov niz funkcija za manipulaciju sa nizovima podataka, te na taj način značajno postiže uštedu vremena programeru (korisniku) koji ne mora sam programirati pojedine funkcije kada se ukaže potreba za njima.

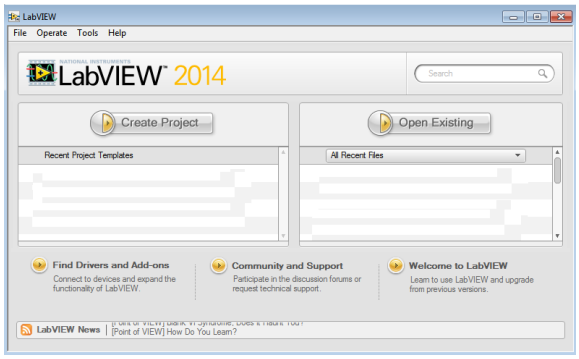


Slika 4.32 Gotove funkcije za manipulaciju sa nizovima podataka

U sljedećem poglavlju biti će izrađen VI-a za prikupljanje i obradu analognih signala te, generiranje digitalnih signala. Objašnjene funkcijske cjeline programa kao i elementi koji su po prvi puta korišteni u konstrukciji programa.

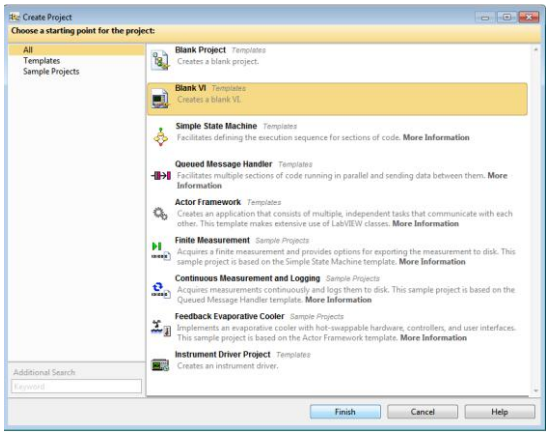
5. IZRADA PROJEKTNOG ZADATKA U SOFTVERU LABVIEW

Programiranje korisničkog programa u LabVIEW-u 2014 započinje klikom na tipku*Create Project*tustartnom sučelju LabVIEW-a slika 5.1.



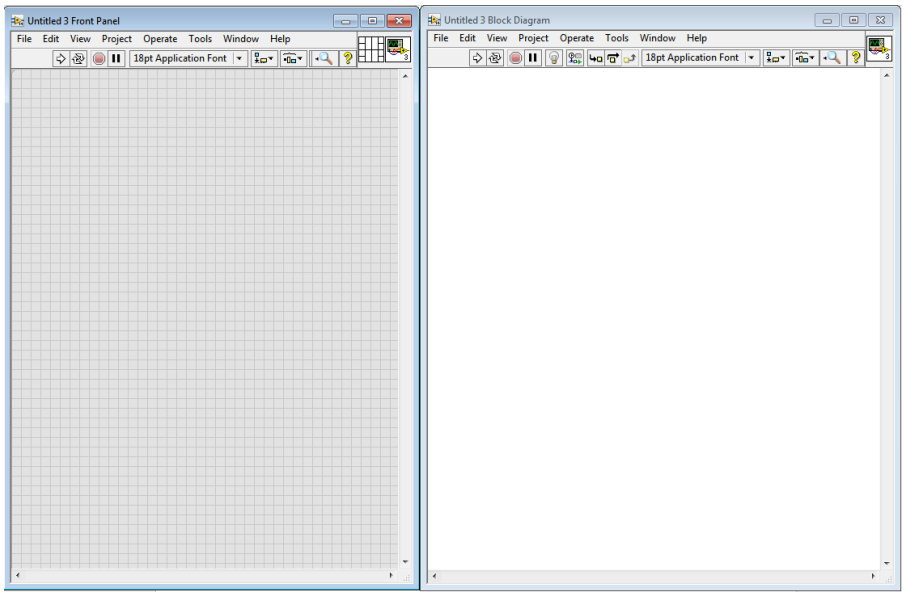
Slika 5.1 Startno sučelje

Zatim u slijednom sučelju dvostrukim klikom na link Blank VI, slika 5.2.



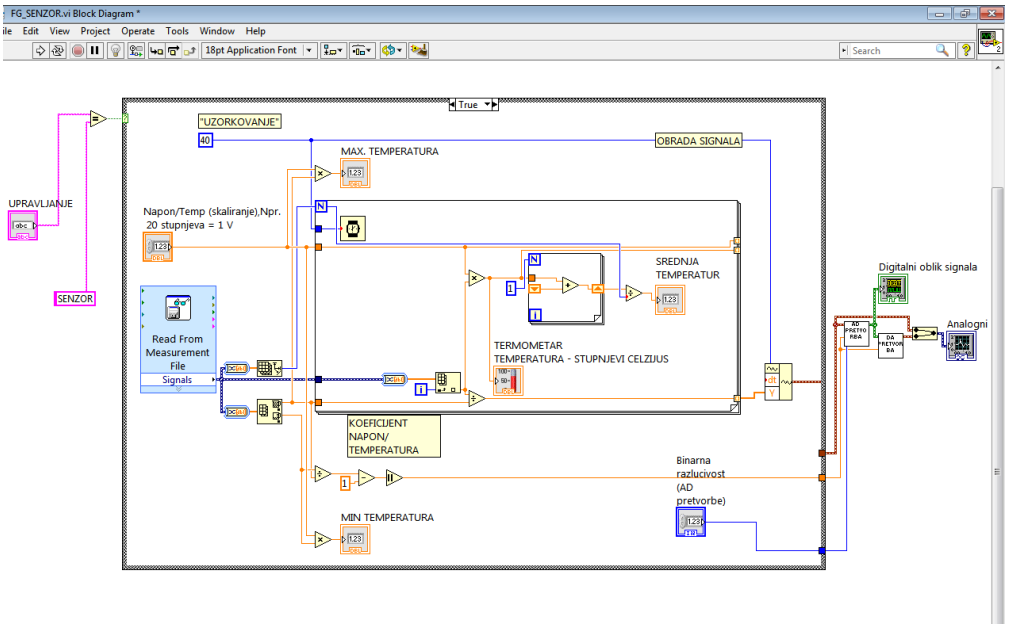
Slika 5.2 Slijedno sučelje LabVIEW-a

Cilj je programirati jedan VI predviđen za prikupljanje i obradu analognih signala te, generiranje digitalnih signala. Nakon slike 5.2 može se započeti s programiranjem programa. Program otvara prazan VI-a koji se sastoji od prednje ploče i blok dijagrama prikazano na slici 5.3, ovo predstavlja sučelje za programiranje (blok dijagram) i prikazivanje rezultata izvršavanja programa (prednja ploča).



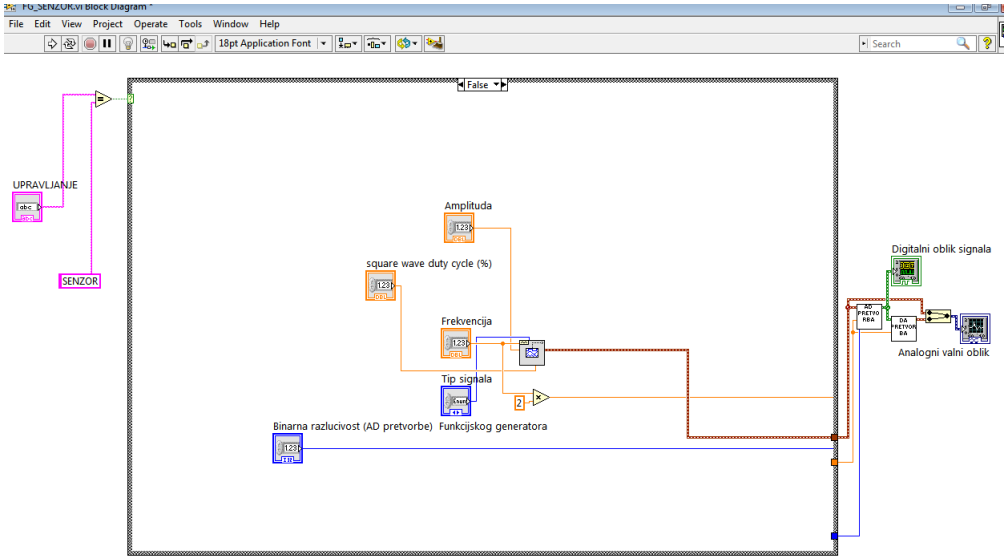
Slika 5.3 Novi prazni VI prednja ploča (lijevo) i blok dijagram (desno)

Dodavanjem i međusobnim spajanjem elemenata unutar blok dijagrama programa dobiva se slika 5.4 a), ukoliko se koristi mogućnost programa da prihvaća podatke iz realnog svijeta te na temelju upravljačkih signala generiranih od strane senzora vršimo obradu signala i generiranje digitalnog zapisa signala.



Slika 5.4 a) Blok dijagram programa upravljanog signalima senzora

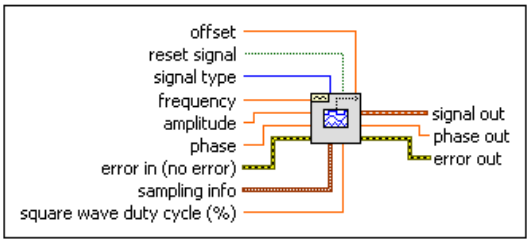
Slika 5.4 b) prikazuje izgled blok dijagrama ukoliko odlučimo koristiti funkcijski generator za generiranje digitalnog signala.



Slika 5.4 b) Blok dijagram ukoliko koristimo funkcijski generator za generiranje signala

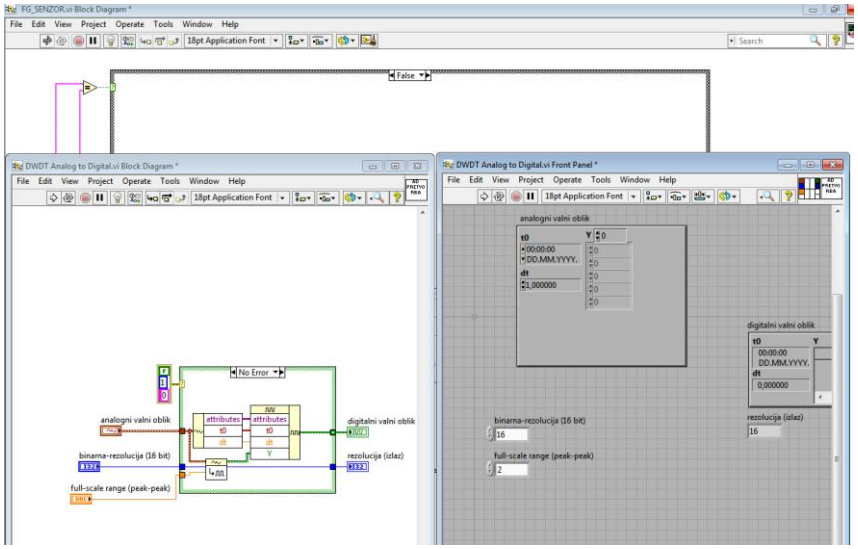
Slike 5.4a) i 5.4b) dio su istog programa u ovisnosti o odluci korisnika programa. Program na temelju uvjetne strukture vrši (objašnjeno u poglavlju 4.6) izvedbu slučaja a) odnosno slučaja b). Uvjetom strukture upravlja se logičkom funkcijom za uspoređivanje poglavlje (4.6). Na ulaze elementa za uspoređivanje dovode se dvije vrijednosti ukoliko su one iste, izlaz elementa je 1, ukoliko ovo nije točno izlaz bloka je 0, npr. ('SENZOR' == 'SENZOR', izlaz=1 ; 'a'=='SENZOR', izlaz=0).Uvjetna struktura dobiva vrijednost na priključak zvan "*Case selector*" te ukoliko je dobivena vrijednost jednaka 1 vrši se slijed programa koji je napisam u slučaju istinitosti slika 5.4a) , također sve isto vrijedi i u suprotnom slučaju slika 5.4b).

Ukoliko se izvršava program prikazan na slici 5.3.b) analogni signal je generiran od strane elementa (eng.*Basic Function Generator*) koji je dio softverskog paketa LabVIEW-a.



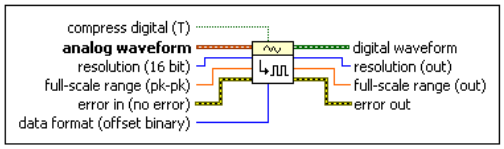
Slika 5.5 Blok koji predstavlja funkcijski generator (eng. BasicFunction Generator)

Element sa slike 5.5 daje nam mogućnost generiranja sinusnih, kosinusnih, pravokutnih i pilastih signala. Također nam daje dodatne mogućnosti odnosno upravljanje amplitudom, frekvencijom, faznim pomakom za sve signale te dodatnu mogućnost u slučaju pravokutnog signala za određivanje postotka pozitivnog dijela pravokutnog valnog oblika. Kao što je spomenuto i prikazano u poglavlju 5.2 svaki VI može imati podprogram*SubVI*-a. Tako VI-a glavnog programa ima *SubVI*-a za analogno-digitalnu pretvorbu elementa koji ga prikazuje na slici 5.3b) zove se "AD PRETVORBA". Blok dijagram *SubVI*-a može se vidjeti na slici 5.6.



Slika 5.6SubVI-a glavnog VI-a za analogno-digitalnu pretvorbu

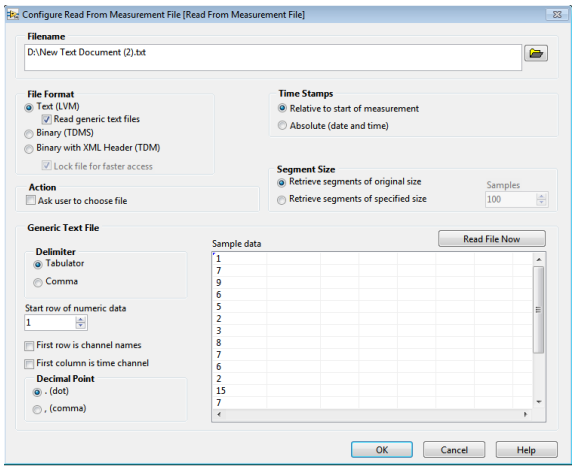
Element za analogno-digitalnu pretvorba slika 5.7u ovom slučaju ima mogućnost podešavanja jedino binarne razlučivosti, ulaz u element “Raspon signala“ ne može se podešavati od strane korisnika nego se koristi za rad programa, a neophodan je za funkcioniranje programa. Koristi se kako bi se odredio raspon signala između najpozitivnijeg i najnegatvnijeg dijela amplitude signala. Ovo je važno kako bi se mogao odrediti pravilan raspon nivoa kvantizacije i pri kodiranju kvantizacijskih nivoa(poglavlje 3.).



Slika 5.7 Element za analogno-digitalnu pretvorbu

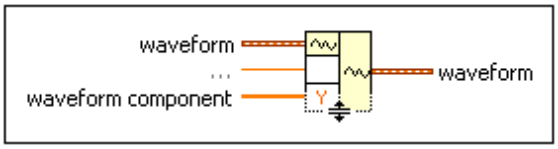
Sve što vrijedi za analogno-digitalnu pretvorbu vrijedi i za digitalno-analognu pretvorbu osim što je digitalno-analogna pretvorba predstavljena elementom “DA PRETVORBA“. Analogno – digitalna pretvorba i digitalno-analogna pretvorba izvan su uvjetne strukture zato što se koriste u oba slučaja izvođenja programa, kao i blokovi za prikazivanje signala na prednjoj ploči *WaveformGraph* i *Digital WaveformGraph* slika.

Ukoliko se korisnik programa odluči za rad u interakciji sa senzorom izvodi se program prikazan na slici 5.4 a). Slijed programa se mijenja. Element *ReadFromMeasurement File* traži podatke na temelju putanje koje mu zadamo (sučelje za upravljanje elementom za učitavanje mjernih podataka prikazano je na slici 5.8), prihvaća niz vrijednosti s definirane putanje te ih dalje prosljeđuje na obradu.



Slika 5.8 Sučelje za upravljanje elementa (eng.ReadFromMeasurement File)

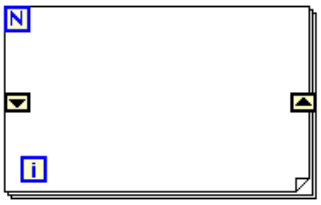
Kako bi se moglo raditi sa nizom vrijednosti koriste se polja (nizovi podataka – skup vrijednosti). Stoga je nužno koristiti petlje. Petlje se koriste kako bi se pristupilo svakoj vrijednosti niza zasebno. Brzinu pristupanja vrijednostima "uzorkovanje" moguće je kontrolirati elementom *Wait* (ms). Manipulacija nizovima podataka kao i korištenje petlje prikazano je i objašnjeno u poglavlju 4.6. Program od ovog trenutka radi s dvije varijable. Jedna varijabla je vrijednost senzora u trenutku u(t), druga varijabla je trenutak (t). Ovo su uređeni parovi od kojih generiramo prvo analogni signal (t, u(t)). Koliko senzor očita vrijednosti toliko će biti ovakvih uređenih parova i toliko koraka petlje. Varijable (t) i u(t) proslijeđujemo na blok za generiranje analognog signala (eng.*BuildWaveform*) blok za generiranje analognog signala kao ulazne vrijednosti koristiti objašnjene uređene parove vrijednosti, prikazan je na slici 5.9.



Slika 5.9 Element za generiranje analognog signala

Nakon čega signal ide na daljnju obradu analogno – digitalnu , digitalno-analognu pretvorbu opisanima na početku ovog poglavlja. Također se sa slike 5.4a) može vidjeti da se koristi ugniježđena petlja. Ugniježđena petljakoristi se za dobivanje srednje vrijednosti signala.

Unutarnja petlja je *shift-register*[4] (slika 5.10) i koristi se kako bi se mogla zbrojiti vrijednost u trenutku (t_n, $u(t)_n$) sa prošlom vrijednosti (prošlog koraka petlje) odnosno (t_{n-1}, $u(t)_{n-1}$). U načelu ovo se može shvatiti kao integriranje.

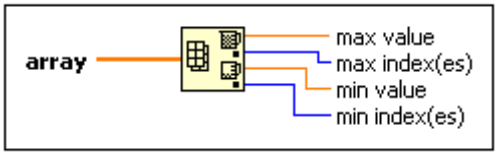


Slika 5.10 Shift-register sa for petljom

For petlja uz pomoć dodatnih elemenata unutar svije strukture (eng.*shift-register*) posjeduje mogućnost prosljeđivanja podatka iz prošle iteraciju u sljedeću iteraciju.

Slika 5.11 Shift-register

Također su prikazana maksimalna i minimalna vrijednost signala koje se dobiju na temelju gotovih funkcija softvera LabView-a za pronalaženje maksimalne i minimalne vrijednosti polja slika 5.12.



Slika 5.12 Funkcija za pronalaženje maksimalne i minimalne vrijednosti niza

Važno je napomenuti pretpostavku da senzor vraća različite naponske nivoe u ovisnosti o temperaturi unutar nekog postrojenja. Stoga se vrši skaliranje ulaznog signala te dobiva faktor na temelju kojeg se uvećava vrijednost očitana od strane senzora te dobiva stvarna temperatura postrojenja. Skaliranje i prilagođavanje podataka vrši se uz pomoć elemenata sa matematičkim funkcijama koji su objašnjeni u poglavlju 4.6

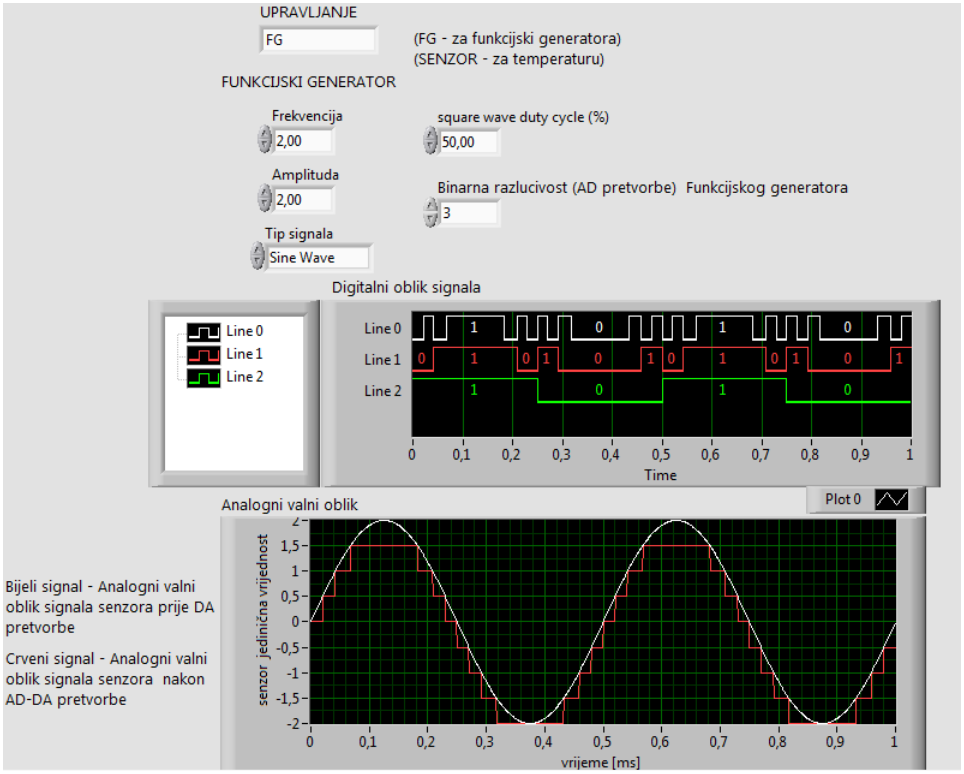
U sljedećempoglavlju biti će prikazan rad programa kao i detaljnije pojašnjenje prednje ploče.

[4]vrsta for petlje

6. IZVEDBA PROGRAMA

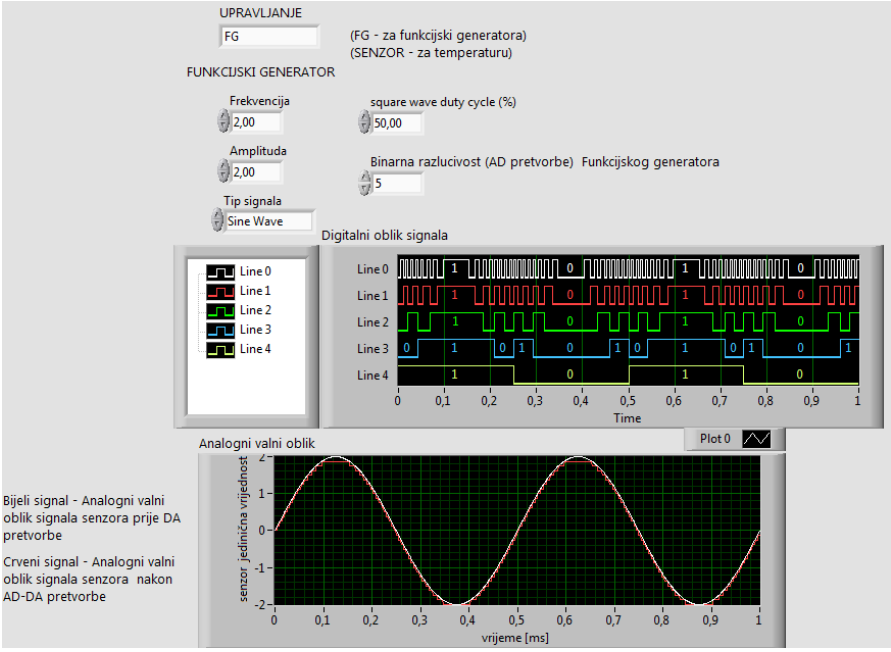
U poglavlju 6 biti će prikazani rezultati izvođenja programa za slučaj a)funkcijski generator i slučaj b) senzor.

Ukoliko korisnik upiše na prednjoj ploči u polju za kontrolu stringom (“ znakovnim nizom“ – *string*, “tipovi podataka objašnjeni u poglavlju 4. kao i indikator i kontroleri“) “FG“ vrši se izvođenje programa upotrebom funkcijskog generatora osnovnih signala slika 5.4 a). Prije pokretanja programa postavljaju se početni parametri signala npr.(tip signala- Sinusni valni oblik, amplituda = 2, frekvencija = 2 Hz, binarna razlučivost signala = 3). Što je binarna razlučivost veća aproksimacija digitalnog zapisa analognog valnog oblika je bolja (naravno ovo zahtjeva i veću opterećenost računala ili mikrokontrolera).



Slika 6.1 Prednja ploča za izvedbu a)

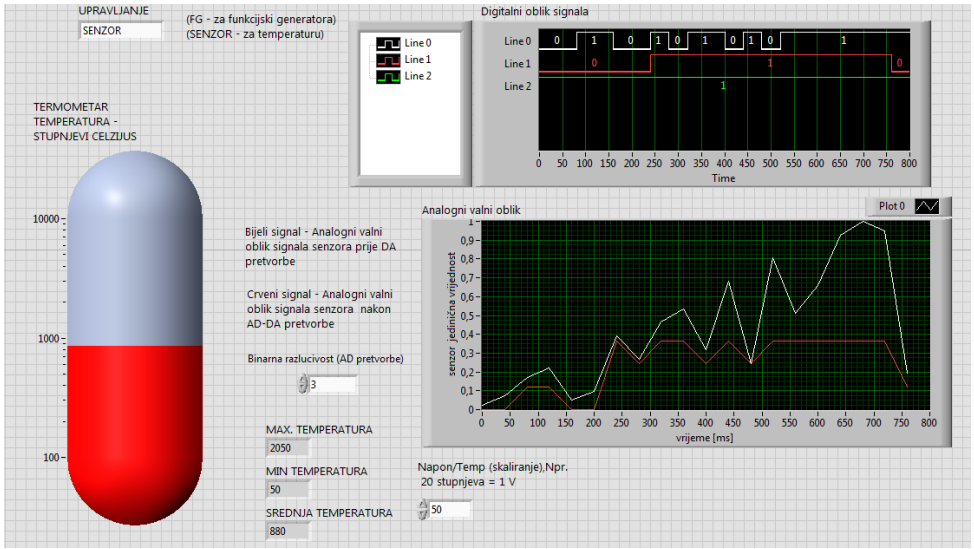
Na slici 6.1 mogu se vidjeti dva valna oblika analognog signala. Bijeli valni oblik je signal prije AD – pretvorbe, crveni signal je isti valni oblik istog analognog signala ali nakon AD-DA pretvorbe za vrijednost binarne-razlučivosti 3 . Ukoliko bi ova vrijednost bila veća npr. 5 razlika signala bi bila manja, prikazano na slici 6.2.



Slika 6.2

Sa slike 6.2 vidljivo je da pri binarnoj razlučivosti = 5, razlike signala prije i nakon pretvorbe skoro i nema (greška kvantiziranja je mala). Zapisi analognog signala u digitalnom obliku također se mogu vidjeti sa slike 6.1 i slike 6.2.

Ukoliko se korisnik odluči na primanje podataka od strane senzora izvodi se slijed programa za slučaj b) prema čemu dobivamo izgled prednje ploče kao na slici 6.3.



Slika 6.3 Prednja ploča VI-a za slučaj b)

Za način rada programa kao sa slike 6.3 korisnik mora na prednjoj ploči u polju za kontrolu *stringom* upisati "SENZOR". U ovom načinu rada može se vidjeti da na slici 6.3 odnosno na prednjoj ploči koristimo i termometar koji simulira rad ili pad temperature u postrojenju što se može vidjeti prilikom izvođenja programa. Također postoji uvid u maksimalnu i minimalnu temperaturu u postrojenju unutar nekog vremenskog intervala kao i srednju vrijednost temperature. Osim kontrole binarne razlučivosti postoji dodatna mogućnost podešavanja skaliranja. Za slučaj sa slike 6.3 ova vrijednost iznosi 50. Ovo zapravo znači da je razlika napona očitavanja senzora od 1 V jednaka razlici od 50 °C. Također, važno je napomenuti da su valni oblici prikazani na slici 6.3 predstavljeni za jediničnu vrijednost amplitude napona. Odnosno ukoliko je maksimalno očitavanje napona od strane senzora npr. 5 V, 5 V je predstavljeno vrijednosti jednakoj 1, također ovo znači da će maksimalna temperatura za 5 V unutar nekog vremenskog intervala iznositi 250 °C . Sve druge naponske vrijednosti senzora mogu se izračunati prema sljedećem izrazu:

$$Napon\ senzora\ [V] = \frac{tražena\ \ temp.}{Max.temp - Min.temp} \times koef.skaliranja \qquad (6 - 1)$$

Za digitalni zapis analognog valnog oblika vrijedi sve isto kao u slučaju a).

7.ZAKLJUČAK

U ovom radu prikazane su prednosti i mane objektno orijentiranih programa. Iz čega se vidi da nema previše smisla uspoređivati i govoriti o prednostima ili manama proceduralnih ili objektno orijentiranih programskih jezika. Korisnikprogramskoga jezika sam izabire način programiranja kao i programski jezik koji će koristit u ovisnosti u projektnom zadatku. Svaki od načina programiranja koristi se u ovisnosti o svojoj svrsi i potrebama kako bi programer zacrtani projektni zadatak izvršio uspješno u što kraćem vremenskom roku. Softver koji je korišten u radu za svrhe programiranja je LabView 2014 firmeNational Instruments (NI). LabView je grafičko orijentirani programski jezik koji je namijenjen za upotrebu od strane inženjera kojima primarna grana djelatnosti nije programiranje. Opisane su i objašnjenje osnovne i specifične osobine softverskog paketa. U poglavlju četvrtom također su prikazane osnovne funkcionalnosti i način programiranja na jednostavnim primjerima. Namjerno su izabrani jednostavni primjeri na kojima su objašnjenje osnove programiranja nužne za rad. Savladavanjem ovih osnova programiranja i njihovim povezivanjem u cjelinu korisnik postaje kompetentan kreirati složene programske zadatke. Jedan od složenijih zadatak koji je istovremeno i projektni zadatak ovog završnog rada je osmisliti i kreirati virtualni instrument za prikupljanje i obradu signala te generiranje digitalnih signala. Kako bi uopće bilo moguće izraditi program koji se bavi signalima nužno je barem osnovno teorijsko poznavanje signala. Teorijske osnove signala predstavljene su u trećem poglavlju ovog rada. Projektni zadatak kao i njegova izvedba predstavljeni su u petom i šestom poglavlju rada. Projektni zadatak uspješno je obavljen. VI instrument koji je izrađen posjeduje dvostruku mogućnost upravljanja. Prva mogućnost je upravljanje od strane senzora. Druga mogućnost predstavlja upravljanje generatorom osnovnih funkcija. Iako blok dijagram VI koji je konstruiran u svrhu prikupljanja , obrade i generiranja signala izgleda komplicirano on je povezana cjelina osnova programiranja. Kreirani VI posjeduje niz mogućnosti. Omogućen je prikaz srednje , minimalne i maksimalne vrijednosti temperature unutar nekog postrojenja za određeni vremenski interval. Na temelju naponskih nivoa senzora generira se kontinuirani analogni signal koji se može prebaciti u digitalni zapis. Generiranje digitalnog zapisa analognog valnog oblika vrši se u tri koraka a to su: uzorkovanje, kvantiziranje i kodiranje. Također je omogućena i inverzna pretvorba, kao i mogućnost usporedbe analognog signala prije AD-DA pretvorbe i nakon AD-DA pretvorbe. Usporedbom signala može se zaključiti da pretvarači unose određeni kvantizacijski šum. Što je kvantizacijski šum niži kvaliteta pretvorbe je veća.

Kvaliteta pretvorbe ovisi o binarnoj razlučivosti AD pretvarača. Binarna razlučivost predstavlja broj bitova koji se mogu koristiti za digitalnu predstavu analognog signala. Binarna razlučivost AD pretvarača ne treba biti niti prevelika. Nakon određene vrijednosti binarne razlučivosti, porast kvalitete signala je mala, a usporavanje rada računala ili mikrokontrolera značajno.

LITERATURA

[1]Z. Valter, Procesna mjerenja, ETF Osijek 2008., ožujak 2016

[2]Uvod u teoriju signala i sistema, https://www.scribd.com/doc/231121219/Uvod-u-Teoriju-Signala-i-Sistema,ožujak 2016

[3]National Instruments, Learn NI LabVIEW Basics 2014., travanj 2016

[4] Tomislav Perković, Primjena osjetila sila i momenata u radu robota, svibanj 2016

[5] Ivan Lujo, LabView grafičko programiranje, svibanj 2016

[6] DAQ proizvodi, http://www.ni.com/data-acquisition/, lipanj 2016

[7] Miloš Đalović, Programski paket LabView u računarski podržanom sistemu merenja i upravljanja,http://weblab.fink.rs/vezbe/3/docs/bsc_mdj.pdf , travanj 2016

[8]FER, Uvod u LabVIEW, https://www.fer.unizg.hr/_download/repository/Uvod_u_LabVIEW_2016._-_slideouts.pdf , lipanj 2016

SAŽETAK

U ovom radu dan je pregled osnovnih osobina programskog paketa NI LabVIEW 2014 razvojne tvrtke National Instruments. Također su prikazane i razlike između objektno-orijentiranih i proceduralnih programskih jezika kao i navedene prednosti programskog jezika LabVIEW 2014. Napisan je kratak uvod u signale, te opisan proces analogno-digitalne pretvorbe signala. Navedene su osnovne komponente korisničkog sučelja , potrebnog alata za programiranje. Prikazani su jednostavni VI na kojima su objašnjeni osnovni elementi i funkcije nužne za rad programa. Isto tako prikazan je glavni cilj ovog rada , tj. kreiranje i izvedba virtualnog instrumenta za prikupljanje, obradu , generiranje te pretvorbu analogno-digitalnih i digitalno-analognih signala.

Ključne riječi: NI LabVIEW 2014, korisničko sučelje, virtualni instrument, analogno-digitalna pretvorba, digitalno-analogna pretvorba, programiranje, signal, valni oblik, prednja ploča,blok-dijagram

Using LabView in collecting, processing and generating characteristic digital signals

ABSTRACT

This graduation work presents an overview of the basic properties of the software package NI LabVIEW 2014 of development company National Instruments. Also the differences between object-oriented and procedural programming languages and the advantages of LabView 2014 programming languages were presented. Short introduction to signals was written as well as conversion from analog to digital signals. The main components of the user interface were listed. Simple virtual instruments were explained, which are necessary in order for a program to function. In the end the main goal was to create a virtual instrument which collects, processes, generates and converts analog to digital signals and vice versa.

Keywords: NI LabVIEW 2014, userinterface, virtual instrument, waveform, analog to digitalconverter, digitalto analogconverter, aquasion

ŽIVOTOPIS

Boris Poprocki rođen je 16. Kolovoza 1993. u Sr.Mitrovici. Školovanje započinje 2000. godine u osnovnoj školi „Siniše Glavaševića“ u Vukovaru , završava 2008. godine , te upisuje Matematičku gimnaziju u Vukovaru , gdje 2012. godine maturira. Iste godine upisuje Elektrotehnički fakultet u Osijeku, stručni studij Elektrotehnike , smjer Elektroenergetika.