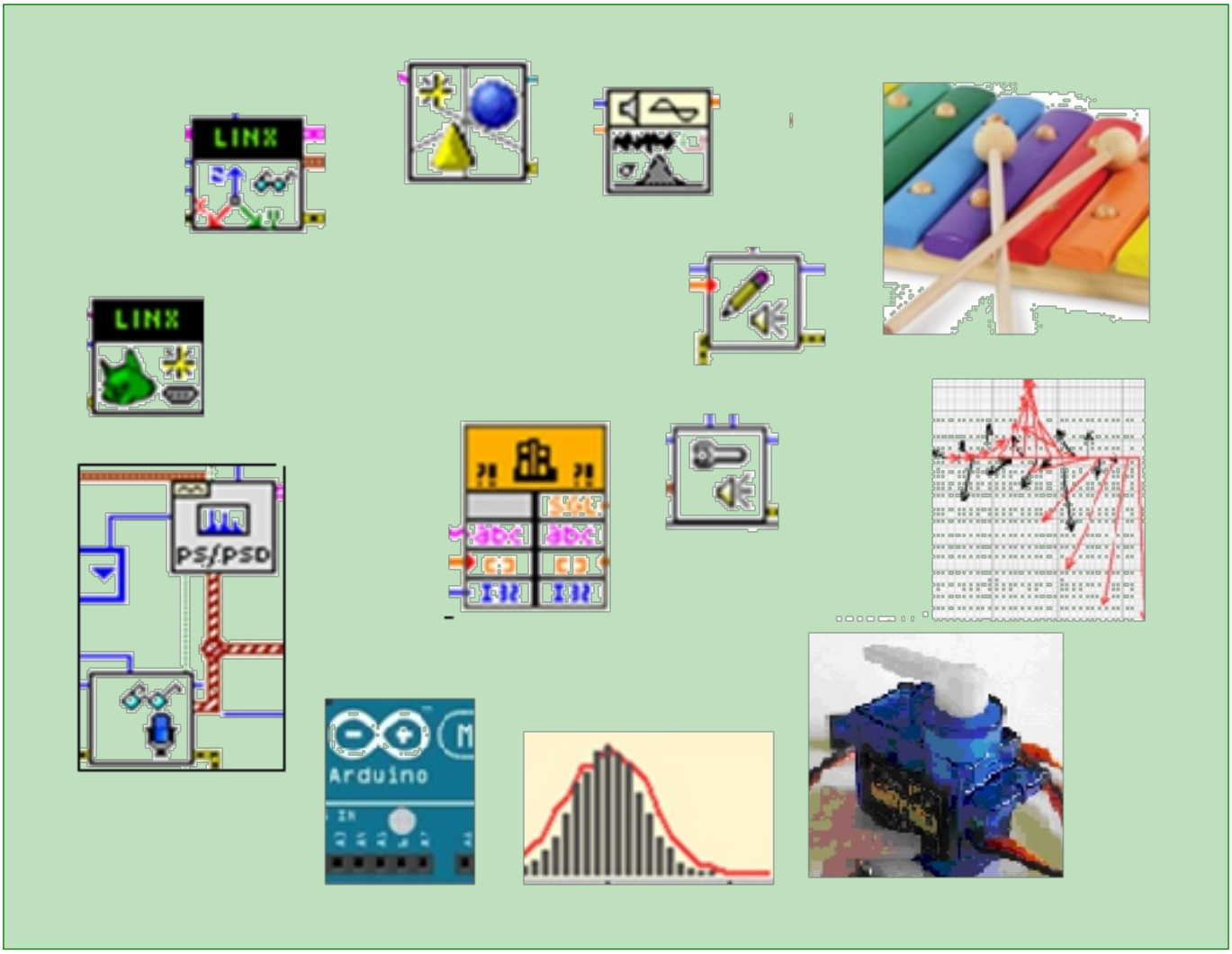
Iulian LUPEA

LabVIEW

APLICATII



UTPRESS
Cluj-Napoca, 2022
ISBN 978-606-737-598-5

Iulian LUPEA

LabVIEW

APLICATII

UTPRESS
Cluj - Napoca, 2022
ISBN 978-606-737-598-5

Editura UTPRESS
Str. Observatorului nr. 34
400775 Cluj-Napoca
Tel.: 0264-401.999
e-mail: utpress@biblio.utcluj.ro
http://biblioteca.utcluj.ro/editura

Director: ing. Dan Colțea

Recenzia: Prof.dr.ing. Horea Hedeșiu

Coperta: Iulian Lupea

Tehnoredactare computerizată: Iulian Lupea

ISBN 978-606-737-598-5
Bun de tipar: 21.11.2022

PREFAŢĂ

În lucrarea de faţă sunt prezentate aspecte de programare vizuală în mediul de programare LabVIEW prin folosirea elementelor grafice în schimbul textului, sunt supuse atenţiei aplicaţii diverse pentru fixarea noţiunilor, dar sunt şi aprofundate aptitudinile de programare. Limbajul grafic pus la dispoziţie de National Instruments Co. prin platforma LabVIEW, plecând de la limbajul grafic G, asigură crearea de instrumentaţie virtuală având ca suport calculatorul, achiziţia de date din mediu prin intermediul traductorilor, analiza şi prelucrarea informaţiei achiziţionate, procesări de imagini, dar şi simularea, controlul şi testarea automată a sistemelor inginereşti. LabVIEW este folosit în special pentru automatizări industriale, achiziţii de date şi control sub diverse sisteme de operare incluzând Windows, Unix, Linux şi macOS.

Denumirea limbajului este abrevierea expresiei englezeşti „Laboratory Virtual Instrument Engineering Workbench”, limbajul fiind conceput prin prima versiune în anul 1986 de firma National Instruments din Austin, Texas, SUA şi dezvoltat ulterior prin noi versiuni, cea mai nouă fiind LabVIEW 2021.

Spre deosebire de limbajele clasice folosite în asistarea achiziţiei de date din mediu prin intermediul plăcilor de achiziţie dedicate, limbajul LabVIEW prezintă o înaltă specializare, o bază remarcabilă de funcţii şi instrumente virtuale clasate pe domenii, asigurând astfel fiabilitatea şi rapiditatea programării.

Asocierea limbajului cu plăci de achiziţie şi alte periferice, face posibilă prelucrarea datelor numerice şi analogice prin simularea unor aparate de măsură şi control ce au primit denumirea de instrumente virtuale. Instrumentaţia virtuală, spre deosebire de instrumentele fizice, este mai flexibilă, permiţând modificare şi adaptare prin programare, şi este mai atrăgătoare din punctul de vedere al costului. Prin implementarea de primitive pentru control, indicaţie şi afişare specifice instrumentelor reale, este posibilă programarea grafică a instrumentului

virtual cu o interfaţă utilizator foarte asemănătoare panourilor frontale ale aparatelor dedicate. Funcţiile de măsurare, reglaj, analiză, control şi memorare ale aparatelor reale sunt implementate în aşa-zisa diagramă bloc a instrumentului virtual prin programare grafică pe baza unor funcţii (instrumente virtuale) existente, predefinite şi a unor funcţii implementate de programator, asigurând astfel o foarte bună modularizare a aplicaţiilor.

Noţiunile: tipuri de date simple şi structurate, expresii, operanzi şi operatori, controlul execuţiei programului prin decizii şi ciclări, controlul timpului, programarea pe evenimente, funcţii, lucrul cu fişiere, programarea obiectuală, servicii Web etc., sunt prezente şi adaptate specificului programării grafice. Limbajul de programare LabVIEW este dedicat atât aplicaţiilor de măsurare şi automatizare industrială, aplicaţiilor de laborator, cât şi nevoilor generale de calcul numeric susţinute de bibliotecile de funcţii matematice. LabVIEW permite integrarea codului scris în Python, C/C++, .NET sau Matlab.

Lucrarea se adresează studenţilor de la secţii cu specific ingineresc: mecatronic, inginerie mecanică, industrială, electrică, ingineria automobilului. Se presupune, în general, că studenţii au noţiuni elementare de programare însuşite în abordări anterioare. Studenţii au acum o flexibilitate şi înţelegere rapidă a noţiunilor de programare ca urmare a informatizării şi digitizării ca acţiuni prezente nu de puţini ani în societatea actuală.

Prezenta lucrare se adresează de asemenea inginerilor, fizicienilor, chimiştilor din sectorul productiv şi de cercetare precum şi cadrelor didactice din învăţământul universitar şi preuniversitar.

Autorul

LABVIEW

Aplicaţii

CUPRINS

I. Instrucţiunea de ciclare While

1. Mediul de programare LabVIEW

Pentru realizarea unei aplicaţii se lansează produsul LabVIEW şi se creează o nouă aplicaţie: File/New VI. Aplicaţia conţine două ferestre: Panoul frontal (PF) şi Diagrama bloc (DB). Se va împărţi ecranul calculatorului de preferinţă în două zone: PF fereastra din stânga şi DB fereastra din dreapta, cu comanda Window/ *Tile Left and Right.*

Paleta de unelte (Fig. 1) folosite pentru dezvoltarea aplicaţiei se vizualizează prin comanda: View/*Tools* Palette.

Fig. 1

Pentru a insera obiecte (controale şi indicatoare) în panoul frontal al aplicaţiei utilizăm comanda View/*Controls* Palette (sau click dreapta în PF). Pentru a realiza programarea grafică şi a insera funcţii în Diagrama bloc a aplicaţiei folosim comanda View/ *Functions* Palette (sau click dreapta din DB). Pentru a primi ajutor şi explicaţii cu referire la funcţiile inserate, la elementele de programare, la tipul datelor care sunt transferate prin firele de legătură sau la controale şi indicatoare se deschide fereastra de Help prin comanda Help/Show Context Help.

Diagrama bloc conţine codul grafic fiind o reţea grafică de noduri conectate prin fire de legătură. Nodurile sunt funcţii, subprograme sau operatori. Firele (în tabelul alăturat se observă grosimea, culoarea şi textura) sunt legături prin care se face transfer de date (variabile) într-un singur sens.

Tip de dată	scalar	1D Array	2D Array
numeric			
logic			
şir caractere			

O funcţie se execută atunci când are disponibile toate datele necesare primite prin firele de intrare (de la funcţii sau controale). Un nod (funcţie) fără intrări se execută imediat. Mai multe noduri pot avea simultan date necesare la intrare, deci se pot executa în paralel. După execuţia unui NOD se generează date de ieşire care se propagă prin fire la nodurile următoare conectate sau la indicatoare în PF. Deplasarea datelor prin noduri şi fire reprezintă execuţia programului.

2. Ciclul While - aplicaţie

2.1. Se dezvoltă o aplicaţie care permite modificarea valorii unui control numeric (Dial) din PF prin acţiunea operatorului şi vizualizarea grafică a variaţiei valorii controlului într-un interval de valori impus prin alte două controale numerice de tip Fill Slide (orizontal) sau Pointer Slide (Fig. 2).

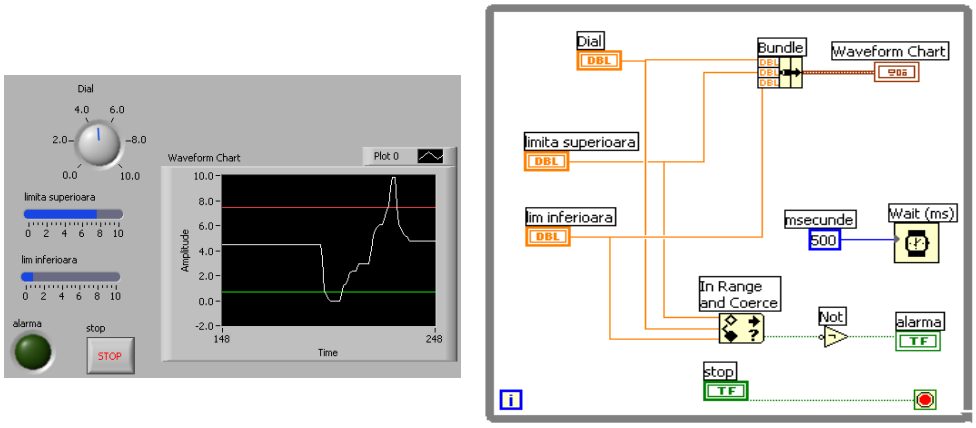


Fig. 2

2.2. Se roteşte controlul Dial cu unealta 'Operate Value - deget indicator'; cât timp valoarea controlului este între *limită superioară* şi *lim inferioară* nu se aprinde indicatorul boolean *alarma*. Pentru o valoare mai mare decât *limită superioară* sau mai mică decât *lim inferioară* se aprinde (schimbă luminozitatea) ledul *alarma*. Fereastra Help/Show Context Help se va deschide; aceasta va ajuta de asemenea, la conectarea pictogramelor în DB.

În PF se observă controale (Dial, stop …) prin care se introduc date în aplicaţie şi indicatoare (alarma, Waveform Chart) prin care aplicaţia returnează date (spre operator). Operatorul Bundle se poate redimensiona permiţând modificarea numărului de intrări; acesta grupează datele de intrare generând o structură de date sau cluster.

Funcţia *In Range and Coerce* prezintă trei (3) intrări şi două (2) ieşiri. Ieşirea de jos returnează valoarea logică True dacă valoarea numerică a controlului Dial este între cele două limite. În acest caz inserarea operatorului Boolean (Logic) NOT asigură stingerea indicatorului boolean 'alarma' de tip Round LED. Indicatorul *Round LED* (eticheta alarma) dacă primeşte True se luminează (On), altfel este stins (Off).

Indicatorul grafic Waveform Chart permite afişare continuă pe grafic (câte trei valori numerice la fiecare repetare a corpului ciclului While) a valorilor celor trei controale (dial şi cele două limite).

Corpul ciclului While se execută repetitiv cu o temporizare de 500 milisecunde (aproximativ două ciclări pe secundă). La fiecare iteraţie se citesc valorile celor trei controale numerice, se afişează aceste valori în indicatorul grafic

(trasând astfel trei curbe), se luminează sau stinge indicatorul logic tip LED, se citeşte valoarea logică a butonului STOP şi în funcţie de valoarea citită se decide repetarea corpului ciclului sau oprirea ciclării. Terminalul condiţional al instrucţiunii de ciclare While are două variante de funcţionare:

a) Stop if True  şi

b) Continue if True  .

Control *OK Buton* (eticheta STOP) la apăsare generează valoarea logică True, altfel generează False. În diagrama aplicaţiei este folosit terminalul condiţional Stop if True. Când operatorul apasă butonul Stop acest buton generează valoarea logică True care valoare ajunsă la terminalul ciclului va opri repetarea ciclului. Observăm faptul că corpul ciclului se va repeta cel puţin o dată deoarece oprirea se realizează după execuţia corpului.

2.3. Pentru vizualizarea valorii datelor care se transferă prin firele din diagramă se activează Highlight Execution - bec galben (Fig. 3).

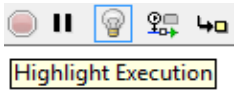


Fig. 3

2.4. Se va rula pas cu pas, cu urmărirea valorilor numerice şi logice prin comanda Start Single Stepping (Fig. 4).

Se observă culorile firelor. Culorile indică tipul datelor care trec prin fire: albastru (numeric - întreg), cărămiziu (numeric – real), verde (logic) şi firul pentru structuri sau cluster (între Bundle şi Waveform Chart).

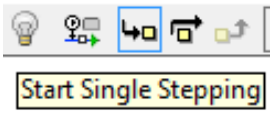


Fig. 4

3. Modificări aduse aplicaţiei

3.1. Să se redimensioneze contoarele şi indicatoarele din PF folosind unealta săgeată (Position/Size/Select) (Fig. 5).

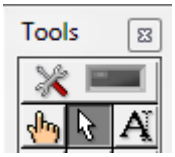


Fig. 5

3.2. Să se afişeze în panoul frontal (PF) ieşirea numerică *coerced(x)* a funcţiei *In Range and Coerce* (Fig. 6).

3.3. Înlocuiţi controalele din PF cu alte controale disponibile în paleta Numeric şi Boolean.

3.4. Afişaţi contorul ciclului While folosind un indicator numeric de tip întreg plasat pe PF.

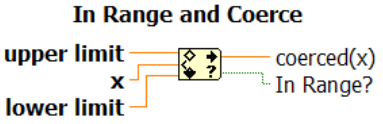


Fig. 6

3.5. Eliminaţi din aplicaţie ciclul While (Remove While Loop). Rulaţi aplicaţia prin apăsare repetată pe săgeata de rulare sau pe pictograma de rulare repetată (continuă) (Fig.7).

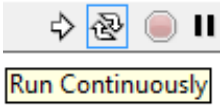
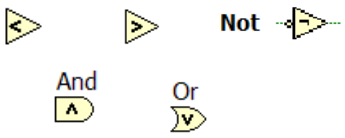


Fig. 7

3.6. Variaţi continuu numărul de milisecunde de temporizare introdus de funcţia Wait (ms) la fiecare ciclu While înlocuind constanta (500 ms) cu un control de tip *dial* plasat în Panoul Frontal (cu limitele 0 şi 1000 ms).

3.7. În locul funcţiei In range and Coerce folosiţi operatori relaţionali:

şi logici:



cu obţinerea unor acţiuni similare prin implementarea expresiilor următoare:

a) Not (Dial < limită superioară And Dial > limită inferioară)
b) (Dial > limită superioară) Or (Dial < limită inferioară)

3.8. Să folosim două leduri de alarmă etichetate *alarmă prea cald* şi *alarma prea rece*.

◦ dacă temperatura senzorului (ts/Dial) este între limite, ambele alarme sunt verzi;

◦ dacă *ts* > *lim sup* => numai led *alarmă prea cald* devine roşu, iar

◦ dacă *ts* < *lim inf* numai led *alarma prea rece* devine roşu.

Pentru stabilirea culorii LED-ului pentru stare True şi stare False procedăm astfel: click dreapta mouse pe LED, alegem Properties, iar din fereastra deschisă modificăm Colors (On = roşu şi Off = verde pentru ambele leduri).

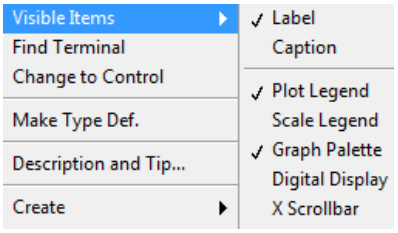


Fig. 8

3.9. Vizualizaţi/dezactivaţi comenzile asociate indicatorului grafic Waveform Chart din meniul Visible Items (Fig. 8).

4. Ciclul While (While loop) - reprezentarea grafică a unei funcţii

4.1. Se va observa un program LabVIEW (Fig. 9) pentru reprezentarea grafică a funcţiei y=sin(x)+x în intervalul [-11, 18.01]. Se foloseşte un ciclu While pentru evaluarea repetitivă a funcţiei pentru valori crescătoare ale abscisei x. Este utilizat contorul *i* pentru calculul abscisei curente cu relaţia:

x= -11+ i * pas

La prima iteraţie când i=0, abscisa devine x= -11. Valoarea funcţiei se calculează în cadrul entităţii *Expression Node*. Nodul expresie permite calculul unei expresii care conţine o singură variabilă (x în acest caz). Valorile rezultate la fiecare iteraţie din Expression Node se grupează la tunelul de ieşire din ciclul While într-un tablou de valori reale deoarece tunelul este setat în modul *Indexing* (Fig. 10). Dacă s-ar fi

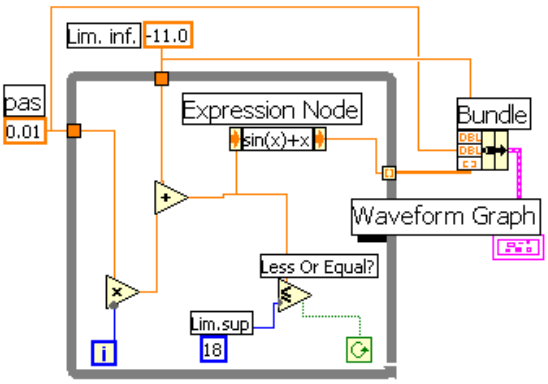


Fig. 9

setat modul *Last Value* numai ultima valoare calculată a funcţiei ar fi părăsit ciclul prin tunelul de ieşire.

Funcţia *Bundle* formează o structură (cluster) cu trei câmpuri aşezate în următoarea ordine: prima abscisă (Lim. inf.), pasul abscisei şi tabloul valorilor funcţiei. Structura formată este vizualizată prin indicatorul grafic Waveform Graph.

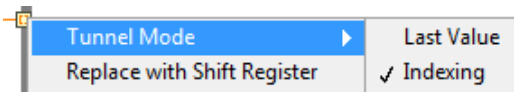


Fig. 10

Corpul ciclului se repetă cât timp terminalul condiţional 'Continue if True' primeşte valoarea logică True. Pentru x=18 se evaluează expresia relaţională 18≤18 de valoare True. Aceasta permite încă o repetare a ciclului. La evaluarea expresiei 18.01≤18 valoarea False rezultată opreşte ciclul. Astfel ultima valoare pentru care se evaluează funcţia este 18.01.

4.2. Utilizarea unui cursor

În Figura 11 observăm graficul funcţiei vizualizat în Panoul Frontal.

Pentru a citi puncte sau perechi (abscisă, ordonată) de pe grafic se poate insera un cursor. Pentru aceasta se bifează Cursor Legend din paleta *Visible Items* (click dreapta pe grafic), apoi Create Cursor (click dreapta de pe suprafaţa legendei cursor) şi se alege varianta Single-Plot care asigură rămânerea cursorului ataşat punctelor de pe curbă. Se poate adăuga al doilea sau al treilea cursor după nevoie.

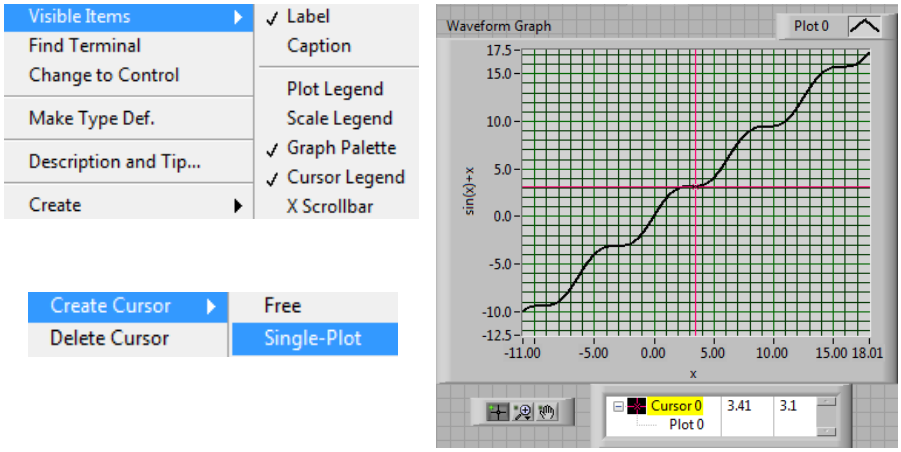


Fig. 11

Schimbarea culorii fondului din indicatorul grafic (implicit background negru) se realizează prin Click dreapta pe grafic având selectată unealta corespunzătoare (pensula) din Tools Palette.

Pentru schimbarea grosimii curbei trasate, din Visible Items se bifează Plot Legend. După click cu mouse-ul pe zona din dreapta textului Plot 0 se selectează Line Width (Plot Visible trebuie sa fie bifat) (Fig. 12). Alte setări cum sunt culoarea curbei sau stilul liniei sunt disponibile în mod similar.

4.3. Modicaţi diagrama aplicaţiei

1. Înlocuiţi constantele numerice Lim. inf, Lim. sup şi pas cu controale în panoul frontal,
2. Modificaţi Lim. inf, Lim. sup şi pasul asociate reprezentării grafice a funcţiei.
3. Ce tipuri de date foloseşte aplicaţia ?
4. Care este rolul contorului *i* şi cum se realizează oprirea ciclului While? (Continue if True sau Stop if True)
5. La ieşirea din While este setat 'Enable Indexing' pentru cumularea valorilor calculate şi vizualizare cu Waveform Graph. Implicit ieşirea valorilor din While Loop este fară indexare, caz în care iese numai ultima valoare.
6. Să se adauge un Waveform Chart în ciclul While şi temporizare 10 ms /ciclu pentru vizualizarea fiecărei valori a funcţiei imediat după calculul valorii în cadrul 'Expression Node'.
7. Reprezentaţi alte funcţii de o variabilă înlocuindu-le în Expression Node (modificaţi limitele variabilei x).

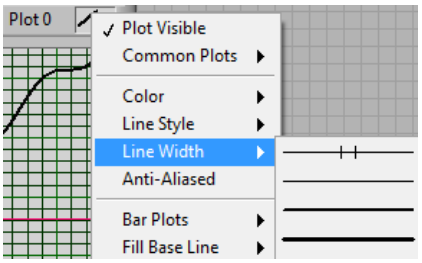


Fig. 12

4.4. Scară liniară versus logaritmică

Se consideră funcţia y=log(x) şi scară logaritmică pentru abscisa x, iar pentru ordonată scară liniară. Se obţine graficul din Figura 13. Limitele sunt de la 0.01 la 1000 pentru abscisa x.

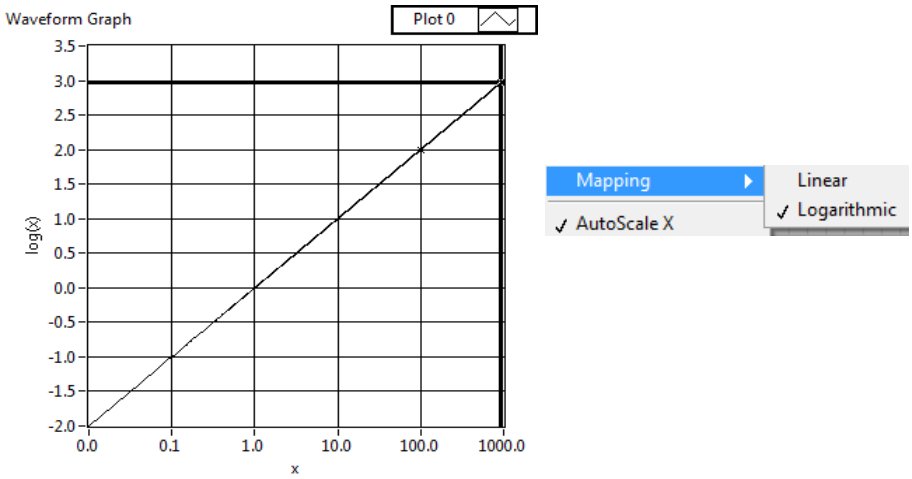


Fig. 13

5. Generarea repetitivă a unui semnal triunghiular

În diagrama din Figura 15 este apelată funcţia Triangle Pattern.vi din paleta Signal Generation. Semnalul generat conţine 40 eşantioane (samples). Spaţierea între eşantioane *dt* este de 0.5 s. Lăţimea (width se referă la bază) triunghiului (Fig. 14) este de 5 secunde, rezultând 10 eşantioane din totalul de 40; *width(s)* este control, astfel parametrul este schimbat interactiv pe măsură ce se repetă ciclul While. Întârzierea sau *delay(s)* este de două secunde (2 s) realizată prin patru (4) eşantioane de valoare zero. Din controlul *asymmetry* se poziţionează vârful triunghiului în mod interactiv. Amplitudinea este implicit 1. În Figura 15 observăm diagrama bloc şi panoul frontal al aplicaţiei prin care se modifică interactiv asimetria şi baza semnalului triunghiular. Programul se opreşte simplu prin apăsarea butonului de Stop.

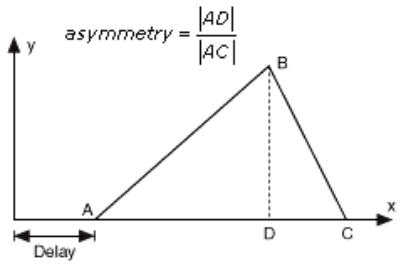


Fig. 14

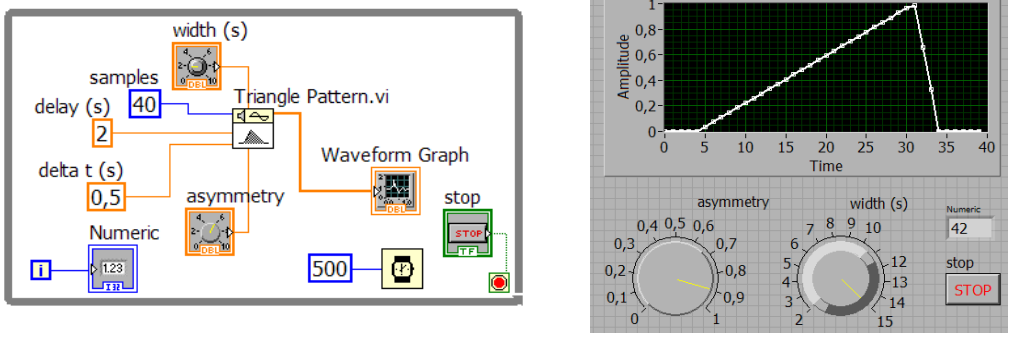


Fig. 15

II. Ciclul FOR, registrul Shift, operatorul SELECT

1. Ciclul FOR şi registrul Shift - suma elementelor din tabloul 1D

1.1. Se programează calculul sumei elementelor reale dintr-un tablou 1D. Este folosită instrucţiunea de ciclare For care asigură repetarea corpului ciclului de un număr fix (N) de ori. Elementele reale ale tabloului 1D intră în corpul ciclului (printr-un tunel) implicit cu indexare, deci la fiecare ciclare în corp intră un nou element din şir şi acesta se adună la suma elementelor precedente. Ciclul For se repetă de atâtea ori câte elemente sunt în tabloul care intră cu indexare. La ultima repetiţie a ciclului intră ultima valoare din tablou şi se obţine suma tuturor elementelor din şir. Terminalul N din colţul stânga-sus al cadrului furnizează o valoare egală cu numărul de elemente din tabloul de intrare (pentru indexare activată).

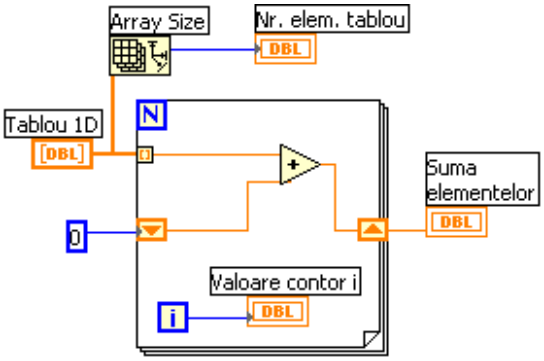


Fig. 1

1.2. Registrul Shift (de transfer) transferă o valoare, şi anume suma parţială, de la o iteraţie la următoarea, din 'sertarul' drept al registrului la iteraţia curentă în cel stâng la iteraţia următoare. Registrul este iniţializat cu valoarea zero, astfel prima suma este dintre primul element din tablou şi valoarea zero din registru. La terminarea celor N iteraţii valoarea din registrul Shift trece in afara ciclului la indicatorul Suma elementelor.

1.3. Observaţi indicaţia contorului ‘i’. Acesta porneşte de la valoarea întreagă 0. Funcţia Array Size returnează numărul de elemente din tablou.

1.4. Memoraţi toate sumele parţiale şi afişaţi-le în Panoul Frontal (PF) la ieşirea din ciclu. Observaţie: în corpul ciclului, valoarea rezultată din adunare legaţi-o printr-un fir la cadrul ciclului; valorile se vor memora (varianta cu indexare) la tunelul de ieşire din ciclu, iar tabloul generat se afişează la ieşirea din ciclu.

1.5. Implicit un tablou de elemente intră în corpul ciclului For prin indexare. Se poate schimba setarea la nivelul tunelului de intrare astfel încât să intre toate elemntele tabloului odată prin setarea Disable Indexing (Fig. 2). În acest caz toate elementele sunt disponibile în ciclu la orice iteraţie.

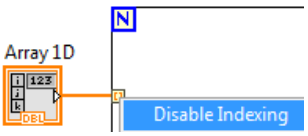


Fig. 2

2. Precizări cu referire la registrul de transfer - Shift register

2.1. La inserarea registrului pe laturile ciclurilor For sau While (Fig. 3), terminalele registrului sunt negre, deci nu este încă stabilit tipul de dată care va fi conţinut în registru**.**

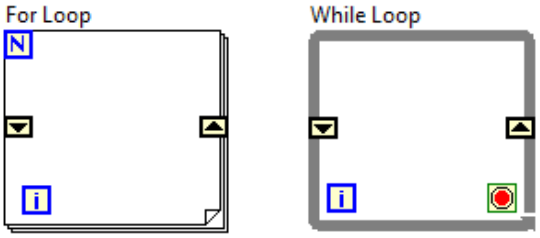


Fig. 3

2.2. Registrul iniţializat cu o valoare numerică

În diagramă registrul este iniţializat cu valoarea 10 de tip întreg (albastru). După adunarea unei valori reale (0.2) culoarea registrului devine cea a tipului de dată mai general deci cărămiziu, corespunzând datei reale în dublă precizie conţinute în registru.

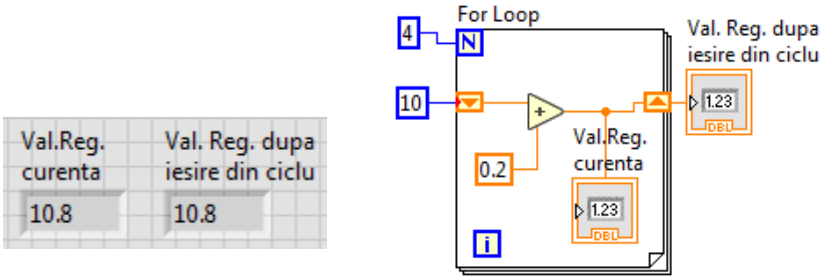


Fig. 4

2.3. Registrul neiniţializat

La terminalul stânga al registrului nu este conectată nici o dată de input astfel registrul este neiniţializat. Dacă terminalul din stânga nu are date de intrare conectate la prima iteraţie a ciclului, valoarea din terminalul stâng va fi valoarea implicită pentru tipul de dată care este memorat în registru.

Ciclului For i se impun patru repetări prin conectarea constantei (4) la colţul stânga sus al ciclului. La prima execuţie a aplicaţiei rezultă valoarea finală 0.8 în registru şi la nivelul fiecărui indicator din PF (Fig. 5). La a doua execuţie a aplicaţiei se porneşte de la valoarea 0.8 memorată în registru şi se continuă adunarea constantei 0.2 la fiecare iteraţie rezultând 1.6 la finalul rulării a doua a aplicaţiei.

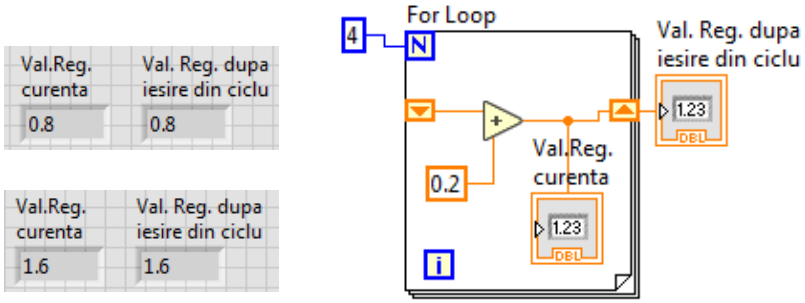


Fig. 5

2.4. Ciclul For cu zero iteraţii

În cazul în care corpul ciclului execută zero iteraţii, registrul Shift transferă valoarea de iniţializare (Fig. 6).

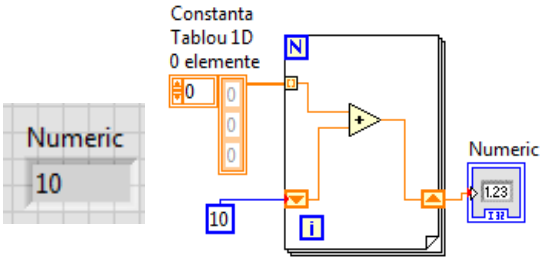


Fig. 6

3. Suma elementelor pozitive dintr-un şir de numere reale, operatorul Select

3.1. Operatorul Select prezintă trei intrări şi o ieşire a datelor (Fig. 7).

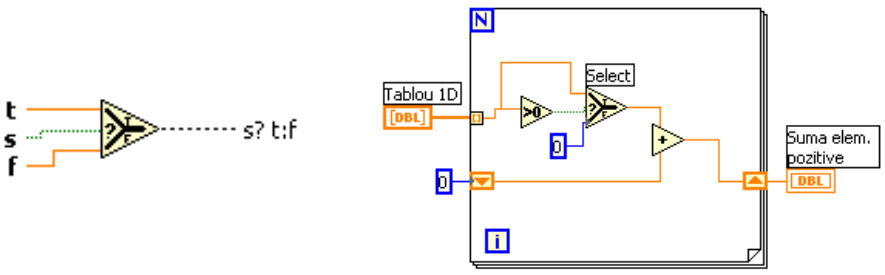


Fig. 7

Dacă valoarea logică primită pe intrarea de selecţie (cea din mijloc) este True, operatorul Select permite trecerea valorii de pe ramura T (True) (intrarea de sus), iar dacă valoarea primită este False permite trecerea valorii de pe ramura F (False) (intrarea de jos).

3.2. Registrul Shift funcţionează ca o memorie locală ciclului permiţând transferul unei date/valori, în acest caz suma parţială a valorilor pozitive (Fig. 7) de la o iteraţie la următoarea iteraţie.

Dacă elementul curent este pozitiv, selectorul, în urma comparaţiei cu zero, primeşte valoarea True şi permite transferul valorii elementului curent în vederea adăugării la suma parţială memorată în registrul Shift. Pentru element curent negativ sau nul se va adăuga valoarea zero.

Suma parţială iniţială este zero, registrul Shift fiind iniţializat din afara ciclului. Dacă registrul nu este iniţializat cu 0 la prima rulare rezultatul este corect. La următoarea execuţie a programului registrul va avea memorată suma de la prima rulare.

4. Exemple cu Select, tablou de şiruri de caractere, tablou de valori logice

4.1. Selecţie între două constante de tip şir de caractere

Se compară elementele reale dintr-un tablou 1D cu zero şi se generează un tablou de şiruri de caractere (indicator) care conţine şirurile 'Nr.Pozitiv' şi 'Nr.<0 sau 0'. Ieşirea din ciclu este în mod 'indexing'; la tunel se cumulează şirurile de caractere de la fiecare iteraţie şi la terminarea ciclului for iese un tablou de şiruri de caractere (Fig. 8).

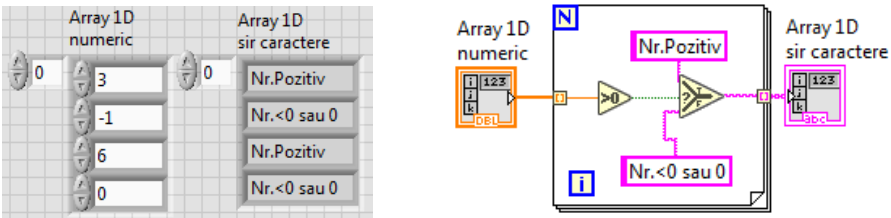


Fig. 8

4.2. Selecţie între două valori logice (constante)

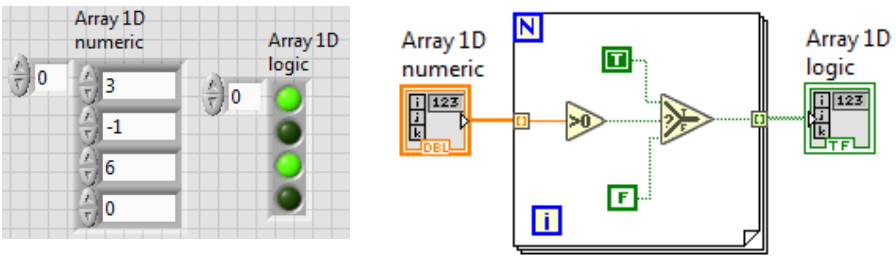


Fig. 9

Se compară elementele reale dintr-un tablou 1D cu zero şi se generează un tablou (indicator) de valori logice. Ieşirea din ciclu este 'indexing'; la tunel se cumulează logice de la fiecare iteraţie şi la terminarea ciclului iese un tablou de valori logice (Fig. 9).

5. Suma elementelor dintr-un tablou 2D

Calculaţi suma elementelor din tabloul 2D (array). Se vor folosi două cicluri For imbricate. Pentru a obţine un tablou (array) 2D în PF adăugaţi o dimensiune (Add Dimension) tabloului 1D deja creat (Fig. 10).

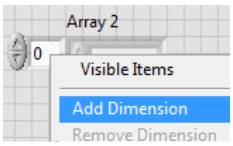


Fig. 10

6. Suma elementelor aflate deasupra DP într-o matrice pătratică (cicluri FOR imbricate + Select)

6.1. Ciclul For exterior (Fig. 11) permite accesul matricei în corpul ciclului, linie cu linie, prin urmare ciclul exterior se repetă de atâtea ori câte linii sunt în matrice. Ciclul For interior permite accesul unei linii în corpul lui, element cu element, astfel pentru fiecare linie ciclul interior se repetă de atâtea ori câte elemente sunt într-o linie (numărul de coloane a matricei). Astfel ambele tuneluri de intrare sunt cu indexare.

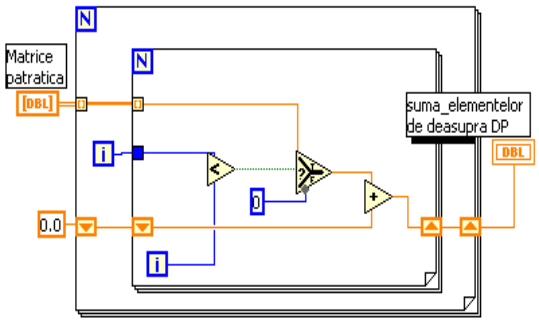


Fig. 11

6.2. Dacă indicele liniei este mai mic decât indicele coloanei, selectorul funcţiei Select primeşte valoarea True şi permite adunarea elementului curent la suma parţială memorată în registrul Shift. Elementele având indicele de linie mai mic decât indicele de coloană sunt situate deasupra diagonalei principale (DP).

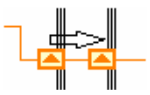


Fig. 12

6.3. Registrul shift al ciclului interior transferă sume parţiale de la o iteraţie la următoarea pe timpul parcurgerii unei linii din matrice. La terminarea elementelor dintr-o linie, terminalul drept al registrului interior trece valoarea în terminalul drept al registrului ciclului exterior (Fig. 12).

7. Registrii shift (de transfer) multipli - şirul Fibonacci

7.1. Relaţia de calcul a elementelor din şir:

$$f_{n+1} = f_{n-1} + f_n , \quad f_0 = f_1 = 1$$

7.2. La prima ciclare cele doua *sertare* din stânga ale unicului registru de transfer primesc valoarea 1; după adunare valoarea (2) se salvează în sertarul (terminalul) dreapta. La

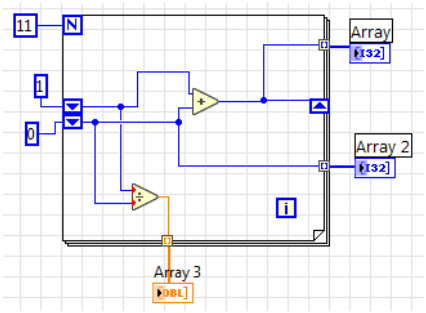


Fig. 13

iteraţia următoare valoarea 2 se transferă în sertarul de sus – stânga, iar valoarea din sertarul sus este salvată automat în sertarul stânga – jos etc.

7.3. Calculaţi şi afişaţi şirul Fibonacci folosind doi regiştri shift.

8. Precizări cu referire la tablouri de date

Tabloul 1D sau 2D (Array 1D sau 2D) este o colecţie de date de acelaşi tip aşezate în tablou pe poziţii referite printr-un indice pe fiecare dimensiune a tabloului. Elementele tabloului pot fi toate de tip numeric sau boolean (logic), şir de caractere, structură, formă de undă etc.

8.1. În Figura 14 observăm un tablou 1D numeric (real dublă precizie) şi unul 2D numeric controale în PF şi terminale asociate în diagramă.

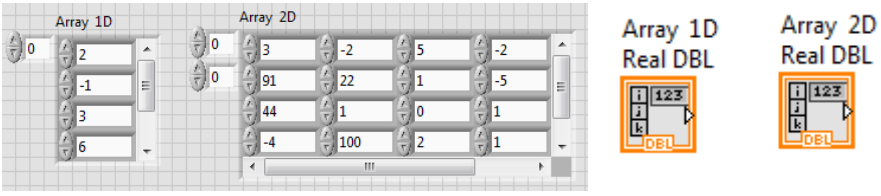


Fig. 14

8.2. În continuare (Fig. 15) observăm un tablou 1D numeric şi unul 2D numeric cu rol de indicator în PF şi terminale asociate în diagramă.

Fig. 15

8.3. Un tablou 1D şi altul 2D cu toate elementele de tip logic şi terminalele asociate în diagrama bloc (DB) se pot observa în continuare în Figura 16. Pentru controlul Array 2D, elementul stânga – sus (True) este plasat pe linia indice 0 şi coloana indice 0, aşa cum indică cei doi indici (de tip control) din stânga.

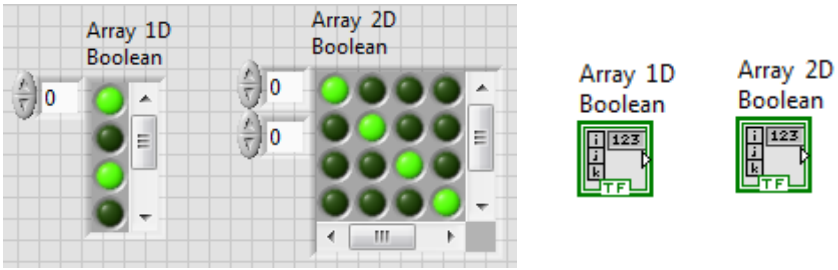


Fig. 16

9. Feedback Node versus registrul Shift

9.1. Pentru a transfera o valoare de la o iteraţie la următoarea (în cazul ciclurilor For sau While) se poate folosi structura de programare Feedback Node din paleta Structures în schimbul registrului de transfer. În Figura 17 observăm folosirea acestei variante pentru calculul sumei elementelor dintr-un şir numeric furnizat printr-un tablou 1D în panoul frontal (PF). Este disponibilă comanda directă *Replace with Feedback Node* aplicată registrului Shift sau vice versa comanda *Replace with Shift Register* aplicată imaginii grafice asociate pentru Feedback Node. Observăm şi aici iniţializarea cu zero a nodului. Dacă aceasta se omite se va păstra valoarea finală a nodului de la o rulare la alta, similar exemplului din Figura 5.

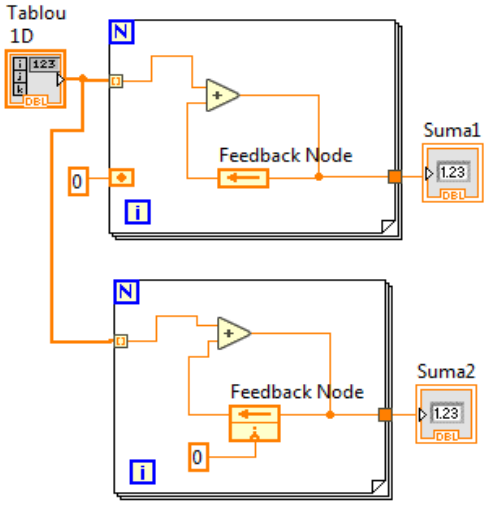


Fig. 17

9.2. Suma elementelor din matrice folosind Feedback Node

În prima diagramă din Figura 18 este calculată suma tuturor elementelor din tabloul 2D fiind folosit Feedback Node pentru ciclul For interior. În a doua diagramă (Fig. 18) este folosită structura Feedback Node pentru ambele cicluri For în scopul transferului de valoare numerică (suma parţială) de la o iteraţie la următoarea.

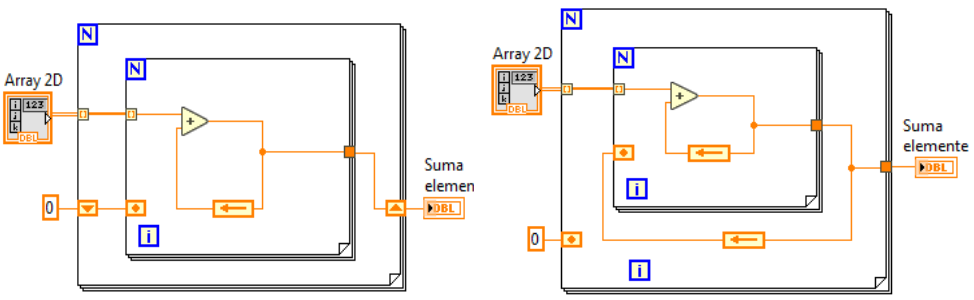


Fig. 18

10. Probleme propuse

10.1. Calculaţi media aritmetică a elementelor dintr-un şir numeric (tablou 1D control în PF) cu relaţia: $(x_1 + x_2 ... + x_n)/n$

10.2. Calculaţi media aritmetică a pătratelor elementelor din şirul numeric (tabloul 1D este control în PF) programând relaţia: $(x_1^2 + x_2^2 ... + x_n^2)/n$

10.3. Calculaţi media aritmetică a elementelor pozitive dintr-un şir numeric. Pentru numărarea elementelor pozitive se poate adăuga al doilea registru shift sau de transfer.

10.4. Calculaţi media aritmetică a elementelor pozitive de pe diagonala principală (DP) a unei matrice pătratice dată printr-un tablou 2D (control în PF).

10.5. Să se genereze un tablou de 5 linii şi 100 coloane de numere aleatoare folosind două cicluri imbricate unul For şi altul While. După generare se va calcula media valorilor pe fiecare linie şi se afişează tabloul mediilor la final. Pentru generarea unui număr aleator se poate folosi funcţia din Figura 19. Distribuţia valorilor la apelul repetat al funcţiei este uniformă, cu valori între 0 şi 1.

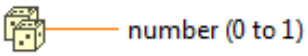


Fig. 19

10.6. Să se apeleze funcţia predefinită Sine pattern.vi. Această funcţie returnează tabloul de valori reale Y. În paralel să se programeze relaţia de calcul asociată $y_i = a \cdot \sin(x_i)$, unde $x_i = \frac{2\pi k}{n} i + \frac{\pi}{180}\varphi_0$ (i=0,1,2,...,n-1) observând generarea aceluiaşi semnal (acelaşi grafic). Intrările în funcţie sunt: samples = n, amplitude = a, phase (în grade) = φ_0 şi cycles = k (Fig. 20).

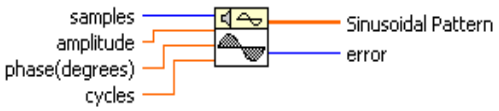


Fig. 20

III. Structura de selecţie multiplă Case şi variabila locala

1. Structura de selecţie multiplă Case

Instrucţiunea Case permite execuţia unui singur caz din mai multe disponibile. Se vor realiza aplicaţii evidenţiind câteva modalităţi de selecţie a cazurilor structurii Case.

1.1. Selector de tip Boolean: sunt disponibile două cazuri: True şi False. Cazurile sunt selectate prin valori logice; putem să nu avem nici un caz implicit (default). Aplicaţia din Figura 1 afişează valoarea maximă dintre două controale numerice. În cazul False (Fig. 1) valoarea controlului B este conectată la tunelul de ieşire din Case.

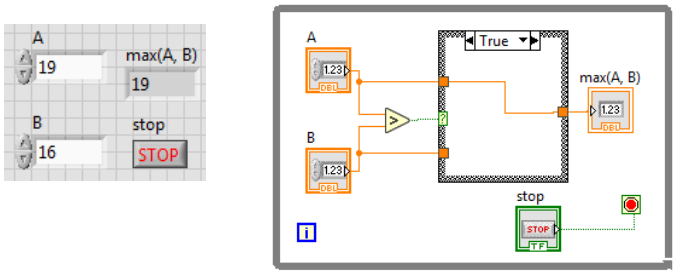


Fig. 1

1.2. Selector de tip întreg

În varianta instrucţiunii Case cu selector de tip întreg sunt permise două sau mai multe cazuri. Cazurile se identifică prin numere întregi pozitive şi negative şi/sau intervale. În exemplul propus (Fig. 2) sunt disponibile următoarele cazuri prin care se aplică prelucrări cu valorile celor două controale a şi b:

Suma (a+b) pentru Selector = 0;
Diferenţă (a-b) pentru Selector = -2;
Produs (a*b) pentru Selector =1..5, 10;
Incrementare (b+1) pentru Selector = ..-4.

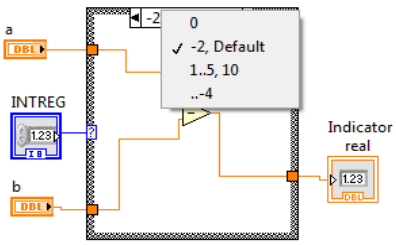


Fig. 2

La o execuţie a instrucţiunii Case se execută un singur caz din cele patru existente, în funcţie de valoarea întreagă primită de selector. Trebuie să fie un caz implicit (Default) - selectat (care se execută) la valori ale selectorului diferite

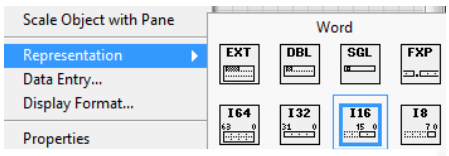


Fig. 3

de cele din listă (de exemplu valoarea -3).

Pentru a avea un control de tip întreg în PF, se poate introduce iniţial un control real şi schimba tipul de dată asociat controlului din *Representation* (click dreapta pe controlul real, Fig. 3) în I16 (întreg pe 16 biţi), I8 etc.

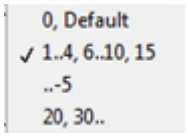


Fig. 4

Pentru exemplificare se prezintă în Figura 4 o altă variantă de selecţie a cazurilor structurii Case.

1.3. Selector de tip Enum

Prin selectorul Enum este selectat un caz din două sau mai multe cazuri disponibile. Se consideră cele patru cazuri din exemplul precedent. Cazurile vor fi identificate prin şirurile de caractere asociate controlului Enum (Fig. 5): “adun”, “scad”, ”mult”, ”+1”. Valorile care ajung la selector sunt întregi (fir albastru), dar acestor valori întregi (0, 1, 2, ...) le corespund şiruri de caractere, conform cu tipul de dată enumerare (prezent şi în limbajul C).

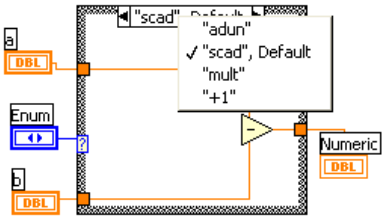


Fig. 5

Controlul de tip Enum se găseşte în paleta Ring&Enum (Fig. 6). Pentru construcţia diagramei din Figura 5, urmăm doi paşi:

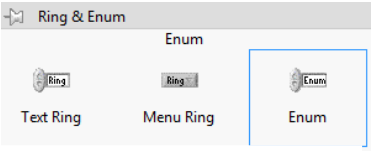


Fig. 6

pas#1: se adaugă articolele (item) *Add Item After* dorite în controlul Enum,
pas#2: se comandă *Add Case for Every Value* din structura Case (Fig. 7).

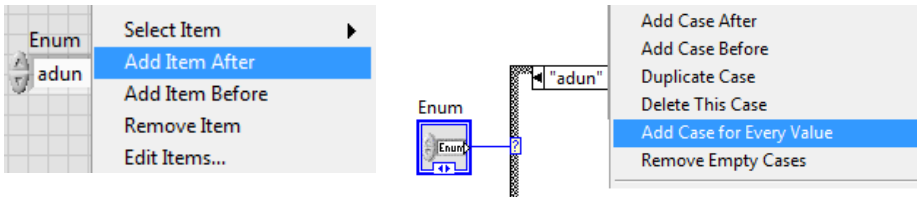


Fig. 7

1.4. Executaţi cazurile de mai sus în mod ciclic

Se va include fiecare exemplu într-o instrucţiune de ciclare While Loop. Oprirea ciclării se va realiza printr-un control de tip buton Boolean STOP (apasat=True) folosind terminalul condiţional *Continue if True* sau *Stop if True*.

2. Suma elementelor pozitive dintr-un şir sau matrice folosind Case

2.1. Calculaţi suma elementelor pozitive dintr-un şir numeric 1D folosind structura de programare Case (Fig. 8) în locul operatorului Select.

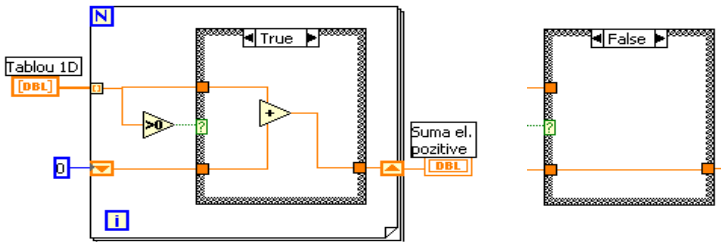


Fig. 8

2.2. Folosiţi aplicaţia de calcul a sumei elementelor pozitive din şir şi extindeţi calculul la o matrice 2D, introducând un ciclu For suplimentar.

3. Suma elementelor de pe anumite linii din matrice

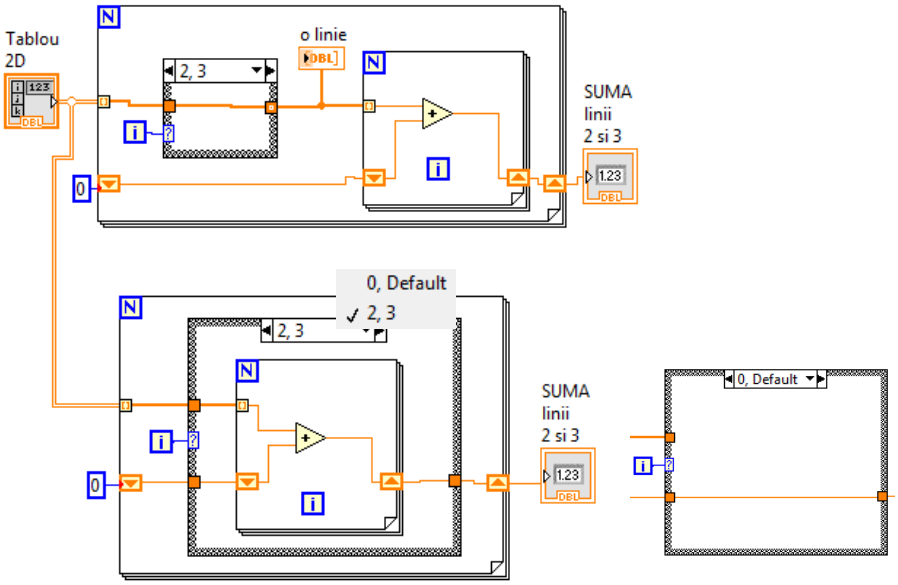


Fig. 9

În Figura 9 sunt două variante de rezolvare. Pentru ambele variante ciclul For exterior se repetă de atâtea ori câte linii sunt în matrice. Contorul *i* este selector pentru instrucţiunea Case.

În prima variantă, liniile 2 şi 3 sunt trecute neschimbate prin Case spre ciclul For. Liniile diferite de 2 şi 3 se transferă de 0 elemente, astfel ciclul For nu se repetă, dar valoarea din registru se transferă (sertar stânga → sertar dreapta).

În a doua variantă pentru liniile indice 2 şi 3 este selectat cazul în care se observă un ciclu For interior (Fig. 9 jos); acesta asigură însumarea elementelor din

liniile indice 2 şi 3 folosind registrul de transfer ataşat. Pentru celelalte linii se intră în cazul *0, Default* în care suma parţială din registrul shift trece neschimbată.

În structura Case, pot fi cazuri care nu prezintă fir de date conectat la tunelul de ieşire din Case. Valoarea implicită transferată pentru aceste cazuri este valoarea *implicită a tipului de dată* (Fig. 10), astfel: pentru numeric este *0;* pentru şir de caractere este *şirul vid*; pentru boolean este *False*.

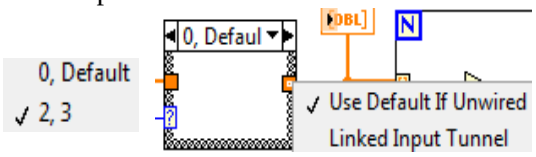


Fig. 10

4. Selectarea elementelor de pe poziţiile dorite din şirul real 1D

În Figura 11 este calculată suma elementelor reale situate pe poziţiile 2,3,4 şi 6 în varianta cu instrucţiunea Case şi în paralel în varianta cu Select.

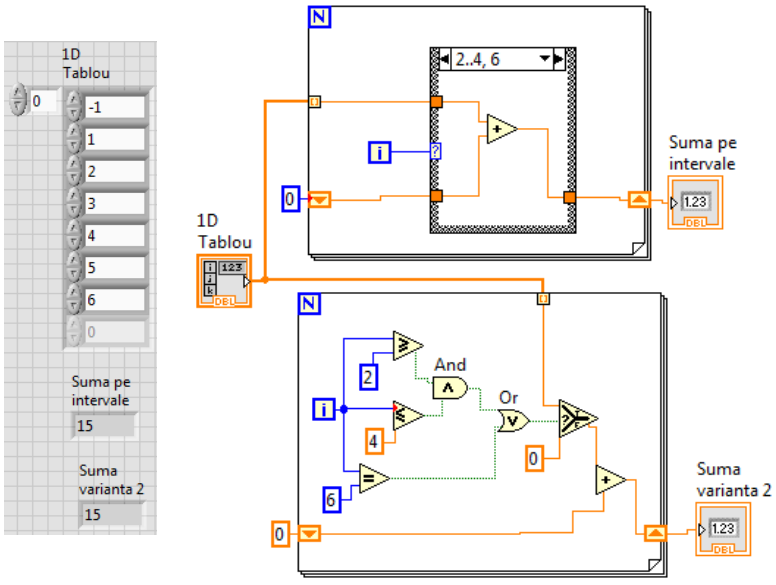


Fig. 11

5. Elementul maxim din matricea 2D şi poziţia acestuia

Se iniţializează registrul *shift* cu valoarea primului element din matrice, indice (0,0) (Fig. 12). Pentru aceasta utilizăm Index Array din paleta Array.

Este parcursă întreaga matrice prin două cicluri For imbricate. La fiecare repetiţie a ciclului For interior, elementul curent se compară cu valoarea maximă memorată în registru:

a) dacă elementul curent este mai mare sau egal, se execută cazul True => se introduce valoarea elementului curent în registrul Shift, iar indicii asociaţi elementului sunt afişaţi în PF,

b) altfel, se selectează cazul False => valoarea maximă din registru rămâne aceeaşi; valorile Col max, Lin max în PF se păstrează.

La finalul parcurgerii elementelor matricei, indicatorul *Max pe parcurs* şi registrul shift exterior vor conţine valoarea maximă (ultima întâlnită în tablou dacă sunt mai multe), iar indicatoarele *Col max* şi *Lin max* indică linia şi respectiv coloana elementului maxim.

Indicatorul *Maxim din array 2D* este setat pe iluminare intermitentă (blink) pentru 5 secunde prin succesiunea de comenzi: Create/Property Node/Blinking; Change to Write; setare pe True prin conectarea constantei True.

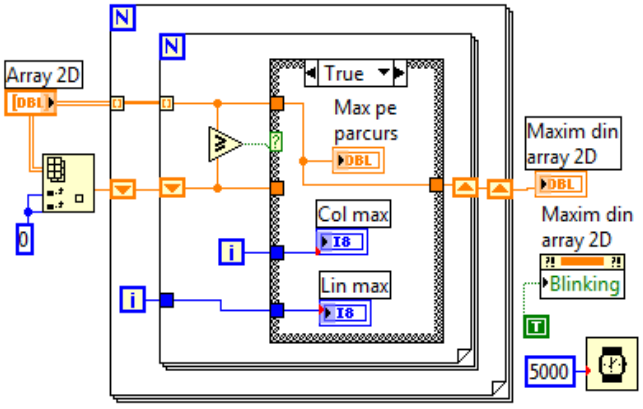


Fig. 12

6. Probleme propuse I

6.1. Pentru matrice pătratică şi diagrama din Figura 12, reprezentaţi grafic linia şi coloana care conţine valoarea maximă.

6.2. Adăugaţi doi registri pentru salvarea liniei şi coloanei elementului maxim şi afişarea la ieşirea din cicluri pentru a avea acces la aceste valori în diagramă din exteriorul ciclurilor For,

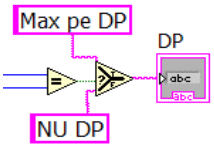


Fig. 13

6.3. Dacă elementul maxim este pe DP afişaţi mesajul ‘Max pe DP’ altfel ‘NU DP’; în paralel, dacă elementul maxim este pe DS afişaţi ‘Max pe DS’ altfel ‘Nu DS’. Se va folosi operatorul Select, iar tipul de dată selectat este şir de caractere (Fig. 13).

6.4. Reprezentaţi grafic linia şi coloana care conţine valoarea maximă (folosiţi Index Array) după ieşirea din cele două cicluri.

6.5. Calculaţi media valorilor pozitive de pe liniile indice 2, 3 şi 5 dintr-o matrice 2D oarecare, folosind Select.

7. Variabila locală - însumarea elementelor din şir/matrice

Se creează un indicator numeric (cu eticheta *Suma2*) care va memora succesiv sumele parţiale ale elementelor şi suma finală (Fig.14). Asociat indicatorului numeric sunt create două variabile locale numite *Suma2* – ambele fac referire la indicatorul numeric (click dreapta mouse pe terminal indicatorul *Suma2*/ Create/Local Variable). Cele două terminale ale variabilei locale sunt plasate astfel: una în ciclul For şi a doua în exteriorul ciclului.

a) variabila locală plasată în corpul ciclului For citeşte valoarea indicatorului şi o trimite la operatorul plus (cadrul *bold*=generează valoare în diagramă).

b) variabila locală din exteriorul ciclului primeşte valoarea 0 de la constantă şi de asemenea atribuie valoarea de iniţializare zero, indicatorului (etichetat *suma2*).

Se observă comenzile Change To Read şi Change To Write în Figura 15, cu acţiune asupra terminalelor variabilei locale.

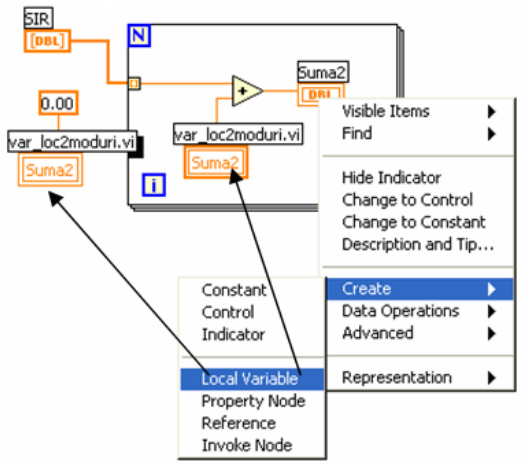


Fig. 14

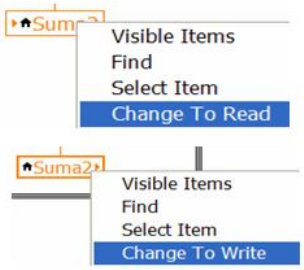


Fig. 15

8. Suma elementelor din matrice - varianta cu variabilă locală

Indicatorul *sume* 2D indică în panoul frontal toate sumele parţiale în timpul procesului de însumare. Variabila *Suma* indică numai suma finală a elementelor tabloului. La intrarea tabloului Tablou 2D în cele două cicluri For observăm indexare pentru ambele tuneluri de intrare. La tunelurile de ieşire din cicluri se asigură transferul ultimei valori, aceasta fiind suma finală (Fig.16).

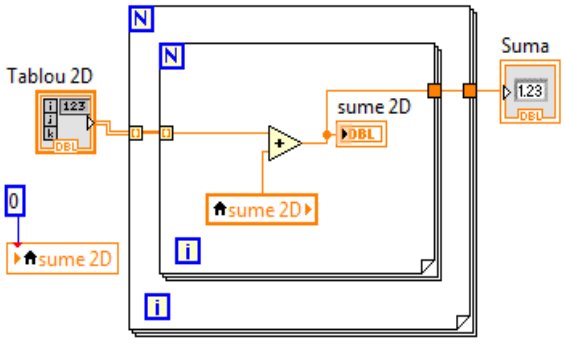


Fig. 16

9. Suma valorilor plasate pe diagonala principală - varianta cu variabilă locală

Calculul sumei elementelor de pe DP prin folosirea variabilei locale asociate indicatorului *x+y* este în diagrama din Figura 17. Observăm controlul Real Matrix (conţine valori reale în dublă precizie) în locul tipului Array 2D de valori reale. Array este tablou de valori de acelaşi tip: reale, logice, şir de caractere, tablou de structuri etc., deci este mai general.

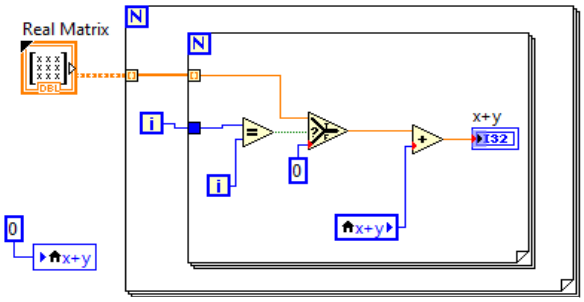


Fig. 17

10. Două cicluri While oprite cu un buton de Stop

În diagrama din Figura 19 sunt două cicluri While care se execută în paralel. Primul generează şi afişează prin Waveform Graph câte un tablou 1D de valori numerice aleatoare. Al doilea ciclu vizualizează imediat fiecare valoare aleatoare generată în Waveform Chart (Fig.18). Un buton de Stop asigură oprirea primului ciclu. Printr-o variabilă locală asociată butonului de Stop se va comanda oprirea şi a celui de al doilea ciclu (starea controlului logic se poate citi şi din corpul ciclului al doilea). Ambele cicluri While sunt oprite prin terminalul condiţional de tip *Stop If True*. Butonul apăsat generează valoarea logică True iar neapăsat generează valoarea False. Controlul Blinking poate impune iluminare intermitentă în Chart.

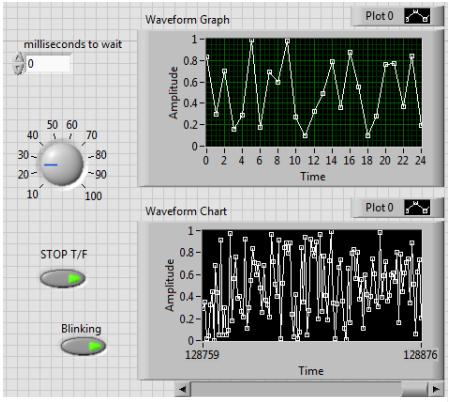


Fig. 18

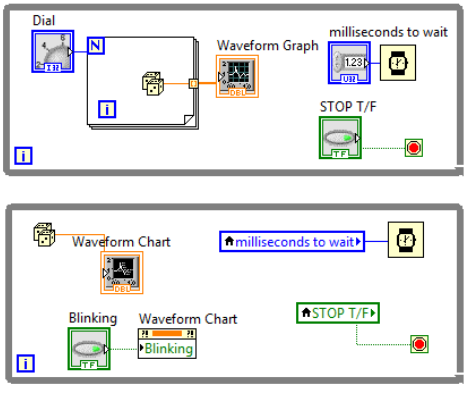


Fig. 19

11. Elementul maxim din şirul numeric - Case şi variabila locală

Max 1D este un indicator numeric. Prima valoare din tabloul Array 1D iniţializează Max 1D prin intermediul variabilei locale asociate (setată pentru scriere). În ciclul For valoarea curentă intrată în ciclu este comparată cu valoarea variabilei locale setate pentru citire (Fig. 20). La selecţia cazului True este actualizat indicatorul Max 1D şi indicatorul *poz* (poziţia maximului în şir). La selecţia cazului False cele două indicatore nu sunt modificate.

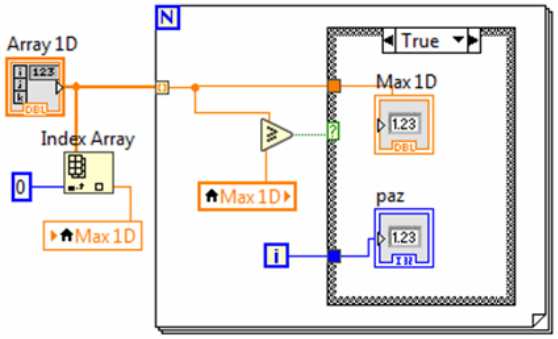


Fig. 20

12. Probleme propuse II

12.1. Calculaţi suma elementelor de pe diagonala principală a unei matrice pătratice folosind instrucţiunea Case şi variabila locală.

12.2. Calculaţi elementul maxim dintr-un şir numeric 1D şi poziţia maximului folosind instrucţiunea Case şi variabila locală.

12.3. O matrice este control în PF. Modificaţi valorile din matrice şi trasaţi interactiv graficul liniei în care se află elementul maxim din matrice.

12.4. Se generează o matrice având pe cele 4 linii câte 100 elemente generate de funcţia Uniform White Noise.vi; se afişează matricea 4x100. În continuare se determină prin program poziţia elementului maxim de pe linia a doua şi se afişează în panoul frontal. Se va programa o variantă folosind registrul shift şi în a doua variantă se va folosi variabila locală.

12.5. Se generează o matrice pătratică de valori aleatoare folosind două cicluri imbricate ambele While. Se va determina elementul maxim de pe diagonala principală şi linia pe care este elementul maxim. Se reprezintă grafic diagonala principală.

12.6. Generaţi un şir A numeric de 20 valori aleatoare cu Gaussian White Noise.vi. Generaţi tabloul B de valori logice astfel: pentru valori pozitive din A să avem True în B, iar pentru valori numerice negative sau nule să avem False în B. Generaţi tabloul P care va conţine numai numerele pozitive din A.

IV. Date de tip Cluster şi structura de control Sequence

1. Date de tip cluster sau structură

Data de tip structură (cluster) este un mănunchi sau o colecţie ordonată formată din unul sau mai multe elemente, sau câmpuri având tipuri diferite de date sau fiind de acelaşi tip. Fiecare element sau câmp din structură are asociat un număr de ordine: 0,1,2,3…. Structura/cluster în panoul frontal (PF) se găseşte în paleta Array, Matrix&Cluster în varianta indicator şi control.

1.1. Schimbarea valorii unei componente a structurii, accesul la componente

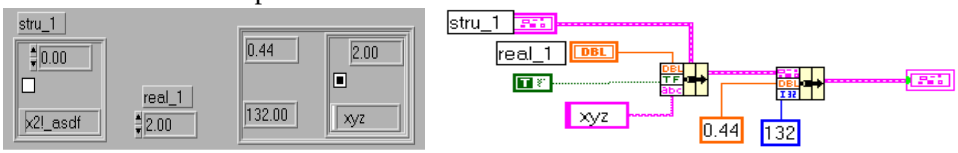


Fig. 1

Pentru modificarea valorii unui element al structurii se conectează structura cu eticheta stru_1 (Fig. 1) la coloana a doua (din mijloc) a operatorului Bundle (Fig. 2). La stânga este prima coloană a operatorului Bundle şi conţine cele trei câmpuri ale structurii: real/DBL, logic/TF şi şir caractere/abc; prin coloana din dreapta structura se conectează mai departe în diagramă, structura având eventual noi valori pentru anumite câmpuri. Calculaţi suma câmpurilor numerice din structura finală (0.44+132+2) şi afişaţi suma în PF (se va folosi operatorul *Unbundle* din Figura 2).

Fig. 2

1.2. Folosirea numelor câmpurilor structurii

Pentru a referi elementele componente ale structurii de date prin numele etichetelor în scopul schimbării unor valori folosim operatorul *Bundle By Name*. Urmăriţi aplicaţia din Figura 3. Pentru a accesa mai uşor o componentă din structură se preferă desfacerea mănunchiului de date cu operatorul *Unbundle by Name* (Fig. 4).

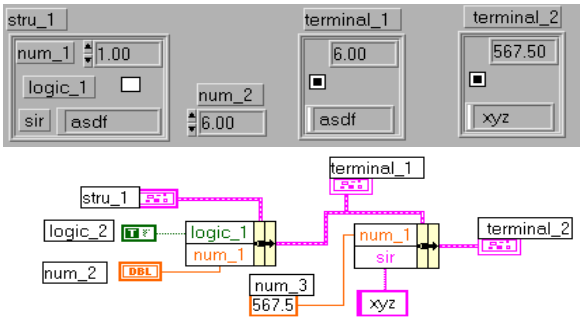


Fig. 3

1.3. Adunaţi câmpul numeric din controlul stru_1 cu cel din indicatorul terminal_2. Rezultatul se adaugă ca un nou câmp într-o structură mai generală care va conţine şi terminal_2. Se afişează ultima stuctură generată (terminal_3).

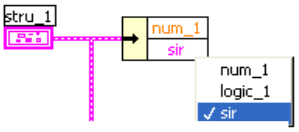


Fig. 4

2. Aplicaţie cu studenţi, discipline, note

2.1. Din tabloul de şiruri de caractere cu eticheta Tablou NUME, intră la fiecare iteraţie câte un nume de student în ciclul For (SESIUNE) (Fig. 5). Se compune o structură cu 4 câmpuri: şir caractere (nume student), Tablou Discipline, tablou note şi media notelor. În continuare se adaugă un tablou cu 5 valori logice, câte una pentru fiecare notă/disciplină. Se generează *cluster 5camp*, o structură cu 5 câmpuri. O altă structură cu eticheta *cluster 4camp+Tabl bool* se afişează şi conţine o structură cu 4 câmpuri şi tabloul cu 5 valori logice.

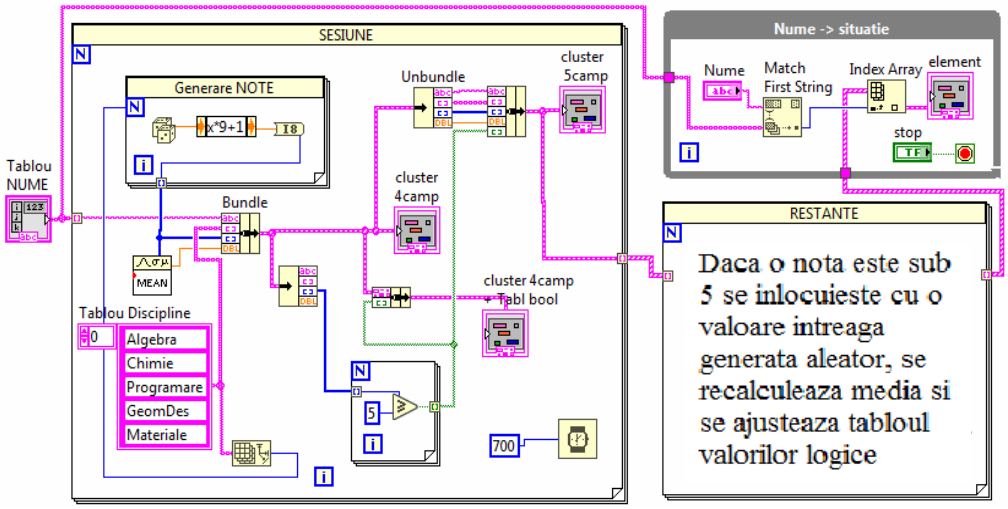


Fig. 5

Din ciclul SESIUNE iese întregul tablou de structuri. Fiecare structură conţine situaţia unui student.

2.2. Se completează ciclul RESTANTE conform cu îndemnul scris în cadrul ciclului.
În ciclul While (Nume -> situaţie) care se repetă până la apăsarea butonului de Stop, se selectează şi afişează situaţia studentului introdus în controlul şir de caractere *Nume* (Fig.6). Funcţia predefinită Match First String.vi returnează poziţia şirului (nume student) din Tablou NUME. Index Array returnează structura (situaţia studentului) selectată, din tabloul de structuri, prin indicele întreg.

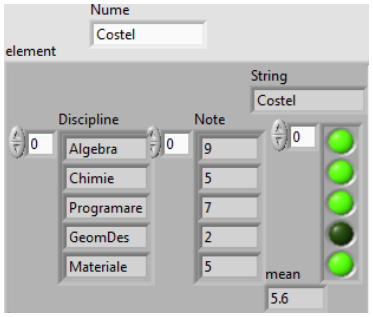


Fig. 6

2.3. Se va folosi Bundle by Name şi Unbundle by Name în loc de Bundle/ Unbundle în cadrul aplicaţiei de mai sus (studenţi, discipline, note).

3. Structura de control Sequence

Structura Sequence conţine una sau mai multe *subdiagrame* sau cadre numerotate (0, 1, 2 ...) care sunt executate secvenţial, în ordine crescătoare. Structura Sequence poate să apară în diagramă în varianta *Flat* pentru care cadrele sunt vizibile toate şi înlănţuite în ordine de la stânga la dreapta şi în varianta *Stacked* sau stivuite în care cadrele sunt suprapuse astfel încât numai codul grafic dintr-un cadru (cel curent) este vizibil, obţinându-se o economie de spaţiu ocupat în diagramă.

3.1. Măsurarea timpului consumat la execuţia unui segment de cod

Se propune funcţia *Tick Count (ms).vi* + structura Sequence cu trei cadre. În diagramă (Fig.7) se măsoară aproximativ timpul necesar unei ciclări care se execută în cadrul/secvenţa din mijloc. Funcţia Tick Count(ms).vi este apelată de două ori. La momentul primului apel, furnizează o valoare în milisecunde (Timer). Valoarea returnată la apelul din secvenţa indice 2 va fi mai mare decât valoarea generată în prima secvenţă (indice 0). Diferenţa între valorile returnate este durata în milisecunde căutată. Când *timerul* atinge valoarea maximă întreagă 2^32–1 următoarea valoare după incrementare este 0.

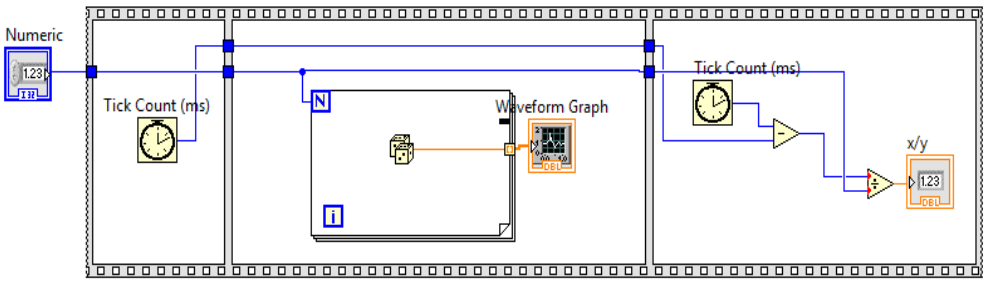


Fig. 7

În exemplul următor (Fig.8), în mod similar se verifică egalitatea dintre timpul măsurat şi timpul de aşteptare stabilit prin funcţia *Wait (ms).vi*.

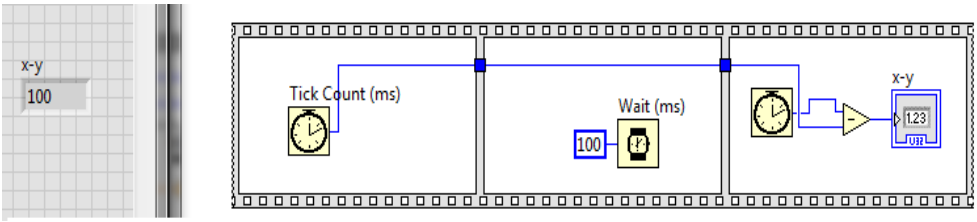


Fig. 8

Observăm valoarea 100 în PF, egală cu milisecundele impuse prin Wait.vi.

3.2. Ordonarea unui tablou 1D, afişarea şi salvarea în format text

În exemplul următor (Fig. 9) este folosită o structură secvenţială cu cadrele înlănţuite (flat). Aceasta va conţine secvenţa indice 0 apoi se adaugă (Add Frame After) o nouă secvenţă (indice 1). Aplicaţia salvează într-un fişier text, pe o linie, valorile numerice din tabloul Array 1D în prima secvenţă, iar în a doua secvenţă în acelaşi fişier este adăugat (append to file?=T) şirul ordonat crescător cu valorile aşezate pe coloană (transpose?=T). La execuţia primei secvenţe se va tasta sau selecta numele fişierului (în cadrul unei ferestre de dialog), iar numele fişierului în care se salvează va fi intrare pentru funcţia Write Delimited Spreadsheet.vi (Fig.11) în a doua secvenţă.

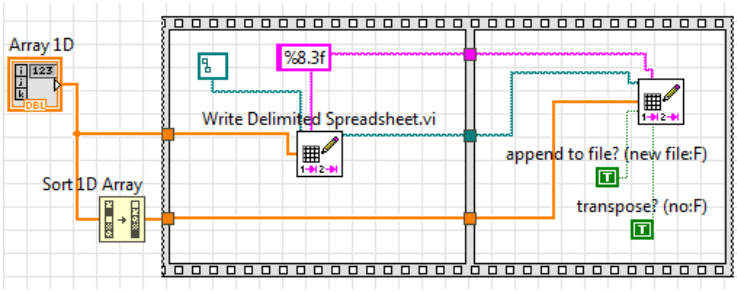


Fig. 9

Funcţia Write Delimited Spreadsheet.vi converteşte un tablou numeric 1D sau 2D având valori reale reprezentate în simplă sau dublă precizie, la tipul de dată *şir de caractere*, şi-l salvează într-un fişier text.

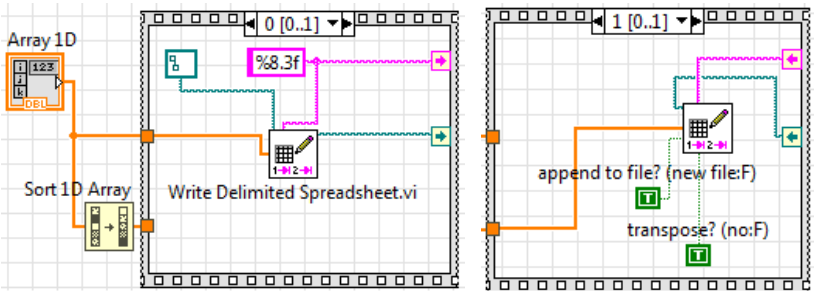


Fig. 10

În figura 10 acelaşi exemplu este refăcut în varianta de structură cu cadre stivuite (suprapuse). Această variantă se poate obţine automat cu comanda *Replace/ Replace with stacked sequence* (click dreapta pe antetul structurii).

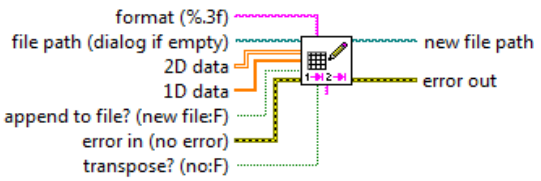


Fig. 11

Salvaţi şirul ordonat crescător şi apoi cel iniţial neordonat. Folosind operatorul *Reverse 1D array* ordonaţi descrescător şi salvaţi în fişierul text. Formatul %8.3f indică salvarea valorii numerice pe un spaţiu de 8 caractere din care trei sunt alocate pentru zecimale, unul pentru caracterul punct, iar restul pentru partea întreagă.

4. Verificarea unor relaţii matriceale

4.1. Verificaţi relaţia matriceală:

$$(A*B)^T=B^T * A^T$$

unde A, B sunt de tip matrice. A şi B pot să nu fie matrice pătratice. Pentru operaţii cu matrice sunt folosite funcţii din subpaleta Functions/ Mathematics/Linear Algebra (Fig. 12). La intrarea şi ieşirea din instrucţiunile de ciclare matricele şi tablourile se comportă în mod similar.

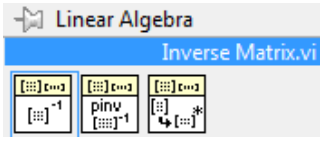


Fig. 12

4.2. Folosiţi tipul de dată tablou/array 2D în locul tipului matrice accesând funcţii din paleta Functions/Array. Observaţi testul de egalitate între tablouri, caz în care rezultă un tablou de valori logice (Fig. 14). Funcţia *Equal?* (Fig. 13) prezintă două moduri de lucru pentru tablouri: a) compară element cu element de pe poziţii similare în tablou şi b) compară întregul tip de dată sau agregatul, caz în care rezultă o singură valoare logică.

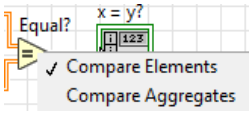


Fig. 13

4.3. Verificaţi de asemenea următoarele relaţii matriceale:

1. $(A*B)^{-1}=B^{-1} * A^{-1}$ *(produsul inversat)*
2. $(A+B)*C=A*C+B*C$ *(distributivitate înmulţire faţă de adunare)*
3. $(A^{-1})^T=(A^T)^{-1}$
4. $\det(A) * \det(A^{-1})=1$

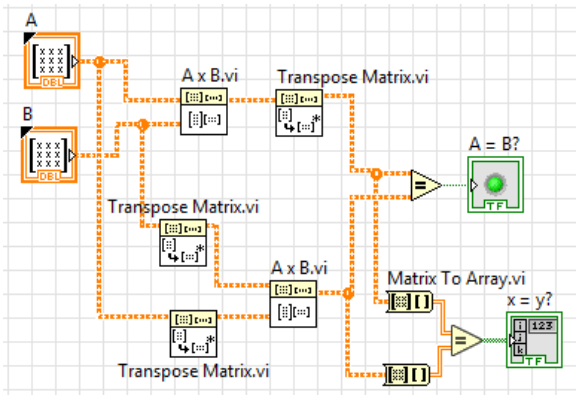


Fig. 14

V. Organizare pe pagini în Panoul Frontal cu Tab Control, funcţia Sine Pattern.vi, semnale aleatoare, histograma

1. Funcţia armonică x(t)=a·sin(t), reprezentare grafică, media, RMS

1.1. Media valorilor (mean) pe una sau mai multe perioade este notată X_{mean} ($\bar{x}$) iar rădăcină din media pătratelor valorilor pe o perioadă (r.m.s. = root mean square) este X_{rms} (1).

$$\bar{x} = \frac{1}{n}\sum_{k=1}^{n} x_k \qquad x_{rms} = \sqrt{\frac{1}{n}\sum_{k=1}^{n} x_k^2} \tag{1}$$

1.2. Să se verifice prin program relaţiile dintre amplitudinea armonicei (a), media valorilor absolute şi rădăcina din media pătratelor valorilor pentru una sau mai multe perioade (2):

$$x_{rms} = 0.7071 \cdot a \qquad x_{mean} = 0.6366 \cdot a \tag{2}$$

1.3. Se va crea o aplicaţie LabVIEW pentru generarea panoului frontal (PF) din Figura 1 (sunt folosite funcţii din subpaleta Mathematics).

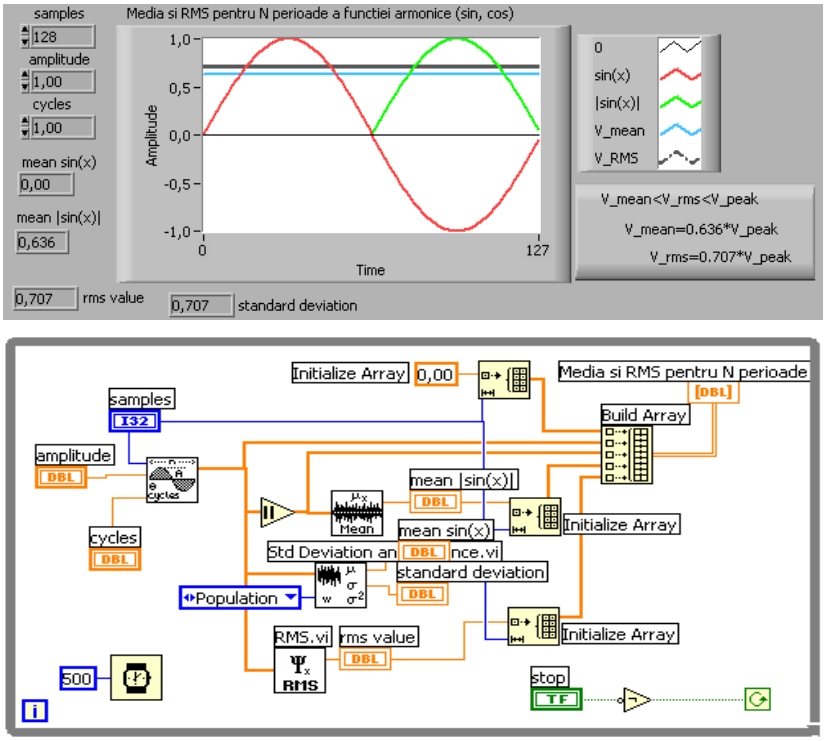


Fig. 1

Se va apela funcţia Sine Pattern.vi (Signal Processing/Signal Generation). Prin intrarea *samples* este specificat numărul de eşantioane (128 implicit) sau valori generate în tabloul de ieşire (*Sinusoidal Pattern*). Prin *cycles* se impune

numărul de cicluri sinusoidale formate în semnalul de ieşire. Pictogramele funcţiilor apelate sunt vizualizate (Fig. 2) cu intrările şi ieşirile asociate.

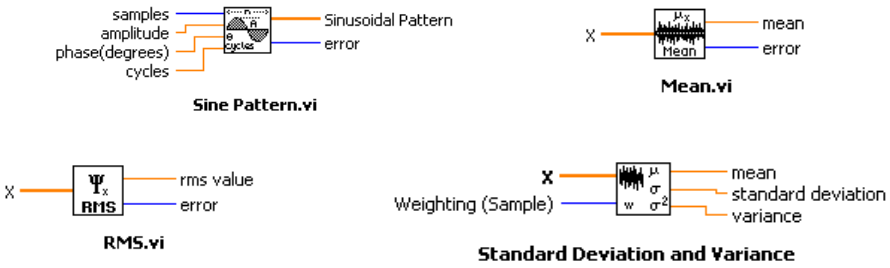


Fig. 2

Sunt generate cinci tablouri 1D, fiecare de câte 128 valori. Acestea prin funcţia Build Array vor forma un tablou 2D de 5 linii şi 128 coloane (0,...,127). Primul tablou 1D va trasa linia de zero (abscisa), al doilea tablou 1D trasează o perioadă a funcţiei sinus, urmează în ordine tabloul valorilor în modul, tabloul liniei asociată mediei modulelor de ordonată 0.6366·a şi la final tabloul care va trasa linia de ordonată 0.707·a. Tabloul 2D este vizualizat grafic în PF prin Waveform Graph. Funcţia Initialize Array generează un tablou 1D având 128 elemente, iar elementele au aceeaşi valoare (cea prescrisă). În Figura 3 se observă cazul în care s-au impus trei cicluri funcţiei Sine Pattern.vi.

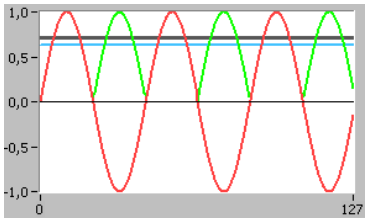


Fig. 3

2. Semnale aleatoare. Uniform white noise (Uwn), Gaussian white noise (Gwn), Periodic random noise (Prn) şi histograma

2.1. Descrierea celor trei funcţii care generează valori aleatoare

Funcţia Gaussian white noise.vi (Gwn, Fig. 4) are intrarea samples = numărul de eşantioane (valori) din semnalul generat (implicit n=128); folosiţi 1024, 2048 etc. Pentru a comanda *împrăştierea* valorilor în jurul mediei se prescrie o valoare pentru intrarea *standard deviation* (σ); aceasta este calculată cu relaţia (3).

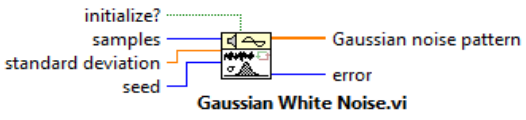


Fig. 4

$$\sigma = \sqrt{\frac{1}{n}\sum_{k=1}^{n}(x_k - \bar{x})^2} \qquad (3)$$

Gwn generează tabloul de n valori (Gaussian noise pattern). Pentru semnalul Gwn, densitatea de probabilitate a amplitudinii semnalului (domeniul timp)

urmăreşte curba lui Gauss (Fig. 5, sus), media aritmetică a valorilor semnalului este zero şi deviaţia standard este notată cu sigma (σ).

Valorile semnalului Uniform white noise.vi (Uwn) prezintă aceeaşi probabilitate de apariţie iar media aritmetică a valorilor semnalului este zero (Fig. 5, jos).

Semnalul Periodic random noise.vi (Prn) este o sumă de sinusoide, fiecare sinusoidă având un număr întreg de cicluri; ciclurile sunt de aceeaşi amplitudine, dar faze aleatoare. Media valorilor semnalului Prn este zero.

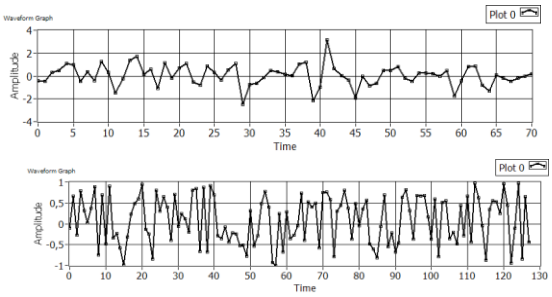


Fig. 5

2.2. Aplicaţie cu histogramă

Histograma h(x) indică câte valori din şirul real de intrare (X) sunt în fiecare din cele *m* intervale. Intervalele au lăţime egală:

$$delta\_x = (\max(X) - \min(X))/m \quad (4)$$

Centrele celor m intervale sunt calculate cu relaţiile:

$$center(i) = min(X) + delta\_x/2 + i * delta\_x, \; i=0,1,2\ldots m-1. \quad (5)$$

Se consideră aplicaţia din Figura 6. *Histogram Graph* reprezintă histograma în grafic de bare, fiind o structură de 2 tablouri 1D. *h(x)* este tabloul ordonatelor barelor sau numărul de valori găsite în fiecare interval, iar *X Values* este tabloul centrelor celor şapte intervale.

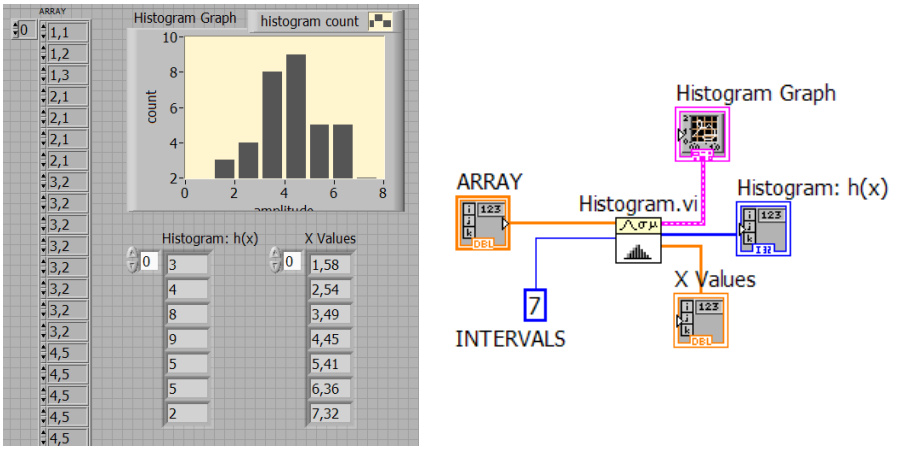


Fig. 6

Sugestiv, în Figura 7 sunt prezentate cele şapte intervale între min(X) şi max(X) şi valori în lungul axei Ox reprezentate prin puncte în cele şapte intervale. Sub abscisă sunt totalizate valorile din fiecare interval.

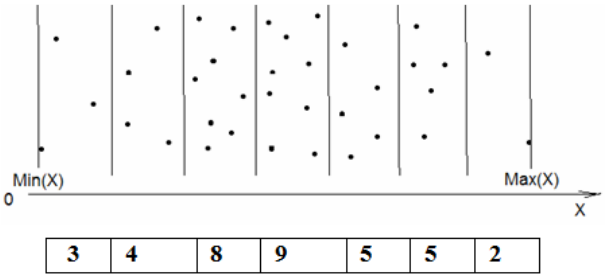


Fig. 7

3. Organizarea pe pagini în Panoul Frontal cu Tab Control

3.1. În Panoul Frontal (PF) se plasează un Tab Control (paleta Containers) cu trei pagini. Prima pagină a aplicaţiei va conţine o scurtă descriere a funcţionării aplicaţiei. A doua pagină, numită SINUS, conţine Panoul Frontal al aplicaţiei din Figura 1, iar în diagrama asociată paginii se găseşte codul deja prezentat, având un ciclu While care se poate opri din pagina SINUS cu un buton de Stop numit Stop SINUS. A treia pagină, numită HISTOGRAME, vizualizează histograme pentru semnalele Gwn, Uwn şi Prn în mod repetitiv (realizat cu al doilea ciclu While care se execută independent de primul), conform cu Panoul Frontal din Figura 8 şi diagrama asociată din Figura 9.

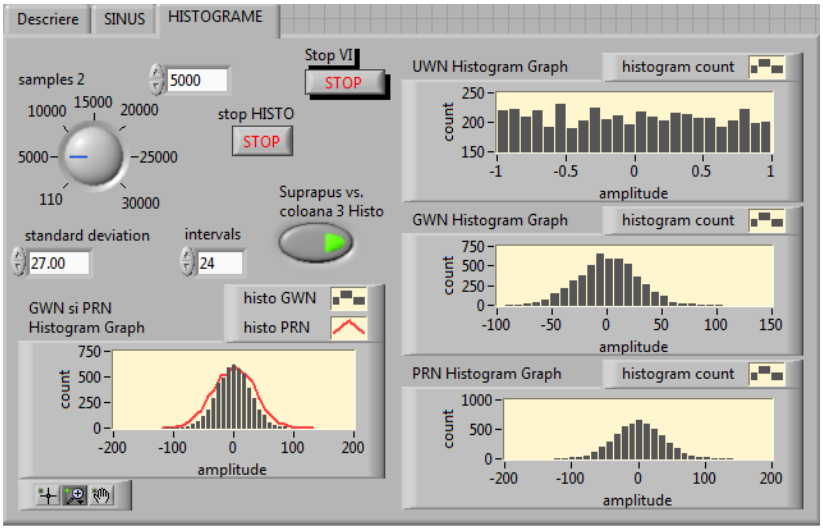


Fig. 8

Selectorul Boolean *Suprapus vs. coloana 3 Histo* activează afişarea coloanei celor trei histograme din dreapta PF sau fereastra grafică *GWN si PRN Histogram Graph* comandând unul dintre cazurile structurii Case din ciclul While (Fig. 9 şi

Fig.10). Suprapunerea celor două histograme din fereastra *GWN si PRN Histogram Graph* se poate realiza prin modificarea (reglarea) deviaţiei standard a semnalului Gwn generat la fiecare iteraţie.

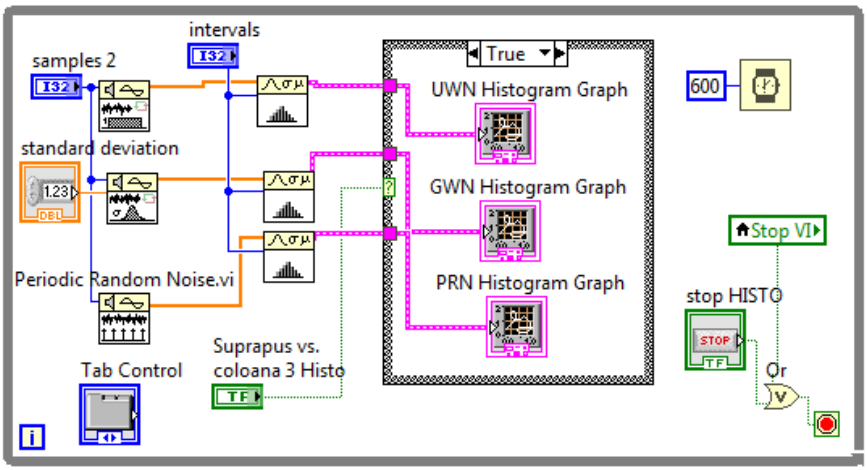


Fig. 9

3.2. Stop HISTO opreşte numai ciclul asociat paginii HISTOGRAME. Stop SINUS opreşte numai ciclul While asociat paginii SINUS. Butonul Stop VI este vizibil în toate cele trei pagini şi opreşte la apăsare *prelungită* întreaga aplicaţie, deci ambele cicluri While din diagramă. Butonul Stop VI este setat pe Properties/Operation/*Switch until released* (Fig. 11); acesta este creat în PF mai întâi în exteriorul suprafeţei Tab Control (TC), apoi este selectat şi mutat (folosind tastele săgeţi) pe suprafaţa TC.

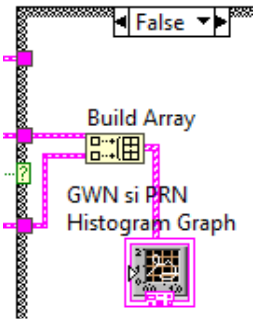


Fig. 10

3.3. Pentru un anumit număr de eşantioane, modificaţi standard deviation (Gwn) până devin apropiate ca formă histogramele din fereastra *GWN si PRN Histogram Graph*, în care sunt suprapuse cele două histograme. Suprapunerea se realizează prin execuţia cazului False al structurii Case, caz prezentat în Figura 10.

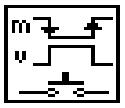


Fig.11

În graficul histogramelor semnalelor Gwn şi Prn am observat profilul de tip clopot (Gauss), iar pentru semnalul Uwn se obţine palier orizontal.

4. Probleme propuse

4.1. Să se verifice egalitatea dintre standard deviation (intrare) în Gaussian white noise.vi şi deviaţia standard (σ **)** a valorilor numerice ce compun semnalul generat de Gaussian white noise.vi

4.2. Se reprezintă grafic histograma şi puterea spectrală a unui semnal Gaussian white noise de 400000 eşantioane. Deviaţia standard (σ) va fi control în Panoul Frontal. Se va calcula folosind ciclări, procentual câte eşantioane sunt în fiecare din intervalele (0 σ), (–σ 0), (σ 2σ), (2σ 3σ), (-σ σ), (-3σ 3σ). (R:34% în primul interval).

4.3. Se generează şi reprezintă grafic un semnal de tip Gaussian white noise. Se va calcula histograma semnalului cu funcţia histogram.vi şi în paralel cu programul propriu care împarte valorile semnalului în *m* intervale după relaţiile (4) şi (5) de mai sus.

4.4. Funcţia Gaussian white noise.vi generează un semnal aleator cu medie zero; deviaţia standard va fi control. Se verifică repetitiv (folosind pentru calcule instrucţiuni de ciclare) corectitudinea relaţiei de mai jos, pentru diverse valori ale deviaţiei standard (Fig. 12).

$$RMS^2 = deviaţia\_standard^2 + media^2$$

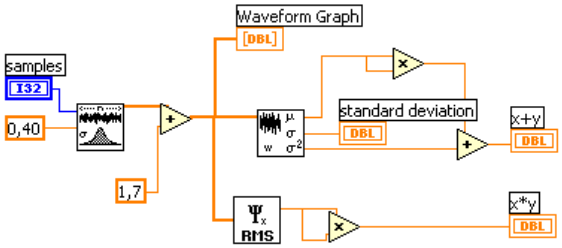


Fig. 12

4.5. Generaţi un tablou de 128 valori cu funcţia Gaussian white noise.vi. Identificaţi valoarea maximă şi poziţia ei în semnal. Reprezentaţi grafic semnalul şi marcaţi cu un pătrăţel valoarea maximă din semnal.

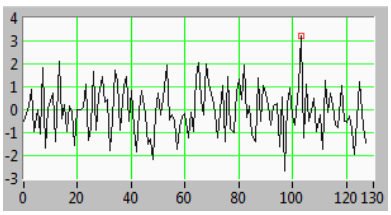


Fig. 13

VI. Prelucrare tablouri numerice, medii, stive

1. Şir de medii aritmetice

1.1. Fie un şir de valori: x1, x2, x3, …, xn, reprezentând citirile succesive de valori de la un senzor, pe un canal de achiziţie. Se va calcula media tuturor eşantioanelor pe măsură ce acestea sunt citite, astfel:

M2 este media primelor 2 valori,
Mi-1 este media primelor i-1 valori,
Mi este media primelor i valori,…,
Mn este media tuturor celor n valori.

Vom calcula Mi folosind Mi-1 şi valoarea curentă xi (1):

$$M_i = [(i-1)*M_{i-1} + x_i\ ] / i \qquad (1)$$

(M0=0)
M1= (0*M0+**x1**) / 1 = x1, iteraţia1
M2= (1*M1+**x2**)/2= (x1+ x2)/2 iteraţia2
M3= (2*M2+**x3**)/3= (x1 + x2 + x3)/3 iteraţia3
……

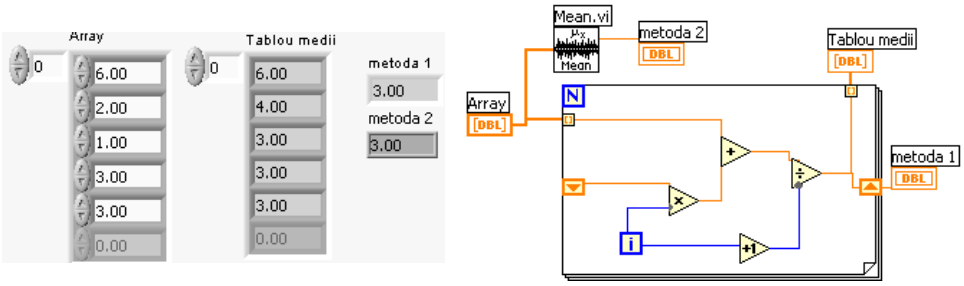


Fig.1

În Figura 1 registrul shift transferă media curentă la iteraţia următoare. Instrumentul *Mean.vi* calculează media aritmetică a celor n valori din tabloul de intrare X, folosind relaţia (2). Media la fiecare nou element indexat (ciclu) este calculată din media de la iteraţia precedentă păstrată în registrul Shift care se înmulţeşte cu i, rezultând suma primelor i elemente; se adună apoi termenul curent şi se împarte cu (i+1).

$$\text{media} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad (2)$$

1.2. Medieri pe mai multe coloane - separat pe fiecare coloană

Se consideră citirea valorilor de la trei senzori prin baleerea repetată a celor 3 canale de achiziţie. În locul senzorilor (Fig. 2) s-a introdus un ciclu For. Acesta la fiecare iteraţie a ciclului While exterior generează trei valori aleatoare în intervalul (0, 1). *Tablou medii* (Fig. 3) conţine pe fiecare coloană mediile pentru un senzor.

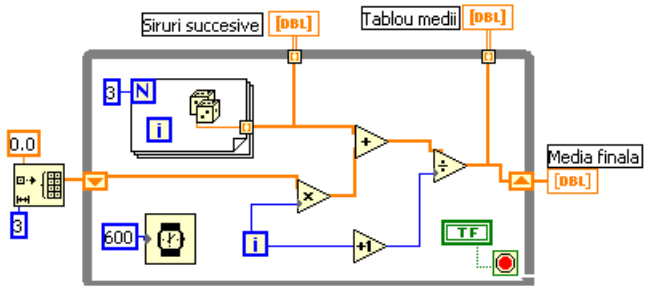


Fig. 2

Fig. 3

1.3. Să se modifice programele pentru calculul mediei pătratelor şi rădăcinii din media pătratelor pe fiecare din cele 3 coloane.

2. Funcţia Mean PtByPt.vi pentru calculul mediei

Se reia diagrama precedentă la care se adaugă (Fig. 4) varianta de calcul care apelează funcţia Mean PtByPt.vi din paleta Signal Processing/Point By Point.

Funcţiile din paleta Point By Point sunt scrise pentru analiza continuă a datelor pe măsură ce valorile sunt disponibile, punct cu punct (valoare cu valoare), spre deosebire de datele memorate în tablouri sau blocuri de date şi procesate astfel în grup. Funcţia (Fig. 5) calculează media aritmetică a valorilor din setul de date de intrare de

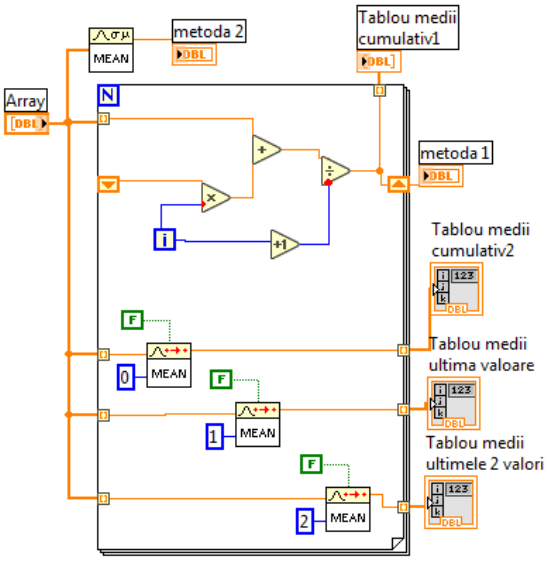


Fig. 4

lungime *sample length* (implicit 100). Dacă *sample length* este 0, funcţia calculează media pe măsură ce valoarea curentă este disponibilă (cumulativ). Dacă *sample length* este mai mare decât zero, funcţia calculează media ultimelor n valori apărute.

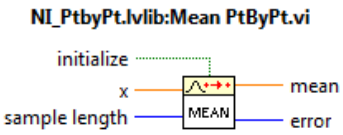


Fig. 5

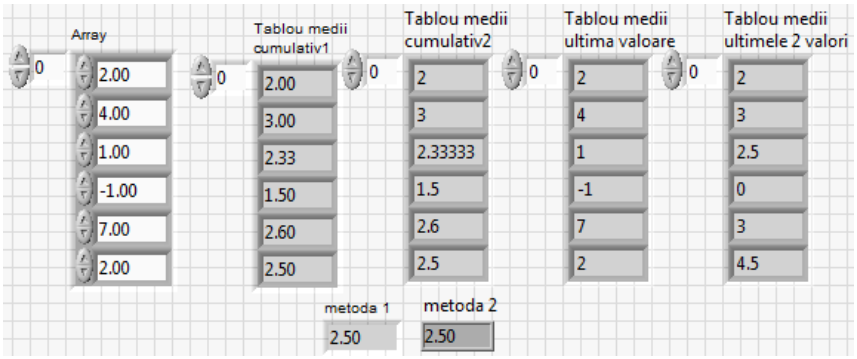


Fig. 6

În Panoul Frontal din Figura 6 se observă identitatea valorilor calculate prin cele două abordări în tablourile Tablou medii cumulativ.

3. Se caută o valoare într-un tablou numeric

Şirul numeric *a* de valori întregi este dat în panoul frontal. În cazul în care elementul căutat este găsit se returnează poziţia elementului în şir. În caz contrar se afişează mesajul "Nu s-a găsit elementul".

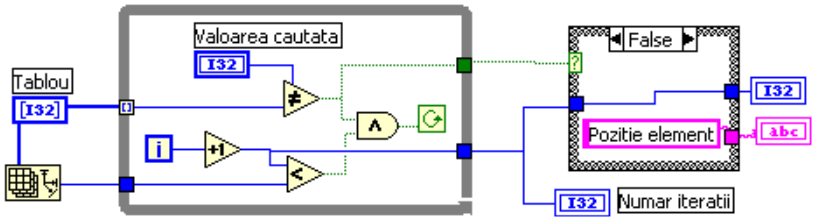


Fig. 7

Folosim un ciclu While pentru indexarea sau preluarea elementelor pe rând din tablou. În corpul ciclului sunt două comparaţii a căror rezultate logice sunt conectate la un AND logic care în continuare este conectat la terminalul ciclului While de tipul Continue if True.

Elementul curent este comparat cu valoarea căutată. Dacă diferă de aceasta şi mai sunt elemente în şir de comparat se trimite valoarea logică True la terminalul ciclului, continuându-se cu iteraţia următoare. Dacă se găseşte valoarea căutată, chiar dacă nu se parcurge întregul şir, se întrerupe ciclarea. Prin tunelul de ieşire

indicele elementului (+1) părăseşte ciclul While şi se afişează în panoul frontal. Valoarea False provenită din evaluarea expresiei logice *Valoarea cautata ≠ a(i)* comandă execuţia cazului False a unei structuri Case pentru afişarea şirului de caractere 'Poziţie element' (Fig. 8).

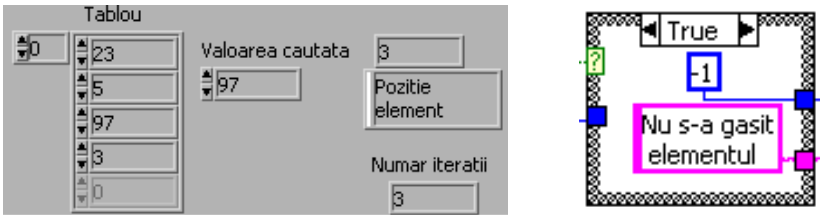


Fig. 8

4. Operaţii cu stivă - funcţii pentru prelucrarea tablourilor

Se implementează o stivă de date numerice întregi. Stiva suportă trei operaţii: adăugare element în stivă (registrul 1D sus), extragere element din stivă (ultimul adăugat) şi iniţializare stivă (ştergerea tabloului din memorie = eliberarea memoriei).

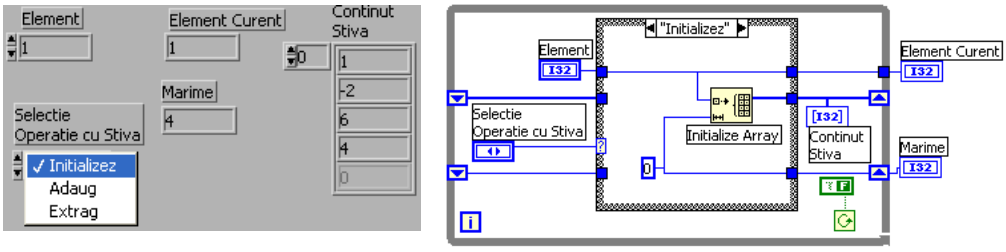


Fig. 9

În panoul frontal (Fig. 9) observăm conţinutul stivei după patru operaţii de adăugare, ultimul element adăugat are valoarea *1*. Un registru (tablou 1D) memorează stiva. Aceasta este actualizată în PF la ieşirea din Case. Registrul de jos conţine numărul de elemente din stivă. Fiecare caz are trei intrări şi trei ieşiri. În continuare sunt descrise succint cele trei cazuri.

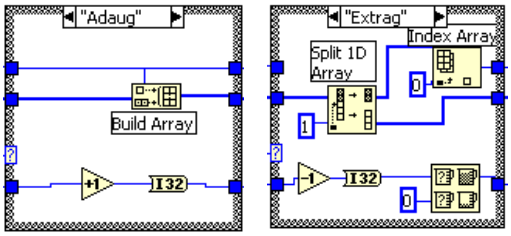


Fig. 10

Case "Initializez": generează (Initialize Array) tablou 1D vid care se va copia în registrul de sus; acesta memorează stiva (Fig. 9). Registrul de jos este iniţializat cu zero.

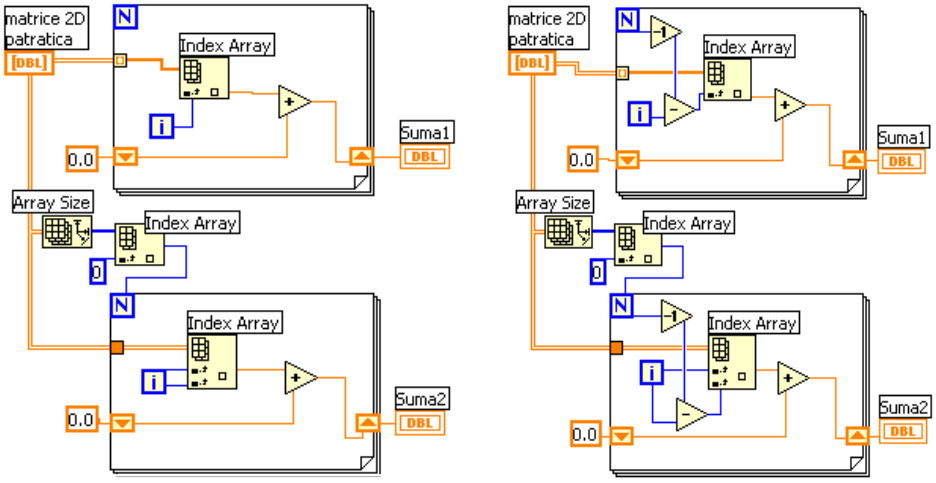
Case "Adaug": Build Arrary (în varianta concatenare) adaugă un element la stivă şi incrementează contorul memorat în registrul de jos (Fig. 10).

Case "Extrag": din tabloul 1D stivă se separă un subtablou de un element din care cu Index Array se extrage unicul element şi se trimite la indicatorul Element Curent. Al doilea subtablou (din Split) sau stivă se copiază în registrul de stivă. Numărul de elemente din stivă decrementează, dar nu scade sub zero.

5. Variante pentru calculul sumei elementelor de pe DP şi DS în matrice

Se accesează direct elementele diagonalei matricei, fără parcurgerea întregii matrice. Este calculată suma elementelor de pe diagonala principală (DP) sau diagonala secundară (DS), ambele prin două variante (Fig. 11). În prima variantă observăm indexarea liniilor matricei la intrarea în ciclu, pentru controlul numărului de iteraţii. În a doua variantă întră în ciclu întreg tabloul.

i, j - indice linie, coloana; DP: i=j; DS: i+j=n-1 => $j=n-1-i$



Suma pe DP Suma pe DS

Fig. 11

6. Separare şir în două subşiruri: unul strict pozitiv, altul strict negativ

Pentru valoarea 0 sunt selectate cazurile False din ambele instrucţiuni Case, iar tablourile trec neafectate.

Pentru *valoare >0*, instrucţiunea Case de sus execută cazul True şi celălalt Case execută caz False.

Pentru *valoare <0*, instrucţiunea Case de jos execută cazul True şi celălalt Case execută cazul False. Şirurile rezultate sunt de lungimi diferite, astfel se va folosi Build

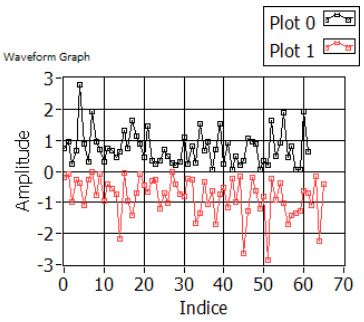


Fig. 12

Cluster Array (Fig. 13) pentru obţinerea unui cluster de şiruri inegale. Urmează vizualizarea grafică cu Waveform Graph.

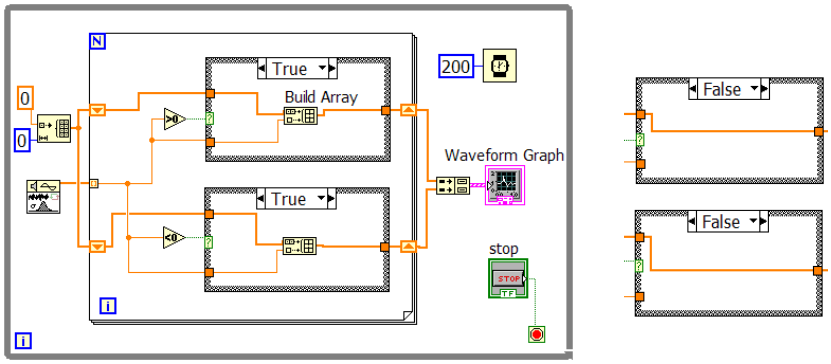


Fig. 13

7. Ordonarea unui şir de numere reale

7.1. Metoda bulelor - bubble sort

Se va ordona descrescător şirul de valori (iniţial) copiat în registrul shift. În ciclul For interior se realizează comparaţii a câte două valori alăturate din şir:

$$a[j] < a[j+1]$$

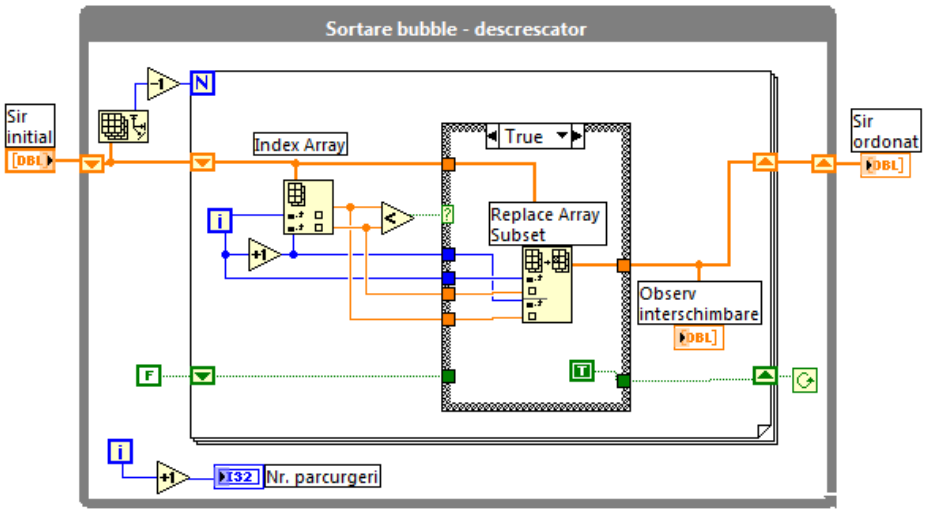


Fig. 14

Dacă a[j] < a[j+1] este True se interschimbă cele două elemente alăturate, se actualizează registrul Shift de sus (conţine tabloul supus ordonării) şi se pune True în registrul Shift Boolean (de jos). Dacă a[j] < a[j+1] este False, elementele comparate rămân pe poziţiile lor, iar registrul Boolean păstrează valoarea logică. După fiecare

comparaţie elementul mai mic va fi pe poziţia cu indice mai mare, adică a[j+1]. Se începe cu j=0, caz în care are loc prima comparaţie a[0]< a[1] şi se termină cu j=N-1. *Replace Array Subset* realizează interschimbarea prin două înlocuiri: valoarea din tablou de la indicele i este pusă la indicele i+1, iar valoarea din tablou de la indicele i+1 este pusă la poziţia i. Când are loc o parcurgere a şirului, fără nici o interschimbare, valoarea False din registrul de jos nu se schimbă şi va opri ciclul While exterior. De fiecare dată ciclul For interior parcurge întregul şir inclusiv subşirul deja ordonat aflat la indici mari; aplicaţia se poate îmbunătăţi prin expresia Array Size-***i***-1 pentru setarea lui N (ciclul For) unde *i* este contor pentru While.

7.2. Metoda inserţiei

Registrul Shift al ciclului For exterior (Fig. 15) conţine şirul pe tot parcursul ordonării descrescătoare a acestuia. Ciclul For exterior primeşte la terminalul N valoarea Array Size -1, astfel se repetă de n-1 ori, unde n este numărul valorilor din şirul de ordonat.

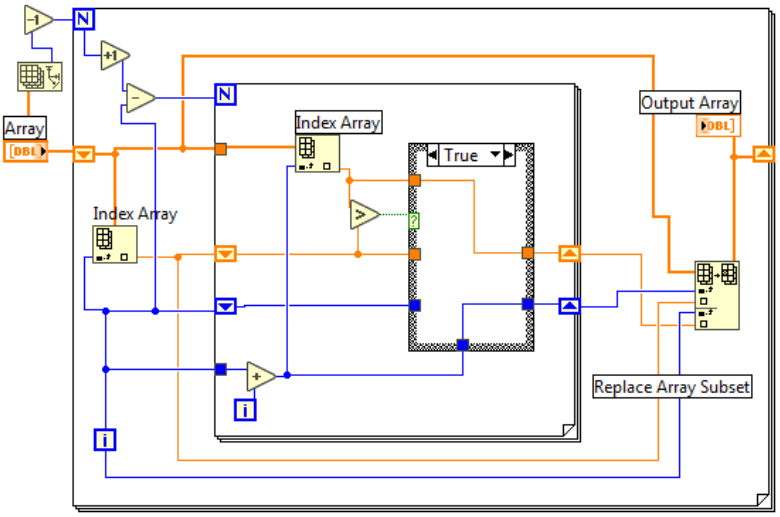


Fig. 15

La fiecare iteraţie a ciclului For exterior, ciclul For interior parcurge un subşir tot mai mic din şirul de ordonat. La prima iteraţie a şirului For exterior, ciclul For interior parcurge întregul şir de ordonat, găseşte elementul maxim şi poziţia acestuia păstrate în cei doi regiştri Shift (ataşaţi ciclului For interior); cazul True este selectat când valoarea curentă extrasă prin Index Array este mai mare decât cea din registrul Shift. În continuare are loc interschimbarea primului element din şir cu cel maxim găsit. Procesul se repetă. Prima poziţie din şir fiind ocupată de cel mai mare element, procesul se repetă pentru subşirul din care este exclus primul element. Astfel, şirul For interior va parcurge subşirul rămas, găseşte elementul maxim şi poziţia acestuia, iar la ieşire din ciclul For interior are lor din nou interschimbarea elementelor. Acest proces în final realizează aranjarea descrescătoare a valorilor numerice din tablou.

8. Extragerea unor linii din matricea 2D şi generarea unei submatrice

Se vor extrage liniile dorite din tabloul numeric Array 2D şi se va forma o nouă matrice cu acestea (Fig. 16, Fig. 17). Se iniţializează registrul Shift cu o matrice (tablou 2D) de zero linii şi zero coloane de tip numeric.

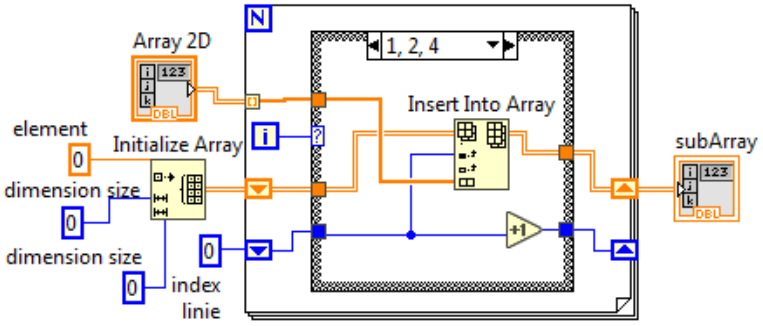


Fig. 16

La fiecare iteraţie a ciclului For linia curentă a tabloului Array 2D (intrată prin indexare în ciclul For) este verificată cu ajutorul structurii Case dacă se află printre liniile selectate de a face parte din noua submatrice. În caz afirmativ cu funcţia *Insert Into Array* se adaugă linia selectată în noul tablou la poziţia dictată de valoarea întreagă *index linie* conţinută în al doilea registru de transfer. Iniţial acest registru are valoarea 0 şi incrementează după fiecare linie adăugată. Cazul *0 Default* asigură trecerea fără modificarea datelor din registrul conţinând matricea şi a celui care conţine indicele liniei.

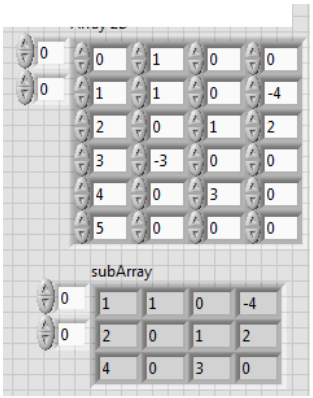


Fig. 17

9. Rezolvarea unui sistem de ecuaţii liniare şi verificarea soluţiei

Fie un sistem de ecuaţii liniare scris matriceal:

$$Ax=B$$

Sistemul este rezolvat prin apelul funcţiei *Solve Linear Equations.vi* (Fig.18). Pentru a eficientiza calculul soluţiei se specifică tipul matricei coeficienţilor sistemului prin controlul *matrix type* şi selecţia unei valori din lista:

general (0),
matrice pozitiv definită (1),
matrice triunghiulară jos (2) sau
matrice triunghiulară sus (3).

În diagrama din Figura 19 se observă, după rezolvarea sistemului, verificarea soluţiei prin înmulţirea fiecărei linii a matricei coeficienţilor (A) cu soluţia coloană (X) şi suma produselor.

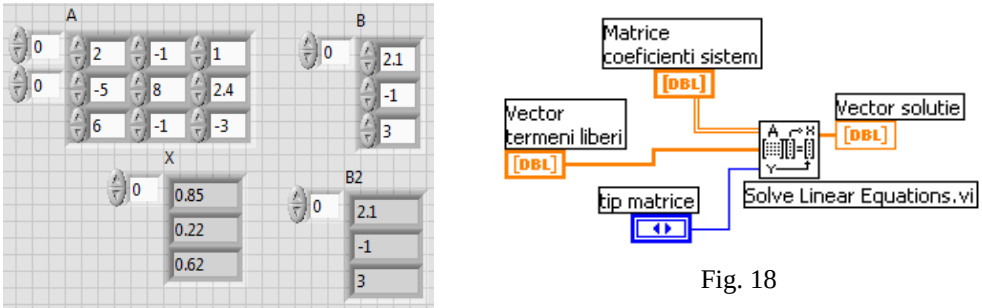


Fig. 18

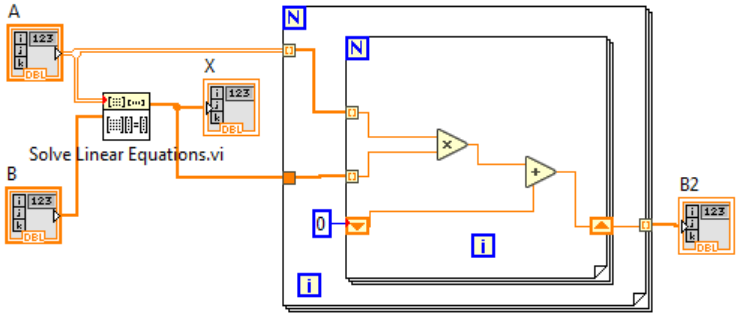


Fig. 19

10. Studiul unei funcţii date prin control şir de caractere - grafice asociate

Aplicaţia urmăreşte un exemplu disponibil în cadrul mediului LabVIEW. Se introduce în PF (Fig. 20) expresia unei funcţii în controlul de tip şir de caractere numit *formula*, aceasta fiind *funcţia de studiu*.

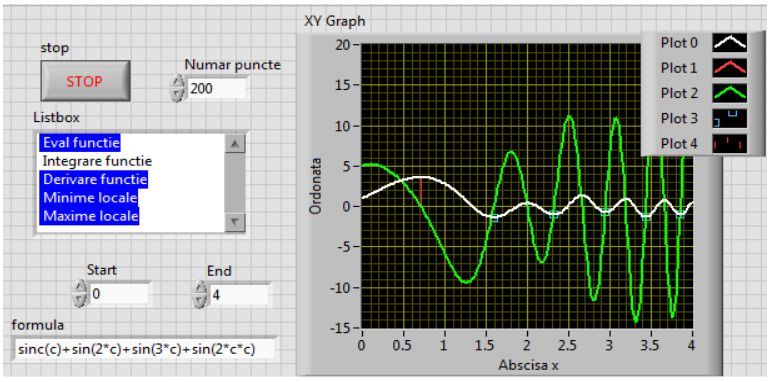


Fig. 20

Evaluarea funcţiei algebrice se face repetitiv în cadrul unui ciclu While, oprit cu butonul de stop. Controlul *Listbox* comandă o structură Case (Fig. 21). Listbox conţine o listă cu următoarele articole: Eval functie, Integrare functie, Derivare functie, Minime locale şi Maxime locale (numerotate de la 0 la 4). Controlul Listbox generează un tablou de valori întregi; tabloul conţine numai poziţiile selectate din listă. Acest tablou intră cu indexare în ciclul For, ciclu care se repetă de atâtea ori câte poziţii sunt selectate în Listbox. Se iniţializează registrul de transfer asociat ciclului For cu un tablou de cinci structuri. Fiecare structură conţine două tablouri 1D de valori reale, unul conţine abscise şi al doilea ordonatele (asociate unei curbe). Iniţial tablourile sunt cu zero elemente. Aceste tablouri vor conţine valori numerice, în urma prelucrării funcţiei de studiu, în vederea reprezentărilor grafice.

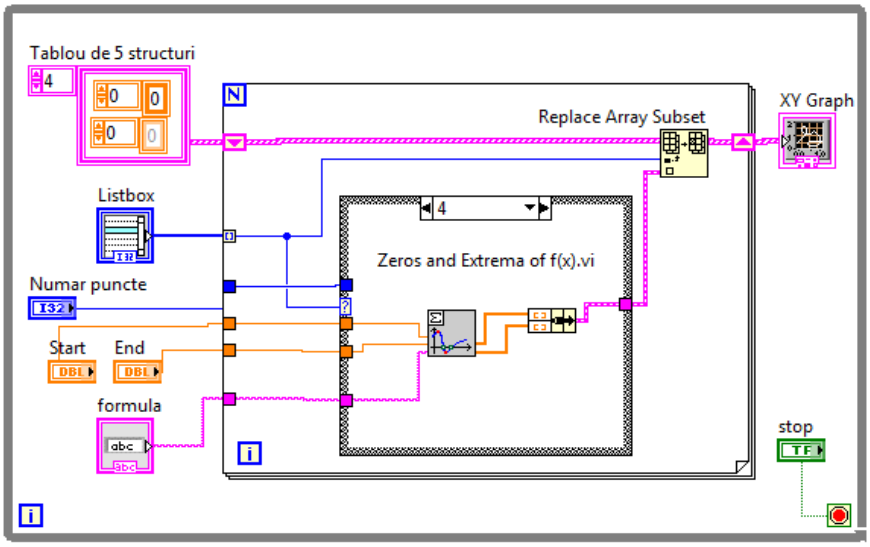


Fig. 21

Funcţiile folosite în cazurile instrucţiunii Case sunt următoarele: *Integration.vi, Differentiation.vi, Eval y=f(x).vi, Zeros and Extrema of f(x)*.vi. Funcţiile sunt plasate în subpaletele: Calculus, Zeros şi 1D & 2D Evaluation observate în Figurile 22 şi 23.

În controlul Listbox (1D array of long [32 bit integer]) se marchează (CTRL+ click mouse) acţiunea sau acţiunile dorite din listă. Modul de lucru al listei este specificat prin *Selection Mode*, acesta fiind setat pe *0 or More Items* (Fig. 24).

Tabloul de structuri este intrarea acceptată de indicatorul XY Graph pentru a trasa mai multe grafice. Prin combinaţia Ctrl+Mouse se adaugă sau elimină articole din Listbox. Tabloul de valori întregi care conţine articolele selectate din Listbox intră cu indexare în ciclul For şi selectează pe rând cazurile care se execută în Case. În fiecare caz din Case este apelată o funcţie din subpaletele amintite. Fiecare funcţie primeşte expresia funcţiei de studiu din controlul formula şi generează o

structură de două tablouri (abscise şi ordonate) care (structură) actualizează tabloul de cinci structuri la poziţa indicată de indicele curent al tabloului generat de Listbox. Tabloul de structuri intră în XY Graph şi astfel se vor vizualiza zero, una sau mai multe curbe.

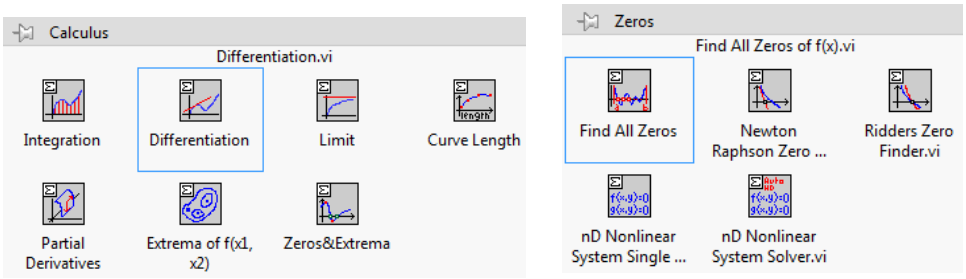


Fig. 22

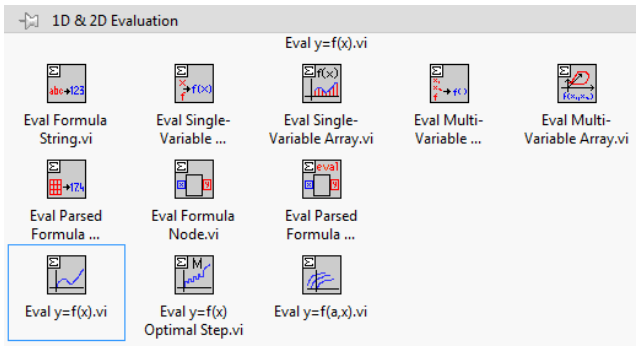


Fig. 23

La fiecare repetiţie a ciclului exterior While, în funcţie de articolele selectate din Listbox, se execută cazurile din Case potrivite (conform cu selecţia făcută), iar structurile corespunzătoare sunt actualizate în cadrul tabloului *Tablou de 5 structuri*. Conform cu prelucrările matematice selectate (evaluare funcţie, integrare funcţie etc.) se vor vizualiza curbele corespunzătoare în XY Graph.

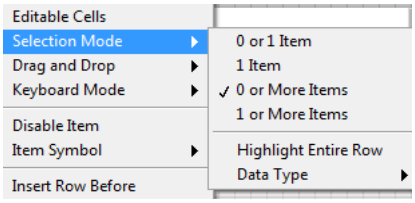


Fig. 24

VII. Formula Node, inserare cod limbajul C, Call Library Function Nod

1. Formula Node, utilizare cod limbajul C

Formula Node este o structură de programare care permite evaluarea de expresii mai complexe scrise cu sintaxa C şi execuţia de programe în limbajul C. Pe marginile structurii se pot adăuga variabile de intrare prin comanda Add Input (Fig. 1). Acestea primesc valoare din exteriorul structurii Formula Node. Prin comanda Add Output se adaugă variabile de ieşire. Acestea primesc valoare în urma evaluării expresiilor de forma: Var_out = expresie;

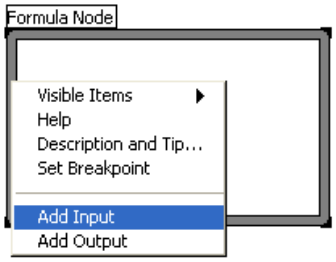


Fig. 1

1.1. Evaluarea unor expresii numerice

În Figura 2, observaţi operatorii din expresii (? : , **, /) şi funcţiile matematice apelate. x1 şi x2 sunt variabile de intrare în Formula Node, iar y1, y2, y3 şi y4 conţin valori de ieşire, calculate. Operatorul ternar este de forma:

variabila = exp_cond ? expT : expF

Se evaluează expresia condiţională. Dacă valoarea expresiei *exp_cond* este True, atunci se evaluează expresia *expT*, iar valoarea obţinută se atribuie variabilei din stânga semnului egal. Dacă *exp_cond* are valoarea False, se evaluează *expF*, iar valoarea obţinută se atribuie variabilei din stânga semnului egal.

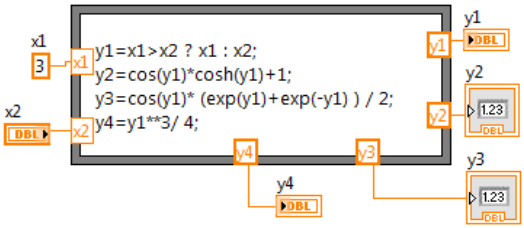


Fig. 2

1.2. Suma valorilor din şirul numeric Array1D

Este folosit ciclul For din LabVIEW şi în paralel cod C scris în Formula Node. Variabilele *x* şi *ne* sunt parametri de intrare în Formula Node (Fig. 3). Variabila *sum* este parametru de ieşire (calculat).

int i; – este declaraţia variabilei *i* de tip întreg,

float sum=0; – este declaraţia variabilei *sum* de tip real şi initializare cu 0.

În ciclul For se însumează elementele din şir astfel:

1) *i=0* – reprezintă *expr1* din Figura 3,

2) dacă *i < ne* (*expr2*) este T → se execută corpul ciclului: *sum=sum+X[i];*
iar dacă *expr2* este F → se iese din ciclul For,

3) *i++* (expr3) creşte variabila *i* (incrementează) cu 1,

4) se execută pasul 2) din nou.

1.3. Suma valorilor numerice din Array (2D), cod C în Formula Node

În diagrama din Figura 4 sunt doi parametri de intrare: *dim* este un tablou 1D conţinând două valori întregi (nl = numărul de linii şi nc = numărul de coloane din tabloul 2D al valorilor numerice de adunat) şi *X* este numele tabloului Array (2D) de valori reale, nume care va fi folosit pentru accesarea valorilor în codul C. Observăm două cicluri for imbricate astfel: ciclul exterior parcurge liniile tabloului 2D, iar cel interior parcurge pe rând elementele liniei curente (deci coloanele).

Parametrii de ieşire *nl* şi *s* sunt valori scalare.

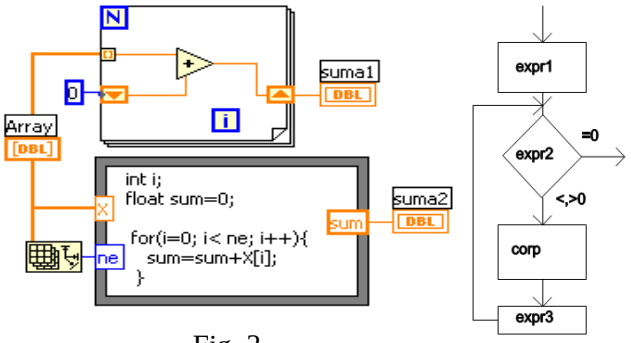


Fig. 3

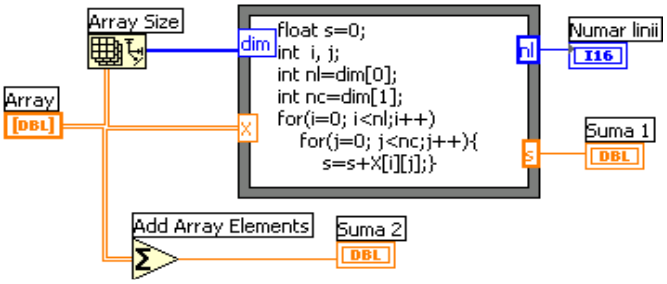


Fig. 4

1.4. Suma anumitor elemente din tabloul (1D) - ciclul While (C)

În Figura 5 este folosit ciclul While condiţionat anterior din limbajul C, pentru parcurgerea întregului şir de valori ale tabloului X. *ne* este numărul de elemente returnat de Array Size. Dacă valoarea elementului curent este în intervalul (0, 10), valoarea este adunată la variabila *sumP*. În variabila *sum* sunt adunate toate elementele tabloului.

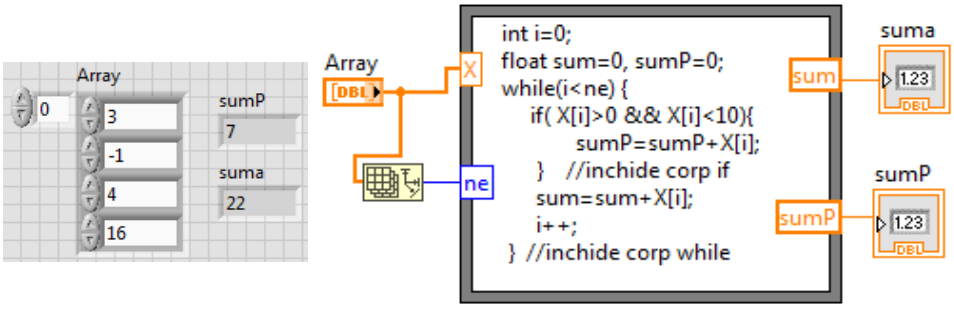


Fig. 5

2. Apel C/C++ DLL din LabVIEW

2.1. Se creează un proiect DLL din CodeBlocks IDE în C/C++
File/ New/ Project... (Fig. 6)*;*
Din fereastră selectăm pictograma Dynamic Link Library (Fig. 6)*;*
Scriem titlul proiectului: SumaTablouDLL;
Selectăm Compiler: GNU GCC Compiler şi bifăm numai Create "Release"...
Se creează fişierul sursă C++: *main.cpp*
şi un fişier header: *main.h.*

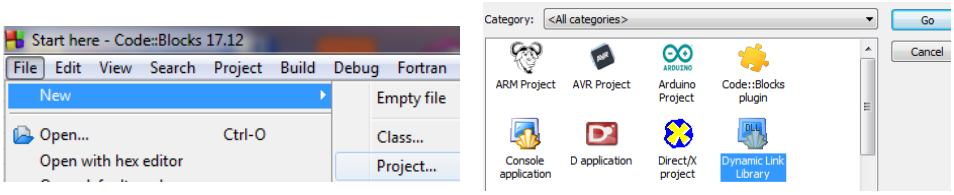


Fig. 6

Fişierul cod sursă *main.cpp* (şablon propus) după adăugarea codului pentru realizarea sumei elementelor pozitive din tablou mai mici decât 10, va conţine:

```
#include "main.h"
// a sample exported function
float DLL_EXPORT  SumPoz(const LPCSTR sometext, float X[],  int ne )
{       // ↓ functie Win32 API pentru configurare Box dialog cu mesaj
   MessageBoxA(0, sometext, "DLL Message", MB_OK | MB_ICONINFORMATION);
     int i=0;  float sum=0;
       while( i<ne ) {
          if( X[i]>0 && X[i]<10)
                sum=sum+X[i];
          i++;
       }
        return sum;
}
.....
```

Fişierul header *main.h* (şablon propus) conţine declaraţia funcţiei SumPoz( ):

```
......
extern "C"
{  #endif
float DLL_EXPORT SumPoz(const LPCSTR sometext,  float X[],  int ne );
#ifdef __cplusplus } ...
```

2.2. Generarea bibliotecii dinamice DLL de programe compilate în format executabil folosind comanda *Build Ctrl+F9* => Output file is bin\Release\ SumaTablouDLL.dll.

Se corectează erorile dacă sunt.

DLL este o bibliotecă dinamică de programe compilate în format executabil (.exe). Programele pot fi folosite prin apelul lor în timpul execuţiei altor aplicaţii.

LIB este bibliotecă statică, conţine programe compilate care se leagă la început în cadrul unui program care le apelează şi vor fi incluse în executabilul acelui program. Dacă două programe A şi B apelează acelaşi program executabil (E) din LIB, ambele (A şi B) vor fi mai mari prin includere la început prin link-editare.

2.3. Call Library Function Nod din LabVIEW

Funcţia Call Library Function Nod din *Connectivity/Libraries& Executables/...* (Fig. 7) apelează cod extern din biblioteca DLL (Dynamic Link Library). Se poate expanda (asemănător cu Bundle) şi configura nodul *Call Library Function Node* pentru a preciza biblioteca, funcţia, parametrii funcţiei şi valoarea returnată.

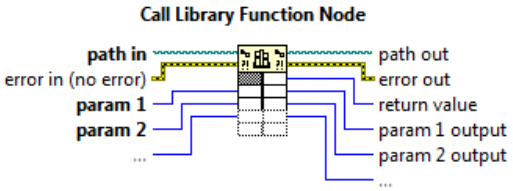


Fig.7

Nodul conţine câte un terminal pereche intrare-ieşire pentru fiecare parametru al funcţiei SumPoz. Prin terminalul din stânga se trimite un parametru (date) în funcţie iar prin cel corespunzător din dreapta se citeşte valoarea parametrului după execuţia funcţiei.

Return value este valoarea returnată a funcţiei. Dacă tipul datei returnate de funcţie este *Void*, terminalul cel mai de sus-dreapta nu este folosit. Implicit 'calling convention' este C.

2.4. Detalii despre pasarea parametrilor

În funcţia din SumPoz.dll cei 3 parametri sunt: şir de caractere/string, tablou/array numeric real (float) şi numeric întreg pe 32 biţi. Funcţia returnează suma elementelor din tablou cu valori între 0 şi 10 (float 4 bytes). În general pentru tipul numeric se pot folosi subtipurile: întreg pe 8, 16, 32, 64-bit, cu semn sau fără semn, şi tipurile real pe 4-byte (single-precision numbers) şi 8-byte (double-precision numbers).

În figurile de mai jos este prezentat modul de configurare prin comanda Configure... (Fig. 8) a parametrilor nodului *Call Library Function Nod*.

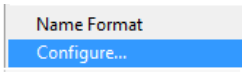


Fig. 8

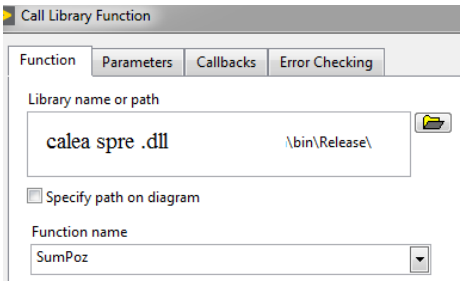


Fig. 9

În secţiunea Function (Fig. 9) se specifică calea ...bin\Release\ *nume*.dll spre biblioteca .dll creată din platforma Code::Blocks. Cu butonul + (secţiunea Parameters) se adaugă trei parametri: arg1, arg2, arg3. În cele patru panouri din Figura 10, se specifică tipul valorii returnate de funcţie şi tipul celor 3 parametri.

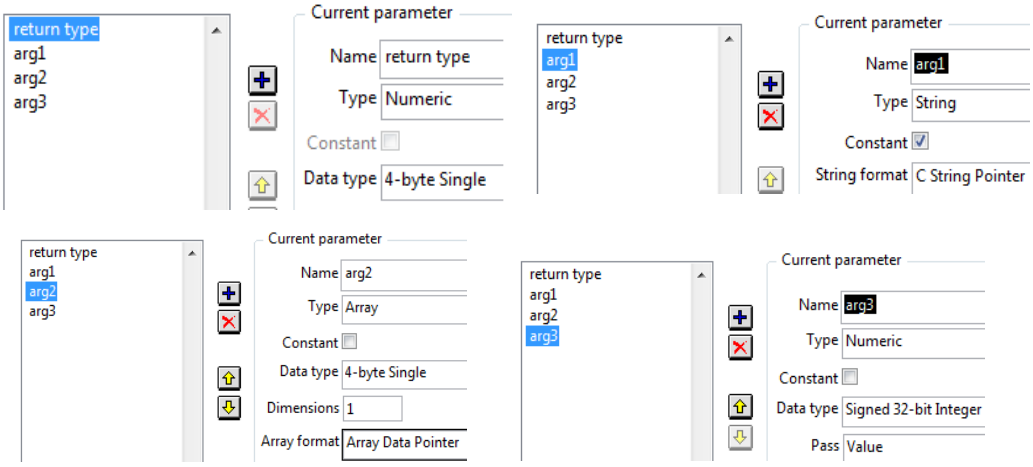


Fig. 10

Funcţia SumPoz primeşte 3 parametri actuali (cunoscuţi) (Fig. 11):
arg1: şirul de caractere 'apel DLL C',
arg2: tabloul de valori reale (control în PF) (20, 1, -3, 7) şi
arg3: numărul de valori reale din tablou, returnat de Array Size.

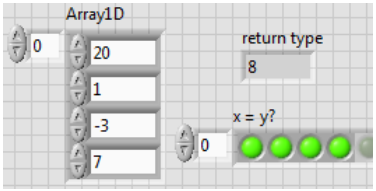


Fig. 12

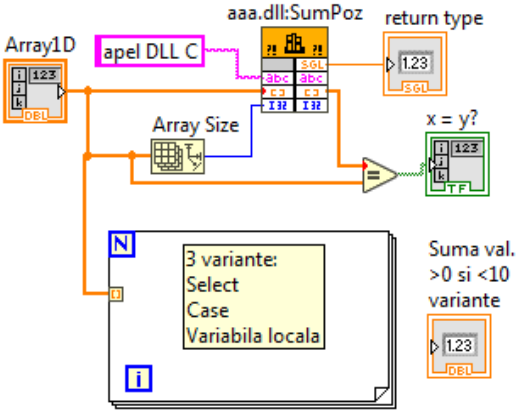


Fig. 11

În Panoul Frontal al aplicaţiei (Fig. 12) indicatorul *return type* afişează suma elementelor cu valori în intervalul (0, 10) şi egalitatea la nivel de elemente între tabloul de intrare şi cel de ieşire din funcţie sub forma tabloului de valori logice (toate sunt True fiindcă tabloul nu se modifică în funcţia apelată).

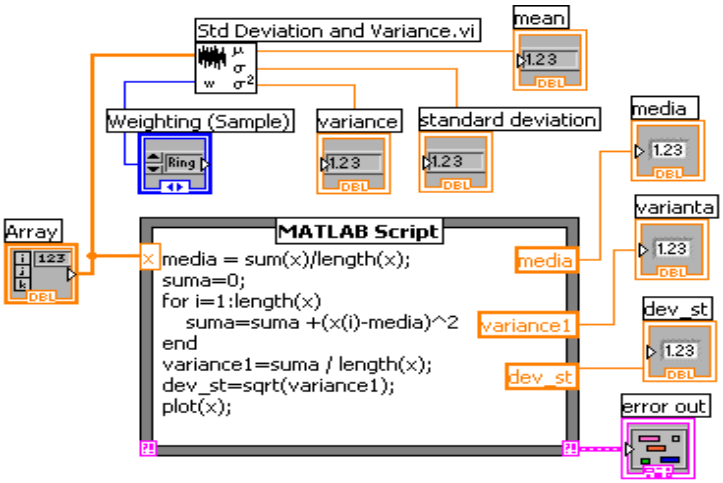
2.5. Să se includă în ciclul For (Fig. 11) codul grafic LabVIEW, pentru calculul sumei elementelor pozitive şi mai mici decât 10, folosind trei variante:

Select în prima variantă, Case pentru selectarea valorilor şi adunarea elementelor în a doua variantă, şi variabila locală în locul registrului Shift în a treia variantă.

3. Apel script MATLAB (MATLAB script node)

Se poate include cod script Matlab în structura de programare *MATLAB script node*. Execuţia codului are loc prin apelul produsului software MATLAB® . Produsul Matlab trebuie să fie instalat în calculator (versiunea 6.5 sau o versiune mai recentă). Se introduce în diagramă structura de programare MATLAB Script din paleta Mathematics/Script&Formula/Script Nodes.

Se va calcula media, deviaţia standard şi varianţa pentru un şir de valori numerice X(i), i=1,…,n. Variabilele de intrare şi ieşire sunt următoarele:
add input: tabloul x,
add output: media, variance1, dev_st.



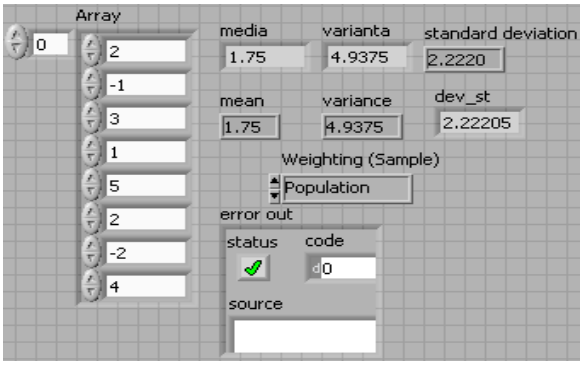


Fig.13

Observăm codul Matlab (Fig. 13): ciclul *for* pentru parcurgerea şirului numeric x, *plot*(x) pentru reprezentarea grafică a şirului într-o fereastră grafică. S-a apelat funcţia LabVIEW Std deviation and Variance.vi pentru valori comparative.

4. Modularizarea programului. Generare subVI ca funcţie apelabilă individual din diagrama altei aplicaţii (apel subVI)

4.1. Aplicaţia din Figura 14 realizează evaluarea repetitivă a unor expresii logice cu operanzii logici (LEDuri) a şi b, şi operatorii logici And, Or, Nand.

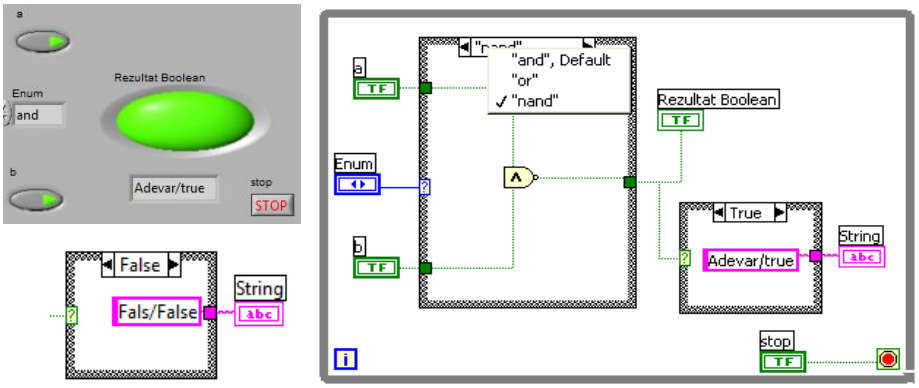


Fig. 14

4.2. Generare SubVI

Se selectează cu mouse o zonă dreptunghiulară (Fig.16) din diagrama unei aplicaţii/VI (de preferinţă nu se includ în zona selectată terminale control sau indicator).

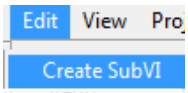


Fig. 15

Cu comanda *Create SubVI*, din meniul diagramei bloc *Edit/Create SubVI* (Fig. 15), se înlocuieşte zona selectată cu o pictogramă (Fig. 16): Untitled 2 (SubVI).

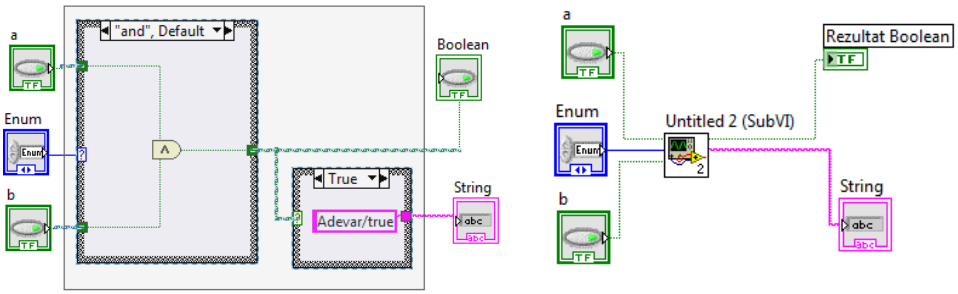


Fig. 16

Prin dublu click pe noul SubVI (sau Open Front Panel, Fig. 17), se deschide şi se observă PF şi diagrama şi apoi se salvează cu numele dorit pentru o viitoare

utilizare ca instrument virtual independent. Pictograma asociată noului SubVI poate fi editată/personalizată prin click dreapa pe colţul dreapta-sus a panoului frontal al aplicaţiei.

Se poate include/apela noul SubVI într-o aplicaţie mai generală cu comanda *Select a VI...* din diagrama aplicaţiei apelantă. Acest SubVI inclus în diagramă pune la dispoziţia operatorului intrările şi ieşirile funcţiei. Astfel, Untitled2 (SubVI) prezintă intrările *a*, *Enum* şi *b*, iar valorile returnate sunt *Rezultat Boolean* şi *String*.

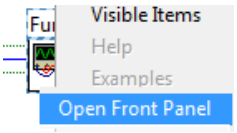


Fig. 17

5. Probleme propuse

5.1. Se generează o aplicaţie care conţine un Tab Control cu trei pagini după modelul din Figura 18. La selectarea unei pagini se selectează un caz al unei instrucţiuni CASE. În fiecare caz se va executa o altă aplicaţie dintre cele deja efectuate în laboratoarele precedente. Fiecare dintre cele trei aplicaţii este în prealabil transformată într-un SubVI cu controalele şi indicatoarele proprii prezente în pagina corespunzătoare a obiectului Tab Control.

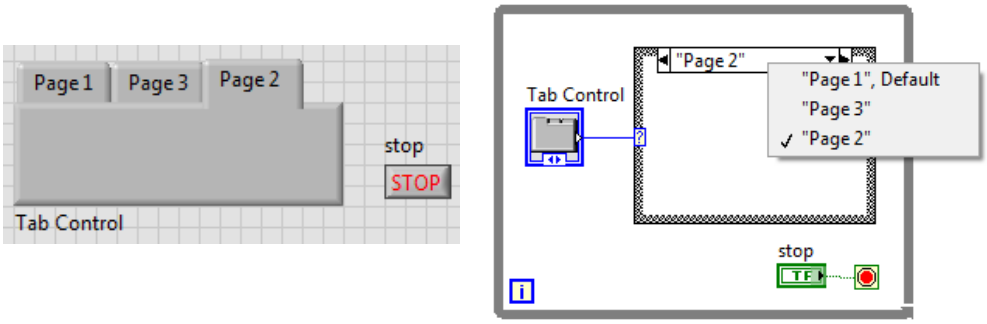


Fig. 18

5.2. Se generează SortariTablou.dll din Code::Bloks limbajul C++ care va conţine funcţia sortari.cpp. Funcţia sortări primeşte următorii patru parametri: un şir caractere pentru generare fereastră cu mesaj, tablou şir de numere reale, număr elemente din tablou şi un număr întreg cu rol de selector. Dacă selectorul este 1 se ordonează şirul crescător, iar daca selectorul este 2 se ordonează descrescător şirul (metoda de sortare este la alegere). Funcţia returnează suma elementelor din tablou şi tabloul sortat. Se foloseşte Call Library Function Nod din LabVIEW.

VIII. Calcule cu Numere Complexe I

1. Reprezentarea numerelor complexe, produsul şi câtul

1.1. Z1 şi Z2 sunt controale numerice, având tipul de dată complex (CDB parte reală + parte imaginară). Se creează iniţial un control numeric real urmat de schimbarea reprezentării în memorie (click dreapta pe controlul numeric + Representation, Fig.1).

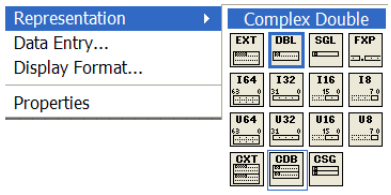


Fig. 1

1.2. Operatorul *Complex To Polar* returnează două numere reale: modulul şi faza. Operatorul *Complex To Re/Im* returnează două numere reale: partea reală şi cea imaginară (Fig.2).

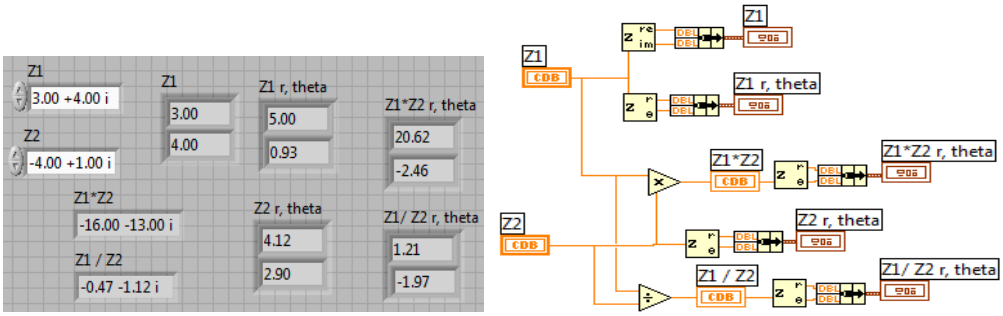


Fig. 2

1.3. În diagrama din Figura 2, observăm *înmulţirea* a două numere complexe Z1·Z2. Rezultă un nou număr complex având modulul egal cu produsul modulelor lui Z1 şi respectiv Z2: 5*4.12=20.6. Faza noului număr complex este suma fazelor (radiani) numerelor Z1 şi Z2: 0.93+2.9=3.83; 3.83-2*π ≈ -2.45.

1.4. Prin *împărţirea* a două numere complexe Z1/Z2 rezultă un nou număr complex având modulul egal cu câtul modulelor numerelor Z1 şi Z2 (5/4.12=1.21) şi având faza (radiani) egală cu diferenţa fazelor: 0.93-2.9=-1.97.

1.5. Se reaminteşte exprimarea unui număr complex z în coordonate carteziene rectangulare: $z = x + j\,y$, unde x şi y sunt numere reale, iar $j = \sqrt{-1}$. De asemenea x este partea reală a numărului complex z, iar $j \cdot y$ este partea imaginară a numărului complex z în planul complex (Fig. 3).

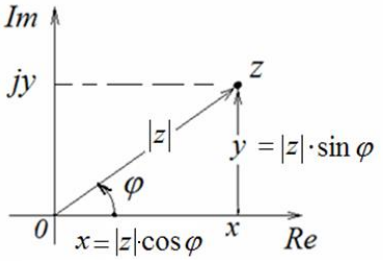


Fig. 3

Numărul complex z exprimat în coordonate

polare (r, φ) (Fig. 3) are forma (1):

$$r\angle\varphi \text{ unde } r=|z|,\ \varphi\in[-\pi,\pi] \qquad (1)$$

sau expresia (2) folosind exprimarea prin funcţia exponenţial complexă (relaţia lui Euler):

$$z=|z|\cdot e^{j\varphi} \qquad (2)$$

Conversia din exprimarea carteziană în cea polară se face cu relaţiile (3):

$$r=|z|=\sqrt{x^2+y^2} \text{ iar } \varphi=a\tan 2(y,x) \qquad (3)$$

Dacă se cunoaşte z în coordonate polare, conversia în coordonate carteziene se face astfel (4):

$$z=|z|\cdot(\cos\varphi+j\sin\varphi) \qquad (4)$$

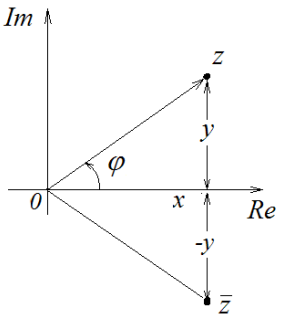


Fig. 4

unde $x=|z|\cdot\cos\varphi,\quad y=|z|\cdot\sin\varphi$.

În Figura 4 sunt vizualizate două numere complex conjugate: $z=x+j\,y$ şi $z^*=x-j\,y$, acestea fiind plasate în planul complex în oglindă faţă de axa reală.

2. Instrumentul *Real FFT.vi*

2.1. Funcţia FFT.vi (Fig. 5) aplică tabloului de intrare X de n valori reale x_i transformata Fourier rapidă (FFT) sau transformata Fourier discretă reală (DFT). Rezultă tabloul FFT{X} de n coeficienţi spectrali complecşi y_k.

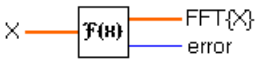


Fig. 5

Se va programa relaţia (5) de calcul a coeficienţilor y_k.

$$y_k=\sum_{i=0}^{n-1}x_i\,e^{-jk2\pi\frac{i}{n}},\quad j=\sqrt{-1} \qquad (5)$$

Folosind relaţia lui Euler (Fig. 6):

$$e^{jx}=\cos(x)+j\cdot\sin(x) \qquad (6)$$

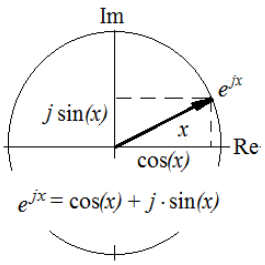


Fig. 6

rezultă:

$$y_k=\sum_{i=0}^{n-1}x_i[\cos(k2\pi\frac{i}{n})-j\cdot\sin(k2\pi\frac{i}{n})] \qquad (7)$$

Semnificaţia coeficienţilor în tabloul de ieşire din funcţie este următoarea:
y_0→ componenta continuă (reală) a semnalului de intrare, de forma:

$$y_0=\sum_{i=0}^{n-1}x_i\,e^0$$

y_1→ prima armonică din componenţa semnalului iniţial,

$\mathbf{y_2}\rightarrow$ a doua armonică,

…

$y_{\frac{n}{2}-1}\rightarrow$ a n/2-1 armonică,

$y_{n/2}\rightarrow$ *armonica Nyquist* de forma:

$$y_{n/2}=\sum_{i=0}^{n-1}x_i[\cos(\pi i)-j\sin(\pi i)]=\sum_{i=0}^{n-1}x_i(-1)^i$$

Urmează coeficienţii complex conjugaţi simetrici de frecvenţe negative:

$\mathbf{y_{n/2+1}}\rightarrow$ a n/2-1 armonică,

…

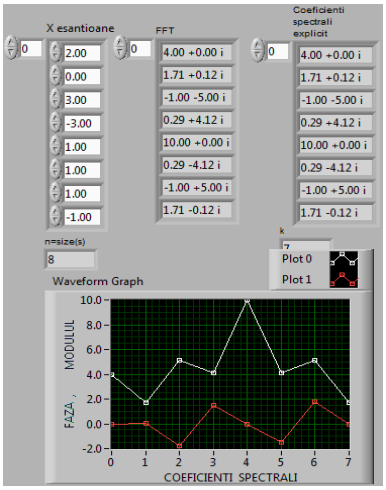
$\mathbf{y_{n-2}}\rightarrow$ a doua armonică,

$\mathbf{y_{n-1}}\rightarrow$ prima armonică.

Coeficienţii complecşi y_k pot fi exprimaţi în varianta modul şi fază:

Modul= $\sqrt{\mathrm{Re}^2+\mathrm{Im}^2}$,

Faza= $a$tan2(Im, Re) $\in[-\pi,\pi]$ (arctangentă în 4 cadrane).



$\mathbf{y_0}$ valoare *reală*
$\mathbf{y_1}, \mathbf{y_2}, \mathbf{y_3}$ valori complexe
$\mathbf{y_4}$ valoare *reală*
$\mathbf{y_5}, \mathbf{y_6}, \mathbf{y_7}$ val. complex conjugate

Fig. 7

2.2. În PF (Fig. 7) observăm tabloul de 8 valori reale (X eşantioane), tabloul coeficienţilor complecşi FFT calculaţi cu funcţia predefinită FFT.vi (Signal Processing/Transforms) şi tabloul coeficienţilor complecşi calculaţi cu programul (puţin eficient) din diagramă (Fig. 8).

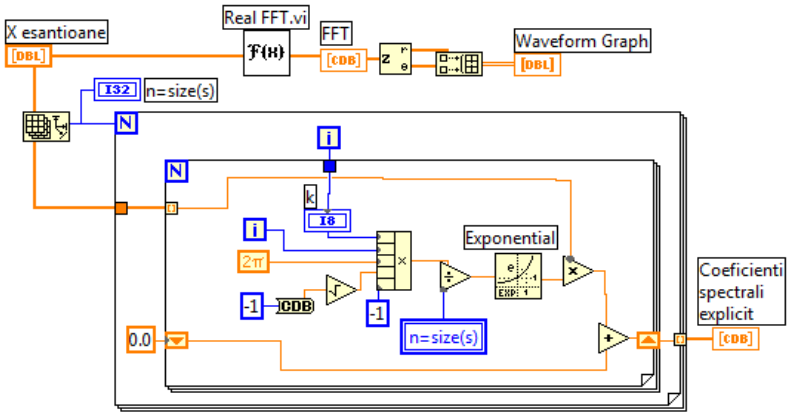


Fig. 8

În diagramă, suma după indicele *i* din relaţia (5) este efectuată de ciclul interior, sumele parţiale fiind gestionate de registrul Shift asociat. La fiecare ciclu exterior se calculează câte un coeficient y_k (k primeşte valoarea contorului ciclului exterior). Funcţia Compound Aritmetic pregăteşte exponentul funcţiei exponenţiale

realizând produs între valorile de intrare. Tabloul *X eşantioane* intră în ciclul For exterior fără indexare, astfel întregul tablou va fi disponibil în corpul ciclului pentru calculul fiecărui coeficient spectral.

n este numărul de eşantioane (valori) pe perioada T a semnalului (T=n·Δt) sau într-un bloc de date supus analizei. $\Delta t = 1/f_s$ este timpul între două valori măsurate sau perioada de eşantionare; f_s este frecvenţa de eşantionare, $\Delta f = f_s / n$ este spaţierea în domeniul frecvenţă între coeficienţii spectrali y_k.

3. Reprezentarea grafică a coeficienţilor spectrali în format vectorial

3.1. Feather Plot şi Compass Plot

Se doreşte vizualizarea unor vectori în plan. Coeficienţii complecşi generaţi de funcţia FFF.vi sunt în general perechi de numere complex conjugate. Pentru reprezentarea grafică se selectează Graph/Feather Plot şi Plot Helper *(Vector sau Complex)*. Se observă grafic perechile de coeficienţi complex conjugaţi şi cei doi coeficienţi reali plasaţi pe axa reală (orizontală).

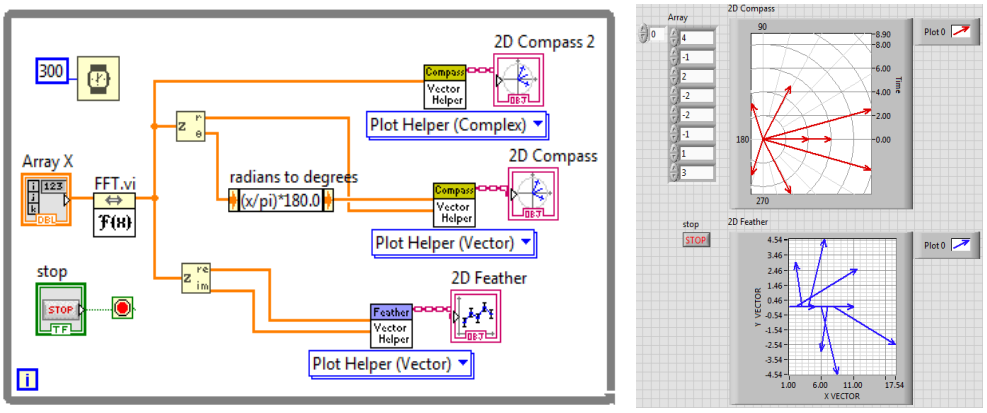


Fig. 9

Modificaţi interactiv valorile din şirul real de intrare şi numărul de valori din vector. Observaţi perechile de vectori simetrici şi cei doi coeficienţi reali.

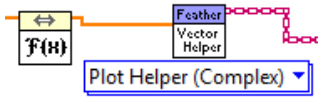


Fig. 10

Varianta în care intrarea funcţiei Feather Plot este tablou Complex este prezentată în Figura 10.

Indicatorul grafic Compass Plot versiunea Plot Helper (*Vector sau Complex*) (etichetă 2D Compass) permite vizualizarea tabloului generat de funcţia Real FFT.vi reprezentând coeficienţii spectrali complex conjugaţi (plus cei doi coeficienţi reali), ca vectori cu aceeaşi origine. Intrările în funcţie sunt în două

variante: a) varianta Vector: tablou module şi tablou unghiuri [grade] şi b) varianta Complex: tablou de numere complexe.

3.2. Feather Plot multiplu - înlănţuire de Feather Plot Helper

Sunt vizualizate, în Figura 11, două seturi de vectori de lungimi diferite în aceeaşi fereastră grafică.

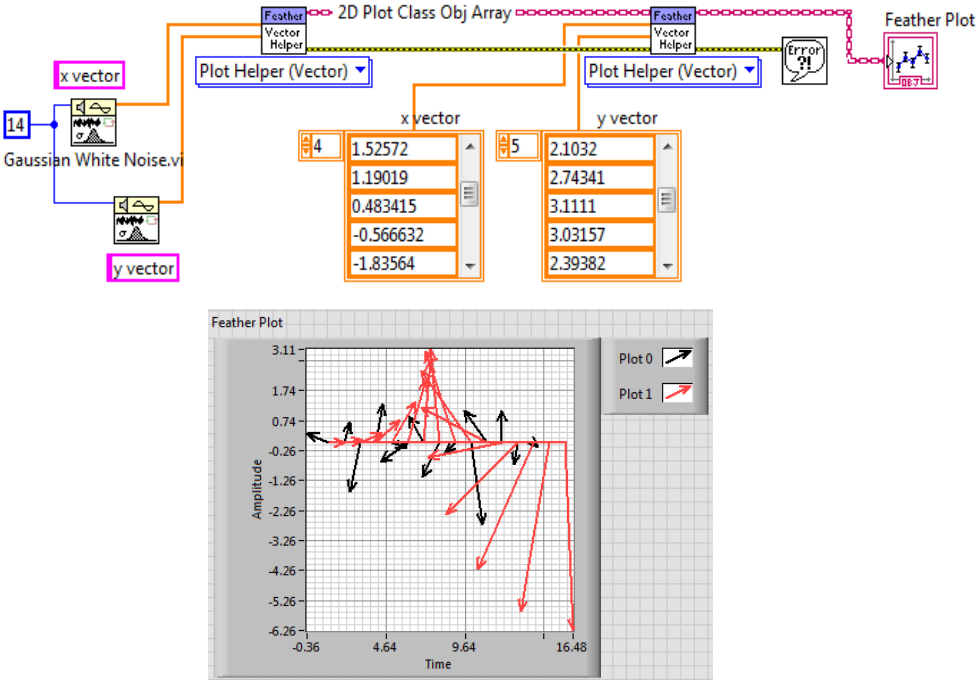


Fig. 11

4. Puterea spectrală bilaterală

Funcţia *Power Spectrum.vi* aplică şirului de intrare X expresia (8):

$$S_{xx} = \frac{1}{n^2} FFT\{X\}^* \cdot FFT\{X\} = \frac{1}{n^2} |FFT\{X\}|^2 \qquad (8)$$

Rezultă puterea spectrală *bilaterală* S_{xx}. n este numărul eşantioanelor de intrare din X, egal cu al puterilor spectrale rezultate. În diagrama din Figura 13 se obţine puterea spectrală în trei moduri de calcul. Este folosită funcţia Power Spectrum vizualizată în Figura 12 (două versiuni ale produsului LabVIEW).

X este tablou real 1D de intrare, conţine n

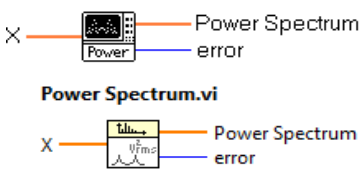


Fig. 12

eşantioane din semnalul de intrare considerat periodic.
FFT{X} este transformata Fourier rapidă sau discretă DFT aplicată şirului X.
Sxx (Power Spectrum) = tabloul de ieşire real 1D, conţine n puteri spectrale.
$Sxx(0)=y_0^2/n^2$ (pătratul coeficientului componentei continue/n^2).
$Sxx(i)=(y_i^* \cdot y_i)/n^2$, i=1,...,n-1.
Produsul a două numere complex conjugate este un număr real.

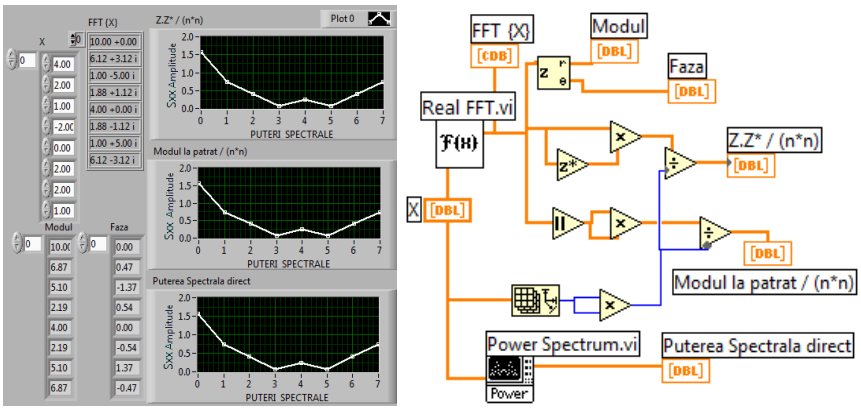


Fig. 13

5. Puterea interspectrală bilaterală complexă Sxy(f)

5.1. Puterea interspectrală dintre două semnale date prin valori reale discrete plasate în cei doi vectori de intrare X şi Y (fiecare de câte n valori reale) se realizează prin instrumentul Cross Power.vi, cu relaţia (9):

$$S_{xy} = \frac{1}{n^2} FFT^*\{X\} \cdot FFT\{Y\} \qquad (9)$$

Rezultă puterea interspectrală complexă bilaterală Sxy(f).
Caracterul asterisc * indică conjugata complexă a valorilor din FFT{X},
n este numărul eşantioanelor reale din X şi Y, egal cu numărul valorilor complexe din tabloul rezultat Sxy.

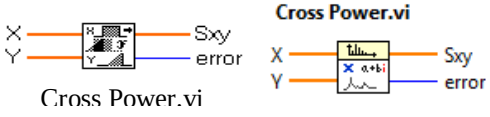


Fig. 14

Dacă $n=2^k$, unde k=1,2,…,23, instrumentul apelează transformata Fourier Rapidă FFT, altfel apelează transformata discretă DFT. Dacă tablourile X şi Y sunt identice rezultă valori reale în Sxy. În diagrama din Figura 15 este calculată puterea interspectrală bilaterală folosind instrumentul Cross Power.vi (Fig. 14) şi în paralel observăm calculul explicit după relaţia (9), constatând identitatea rezultatelor şi simetria valorilor modulelor faţă de componenta centrală Nyquist.

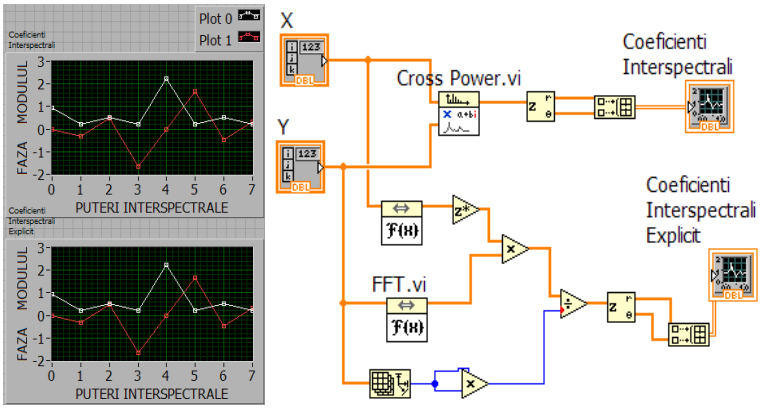


Fig. 15

5.2. Puterea spectrală unilaterală poate fi vizualizată printr-un grafic de bare la ascultarea unui fragment audio (muzică) folosind un CD player (Fig. 16).

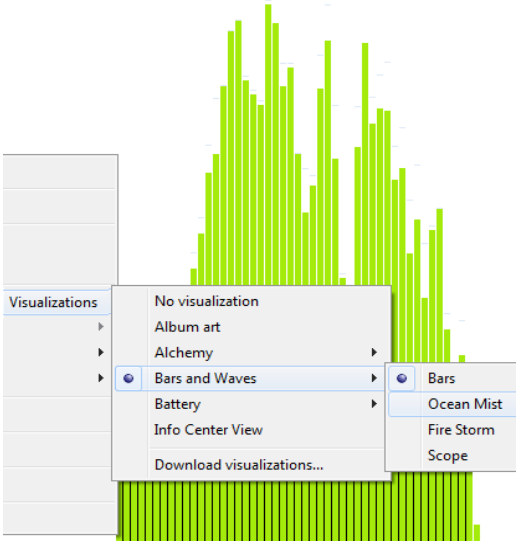


Fig. 16

6. Generarea unui fişier executabil - aplicaţie LabVIEW

6.1. Se doreşte realizarea unui fişier executabil plecând de la un instrument virtual dezvoltat în LabVIEW. Se va putea executa aplicaţia în alt sistem de calcul (calculator) în care nu este instalat LabVIEW, dar trebuie să fie instalat modulul *LabVIEW Runtime Engine*

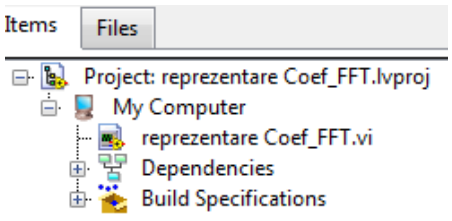


Fig. 17

RTE. Pentru generarea executabilului este folosit modulul *Application Builder*. Acesta este inclus în LabVIEW, varianta Professional Development System.

În prima etapă este generat şi salvat proiectul (reprezentare Coef_FFT. lvproj, Fig. 17); acesta conţine aplicaţia (reprezentare Coef_FFT.vi) sau aplicaţiile (.vi). În continuare, prin mouse *click dreapta*, pe câmpul *Build Specifications* (în Project Explorer, Fig. 18) se selectează New/*Application (.exe)* (sau folosim meniul Tools). Dacă este necesar, în prealabil sunt selectate folosind săgeţi, fişierele sursă care se vor compila (Fig. 19).

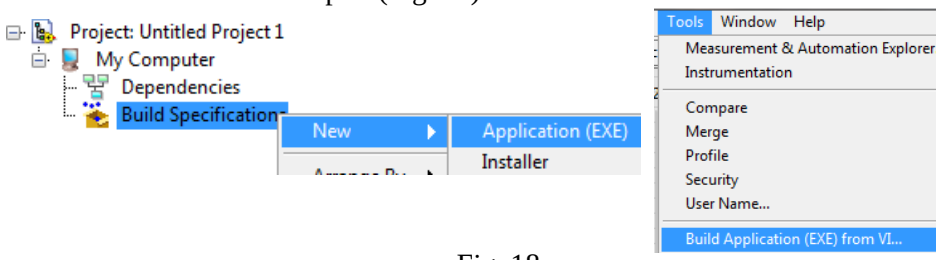


Fig. 18

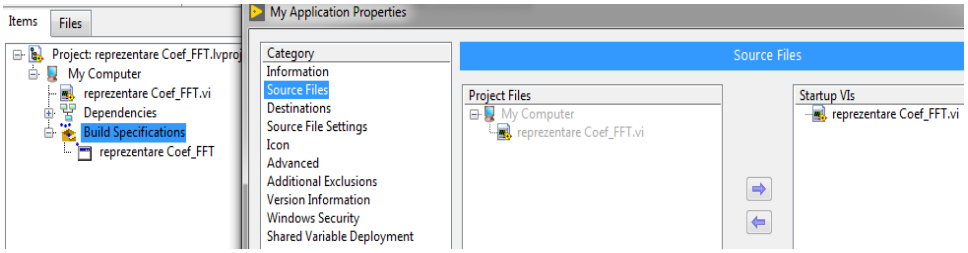


Fig. 19

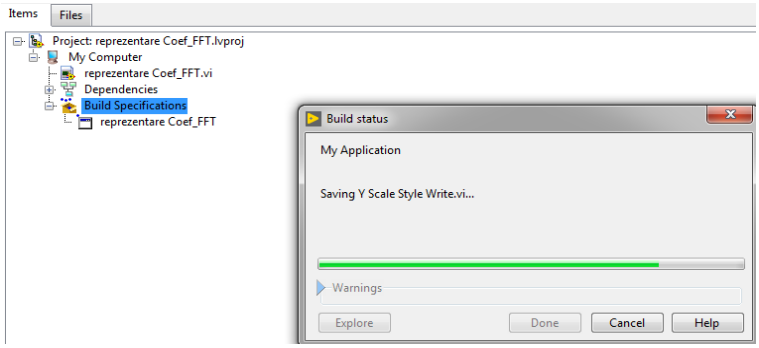


Fig. 20

Se urmăreşte în desfăşurare procesul de compilare (Fig. 20).

Este verificat conţinutul directorului unde s-a depus aplicaţia executabilă. S-a creat un director *build* în care sunt depuse fişierele executabile (Fig. 21).

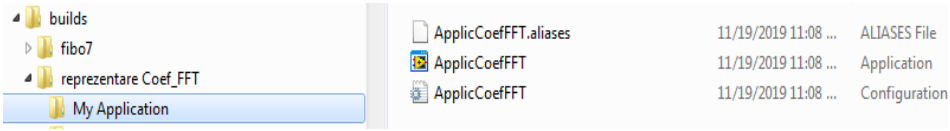


Fig. 21

Este lansată în execuţie aplicaţia executabilă (Fig. 22). S-a făcut referire la aplicaţia reprezentare Coed_FFT.vi descrisă mai sus.

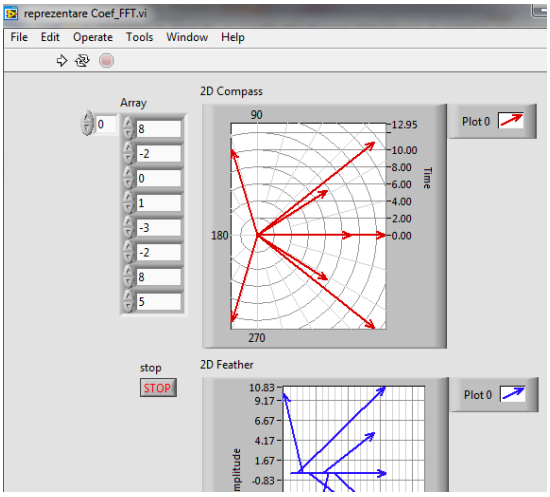


Fig. 22

6.2. Un proiect LabVIEW grupează instrumente virtuale (.vi) şi fişiere asociate lor, diverse resurse, referinţe la fişiere salvate pe disc, variabile, hyperlinkuri etc., cu scopul realizării unei aplicaţii mai ample. Aceste informaţii colaborative sunt grupate într-un fişier proiect cu extensia .lvproj. Gestionarea proiectului se realizează prin utilitarul LabVIEW Project Explorer. Din acesta se pot creea noi fişiere .vi aparţinând proiectului, prin comanda New/VI (comanda deschide un fişier blanc de tip .vi, Fig. 23), noi variabile, clase, biblioteci etc.. Pot fi adăugate aplicaţii .vi existente (comanda Add). Secţiunea *Dependencies* a fişierului proiect conţine zona de fişiere *vi.lib* şi zona de *subVI*-uri asociate unor fişiere .vi incluse în

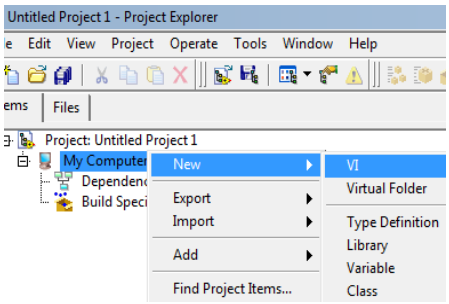


Fig. 23

proiect. De asemenea sunt adăugate fişiere .dll asociate unor fişiere .vi aflate în secţiunea *MyComputer*. O bibliotecă sau library file (LLB) se poate creea plecând de la comanda: Build Specifications/ New/ Source Distribution (Fig. 24). Lista posibilităţilor oferite de LabVIEW Project Explorer poate continua.

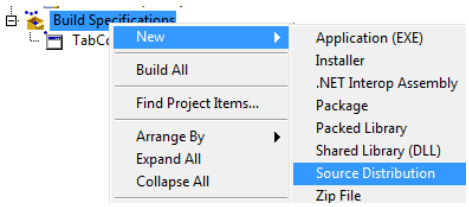


Fig. 24

7. Probleme propuse

7.1. Calculaţi $S'_{xy} = \frac{1}{n^2} FFT\{X\} \cdot FFT^*\{Y\}$ şi comparaţi cu Sxy (rezultă conjugata complexă a vectorului Sxy).

7.2. Reprezentaţi vectorial Sxy(f) apelând Feather Plot şi comparaţi cu S'xy(f).

7.3. Observaţi simetria din Y, Sxx şi Sxy (unde Y=FFT{X}) faţă de termenii: $y_{n/2}$, Sxx[n/2] şi respectiv Sxy[n/2] (excepţie primul element).

7.4. Să se calculeze media aritmetică $\bar{y}$ asociată liniilor spectrale ale puterii spectrale unilaterale (Auto Power Spectrum) în dB, conform relaţiei (10). n este numărul liniilor spectrale generate de funcţia Auto Power Spectrum (Fig. 25) fiind jumătate din numărul eşantioanelor semnalului de intrare *Signal (V)*. A_i este amplitudinea liniei spectrale indice i, unde i=0,...,n-1. Funcţia Auto Power Spectrum este plasată în paleta Signal Processing/Spectral Analysis.

$$\bar{y} = 20\log\left(\frac{1}{n}\sum_{i=0}^{n-1}\frac{A_i}{10^{-5}}\right) \tag{10}$$

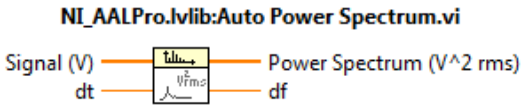


Fig. 25

7.5. Să se calculeze media geometrică $\hat{y}$ asociată liniilor spectrale ale puterii spectrale unilaterale (Auto Power Spectrum) în dB, conform relaţiei (11) unde 10^{-5} este o mărime de referinţă la care se raportează coeficientul spectral A_i.

$$\hat{y} = 20\log\left(\left(\prod_{i=0}^{n-1}\frac{A_i}{10^{-5}}\right)^{1/n}\right) = \frac{1}{n}\sum_{i=0}^{n-1}20\log\left(\frac{A_i}{10^{-5}}\right) \tag{11}$$

IX. Calcule cu Numere Complexe II

1. Raportul a două şiruri complexe - funcţia de transfer

Se va observa o aplicaţie care împarte element cu element două tablouri 1D de valori complexe (A şi B) şi selectează prima jumătate din valorile rezultate.

1.1. A (stimul sistem) şi B (răspuns sistem) sunt două tablouri 1D de *n* valori reale fiecare; acestea pot fi considerate semnale (măsurate) cu spaţiere în timp *dt* între eşantioane succesive. Timpul total sau perioada T a semnalelor (semnalele sunt considerate periodice) este:

$$T=n\cdot dt \qquad (1)$$

Frecvenţa armonicei fundamentale egală cu spaţierea armonicelor în domeniul frecvenţă este:

$$df=1/T \qquad (2)$$

1.2. Funcţia Real FFT.vi primeşte un tablou de *n valori reale* şi returnează un tablou de *n valori de tipul complex*. În Figura 1 observăm diagrama funcţiei de de transfer (unilateral) predefinită în LabVIEW.

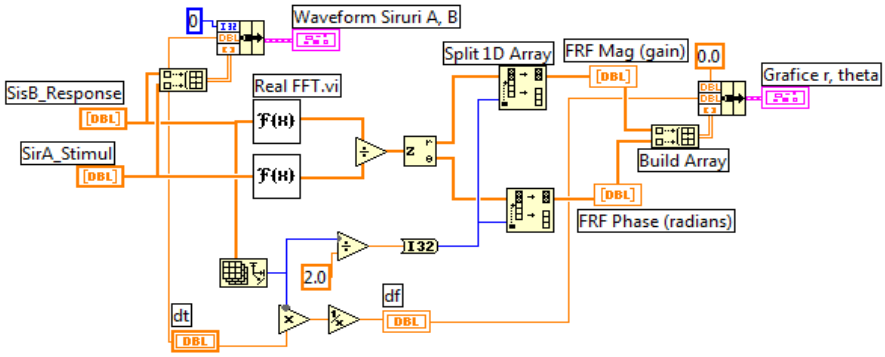


Fig. 1

1.3. Cele două şiruri de numere complexe se împart element cu element (B(i)/A(i), i=0,…,n-1) prin operatorul Divide (Fig. 2) şi rezultă tabloul 1D complex (Transfer Function, TF):

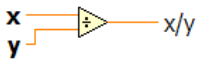


Fig. 2

$$TF=\frac{FFT\{B\}}{FFT\{A\}} \quad <=>$$

$$TF=\frac{FFT\{B\}(i)}{FFT\{A\}(i)}, \quad i=0,\ldots, n-1 \qquad (3)$$

Sunt convertite în format polar (r-modul şi θ-faza în intervalul [-π, π]) toate cele n valori complexe rezultate. Prin funcţia *Split 1D Array* sunt selectate primele n/2 valori. Funcţia *Split 1D Array* returnează două subşiruri:

Split 1D Array **array** — **index** — first subarray — second subarray	subşir sus: array[0] ... array[index-1] din TF subşir jos: array[index] ... array[n-1]

La final sunt afişate, în acelaşi grafic, modulele şi fazele primelor n/2 valori complexe cu spaţiere între valori succesive *df* =1/T, în domeniul frecvenţă (Fig. 3).

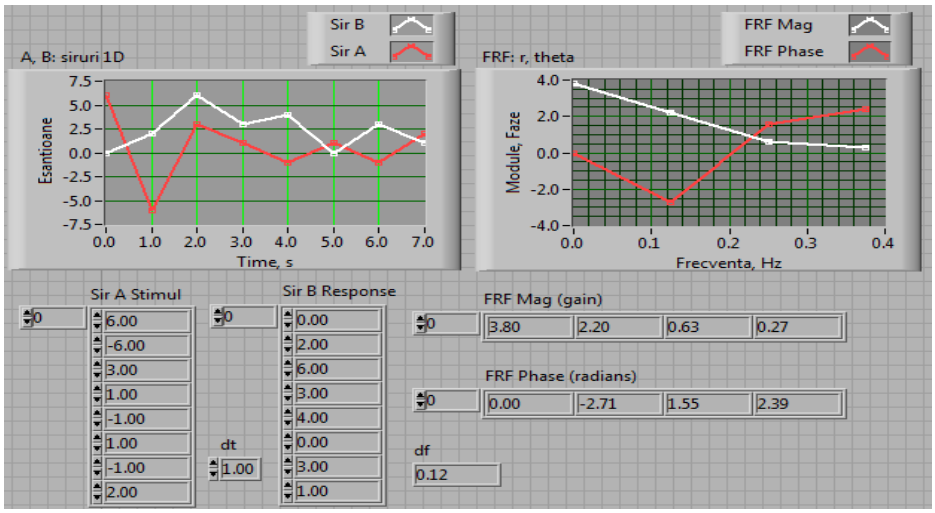


Fig. 3

1.4. Modificaţi aplicaţia/diagrama din Figura 1, realizând împărţirea perechilor de numere complexe în cadrul unui ciclu FOR (noua aplicaţie şi cea iniţială vor afişa grafice similare). În PF (Fig. 3) observăm cele două tablouri reale iniţiale, graficul lor, *dt*=1, *df*=1/n, n=8 valori şi graficul primei jumătăţi din numărul modulelor şi fazelor cu spaţiere *df* între valori.

2. Funcţii care returnează tablouri complexe: Cross Power Spectrum, Frequency response

Funcţiile abordate în diagrama din Figura 4 sunt unele complexe: Cross Power Spectrum(avg) şi Frequency response (avg) iar altele reale: Coherence Function şi Impulse Response.

2.1. Diagrama funcţiei predefinite Network Functions.vi

La fiecare iteraţie a ciclului For (Fig. 4) câte o linie (indice *i)* din tablourile *2D (Semnal 2D Stimul* şi *Semnal 2D Raspuns)* participă la calculul funcţiilor Power Spectrum.vi şi Cross Power.vi. La iteraţia indice 0 este selectat Cazul *..0;* la ieşirea din acest caz sunt iniţializaţi cei trei regiştrii de transfer. Registrul de sus (#1)

conţine un tablou de valori complexe (puteri interspectrale), iar registrul#2 şi registrul#3 conţin tablouri de valori reale (puteri spectrale). La iteraţiile indice 1, 2,…,n-1 este selectat Cazul *1*.. în care se însumează tablourile (nu există caz Default). La ieşirea din ciclu observăm medierea cu n, unde n este numărul de linii ale matricelor 2D de intrare. Funcţia Power Spectrum.vi calculează: $S_{xx} = |FFT\{X\}|^2 / n^2$, iar funcţia Cross Power.vi calculează $S_{xy} = FFT^*\{X\} \cdot FFT\{Y\} / n^2$. Puterile spectrale şi interspectrale vor participa în continuare la calculul celor patru funcţii de ieşire.

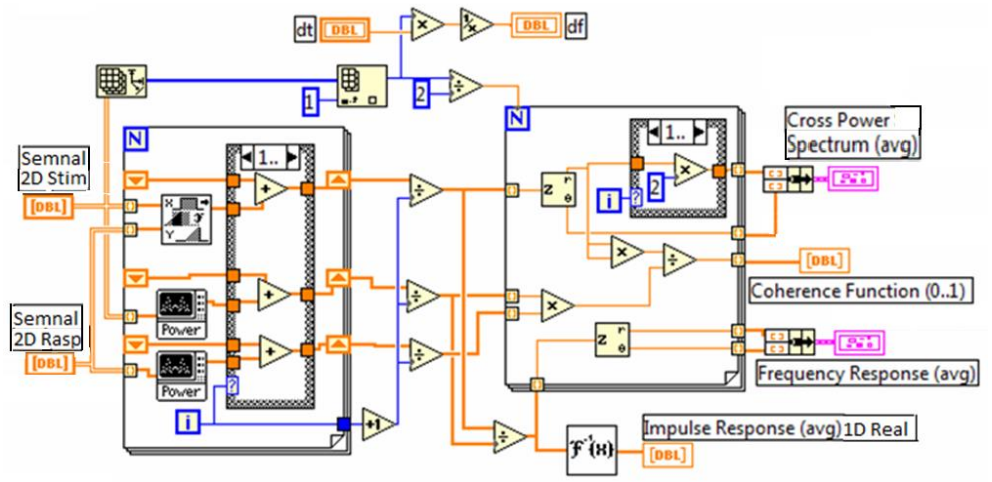


Fig. 4

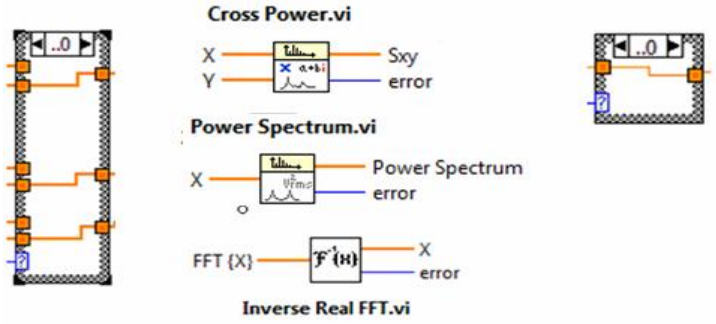


Fig. 5

Al doilea ciclu For se repetă de n/2 ori. La ieşirea din acest ciclu rezultă trei funcţii la care se adaugă funcţia Impulse Response (răspunsul sistemului la un semnal de tip impuls), astfel:

a) Funcţia complexă Cross Power Spectrum (avg) Gxy (unilateral: n/2 valori) dată prin două tablouri: tabloul modulelor (n/2 valori) şi tabloul fazelor (n/2 valori), unite într-o structură (cluster).

b) Funcţia complexă Frequency Response (avg) dată prin două tablouri: tabloul *modulelor* (n/2 valori) şi cel al *fazelor* (n/2 valori), unite într-o structură.

c) Funcţia reală Coherence Function (0..1) cu n/2 valori între 0 şi 1:

$$coherence = \left|avg\ S_{XY}\right|^2 / \left[avg\ S_{XX} \cdot avg\ S_{YY}\right] \quad (4)$$

d) Funcţia reală Impulse Response (avg) (n valori) este calculată folosind Transformata Fourier Inversă (Inverse Real FFT.vi) aplicată funcţiei de transfer mediată (Avg Transfer function). Astfel, sunt parcurse două etape de calcul:

1. Funcţia *Transfer Function TF* (n valori complexe calculate în etapa 1):

   $avg\ \ Transfer\ \ Function = avg\ S_{XY}(f) / avg\ S_{XX}(f)$

2. $avg\ Impulse\ Response = inverse\ \ FFT(avg\ Transfer\ Function)$

Funcţia *Inverse Real FFT.vi* primeşte un tablou 1D de n valori complexe şi returnează un tablou 1D de n valori reale.

Funcţiile, de mai sus, sunt calculate de aplicaţia predefinită Network Functions (avg).vi după relaţiile deja menţionate. În Figura 5 sunt aduse completări la diagrama din Figura 4, fiind afişate cazurile *..0* ale celor două structuri Case din Figura 4 şi alte două variante de pictograme pentru S_{XX} şi S_{YY}.

Spaţierea *df* în domeniul frecvenţă se calculează în mod similar cu aplicaţia din Figura 1, cu observaţia că din tabloul de două valori generat de funcţia Array Size este extras numărul de coloane (numărul elementelor din fiecare semnal al setului de semnale care participă la mediere).

2.2. Creaţi un SubVI cu trei intrări şi cinci ieşiri, similar cu funcţia Network Functions(avg).vi din Figura 6, plecând de la diagramele prezentate în Figurile 4 şi 5. Se va edita o altă pictogramă pentru SubVI-ul creat.

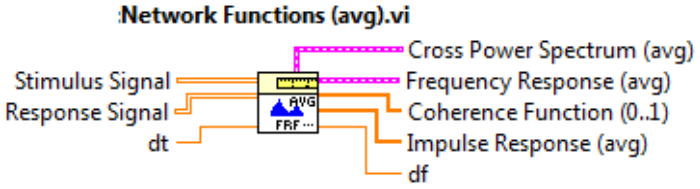


Fig. 6

3. Probleme propuse

3.1. Verificaţi printr-un program dacă rapoartele FFT{Y}/FFT{X} şi Sxy/Sxx aplicate şirurilor de intrare X şi Y sunt egale (aceleaşi valori pentru funcţia Frequency Response nemediată). Folosiţi un număr mare de valori de intrare pentru diagrama din Figura 7.

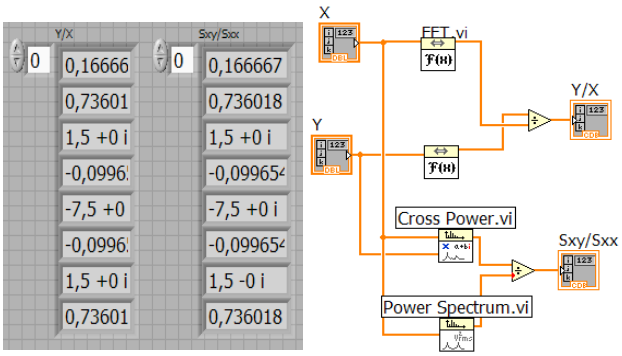


Fig. 7

3.2. Reprezentaţi grafic funcţiile Cross Power Spectrum(avg), Coherence Function(0..1) şi Frequency Response(avg) spaţiind valorile acestora în domeniul frecvenţă (abscisa) cu *df* , începând de la 0 Hz.

3.3. Explicaţi obţinerea funcţiei AutoPower Spectrum.vi (single-sided) din funcţia Power Spectrum.vi (Sxx, bilaterală) observând diagrama funcţiei AutoPower Spectrum.vi din paleta Signal Processing/Spectral Analysis.

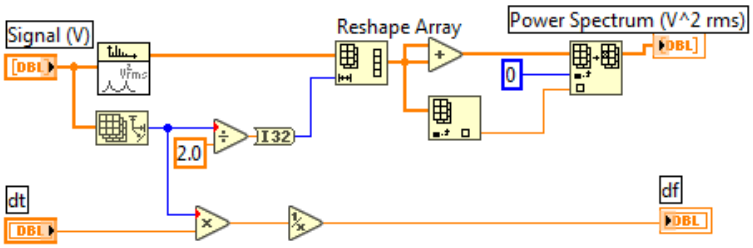


Fig. 8

X. LabVIEW + cod Matlab şi C - semnal ponderat (windowing)

1. Prelucrarea semnalului prin ponderarea valorilor

Se generează un semnal sinusoidal discret apelând funcţia Sine Wave.vi. Sunt folosite controale în Panoul Frontal pentru intrările: samples=*n*, frequency şi phase in. Semnalul va fi prelucrat prin una din cele patru ferestre de ponderare predefinite: Hanning, Hamming, Exponential şi Force selectate din paleta Signal Processing/Windows.

Fiecare eşantion din semnalul (tablou 1D) în domeniul timp, este înmulţit (ponderat) cu câte un coeficient specific ferestrei, în total *n* înmulţiri. Coeficienţii unei ferestre sunt calculaţi prin relaţii de calcul asociate ferestrei.

În aplicaţia din Figura 1, semnalul este prelucrat automat de o fereastră selectată de operator şi în paralel acelaşi semnal este înmulţit (prin program) cu coeficienţi generaţi după relaţia de calcul, rezultând două semnale identice suprapuse.

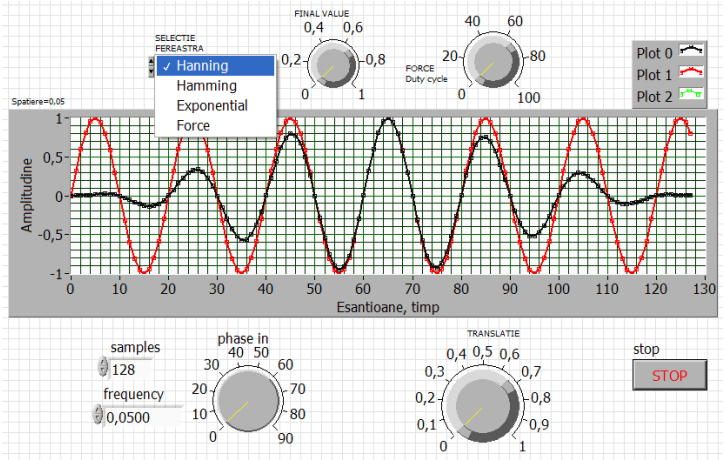


Fig. 1

2. Prelucrarea unui tablou prin fereastra Hanning - Matlab

2.1. În Figura 2 observăm curba sinus iniţială de amplitudine constantă şi două curbe identice, una prelucrată de fereastra Hanning (cu amplitudine zero la început şi la sfârşit) şi respectiv a doua obţinută separat prin programarea formulei de calcul (1). y_i sunt valorile prelucrate prin fereastră, iar x_i sunt valorile din semnalul iniţial.

$$y_i = 0.5x_i(1-\cos(w)),\ \ w = 2\pi i / n,\ \ i = 0,1,2,...,n-1 \qquad (1)$$

Controlul TRANSLATIE permite separarea celor două curbe identice. Prin aceasta se verifică egalitatea între rezultatele provenite din programarea explicită după relaţiile de calcul şi rezultatul aplicării ferestrei predefinite. Ponderarea cu coeficienţii ferestrei Hanning se face într-un caz separat printr-un segment de cod/script scris în Matlab (trebuie să existe produsul Matlab instalat în acelaşi calculator).

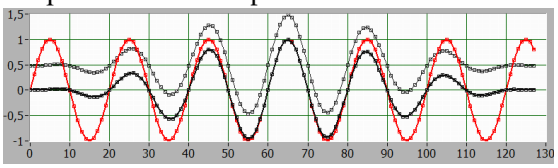


Fig. 2

În aplicaţia generală sunt folosite două instrucţiuni Case (4 cazuri în fiecare) şi un selector comun Selectie Fereastra de tip Enum (Fig. 3). Build Array uneşte trei semnale de tip tablou 1D într-un tablou/array 2D (3 linii şi 128 coloane).

2.2. Rescrieţi formulele Hanning în LabVIEW sau cod C.

3. Fereastra Hamming - cod LabVIEW

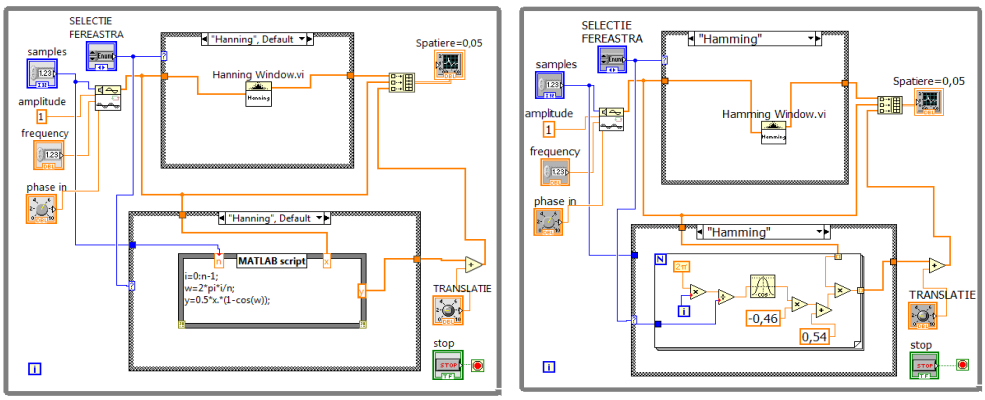


Fig. 3

Ponderarea cu fereastra Hamming se face în paralel într-un alt caz folosind programare LabVIEW şi relaţia (2). Pentru y_0 şi y_{n-1} se obţine atenuare maximă.

$$y_i = x_i(0.54 - 0.46\cos(w)),\quad w = 2\pi i/n,\quad i = 0,1,2,...,n-1 \qquad (2)$$

Fereastra Hanning anulează prima valoare din semnal, iar Hamming n-o anulează. Modificaţi controlul *faze in* (grade) pentru a porni semnalul sinusoidal de la o valoare nenulă şi a observa comparativ semnalele ponderate.

4. Fereastra Exponential - cod C în formula Node

La ponderarea semnalului cu fereastra Exponenţială în aplicaţia curentă se apelează la *formula nod* folosind codul limbajului C. y_i sunt valorile prelucrate prin fereastră conform cu relaţia (3). *Final Value* (f) este valoarea finală minimă, controlată din panoul frontal în limitele 0..1; micşorând-o, mărim atenuarea.

$$y_i = x_i \exp(a \cdot i),\ \ a = \ln(f)/(n-1),\ \ i = 0,1,2,...,n-1 \qquad (3)$$

În Figura 4 valoarea minimă este 0.12. Modificaţi controlul Final Value asociat ferestrei Exponential şi observaţi efectul.

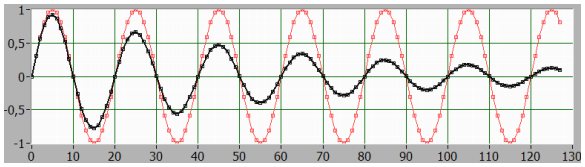


Fig. 4

5. Fereastra Force - cod LabVIEW

La ponderarea cu fereastra Force se apelează la programare în LabVIEW. Prin aplicarea ferestrei Force, semnalul iniţial este păstrat nemodificat în procentul specificat prin controlul FORCE Duty cycle (%) acesta putând fi în intervalul 0%...100% din lungimea semnalului, conform cu relaţia (4).

$$w = \begin{cases} x_i & (daca\ \ 0 \le i \le d) \\ 0 & altfel \end{cases} \qquad \text{unde: } d = (0.01) \cdot n \cdot (duty\ cycle) \qquad (4)$$

În Figura 5 se observă aplicarea feresterei Force având *duty cycle* = 65%.

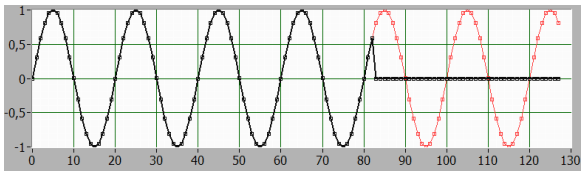


Fig. 5

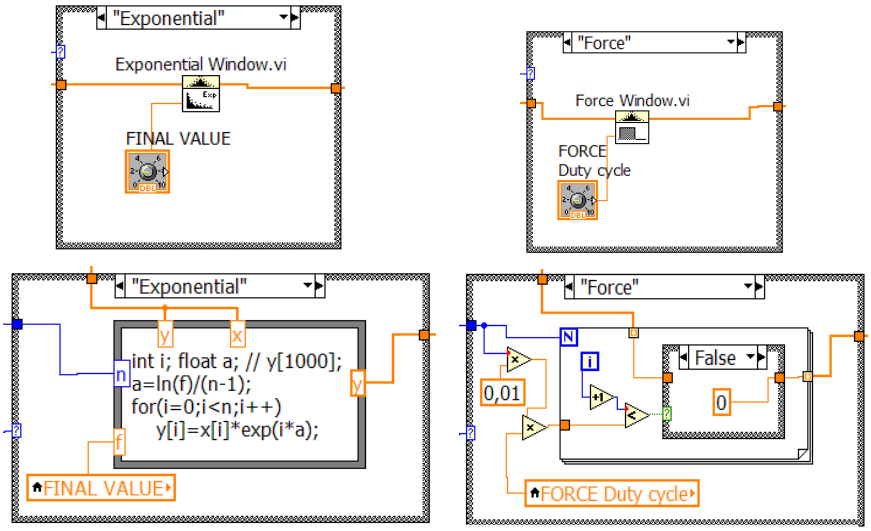


Fig. 6

Modificaţi controlul FORCE Duty cycle asociat ferestrei Force şi observaţi efectul. Se va prelucra semnalul cu ferestre de pondere (windowing) înainte de aplicarea transformatei Fourier (diminuează *spectral leakage*).

6. Probleme propuse

6.1. Introduceţi posibilitatea de a selecta dintr-o listă semnalul de prelucrat. Lista conţine trei semnale Sine wave.vi, Square wave.vi şi un semnal aleator.

6.2. Folosiţi o singură instrucţiune CASE în aplicaţie unificând cazurile omoloage. Astfel, fereastra predefinită şi relaţiile de calcul explicite asociate se vor regăsi în acelaşi caz din instrucţiunea CASE.

6.3. Reprezentaţi grafic tabloul coeficienţilor cu care se înmulţeşte semnalul pentru ferestrele Hanning şi Exponenţial.

6.4. Adăugaţi noi ferestre de ponderare (fereastra Blackman - cazul 5 etc.)

6.5. Aplicaţi funcţia Auto Power Spectrum (APS) atât semnalului iniţial sinusoidal, cât şi semnalului ponderat prin ferestrele Hanning sau Hamming, şi vizualizaţi rezultatele grafic comparativ într-o nouă fereastră în PF (Fig. 7). Graficul prin aplicarea funcţiei APS (Fig. 8) va avea abscisa în domeniul frecvenţă; setaţi ordonata în scară logaritmică şi modificaţi intrarea *frequency* (Fig. 7) a semnalului sinusoidal. Se observă diminuarea valorilor spectrului în jurul unicului vârf spectral al semnalului sinusoidal, după aplicarea ferestrei de ponderare, comparativ cu spectrul semnalului iniţial. Auto Power Spectrum (Fig. 8) calculează puterea spectrală unilaterală plecând de la Sxx (puterea spectrală bilaterală, Fig. 9).

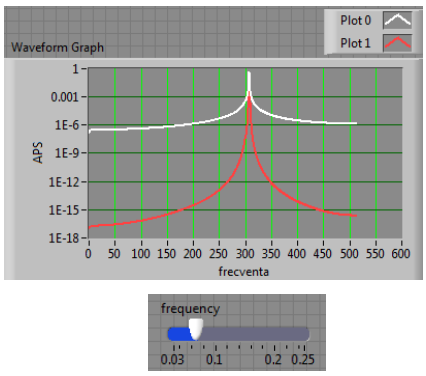


Fig. 7

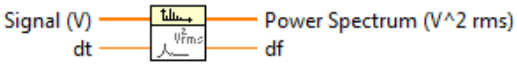


Fig. 8

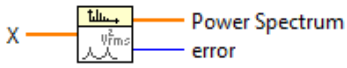


Fig. 9

XI. Programarea pe evenimente

Instrucţiunea Event Structure conţine unul sau mai multe cazuri (asemănător cu Case structure), fiecare caz fiind executat la apariţia unui eveniment sau grup de evenimente. Evenimente pot fi: schimbarea valorii unui control, intrarea sau ieşirea pointerului mouse într-o/dintr-o zona de pe Panoul Frontal, apăsarea unui buton mouse, închiderea/schimbarea mărimii unei ferestre, apăsarea unei taste, timeout etc. În general Event Structure se plasează într-un ciclu While.

1. PF interactiv - Ciclul While (polling) versus Event Structure

Efectuăm produsul a două controale numerice A şi B cu observarea permanentă a modificării valorilor de intrare şi a produsului valorilor.

1.1. În prima variantă produsul este plasat în corpul ciclului While (Fig. 2). Când nu se modifică valorile A sau B ciclarea continuă consumând resurse de calcul. Observaţi contorul *i* în PF din Figura 1.

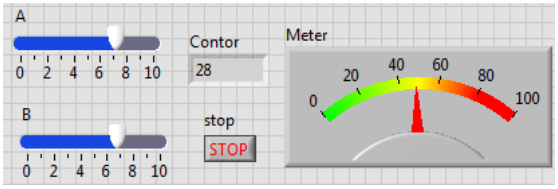


Fig. 1

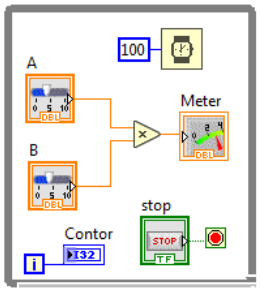


Fig. 2

1.2. Varianta de programare pe evenimente - Event Structure

În aplicaţia precedentă păstrăm Ciclul While, dar produsul valorilor numerice este plasat în primul caz *[0]"A","B":Value Change* al structurii Event, creat prin modificarea cazului *0 Timeout* cu comanda Edit Events Handled by This Case.

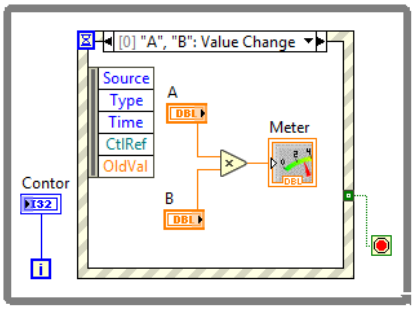


Fig. 3

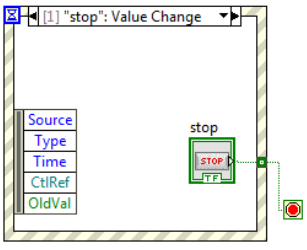


Fig. 4

Se deschide o fereastră de configurare/editare evenimente (Fig. 5) folosind comanda Edit Events Handled by This Case şi se adaugă două evenimente de tip Value Change, sursa evenimentelor fiind controalele A şi B (Fig. 3).

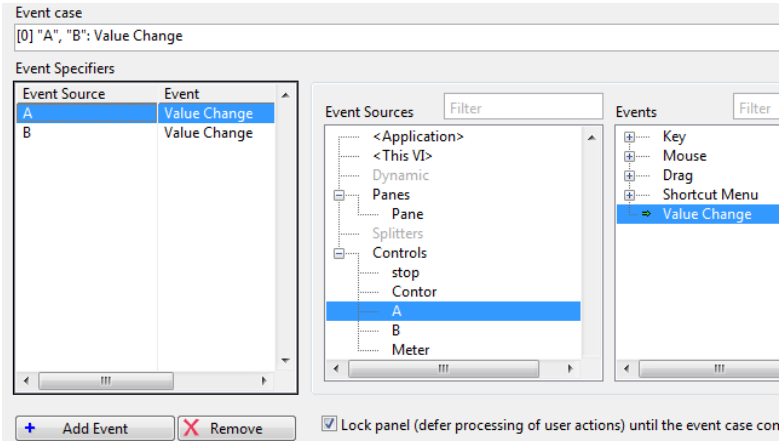


Fig. 5

Oprirea ciclului şi a aplicaţiei este posibilă dacă se introduce un al doilea caz de tip Value Change în structura Event (Fig. 4) care va conţine terminalul butonului de STOP. Acest caz se execută la apăsarea butonului STOP în PF, iar la ieşirea din caz (după tratare Event) valoarea True a butonului opreşte ciclul While.

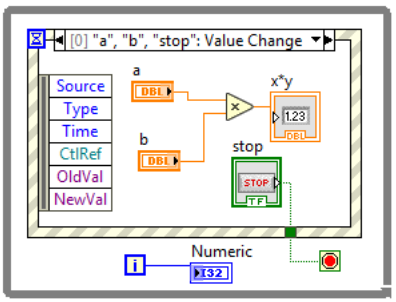


Fig. 6

La execuţie se observă faptul că ciclarea este suspendată (contorul nu creşte) când nu sunt înregistrate evenimente de tipul modificării valorii controalelor A sau B. Când schimbăm din PF valorile controalelor A sau B, contorul ciclului creşte (incrementează), iar înmulţirea se efectuează imediat. Se pot trata cele trei evenimente de tip Value Change într-un singur caz (Fig. 6).

1.3. Cazul Timeout al structurii Event

1.3.1. Aplicaţia din Figura 7 conţine două cazuri: cazul Timeout şi cazul tip Value Change pentru butonul de STOP al ciclului.

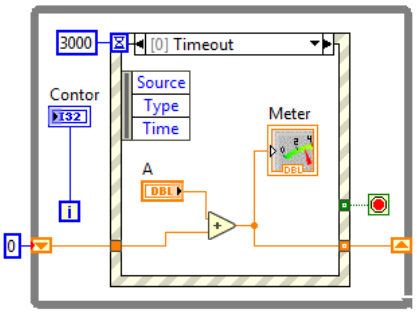


Fig. 7

Se conectează constanta numerică 3000 la clepsidra din colţul stânga-sus a structurii. La fiecare 3000 ms de aşteptare, dacă nu apare un eveniment, se execută automat cazul 'Timeout' în care se adună valoarea lui A la registrul Shift şi se afişează. Dacă schimbăm valoarea controlului A din PF nu se întâmplă nimic imediat, doar după

împlinirea celor 3000 ms se execută cazul şi se adună valoarea schimbată a lui A (schimbarea lui A nu este considerată ca fiind eveniment declanşator).

1.3.2. Adăugaţi un caz nou pentru sesizarea schimbării valorii lui A şi actualizarea imediată a sumei (variabile locale pentru A şi Meter).

1.3.3. Includeţi în cazul Timeout şi evenimentul 'schimbare valoare A' (Value Chance) fără a avea un caz separat pentru schimbarea lui A. De asemenea adăugaţi şi evenimentul de schimbare a valorii butonului Stop în cazul Timeout, astfel va exista un singur caz în aplicaţie (Fig. 8). La apăsarea butonului de Stop are loc incrementarea registrului cu valoarea lui A, afişarea în Meter şi oprirea ciclului.

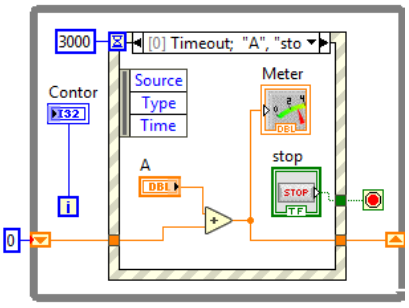


Fig. 8

2. Eveniment static - schimbarea valorii unui control şir de caractere

Se include Event Structure în diagrama bloc. Implicit există cazul '[0] Timeout'. Se adaugă cu comanda 'Add Event Case' un nou caz pentru tratarea unui eveniment de tipul 'Value Change' - caz înregistrat static (Static Event Registration). Se va urmări schimbarea valorii unui control String. La apariţia evenimentului se execută codul din cazul asociat vizualizat în Figura 9.

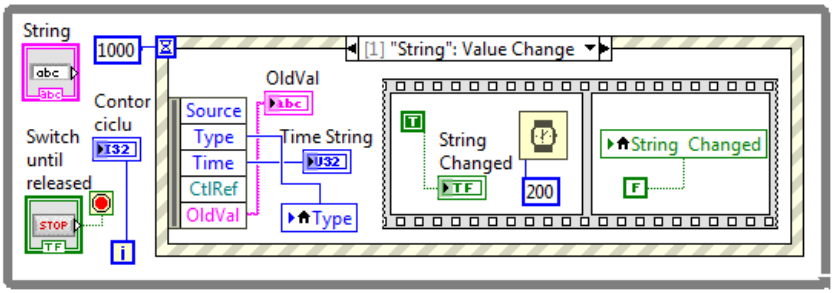


Fig. 9

Lansăm aplicaţia. Se execută repetitiv ciclul While pe cazul '[0] Timeout' (Fig. 10) de fiecare dată după aşteptarea unui număr de 1000 ms. Evenimentul are loc dacă se tastează un şir de caractere sau se modifică un şir existent (fară Enter) în controlul String, urmat de click-mouse pe suprafaţa PF. Acest eveniment declanşează execuţia codului din caz [1]:

a) se afişează în indicatorul OldVal vechiul şir (Fig. 11) care a fost modificat,

b) se execută secvenţa indice *0* din structura Sequence (Fig. 9) în care LED-ul indicator String Changed se luminează (T) pentru 200 ms după care, în secvenţa indice *1*, LED-ul este stins (F) şi se iese din cazul [1] al Event Structure.

Aşteptarea apariţiei unui eveniment (schimbare şir caractere) durează 1000 ms. Dacă nu s-a schimbat şirul se execută evenimentul *Timeout* după care se

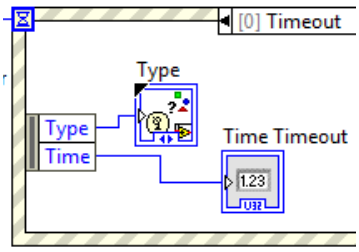


Fig. 10

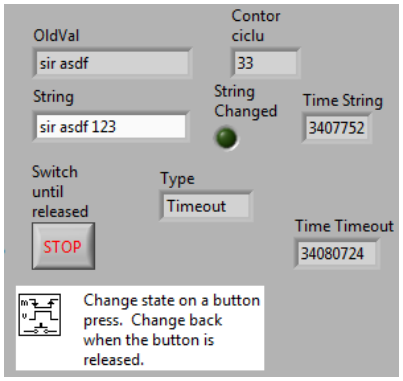


Fig. 11

părăseşte Structura event şi se repetă ciclul While. 'Contor ciclu' incrementează la fiecare secundă (1000 ms) dacă nu apare evenimentul (modificarea şirului).

La execuţia fiecărui caz se afişează în indicatorul unic, realizat prin variabila locală asociată indicatorului Type, mesajul Timeout sau Value Change indicând tipul evenimentului care a avut loc. Indicatoarele Time Timeout şi Time String indică momentul apariţiei evenimentului respectiv. Aceste două indicatoare pot fi tratate într-un singur indicator similar cu indicatorul Type.

Butonul de STOP nu este tratat ca un eveniment, motiv pentru care oprirea ciclului While se realizează prin apăsarea mai de durată a butonului STOP pentru a fi 'sesizată apăsarea' şi transmisă valoarea True la terminalul ciclului.

3. Schimbare parametri Sine Patern.vi - grafic sinus - buton dialog

La schimbarea unui parametru din cei trei (amplitudine, fază, cicluri) a funcţiei Sine Patern.vi (Fig. 12) se actualizează graficul sinus din Waveform Chart (Fig. 13) şi incrementează contorul ciclului While.

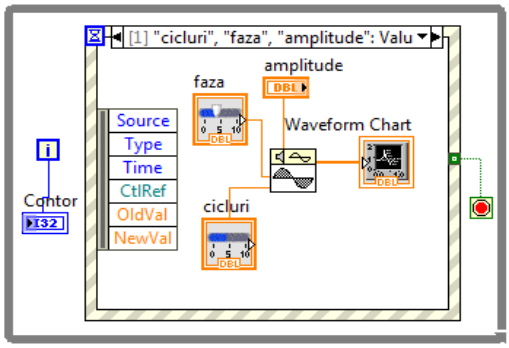


Fig. 12

La apăsarea butonului Stop, acest eveniment declanşează cazul [0]. În acest caz, butonul Stop nu este legat direct la terminalul ciclului While (Fig. 14). Se alege mai întâi între mesajele Opresc (T) şi Continuă (F) asociate funcţiei Two Button Dialog.vi şi astfel se trimite una dintre valorile logice T respectiv F la terminalul *Stop if True* al ciclului.

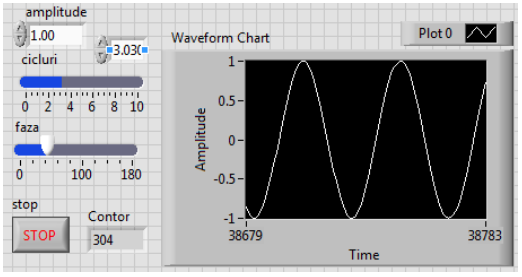


Fig. 13

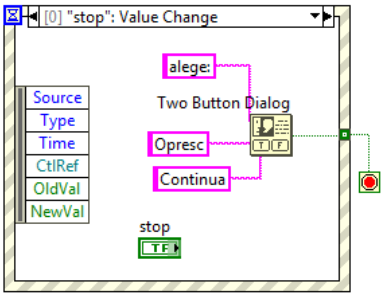


Fig. 14

4. Evenimente Mouse Enter, Mouse Leave şi Value Change

Aplicaţia din Figura 15 generează un tablou de *Nr val random* valori aleatoare, trasează graficul, trece graficul în mod Blink şi revine, afişează (şi apoi şterge) în indicatorul MESAJE etichetele unor indicatoare şi controale când se trece cu pointerul mouse peste zona alocată lor din Panoul Frontal.

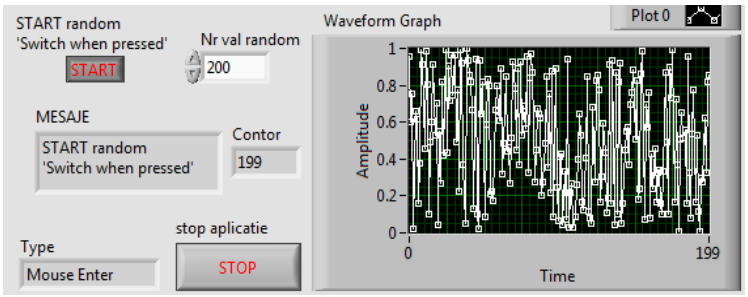


Fig. 15

În continuare sunt prezentate evenimentele şi cazurile asociate lor:

[0] La evenimentul de tip apăsare buton START are loc schimbarea valorii logice T→F sau F→T şi, de asemenea, se generează un număr de valori aleatoare (Nr val random=200). Este afişat semnalul de valori aleatoare în fereastra grafică (Fig. 15 şi 16); butonul nu este conectat în diagramă fiind doar declanşatorul evenimentului.

[1] Când pointerul *mouse* intră în zona indicatorului grafic Waveform Graph, graficul trece în mod *Blink*; o constantă şir de caractere (mesaj) se afişează în indicatorul MESAJE (Fig. 17); evenimentul este de tip Mouse Enter.

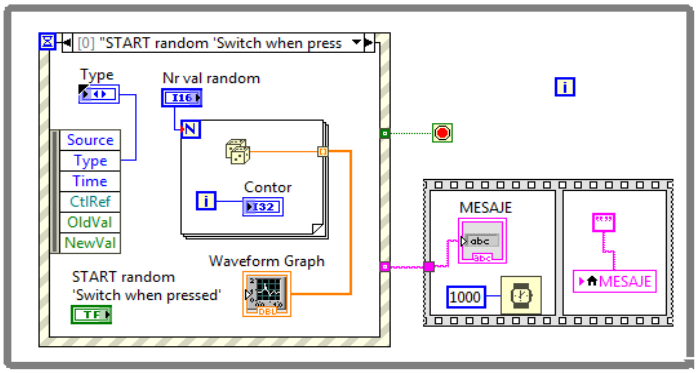


Fig. 16

[2] Când pointerul *mouse* iese (eveniment Mouse Leave) din zona indicatorului Waveform Graph se opreşte modul *Blink* al graficului şi constanta şir de caractere *'Blink se opreste cand iesi din graph'* trece în indicatorul MESAJE (Fig. 17).

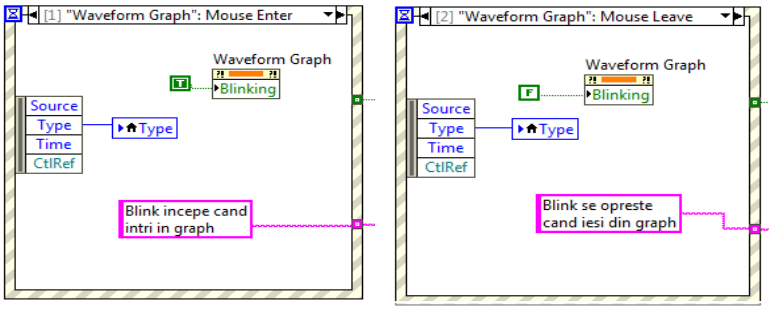


Fig. 17

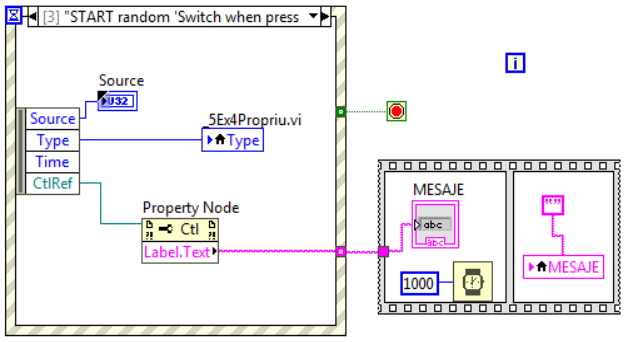


Fig. 18

[3] Când pointerul Mouse intră în zona controlului *START random 'Switch when pressed* ' sau în zona controlului '*stop aplicatie*' sau '*Nr val random*', se afişează

eticheta (şir caractere) acestor controale (CtlRef→ Property Node→ Label.Text) în indicatorul şir de caractere MESAJE pentru o secundă (Fig. 18). Evenimentul returnează CtlRef, iar PropertyNode citeşte o *proprietate a unei referinţe* care poate fi obiect, clasă, metodă, VI (aici Property for Control Class).

[4] La apăsarea butonului STOP (Fig. 19) are loc un eveniment de tip Value Change, se opreşte ciclul While şi aplicaţia curentă.

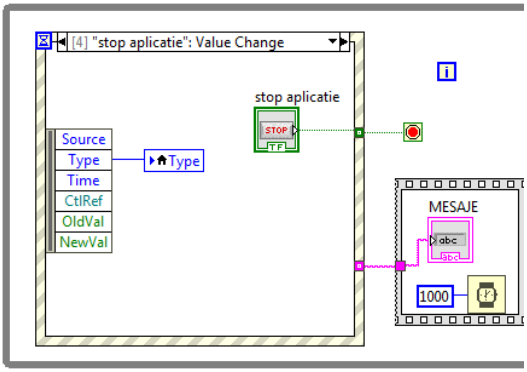


Fig. 19

Pentru trei cazuri [1], [2], [3] ale instrucţiunii Event Structure secvenţa indice 0 (din structura Sequence) afişează mesajul pentru 1000 ms după care, în secvenţa următoare (indice 1), mesajul este şters prin afişare şir vid (Fig. 18). În Figura 20 sunt listate cele cinci cazuri.

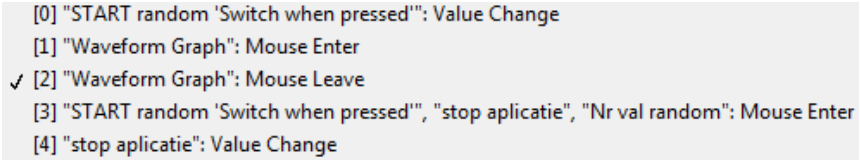


Fig. 20

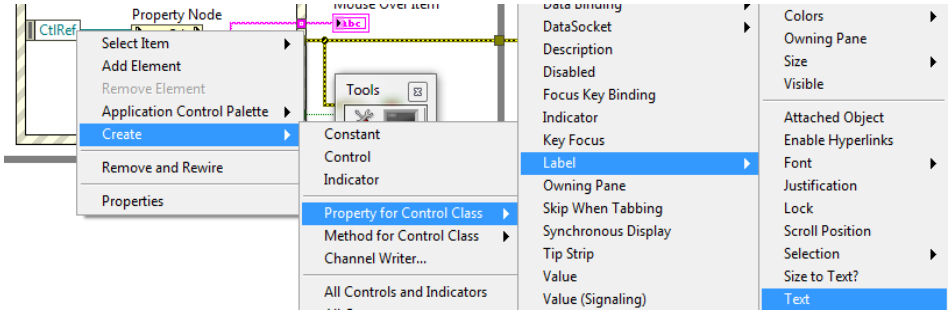


Fig. 21

Indicatorul Type (unic prin folosirea variabilei locale) indică tipul fiecărui eveniment (Value Change, Mouse Enter, Mouse Leave) care are loc.

Pentru setarea PropertyNode (cazul [3]) sunt folosite meniurile din Figura 21, unde iniţial se poziţionează pointerul *mouse* pe CtlRef al nodului Event Data (plasat interior contur stânga), urmat de *click* buton dreapta al mouse/Create etc.

5. Filtrare evenimente (Filter Event)

Anumite evenimente pot fi anulate automat sau este consultat operatorul înaintea efectuării acţiunii. Astfel sunt evenimentele de tip *Panel Close?* şi *Menu Activation?* care vor fi exemplificate în continuare.

Să presupunem că a apărut un eveniment şi anume *click mouse* pentru a închide Panou Frontal (PF). Acţiunea de închidere efectivă a ferestrei poate fi anulată (dacă de exemplu rulează o aplicaţie) prin tratarea evenimentului [0] Panel Close? (Fig. 22). Acţiunile pot fi anulate sau modificate.

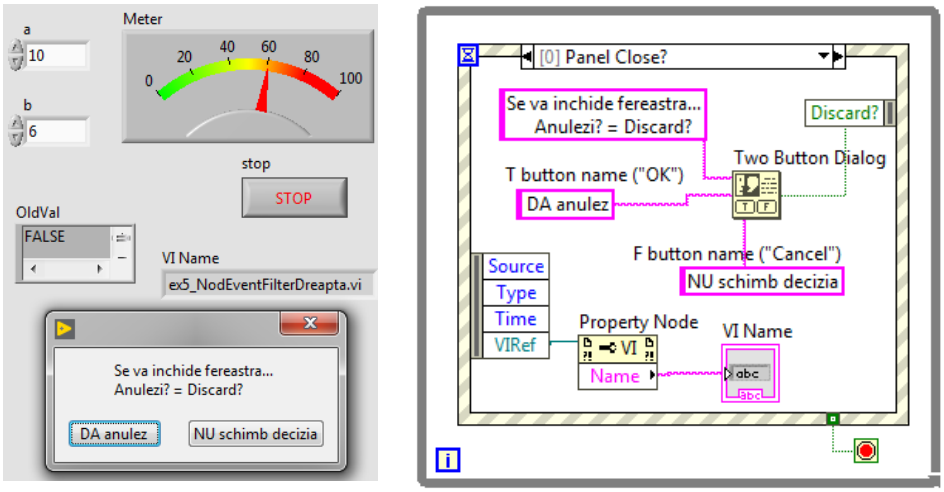


Fig. 22

Se reia aplicaţia din Figura 3, pentru înmulţirea a două valori reale şi indicarea produsului cu un Meter; se adaugă două evenimente tratate în cazurile [0] (Fig. 23) şi [2] (Fig. 24). Pentru început analizăm cazul [0] Panel Close? de tip filtrare evenimente.

Se presupune că în timpul rulării aplicaţiei curente *nume.vi*, cu mouse-ul se acţionează pentru închiderea ferestrei PF a aplicaţiei curente *nume.vi*. Prin câmpul *Discard?* al nodului Event Filter (dreapta) se poate anula comanda de închidere a ferestrei dacă se alege butonul *DA anulez* sau se acceptă decizia de închidere a aplicaţiei dacă se alege butonul *NU schimb decizia*. Câmpul *Discard?* apare automat pe marginea dreaptă a subdiagramei (cazului) când se selectează evenimentul Panel Close? în fereastra de configurare.

Prin *VIRef* din nodul Event Data (stânga) se extrage numele aplicaţiei curente *nume.vi* (Fig. 23). Numele aplicaţiei (a cărui Panou Frontal a fost acţionat pentru închidere) se afişează în indicatorul şir de caractere VI Name (Fig. 22).

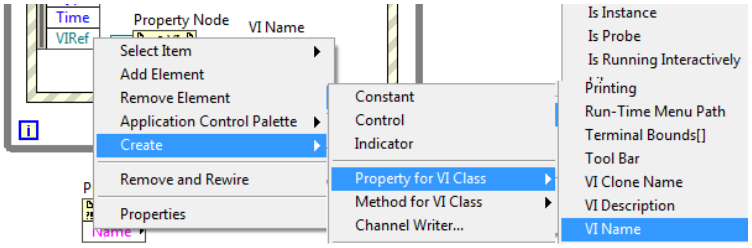


Fig. 23

Tratarea acestui *Event case* se observă în fereastra de configurare (Fig. 24), unde sursa evenimetului (Event Sources) este: *<This VI>*, iar tipul evenimentului (*Events)* este: →*Panel Close?*.

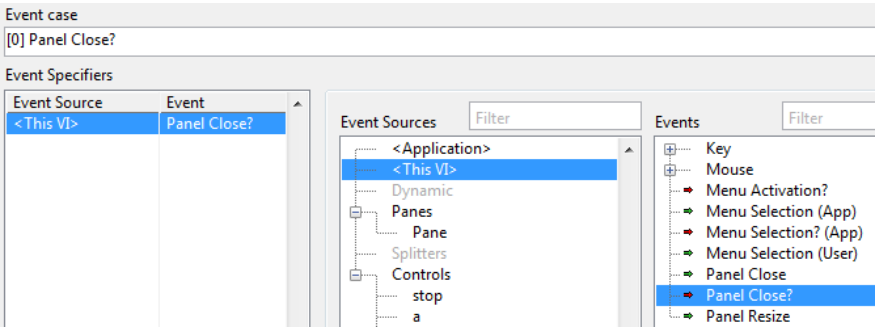


Fig. 24

În continuare este observat cazul [2] din Figura 25:

[2] Filter Event: Menu Activation?

În mod similar se creează cazul de tratare a activării meniului Panoului Frontal (PF) pentru a selecta o comandă în timpul rulării programului. Se poate anula sau continua acţiunea demarată răspunzând la fereastra dialog care apare.

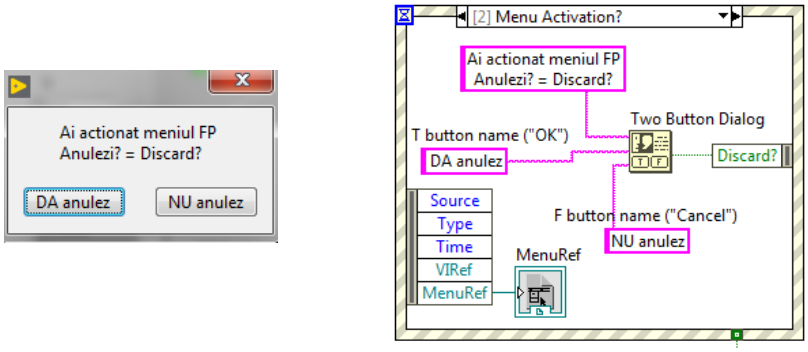


Fig. 25

[1] "stop","a","b": Value Change → eveniment de notificare

1. Se modifică valoarea unui control din lista: a, b sau stop. Prin articolul *OldVal* din nodul Event Data se afişează valoarea veche a controlului modificat. Indicatorul OldVal (Fig. 26) este de tip *variant* (tip generic pentru date simple şi structurate) fiind capabil să afişeze atât valori numerice (a, b) cât şi logice (False).

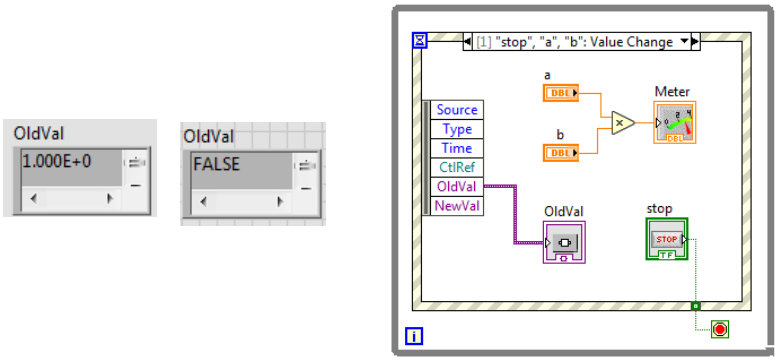


Fig. 26

6. Evenimente dinamice declanşate prin program din diagramă

În aplicaţia propusă, pentru început sunt calculate 300 de valori ale unei funcţii în intervalul [0, 15] şi este reprezentat grafic tabloul valorilor funcţiei (Fig. 27). În continuare sunt configurate două evenimente dinamice cu declanşare pe valorile funcţiei astfel: dacă valorile funcţiei intersectează abscisa are loc generarea unui eveniment logic, iar pentru fiecare valoare a funcţiei în intervalul [1, 3] este generat câte un eveniment dinamic numeric. La generarea evenimentelor şi tratarea lor coparticipă funcţii din paleta Dialog&User Interface/Events. Astfel, cele două

funcţii Create User Event (Fig. 30) prezintă fiecare ieşirea *user event out* prin care se leagă la funcţiile Generate User Event şi Register For Events (Fig. 27).

Evenimentul dinamic aşteptat trebuie să aibă *tip* şi *nume*. Prin intrarea *user event data type* a funcţiei Create User Event se stabileşte numele evenimentului, în cazul aplicaţiei analizate *Eve logic* (Eveniment Logic) dat de utilizator şi tipul de dată al evenimentului este cel *logic* (valoarea True primită, nu are importanţă). Similar evenumentul numeric (al aplicaţiei analizate) va avea numele *Eve numeric* şi tipul *numeric*.

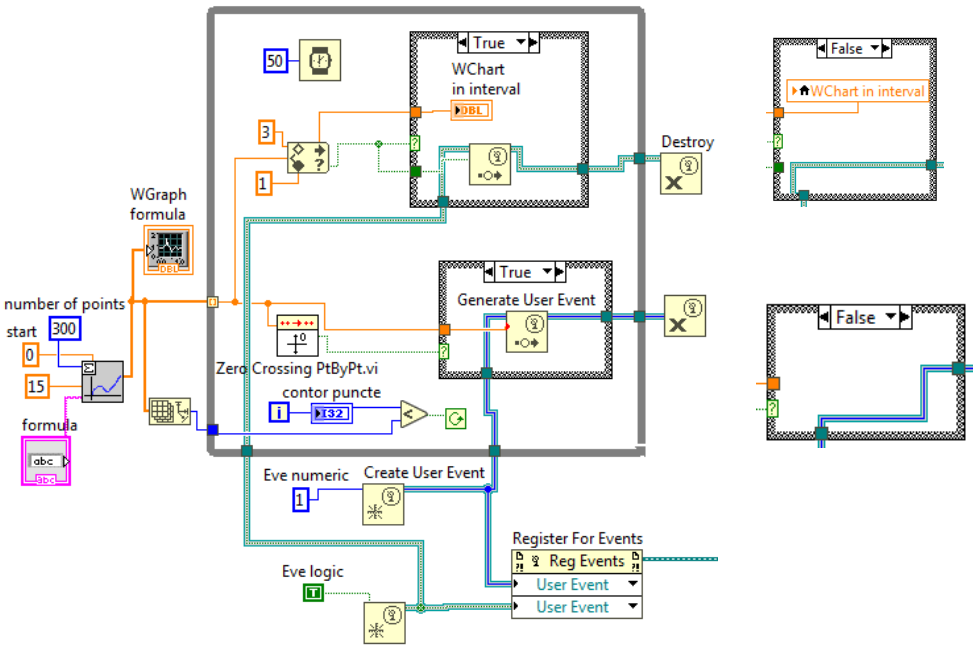


Fig. 27

Când funcţia *Zero Crossing PtByPt.vi* sesizează o trecere prin zero a valorilor, returnează valoarea logică True, iar în cazul True al structurii Case se execută funcţia Generate User Event.vi declanşând astfel cazul [2] *<Eve logic>: User Event* al structurii Event Case (Fig. 28). Similar pentru o valoare în intervalul [1, 3] a semnalului, funcţia *In range and Coerce.vi* returnează True şi este executat *Generate User Event.vi* declanşând astfel cazul [1] *<Eve numeric>: User Event* al structurii Event Case (Fig. 29).

Funcţia Create User Event (Fig. 30) returnează o referinţă la un User Event.

Funcţia Register For Events înregistrează evenimentul în cadrul structurii Event (aceasta prezintă 3 cazuri în aplicaţia curentă).

Funcţia Generate User Event trimite evenimentul la toate structurile de tip Event Structure care au fost înregistrate pentru acel eveniment sau care conţin *un caz* pentru tratarea acestui eveniment.

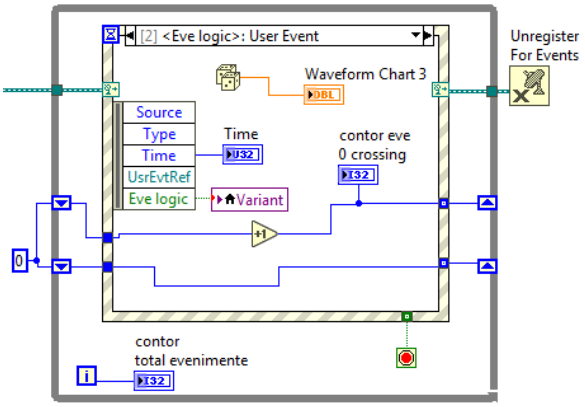


Fig. 28

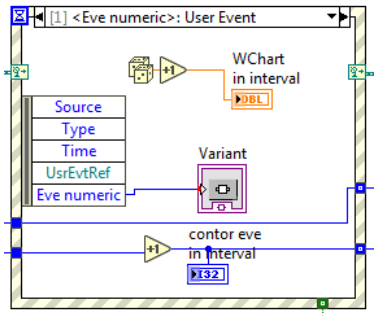


Fig. 29

Se va crea în diagramă o structură Event cu 3 cazuri.

Fig. 30

Event Structure este pusă în ciclul While deoarece după apariţia unui eveniment structura Event Structure se consumă (termină) şi trebuie din nou executată pentru a trata un nou eveniment.

1. Cazul [0] *Stop contorizare evenimente*: Value Change este un caz înregistrat static (Fig. 31). Butonul de oprire a ciclului care înscrie structura Event are acţiunea mecanică prezentată în Figura 32, poziţia (1,1) în matricea de acţiuni posibile.

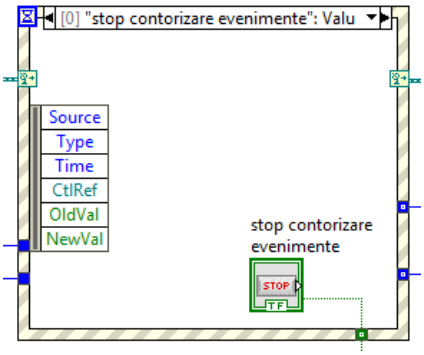


Fig. 31.

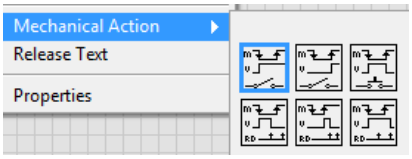


Fig. 32

Pentru a putea adăuga evenimente dinamice se activează (bifează) *Show Dynamic Event Terminal* (Fig. 33), la care se leagă ieşirea *1* a funcţiei *Register For Events*, astfel încât la crearea unui nou caz să apară în fereastra de editare (Fig. 34) posibilitatea de a crea un eveniment dinamic.

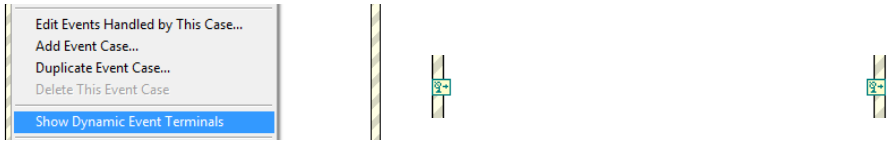


Fig. 33

2. Cu comanda Add Event Case... se creează al doilea caz *[1]<Eve numeric>:User Event*. În fereastra deschisă (Event case, Fig. 34) se selectează Event Sources: Dynamic/.... În cazul apariţiei acestui eveniment se incrementează valoarea din Registrul de transfer cu vizualizare în indicatorul *contor eve in interval* şi se generează un număr aleator care se vizualizează în fereastra *WChart in interval* plasată în PF.

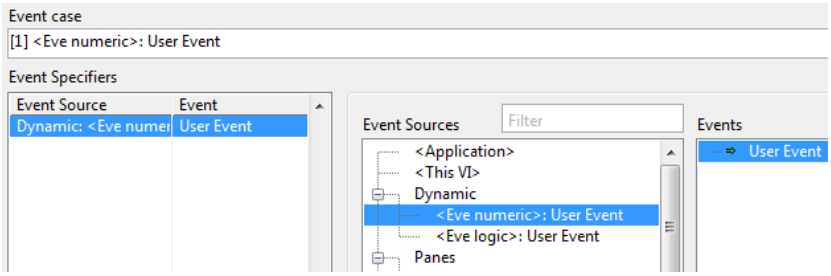


Fig. 34

3. Cu comanda Add Event Case... se creează al treilea caz numit [2]<Eve logic>:User Event. În cazul apariţiei acestui eveniment se incremenetează valoarea din Registrul de transfer cu vizualizare în indicatorul *contor eve 0 crossing* şi se generează un număr aleator cu vizualizare în fereastra Waveform Chart (3) în PF.

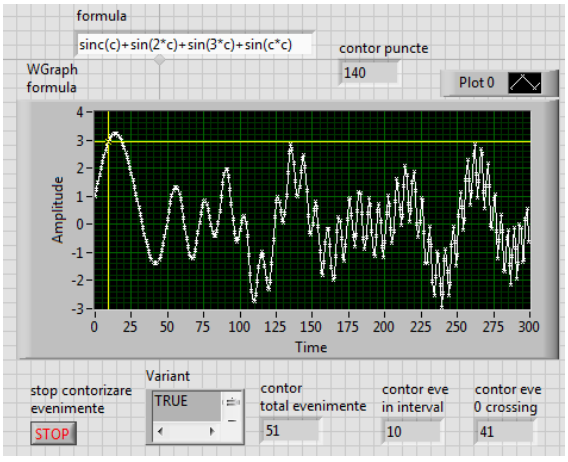


Fig. 35

Observăm continuu în PF egalitatea între numărul total de evenimente care au loc şi suma celor două tipuri de evenimente. În Figura 35, la un moment dat se surprind în timpul rulării 10 evenimente (valori în interval) plus 41 evenimente (0 crossing) rezultând 51 numărul total de evenimente. Numărul total de evenimente este egal cu numărul iteraţiilor ciclului While asociat structurii Event.

Ferestrele grafice *Waveform Chart 3* şi *WChart 2 in interval* vor conţine fiecare atâtea puncte generate aleator câte evenimente au lor din fiecare tip. Pentru ştergerea celor două ferestre (clear history) de la o rulare (execuţie) la alta a aplicaţiei se foloseşte codul grafic din Figura 36, inserat în afara ciclurilor diagramei. Succesiunea de comenzi pentru a avea acces la proprietatea History Data a obiectului fereastră grafică şi iniţializarea acesteia cu tabloul vid de valori întregi este de asemenea abservabilă în Figura 36. Acelaşi efect se obţine prin comanda Data Operations/Clear Chart (sau Clear Graph) de pe suprafaţa ferestrei grafice. Fereastra *WChart 1 in interval* cu terminalul aflat în cazul True din Figura 27 prezintă valorile efective care declanşează evenimentul de tip în valoare în intervalul [1, 3].

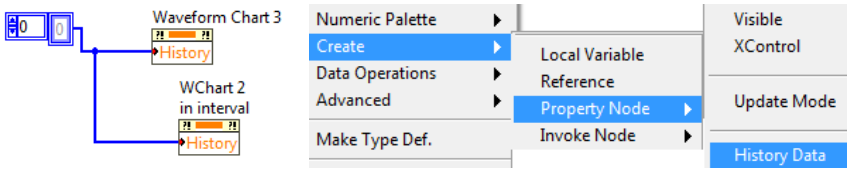


Fig. 36

7. Precizări cu privire la Event Structure

7.1. Aspecte generale

Programarea pe evenimente (Event-Driven Programming) este utilă în special pentru dezvoltarea interfeţelor utilizator (UI). Tradiţional, este folosită tehnica *polling* sau citirea repetată în cadrul unui ciclu a controalelor din Panoul Frontal pentru a sesiza o modificare de valoare, fapt care se traduce în utilizarea de resurse CPU mari. Ciclarea la frecvenţă mai mică face posibilă neglijarea unor evenimente sau a ordinii apariţiei lor. În programarea condusă de evenimente, spre deosebire de programarea procedurală, aplicaţia care rulează sub sistemul de operare (aplicaţia LabVIEW, de exemplu) recepţionează şi tratează corespunzător mesajul primit de la sistemul de operare la producerea unui eveniment. Evenimentele de genul interacţiunii prin tastatură sau mouse sunt identificate de sistemul de operare, este transmis mesajul asociat, care poate fi recepţionat de aplicaţie, urmând să se execute un cod scris special pentru acel eveniment, în cazul de faţă codul aparţine unui caz al Event Structure.

Fluxul firesc al datelor în diagrama unei aplicaţii presupune execuţia unui nod al diagramei la momentul în care nodul are la dispoziţie date la toate intrările conectate. După execuţie, nodul generează (returnează) date care trec prin fire spre nodurile următoare conectate conform cu fluxul de date din diagramă.

Programarea condusă de evenimente permite modificarea execuţiei fireşti la apariţia unui eveniment urmată de tratarea acelui eveniment. Evenimentele pot fi statice sau dinamice. Exemple de evenimente statice sunt: schimbarea valorii unui control, intrarea sau ieşirea pointerului *mouse* într-o/dintr-o zona de pe PF, apăsare

buton mouse, închiderea sau schimbarea mărimii unei ferestre, apăsarea unei taste, timeout etc. Evenimentele dinamice spre deosebire de cele statice prezentate în exemplele anterioare, sunt declanşate prin program din Diagrama bloc, fără intervenţia utilizatorului în PF. Programul construit pe baza structurii Event (Event Structure) aşteaptă apariţia unui eveniment, tratează evenimentul apărut şi revine la aşteptare.

7.2. Descrierea structurii Eveniment

Structura Eveniment (Event Structure) poate avea una sau mai multe subdiagrame sau cazuri şi se plasează într-un ciclu While. Fiecare subdiagramă sau caz gestionează unul sau mai multe evenimente. Cât timp se aşteaptă apariţia unui eveniment nu se consumă resurse de calcul CPU. Evenimentele se pun în coada de aşteptare dacă nu pot fi tratate imediat (de exemplu, un eveniment apare când un alt eveniment se execută). Se pot şterge elemente din coadă, folosind funcţia Flush Event Queue.

Elementele specifice structurii Eveniment sunt prezentate în Figura 37 (preluată din Help LabVIEW) în care se observă un eveniment de tip Key Down? Astfel, observăm:

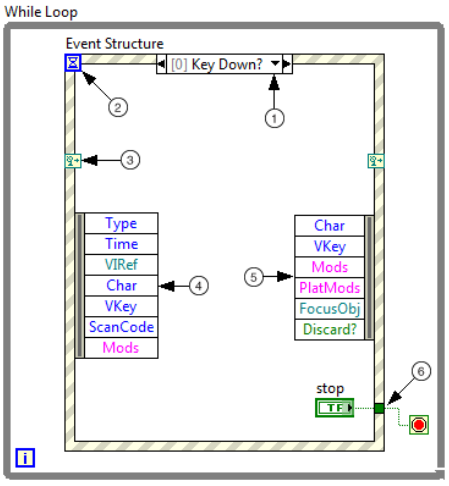


Fig. 37

1. selectorul de evenimente - acesta conţine eticheta sau antetul cazului şi descrie tipul evenimentelor urmărite prin acel caz.

2. Terminalul Timeout (clepsidra) specifică numărul de milisecunde de aşteptare după care se execută cazul Timeout; dacă valoarea conectată la clepsidră este -1 rezultă o aşteptare nelimitată. În cazul în care se conectează la clepsidră o valoare întreagă (în msec) trebuie să existe şi un case Timeout, altfel apare o eroare.

3. Setare Terminal pentru tratarea unui eveniment dinamic; se bifează *Show Dynamic Event Terminal* (click dreapta mouse pe cadrul Event Structure) la care se leagă ieşirea *1* a funcţiei *Register For Events* astfel încât la crearea unui nou caz să apară în fereastra de editare posibilitatea de a crea un eveniment dinamic.

4. Nodul Event Data este plasat în interiorul cadrului ataşat pe latura stângă. Acesta permite acces la datele pe care LabVIEW le returnează la apariţia unui eveniment, fiind date despre evenimentul curent. Asemănător cu Unbundle By Name, se poate redimensiona numărul de terminale din nod pe verticală şi se poate selecta articolul dorit. Nodul Event Data permite acces la câmpuri cum sunt: Type

şi Time care sunt elemente comune tuturor evenimentelor sau elemente Char and VKey, acestea fiind specifice evenimentului apărut.

5. Nodul Event Filter (plasat pe latura dreaptă a cadrului) identifică date ale evenimentului pe care le poţi modifica înainte ca interfaţa utilizator (UI) să poată prelucra acea dată. Nodul apare numai în cazuri de tip *filter events* (având caracterul ? la urmă). Dacă se doreşte schimbarea *event data*, se poate lega la dreapta şi modifica articolul cu data în cauză sau lega data provenită din Event Data Node la Event Filter Node. Astfel, se pot schimba date ale evenimentului cu noi valori şi anume cele dorite de programator. De exemplu, pentru eliminarea/anularea unui eveniment deja cerut se leagă valoarea True la terminalul *Discard?*. Dacă nu se conectează o valoare la un terminal de date aparţinând nodului Event Filter, acea dată îşi păstrează valoarea.

6. Event Structure suportă tuneluri de ieşire, dar nu sunt necesare legături la tunel din fiecare caz - aşa cum în general este necesar la Structura Case.

8. Probleme propuse

8.1. Folosiţi Event Structure pentru umărirea unui eveniment de tip mişcare pointer mouse (Mouse Move) în PF (Fig. 38). Sunt returnate (în case) şi stocate coordonatele mouse-ului în timp ce este deplasat. După oprirea ciclului While (care conţine Event Structure) să se traseze mişcarea efectuată de pointerul *mouse*, conform cu Figura 39. Pe un alt caz (subdiagramă) să se anuleze automat încercarea de a închide sau redimensiona Panoul Frontal.

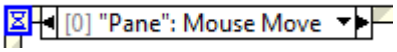


Fig. 38

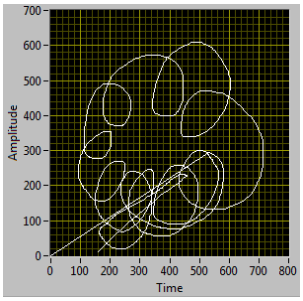


Fig. 39

8.2. Salvaţi într-un tablou de structuri timpul apariţiei şi tipul fiecărui eveniment care a avut loc pe durata rulării unei aplicaţii.

8.3. Se va ilumina un LED verde (indicator boolean) la fiecare 4 secunde. Se contorizează şi afişează, de asemenea, timpul total la fiecare iluminare a LED-ului.

XII. Tipul de dată waveform - generare, salvare, concatenare

1. Tipul de dată waveform, funcţia Build Waveform

1.1. Funcţiile de generare/manipulare a datelor de tip waveform (formă de undă) se găsesc în Paleta Functions/Waveform. Tipul Waveform este o dată de tip structură, se asociază cu un semnal măsurat/generat şi se compune din trei câmpuri: *t0* momentul de start, *dt* spaţierea între valori sau eşantioane succesive şi *Y* tabloul de valori/eşantioane. În aplicaţia din Figura 1 cu funcţia Build Waveform.vi s-a creat o formă de undă analogică. Funcţia Get Date/Time In Seconds.vi obţine data

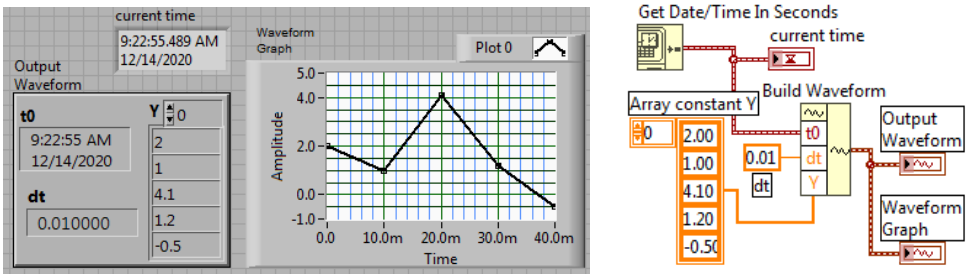


Fig. 1

şi timpul curent. Se poate folosi, de asemenea, funcţia To Time Stamp (Fig. 2) pentru a obţine o valoare pentru t0. În graficul din Panoul Frontal observăm spaţierea de 10 milisecunde între eşantioane succesive şi durata semnalului de 40 milisecunde.

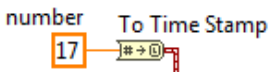


Fig. 2

1.2. Conectaţi forma de undă generată, mai sus, la prima intrare (waveform) a unei noi funcţii Build Waveform (Fig. 3) şi modificaţi spaţierea (dt=0.03) dintre valorile formei de undă. Observaţi în grafic durata modificată a formei de undă.

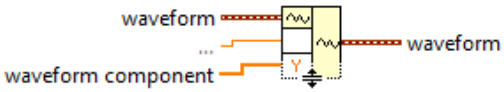


Fig. 3

2. Operaţii cu forme de undă

Două forme de undă se pot aduna, scădea, înmulţi sau împărţi; cele două trebuie să aiba aceeaşi spaţiere *dt* între eşantioane**.**

În diagrama din Figura 4 operaţia care se aplică formelor de undă este aleasă printr-un control de tip Enum (Fig. 5); acesta selectează un caz dintre cele patru cazuri ale instrucţiunii Case. În Panoul Frontal observăm forma de undă Sine Waveform peste care s-a suprapus forma de undă Gaussian White Noise Waveform.vi. Ambele forme de undă prezintă acelaşi Sampling Info (implicit):

Fs=1000 (eşantioane pe secundă) rezultând acelaşi *dt*=0.001 iar numărul total de eşantioane în semnal este #s=1000, rezultând o durată de o secundă a formei de undă. Frecvenţa semnalului sinusoidal generat este implicit de 10 Hz.

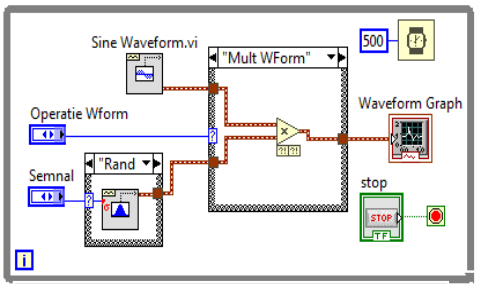


Fig. 4

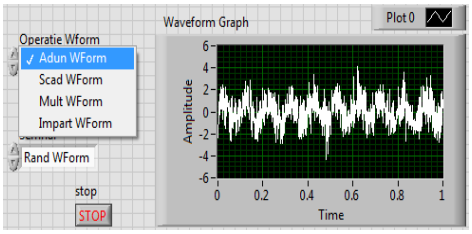


Fig. 5

Funcţia Sine Waveform.vi prezintă intrările observabile în Figura 6. Intrarea *sampling info* (de tip structură) a funcţiei Sine Waveform.vi conţine componentele Fs (implicit Fs=1000 eşantioane/secundă) şi #s (implicit #s=1000 eşantioane); astfel, semnalul sau forma de undă are durata de o secundă, conţine 1000 eşantioane şi dt=0.001 secunde.

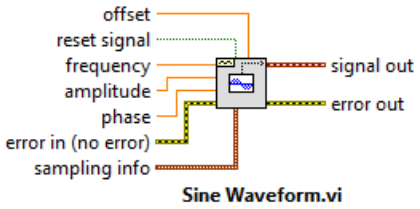


Fig. 6

3. Tablou de forme de undă, salvare în fişiere şi vizualizare Putere spectrală

3.1. Se generează la fiecare iteraţie un semnal formă de undă sinusoidal (Sine Waveform.vi) peste care se suprapune perturbaţia sau zgomotul dat de forma de undă Uniform White Noise Waveform.vi (Fig. 10).

Formele de undă cu zgomot ataşat generate la fiecare iteraţie se adună, iar la ieşirea din ciclu (după apăsarea butonului Stop) se mediază aritmetic. Cazul indice 0 al structurii Case asigură iniţializarea registrului Shift cu prima formă de undă generată, iar la iteraţiile următoare se însumează forma de undă curentă cu suma formelor de undă precedente puse la dispoziţie de acelaşi registru de transfer.

Sunt salvate forme de undă în fişiere la fiecare iteraţie astfel: în format text (f2.txt - coduri ASCII) apelând *Export Waveforms To Spreadsheet File (2D).vi* şi în format *binar* (f1.bin) apelând *Write Waveforms to File.vi*. Fişierele de tip 'Spreadsheet' conţin datele în celule aranjate pe linii şi coloane. O celulă poate fi de tip şir de caractere, valori numerice etc. (de exemplu, fişierele Excel.xlsx).

Intrarea *sampling info* a funcţiei Sine Waveform.vi conţine componentele Fs şi #s cu valori implicite, astfel semnalul durează o secundă, conţine 1000 eşantioane şi dt=0.001 secunde. În exemplul curent modificaţi Fs=100, #s=100.

pentru ambele forme de undă Sine Waveform.vi şi Uniform White Noise Waveform.vi

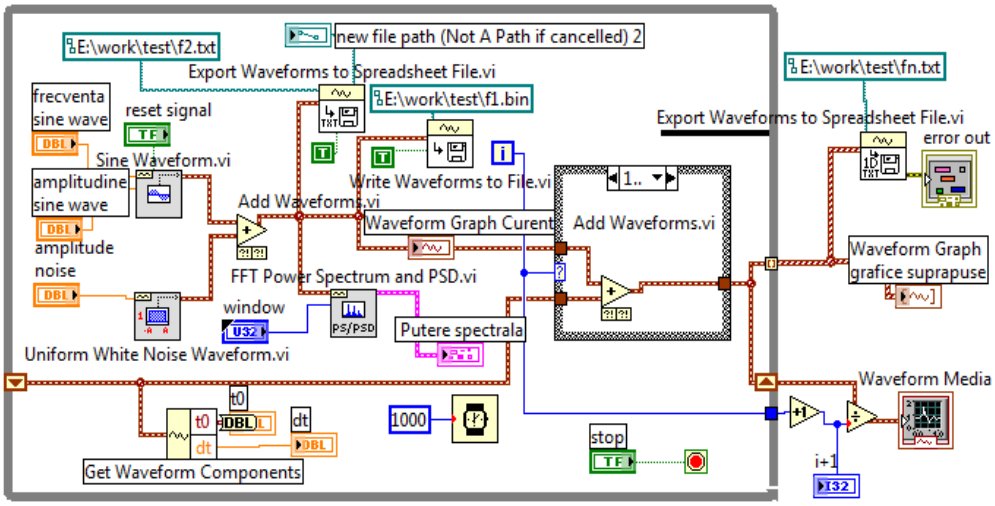


Fig. 7

3.2. Adăugaţi FFT Power Spectrum şi PSD.vi din paleta Waveform Measurements şi vizualizaţi puterea spectrală (PS) a formei de undă generate la iteraţia curentă. Observaţi translatarea stânga-dreapta (pe abscisa frecvenţă) a vârfului puterii spectrale a semnalului când se modifică frecvenţa semnalului generat de Sine Waveform.vi. Folosiţi un cursor pentru citirea vârfului PS. Abscisa vârfului este egală cu frecvenţa semnalului Sine Waveform.vi generat.

3.3. După oprirea ciclului While, într-un indicator Waveform Graph este reprezentat grafic tabloul de forme de undă obţinut la tunelul de ieşire cu indexare conţinând grafice suprapune (Fig. 9) şi în alt indicator grafic este vizualizată media formelor de undă generate.

waveform	[0]	[1]	[2]	[3]
t0	1/1/1904 03:00:00.000000		1/1/1904 03:00:01.000000	1/1/1904 03:00:02.000000
delta t	0.001000	0.001000	0.001000	0.001000

time	Y[0]	Y[1]	Y[2]	Y[3]
1/1/1904 03:00:00.000000	1.170654E-1	1.103271E-1	1.626221E-1	7.844238E-2
1/1/1904 03:00:00.001000	2.188086E-1	5.486205E-2	8.740837E-3	1.155914E-2
1/1/1904 03:00:00.002000	3.132239E-1	1.694494E-1	2.224524E-1	1.533484E-1
1/1/1904 03:00:00.003000	3.555332E-1	2.795871E-1	1.281131E-1	2.006427E-1
1/1/1904 03:00:00.004000	2.628989E-1	9.076752E-2	2.347434E-1	2.999075E-1
1/1/1904 03:00:00.005000	4.894003E-1	2.610495E-1	3.725454E-1	4.442968E-1
1/1/1904 03:00:00.006000	5.251802E-1	5.397310E-1	2.647432E-1	2.460237E-1

Fig. 8

Tabloul formelor de undă este salvat în fişierul text *fn.txt*. Stabiliţi calea potrivită pentru salvare în fişiere conform structurii proprii de directoare. Tabloul

de structuri conţine atâtea forme de undă câte iteraţii au avut loc. Acest fişier se poate deschide în Excel unde vom observa (Figura 8) pe fiecare coloană câte o formă de undă din tabloul de forme de undă. Sunt patru forme de undă; primele 3 linii ale fişierului conţin 4 valori t0, 4 valori delta t =0.001000 urmate de 5 coloane: timp, Y[0], Y[1], Y[2], Y[3].

Observaţi efectul variaţiei amplitudinii semnalului Uniform White Noise Waveform.vi în intervalul 0 - 0.5.

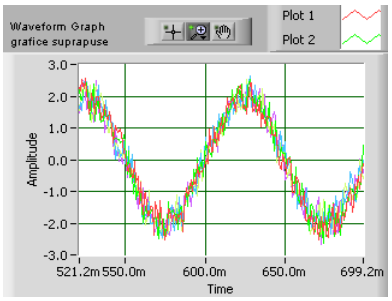


Fig. 9

4. Concatenare forme de undă, controlul Reset Signal

Plecând de la aplicaţia precedentă se vor concatena, în corpul ciclului while, formele de undă generate succesiv. Observaţi *punctele de concatenare* pentru rularea aplicaţiei precedente cu intrarea Reset Signal = T şi pentru Reset Signal = F (Fig. 6).

Scrieţi cod (Fig. 10) pentru concatenare (în ciclul While) de semnale generate de Sine Waveform.vi (Noise=0), Sampling info: Fs=#s =100. Sunt concatenate (apelând Append Waveforms.vi) formele de undă generate la câte trei iteraţii succesive (0, 1, 2), (1, 2, 3) etc. (adăugarea fiecărui semnal se face la urmă) şi vizualizare grafică. Frecvenţa (Hz) nu este număr întreg. Cazul 0..2 (exterior) asigură concatenarea primelor trei forme de undă într-un singur tronson în colaborare cu cazurile *0* şi *1,2, Default* (interior). Toate cazurile sunt comandate de contorul i al ciclului While.

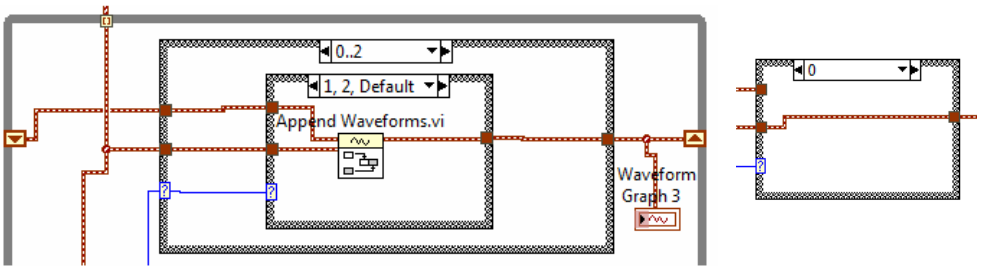


Fig. 10

Pentru iteraţiile următoare (Cazul 3..) s-a ales varianta de a extrage cele 300 de eşantioane cu funcţia *Get Waveform Subset.vi* (Fig. 11) folosind intrările *start* şi *duration* pentru selectarea eşantioanelor extrase.

Pentru *Reset Signal=F* se obţine semnal continuu. Schimbaţi la fiecare ciclare frecvenţa formei de undă sinus şi observaţi din nou punctele de concatenare.

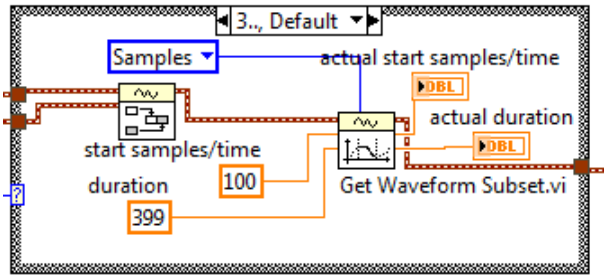


Fig. 11

Pentru *Reset Signal=T* se obţine semnal discontinuu totalizând trei generări de forme de undă fiecare din câte 100 eşantioene (Fig. 12), faza semnalului fiind resetată la valoarea prescrisă prin intrarea *phase* la fiecare generare. Astfel, pentru generarea unui semnal de durată, fără discontinuităţi, se va seta *Reset Signal=F*.

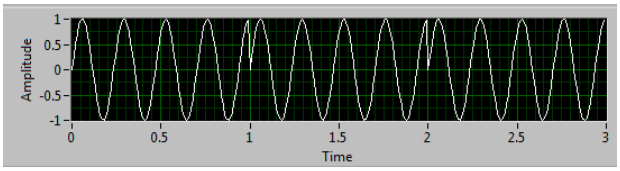


Fig. 12

5. Controlul Sampling Info şi frecvenţa - problema propusă

Puneţi Sampling Info control comun pentru Sine Waveform şi Uniform White Noise Waveform.vi şi frecvenţa de valoare 8. Observaţi semnalul generat la următoarele perechi din Sampling Info: (Fs=100, #s=100), (Fs=100, #s=200), (Fs=200, #s=100), (Fs=300, #s=100), (Fs=400, #s=100). Pentru fiecare pereche găsiţi valorile mărimilor dt şi timpul total în fereastră; calculaţi câte perioade sunt în fereastră şi câte sunt într-o secundă. Prezentaţi tabelar observaţiile. Se modifică frecvenţa (număr cicluri pe secundă) semnalului generat? Pentru numărare corectă, dacă este nevoie, setaţi Reset Signal = T/F.

6. Tablouri de numere reale (DBL) salvate în fişier text

Aplicaţia din Figurile 13 şi 14, salvează tablouri/array de numere reale (dbl) în fişier text apelând Write delimited Spreadsheet.vi şi apoi citeşte din fişier. În prima secvenţă a structurii Sequence este scrisă o matrice 4x3 [DBL] într-un fişier text, iar în secvenţa a doua sunt citite primele trei linii din fişierul text creat (prin selecţia fişierului în fereastra de dialog).

Se specifică formatul (%8.3f) de scriere în fişierul text unde 8 este numărul de caractere alocate unei valori din care trei sunt zecimale. La citirea din fişierul

text este furnizat numărul liniilor de citit (dar şi caracterul de la care să înceapă citirea). Vizualizaţi fişierul creat în Notepad şi în Excel.

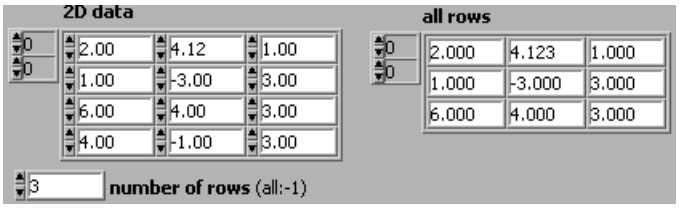


Fig. 13

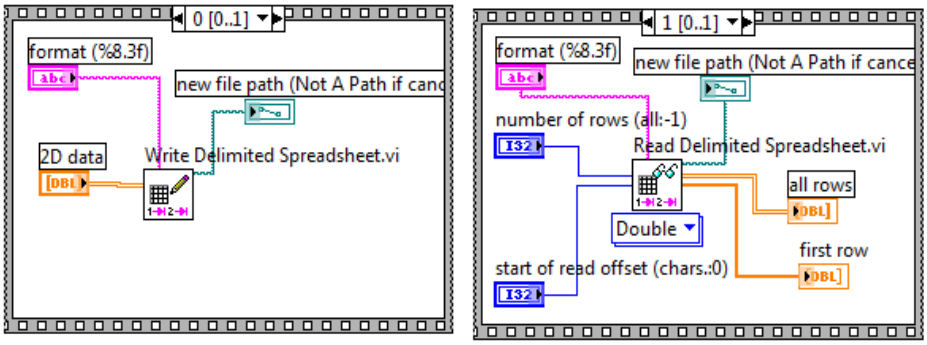


Fig. 14

7. Scrierea/citirea în/din fişier a unui şir de caractere

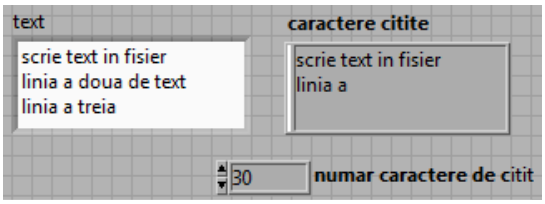


Fig. 15

Aplicaţia din Figurile 15 şi 16, conţine o structură de tip secvenţă cu două cadre (secvenţe). În prima secvenţă se apelează funcţia *Write to Text File.vi* pentru scrierea în fişier a unui şir de caractere plasat în controlul etichetat *text*. În a doua secvenţă se citeşte din fişierul tocmai creat numărul de caractere specificat în controlul *numar caractere de citit* care apoi se afişează în Panoul Frontal.

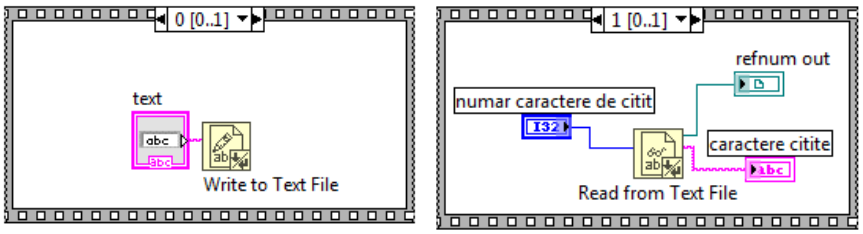


Fig. 16

XIII. Cursori în Waveform Graph

1. Operaţii cu cursori

1.1. Aplicaţia din Figura 1 trasează graficul funcţiei: f(x)=sin(x)+x. Asociat indicatorului grafic Waveform Graph se adaugă doi cursori (single-plot) ataşaţi curbei trasate. Fiecare cursor se poziţionează pe câte un punct de pe curbă. Se doreşte afişarea în PF a diferenţei dintre abscisele (x0-x1) şi ordonatele (y0-y1) punctelor pe care sunt plasaţi cursorii. Fiecare cursor poate fi deplasat cu mouse-ul în alte puncte de pe grafic, cu afişarea imediată a diferenţelor menţionate. Aplicaţia este oprită cu butonul de Stop.

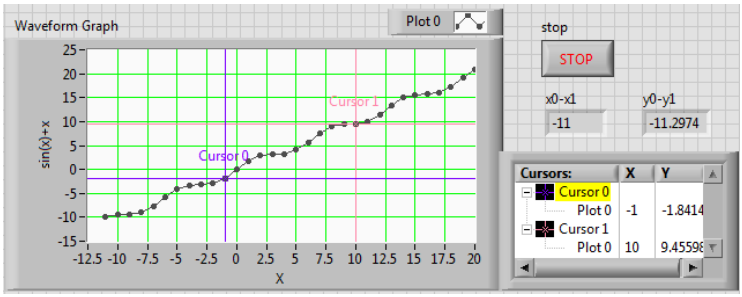


Fig. 1

În prima etapă este trasat graficul funcţiei, după care se intră într-un ciclu While pentru citirea repetată a poziţiei cursorilor. Este generată o referinţă la obiectul indicator grafic 'Waveform Graph': *mouse click-dreapta* pe indicatorul grafic sau pe terminalul asociat din diagramă (Fig. 2). Referinţa poate fi folosită pentru a avea acces la proprietăţile sau metodele (funcţiile) indicatorului grafic (în general referinţa poate fi la VI-ul curent sau la orice control sau indicator).

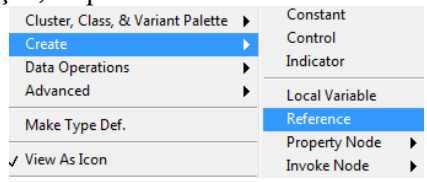


Fig. 2

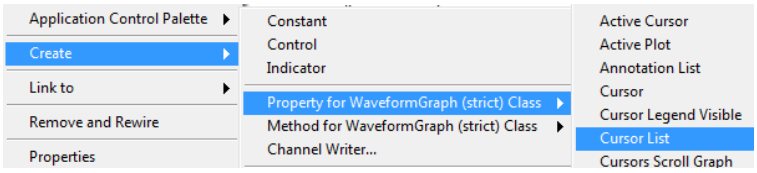


Fig. 3

Se plasează în diagramă (şi se conectează la referinţă) lista cursorilor asociaţi ferestrei grafice, prin comanda din Figura 3: Create/Property for.../Cursor List. Este conectat tabloul de două structuri (doi cursori) cu indexare la tunelul de intrare în

ciclul For. Din fiecare structură intrată în corpul ciclului prin *Unbandle by Name* sunt extrase valorile (coordonatele) X şi Y ale punctului pe care cursorul este poziţionat. Prin Index Array se extrag cele două abscise şi respectiv cele două ordonate pentru calculul diferenţelor. Zona din diagramă asociată prelucrării datelor, furnizate de cei doi cursori, este vizualizată în Figura 4. Codul grafic, pentru calculul funcţiei şi trasarea graficului, nu este vizibil.

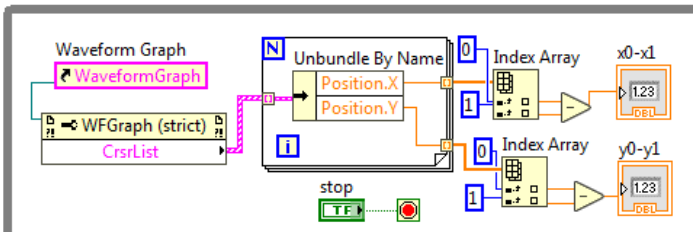


Fig. 4

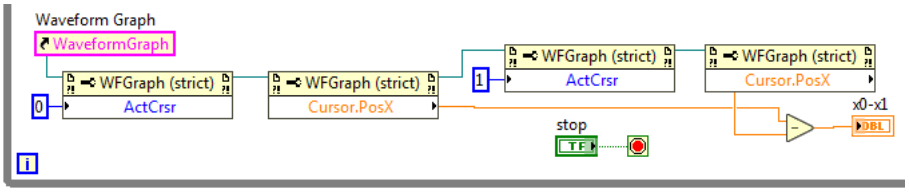


Fig. 5

1.2. O metodă alternativă (Fig. 5) pentru exemplul de mai sus este de a pune pe rând fiecare cursor în stare activă şi a extrage succesiv poziţia cursorului cursor.PosX, urmând să se calculeze diferenţa valorilor. Codul este în corpul unui ciclu While, astfel acţiunea se poate continua prin mutarea unui cursor pe alt punct şi afişarea imediată a noii diferenţe dintre abscisele punctelor.

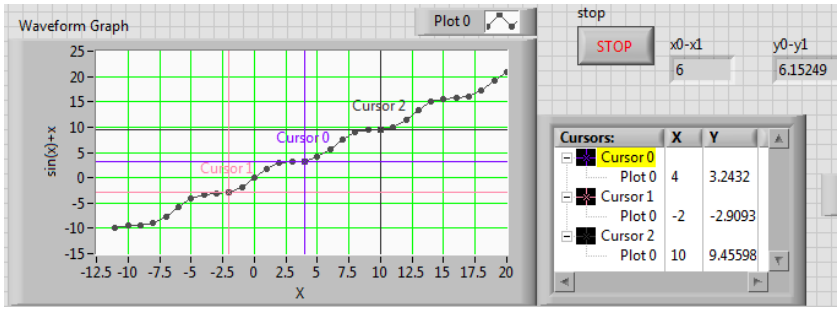


Fig. 6

1.3. Deplasarea unui cursor poate fi comandată sau este dependentă de deplasarea altui cursor. Se va corela deplasarea cursorului 2 cu deplasarea cursorului 0 şi afişarea diferenţei ordonatelor punctelor indicate de cursori (Fig. 6).

La deplasarea cu mouse-ul a cursorului 0, cursorul 2 se deplasează păstrând distanţa specificată prin controlul *'distanta cursori'* (Fig. 7).

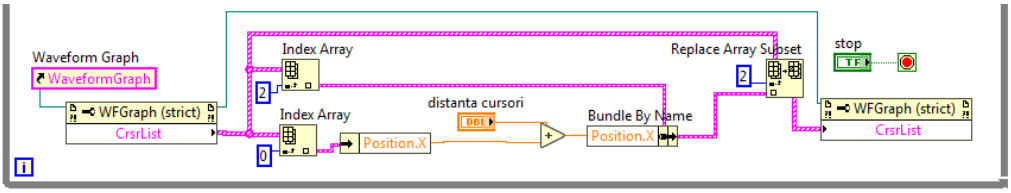


Fig. 7

În diagrama asociată observăm extragerea prin Index Array a structurii cursorului dependent (indice 2), înlocuirea (prin Bundle By Name) a valorii câmpului Position.X cu valoarea poziţiei cursorului indice 0 la care se adună controlul 'distanta cursori'. Urmează actualizarea tabloului de structuri CrstList.

1.4. Sunt amintite în continuare câteva noţiuni despre cursorii ataşaţi ferestrei grafice Waveform Graph. Lista proprietăţilor unui cursor este dată în Figura 8. Acestea sunt conţinute într-un cluster (structură) cu 16 câmpuri (elemente). Mai mulţi cursori formează un tablou 1D de clustere. Cursorul permite citirea coodonatelor punctelor de pe curbele trasate în fereastra grafică (proprietatea Position). La crearea unui cursor se specifică unul din cele trei moduri de lucru (Cursor Mode): Free, Single-Plot sau Multi-Plot; acesta nu se poate schimba ulterior (Fig. 9). În modul *Free* cursorul poate fi deplasat (folosind mouse-ul) la orice punct de pe zona grafică, punct aflat pe o curbă sau în afara curbelor. În modul *Single-Plot* cursorul citeşte puncte numai de pe o curbă din setul de curbe din fereastra grafică; cursorul poate fi mutat pe altă curbă (prin comanda Snap To) după care va prelua puncte numai de pe noua curbă (la deplasarea cursorului cu mouse-ul). În modul *Multi-Plot* cursorul citeşte şi afişează simultan puncte de pe mai multe curbe. Astfel, în Figura 10, cursorul indice 2 afişează trei puncte, câte unul de pe fiecare din cele trei curbe, punctele având aceeaşi abscisă.

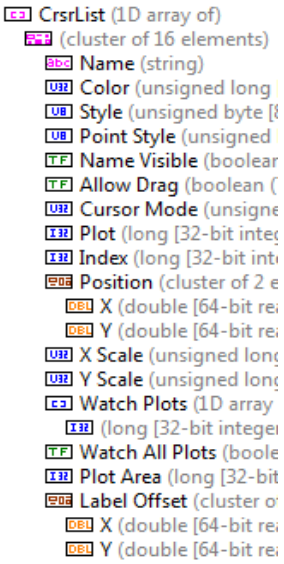


Fig. 8

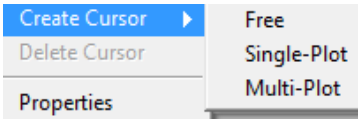


Fig. 9

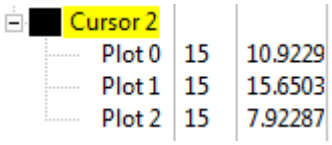


Fig. 10

2. Eveniment de tip 'Cursor Move'

2.1. Se va programa un eveniment de tip 'Cursor Move'. La deplasarea cursorului unic, asociat ferestrei 'Waveform Graph', se creează al doilea cursor plasat la abscisă dublă faţă de primul cursor. În Panoul Frontal (Fig. 11) observăm abscisa cursorului indice 0 la valoarea 7.5, iar cursorul 2X comandat este plasat la abscisa 15.

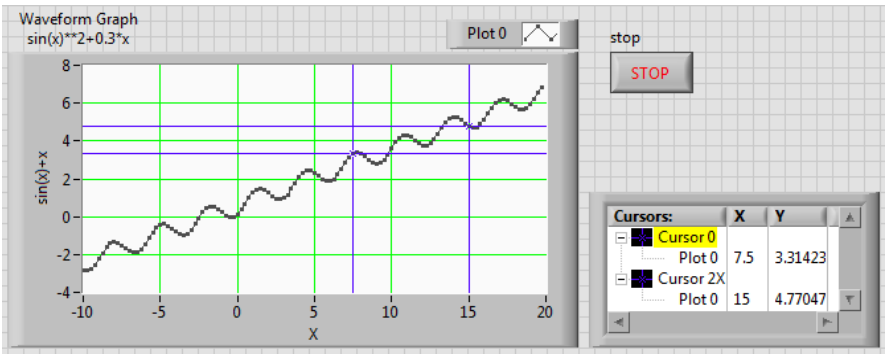


Fig. 11

La deplasarea la stânga sau la dreapta cu mouse-ul a cursorului indice 0, cursorul al doilea, notat 2X, se va deplasa imediat la dublul abscisei primului cursor.

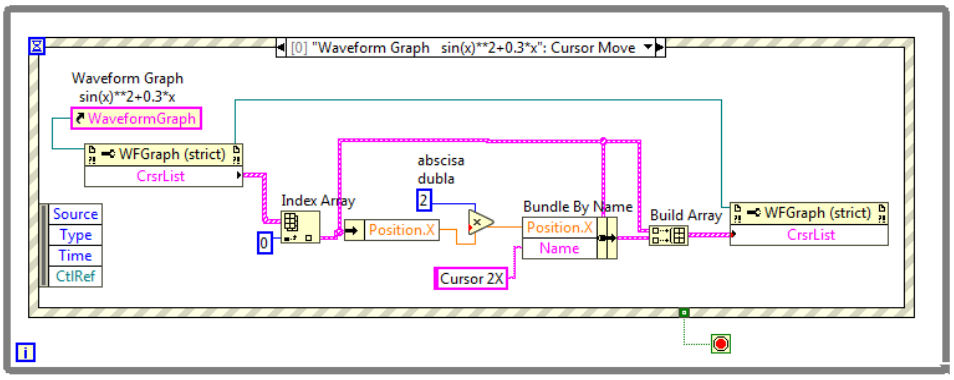


Fig. 12

În diagramă (Fig. 12) se realizează copierea structurii, compusă din 16 câmpuri, ataşate primului cursor (indice 0), şi modificarea în structura copiată a numelui şi a poziţiei cursorului. Sunt astfel modificate două câmpuri ale structurii copiate. Noul nume este Cursor 2X, iar poziţia pe axa absciselor, la fiecare eveniment de tip 'Cursor Move', se obţine prin înmulţirea cu 2 a abscisei primului cursor. Se construieşte un tablou 1D ce conţine cele două structuri asociate celor doi cursori şi se actualizează sau se impune noul tablou obiectului grafic 'Waveform Graph'.

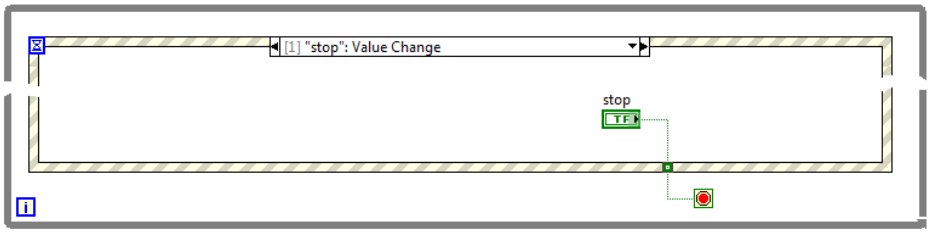


Fig. 13

Evenimentul prin care este oprită aplicaţia este apăsarea butonului de Stop. Schimbarea valorii (Value Change) butonului din False (neapăsat) în True asigură oprirea ciclului While prevăzut cu terminalul Stop if True (Fig. 13).

2.2. Fereastra pentru setarea evenimentului (în cadrul structurii Event Structure) asociat deplasării cursorului, este vizualizată în Figura 14.

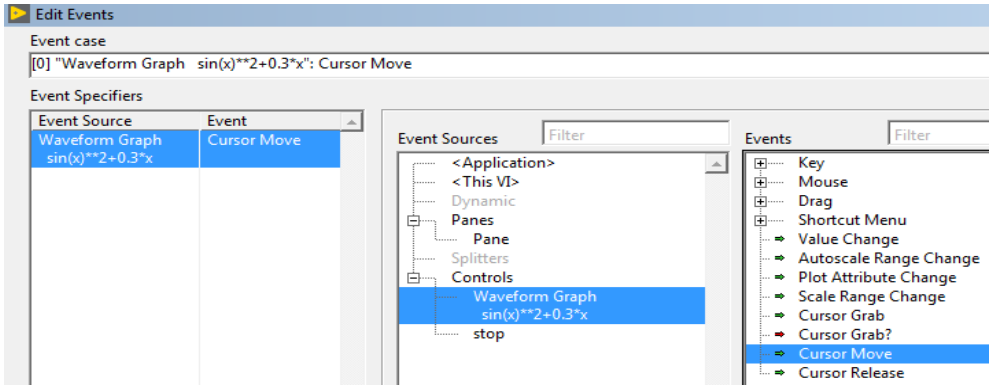


Fig. 14

XIV. Aplicaţii LabVIEW pentru prelucrarea sunetelor prin placa de sunet

1. Funcţii pentru achiziţie de sunet prin placa de sunet

Sunt prezentate în continuare funcţiile de bază pentru achiziţie de sunet (Fig. 1) plasate în paleta Sound (Fig. 2). Funcţia *SI Config.vi* (Fig. 3) pregăteşte placa de sunet pentru achiziţie, alocă un buffer RAM de memorie intermediar (de exemplu, alocă 8192 bytes, apoi trimite datele în buffer-ul intermediar de memorie). Intrarea *number of samples/ch* specifică numărul de eşantioane pe fiecare canal care se alocă în buffer. Intrarea *sample mode* pentru valoarea *Finite Samples* determină achiziţia unui număr de eşantioane specificat şi se opreşte (achiziţia), iar pentru *Continuous Samples* se va realiza o achiziţie continuă (funcţia SI Read.vi se va apela repetitiv).

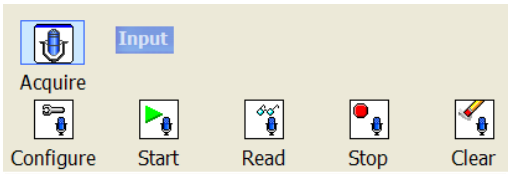


Fig. 1

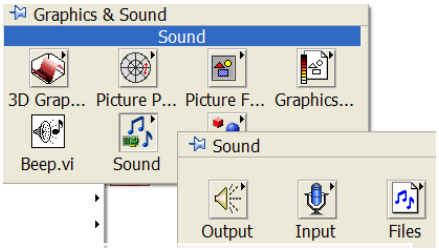


Fig. 2

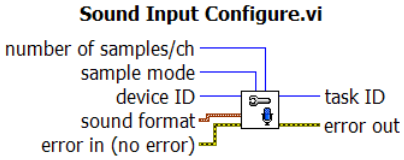


Fig. 3

Intrarea *device ID* pentru valoarea 0 (implicit) specifică numărul dispozitivului de achiziţie (placa de sunet internă sau altă placă conectată pe USB). Intrarea *sound format* (Fig. 4) propune formatul sunetului achiziţionat. Aceasta fiind o structură de date prezintă următoarele câmpuri: 1) rata de eşantionare 44100 eş/sec (Hz), 22050 (Hz implicit) sau 11025 (Hz);
2) numărul de canale de achiziţie 1(mono)/2(stereo);
3) 8/16 biţi/eşantion (biţi necesari pentru memorarea unui eşantion, implicit 16 biţi). Valorile 11.025 kHz şi 22.05 kHz sunt frecvenţe de eşantionare folosite în fişiere WAV. 44.1 kHz sampling rate şi 16 bits/sample sunt comune pentru compact disc digital audio (CD-DA). Frecvenţa de eşantionare de 44.1 kHz este folosită şi de formatul MP3 (standard Sony). 88.2 kHz şi 176.4 kHz sunt frecvenţe de eşantionare înalte folosite în DVD-Audio.

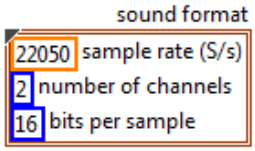


Fig. 4

Funcţia SI Config.vi returnează: *task ID* - reprezintă un număr întreg de identificare a sarcinii primite (task) şi structura de date *error out*. Aceasta din urmă

(Fig. 5) conţine un cluster cu 3 câmpuri: *status* de tip boolean având valoarea True dacă a apărut o eroare şi valoarea False altfel; *code* de tip numeric întreg conţinând codul erorii apărute dacă *status* = true; *source* de tip şir de caractere specifică sursa erorii sau atenţionării (şirul reprezintă numele nodului care a generat eroarea).

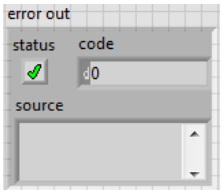


Fig. 5

Funcţia *SI Read.vi* (Fig. 6) transferă/citeşte datele (tablou de waveforms) din buffer-ul RAM în memoria calculatorului. Intrarea *number of samples/ch* specifică numărul de eşantioane pe canal de citit din buffer, la o citire. Funcţia returnează *data* de tip tablou de forme de undă. Fiecare waveform reprezintă semnalul achiziţionat de pe un canal şi conţine câmpurile: *t0* = momentul de start, *dt* = 1/rata eşantionare (de exemplu, Fs = 22050 Hz) şi *Y* = tabloul de eşantioane în interval [-1, +1] dacă tipul de dată este DBL sau SGL.

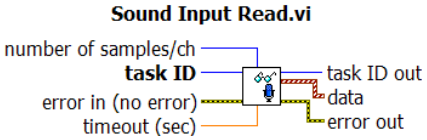


Fig. 6

Funcţia *SI Stop.vi* (Fig. 7) opreşte achiziţia de la Placa Sunet. Funcţia *SI Start.vi* (Fig. 7) reporneşte achiziţia de la placa de sunet (pune eşantioane în bufferul de memorie RAM alocat) şi se apelează numai dacă anterior s-a apelat SI Stop.vi. Funcţia *SI Clear.vi* eliberează memoria de sarcina de achiziţie, inclusiv bufferul alocat.

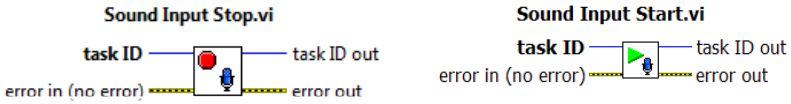


Fig. 7

2. Achiziţie de sunet pe durata finită

Este apelată funcţia Acquire Sound plasată în paleta Sound (Fig. 1). Prin această aplicaţie (Fig. 8) este achiziţionat semnal de la microfon fiind folosită placa de sunet a calculatorului. Sunt stabiliţi parametri de achiziţie prin controale: dispozitivul (placa de sunet), rata de eşantionare (22050 eşantioane/secundă), numărul de canale (un canal sau două canale), rezoluţia unui eşantion (16 biţi) şi durata în secunde a achiziţiei. Durata achiziţiei este stabilită/cunoscută de la apelul funcţiei Acquire Sound. Este şters automat taskul după achiziţia semnalului.

Funcţia returnează semnalul achiziţionat (patru bătăi din palme) în format dinamic. Prin funcţia *Convert from Dynamic Data* (paleta Express/Signal manipulation, Fig. 8) se realizează conversia de la tipul Dynamic Data la tablou de forme de undă, câte una pentru fiecare canal de achiziţie. O funcţie utilă pentru a extrage informaţii utile din semnalul dinamic este *Get Dynamic Data Atributtes* (Fig. 10).

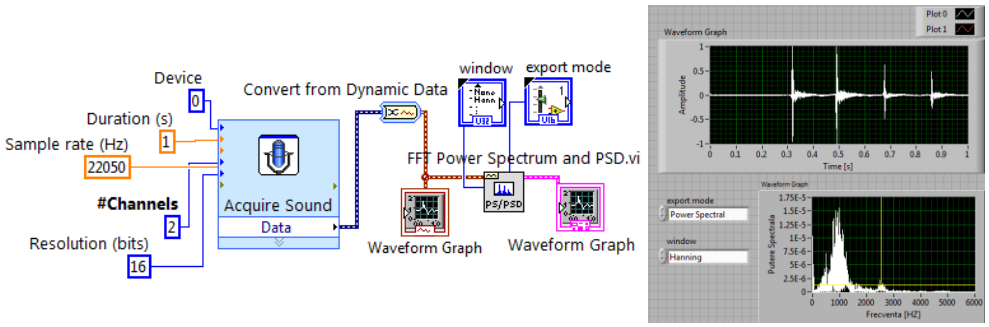


Fig. 8

Funcţia *FFT Power Spectrum and PSD.vi* permite selecţia unei ferestre de ponderare (s-a ales Hanning) şi alegerea funcţiei spectrale PS sau PSD prin intrarea *export mode*.

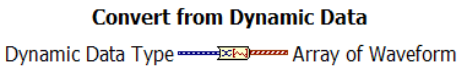


Fig. 9

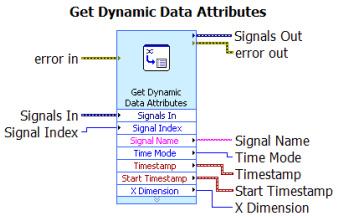


Fig. 10

3. Achiziţie continuă de sunet şi calcul putere spectrală

3.1. Sunt apelate trei funcţii prin care este gestionată achiziţia de sunet.

Funcţia *Sound Input Configure.vi* (Fig. 11) realizează configurarea unei sarcini de achiziţie sunet. Sound Format este o structură cu trei câmpuri. Câmpul *sample rate* (S/s) stabileşte numărul de eşantioane (S) achiziţionate pe secundă; acestea se depun în bufferul intermediar de unde sunt citite de funcţia SI Read.vi. Se alege achiziţie continuă selectând Continuous Samples. Controlul Number of Samples/ch stabileşte mărimea bufferului intermediar de memorie unde se depun temporar eşantioanele provenite de la convertorul analogic-digital. Implicit placa de sunet internă corespunde pentru intrarea device=0. Dacă se adaugă o placă de sunet suplimentară, de exemplu, conectată pe USB, se va selecta valoarea 0 sau 1 în funcţie de placa care se doreşte a fi utilizată în cadrul aplicaţiei.

Funcţia *Sound Input Read.vi* realizează citirea reperată din bufferul de memorie, în cadrul ciclului While, a unui număr de eşantioane (stabilit prin Number of Samples/ch). Funcţia returnează un tablou de două forme de undă (1D array of waveform), câte o formă de undă pentru fiecare canal de achiziţie.

Funcţia *Sound Input Clear.vi* dealocă memoria folosită pentru sarcina de achiziţie configurată prin SI Configure.vi.

3.2. La o ciclare sunt achiziţionate şi afişate 5000 eşantioane. Spaţierea între eşantioane este 1/Sample rate (secunde), iar durata formei de undă este 5000*1/Sample rate (secunde). Semnalul de la fiecare canal este trasat în indicatorul grafic Waveform Graph (Raw Data) cu culoarea proprie (albastru, roşu).

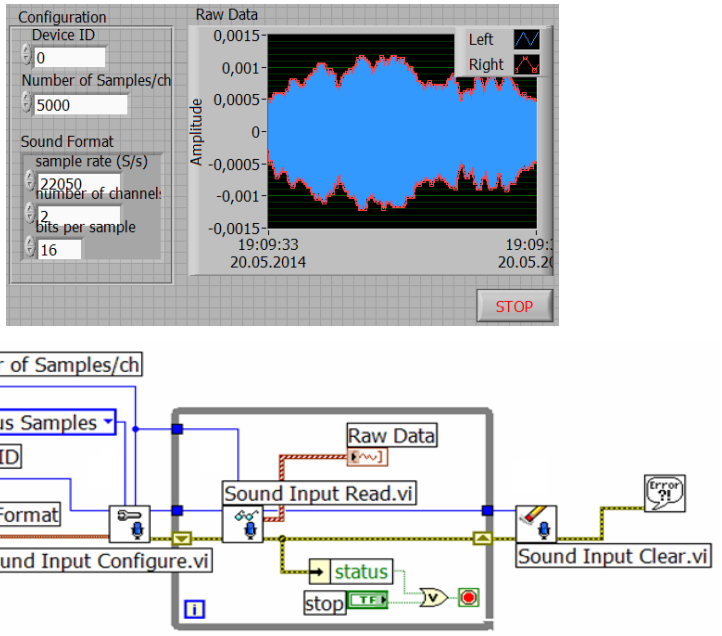


Fig. 11

3.3. Ciclul se opreşte în două situaţii observabile în diagrama din Figura 11: a) apare eroare la execuţia unei funcţii (semnalată prin câmpul *status* al structurii setat pe True) sau b) se apasă controlul STOP.

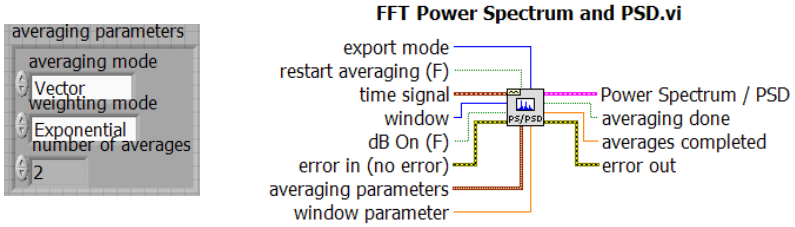


Fig. 12

3.4. Se adaugă calculul puterii spectrale (PS) a semnalului prin funcţia FFT Power Spectrum and PSD.vi (Fig. 12). Asociat funcţiei selectăm fereastra Hanning pentru pregătirea semnalului înainte de aplicarea transformatei Fourier (FFT.vi) şi controlul *Averaging parameters*.

Se vor măsura pe indicatorul grafic asociat diagramei din Figura 13, folosind un cursor (Fig. 15), frecvenţele unor vârfuri ale puterii spectrale obţinute la vibraţia diapazonului (440 Hz), lovirea uşoară a unui pahar de sticlă, bătutul din palme, ciupirea unor coarde ale instrumentelor muzicale şi lovirea unor lamele ale xilofonului (Fig. 15).

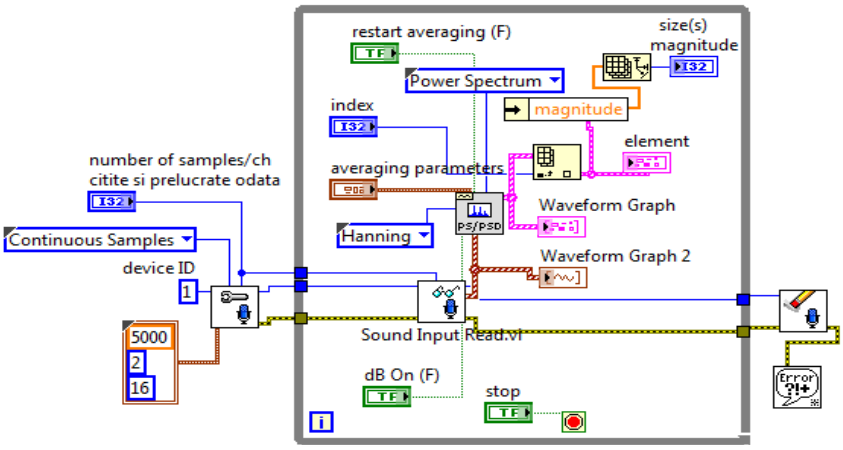


Fig. 13

Pentru reţinerea vârfurilor puterii spectrale observabile în Figura 14 (în vederea măsurării unor frecvenţe cu cursorul) se va selecta Peak hold pentru Averaging mode în cadrul controlului *Averaging parameters* al funcţiei FFT Power Spectrum and PSD.vi.

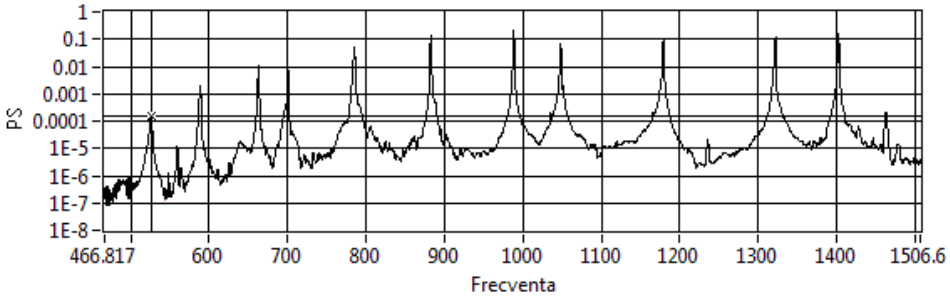


Fig. 14

Frecvenţe rezultate din măsurare, asociate vârfurilor puterii spectrale la lovirea lamelelor xilofonului, sunt prezentate în continuare.

C5	D	E	F	G	A	B	**C6**	D	E	F	G
527	589	663	701	785	883	988	**1049**	1179	1322	1402	1569

În cazul în care Fs = 2000eş/s, iar Nr. eş/ch = 4000 rezultă realizarea unui ciclu la 2 secunde deoarece SI Read.vi aşteaptă 2 secunde pentru a primi 4000 eşantioane (deoarece sunt preluate numai 2000 eşantioane pe secundă de la convertorul analogic digital al plăcii de achiziţie). Cele 4000 eşantioane vor intra în PowerSpectrum. Spectrul este actualizat o dată la 2 secunde (deci rar), df=0.5 Hz, iar lăţimea spectrului este 0-1000 Hz.

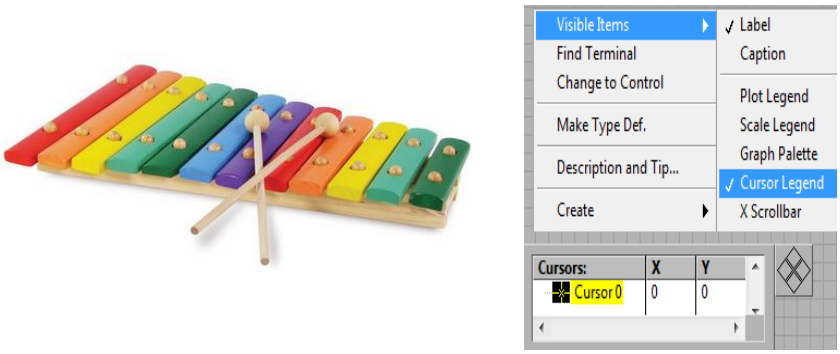


Fig. 15

Dacă Fs = 4000 eş/s, iar Nr. eş/ch = 2000 => în 0.5 secunde vin cele 2000 eşantioane la SI Read.vi şi apoi sunt trimise la PowerSpectrum. Deci vor fi 2 cicluri pe secundă, df = 2 Hz şi spectrul este actualizat de două ori pe secundă; lăţimea spectrului este 0-2000 Hz

Pentru orice Fs şi Nr.eş/ch, frecvenţa tonului (diapazon) este corect determinată.

Rezoluţia în frecvenţă este df=Fs/N, unde Fs este frecvenţa de eşantionare iar N este numărul de eşantioane analizate deodată.

Frecvenţa maximă din spectrul rezultat este f_{max} = N/2*df = Fs/2.

În gama Do major nota Do(C) ≈ 262 Hz, iar nota La(A) = 440 Hz.

4. Funcţii de bază pentru generarea sunetului

Setul funcţiilor de bază utilizate pentru generarea sunetului prin placa de sunet şi trimiterea semnalului la difuzor este prezentat în Figura 16.

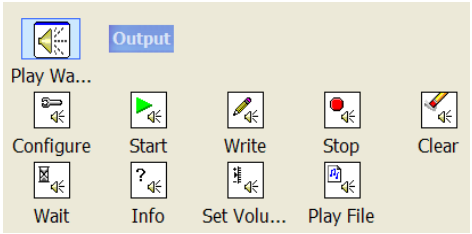


Fig. 16

Funcţia *SO Configure.vi* (Fig. 17) configurează placa de sunet (convertorul digital/analogic) pentru generare de semnal, impune numărul de eşantioane/canal de alocat în bufferul intermediar şi stabileşte modul

de generare (număr finit de eşantioane sau generare în mod continuu). Intrarea *sound format* stabileşte a) rata de trimitere a eşantioanelor (44 kHz, 22 kHz sau 11 kHz), b) numărul de canale (1 = mono, 2 = stereo) şi c) numărul de biţi alocaţi unui eşantion (16 sau 8 biţi/eşantion).

Funcţia *SO Write.vi* (Fig. 17) scrie informaţia furnizată prin intrarea *data* (tablou de waveforms, câte un waveform/canal) la dispozitivul de ieşire (în bufferul de memorie alocat). Momentul t0 se neglijează, *dt* se neglijează, iar tabloul *Y* este trimis la SO Write; astfel, cele două variante de cod din Figura 18 generează acelaşi efect deoarece t0 şi dt sunt ignorate. Tabloul Y poate fi SGL, DBL cu valori [-1 ...1], U8 [0...255], I16 etc.

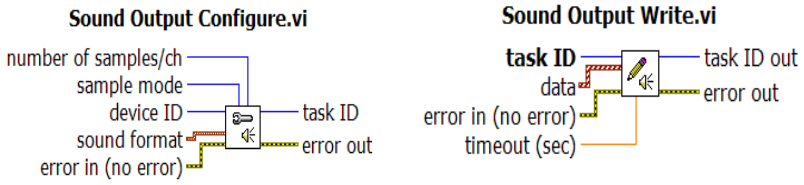


Fig. 17

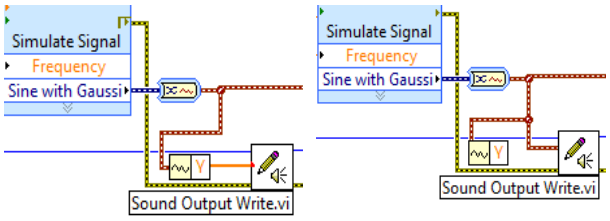


Fig. 18

Funcţia SO Stop.vi opreşte placa de sunet să ia eşantioane din bufferul intermediar. Funcţia SO Start.vi reporneşte generarea de sunet după oprirea acestuia cu SO Stop.vi; se foloseşte numai după SO Stop.vi. SO Wait impune aşteptare până când întregul sunet este trimis la dispozitivul de ieşire.

5. Generare sunet (ton) în mod continuu

5.1. Sunt apelate cinci funcţii prin care este coordonată generarea de sunet prin placa de sunet a calculatorului.

Funcţia *SO Configure.vi* stabileşte prin controlul Sound Format rata de trimitere a eşantioanelor (44100 eşantioane/secundă) la placa de sunet, numărul de eşantioane trimise pe fiecare canal într-o tranşă (valoare care stabileşte şi mărimea bufferului intermediar de memorie) şi modul continuu de generare a sunetului (Continuous Samples). Este returnat un număr întreg **(**task ID**)** pentru identificarea

sarcinii de generare de sunet, număr care este trimis şi la următoarele funcţii care gestionează sarcina de generare.

Funcţia *SO Write.vi* apelată repetitiv trimite efectiv tranşa de 5000 de eşantioane (în exemplul curent), date de tip tablou de forme de undă, la convertorul digital-analogic al plăcii de sunet.

Funcţia *SO Set Volume* apelată repetitiv permite modificarea volumului (amplitudinea) sunetului generat.

Funcţia *SO Clear.vi* este apelată după oprirea ciclului While realizând ştergerea sarcinii de generare a sunetului.

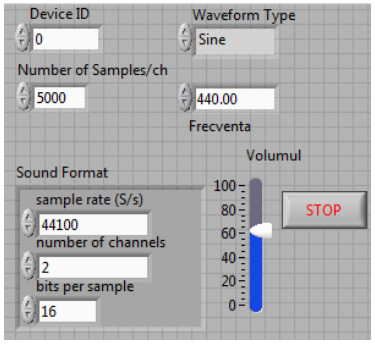


Fig. 19

5.2. Forma de undă primită de SO Write.vi este generată de funcţia Sine Waveform.vi. Aceasta se va apela repetitiv astfel încât cele 44100 eşantioane/secundă (Sample rate) să fie asigurate. Se corelează astfel controlul *sampling info* cu *Sound Format*, asigurând frecvenţa dorită a formei de undă sinusoidale şi implicit frecvenţa tonului generat de placa de sunet. Structura *sampling info* a funcţiei Sine Waveform conţine iniţial valorile (0, 0), iar prin *Bundle by name* se modifică la (Fs = 44100, #s = 5000). *Reset signal* se pune pe False pentru ca secvenţele succesive de semnal generate de Sine Waveform.vi să nu fie cu întreruperi.

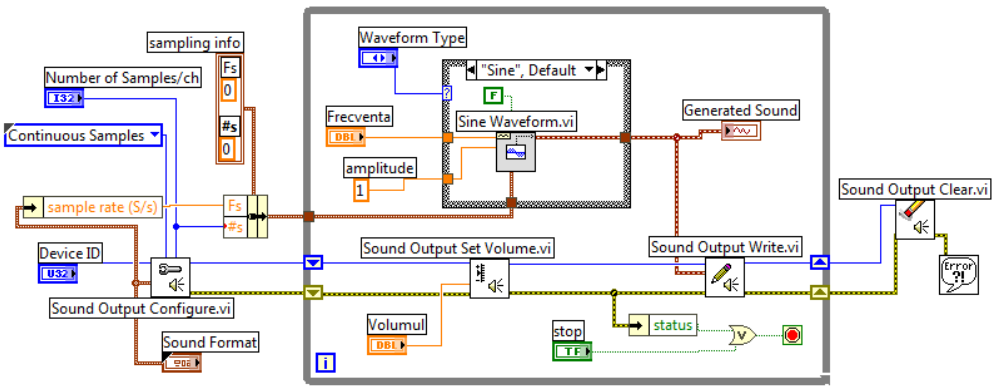


Fig. 20

Semnalul sinusoidal prezintă 440 perioade dacă ar conţine 44100 eşantioane; fiind numai 5000 eşantioane va fi alcătuit din mai puţine perioade de sinus în mod proporţional:

44100 eş ... 440 cicluri (Hz)

5000 eş ... x cicluri => x = 49.8866 cicluri

şi va dura proporţional mai puţin de o secundă:

44100 eş ... 1 sec.

5000 eş ... y sec. => y = 0.1134 secunde.

Ciclul While asigură prin generare repetitivă continuu semnal pentru Sound Output Write.vi. Astfel, SO Write.vi la o execuţie generează sunet de 49.88 cicluri timp de 0.113 secunde (Fig. 21 şi Fig. 22) prin convertorul digital-analogic spre difuzor.

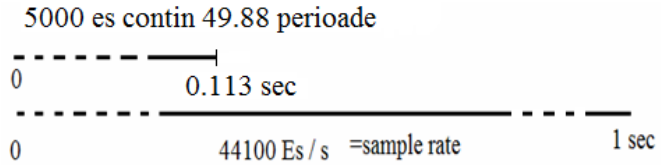


Fig. 21

5.3. În timpul rulării aplicaţiei modificaţi frecvenţa semnalului sinusoidal (Fig. 19). Măsuraţi frecvenţa sunetului generat prin placa de sunet cu aplicaţia Advanced Spectrum Analyzer PRO de pe mobil sub Android (sau alte aplicaţii similare). În această aplicaţie se poate seta Logarithmic scale (sau scara liniară) pentru axa absciselor (frecvenţa), se poate stabili tipul ferestrei de ponderare (Hanning, Hamming etc.), numărul de eşantioane care participă la FFT (512 ,..., 16384), frecvenţa de eşantionare a semnalului de la microfon (44100 sau 48000) şi alte setări.

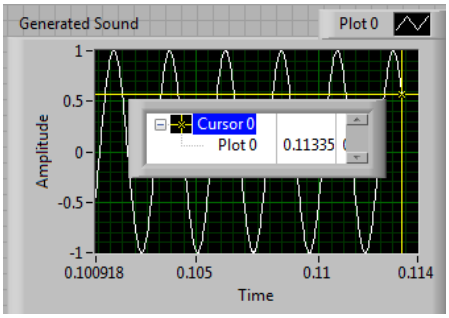


Fig. 22

5.4. Instrucţiunea Case selectează tipul semnalului de ieşire la frecvenţa de 440 Hz şi amplitudinea unitară. Se vor putea selecta şi alte forme de undă prin instrucţiunea Case. În graficul Generated Sound din Panoul Frontal se observă semnalul de trimis la SO Write.vi, durata şi frecvenţa ce caracterizează semnalul sinusoidal.

6. Generare sunet pe două canale - tablou de două forme de undă

Sunt generate cu funcţia *Simulate Signal.vi* două forme de undă: un semnal dreptunghiular (Square) plus zgomot pentru canalul indice zero şi un semnal sinusoidal (Sine) plus zgomot pentru canalul indice unu (Fig. 23).

Fiecare waveform conţine 3000 eşantioane scrise la Sample rate: 44100 Hz rezultând durata waveform = 0.068sec, ($\frac{1}{44100} \cdot 3000$). Funcţia Sound Output Write.vi primeşte tabloul de două forme de undă generat cu *Build Array* pentru cele două canale. Afişarea în PF a semnalelor (Fig. 24) este realizată cu zoom pe durata [0,...,0.01] secunde pentru a distinge semnalele. Frecvenţa semnalului dreptunchiular (1000 Hz) este de două ori mai mare decât frecvenţa semnalului

sinusoidal (500 Hz). Observăm în Figura 24, 10 perioade ale semnalului dreptunchiular în 0.01 secunde, rezultând 1000 perioade pe secundă. Se pot modifica şi observa frecvenţele şi amplitudinile semnalelor cu aplicaţia Advanced Spectrum Analyzer PRO sub Android sau altele similare. Prin apropierea telefonului mobil de un difuzor sau de celălalt difuzor al calculatorului (stereo) observăm variaţia amplitudinii semnalului generat de fiecare difuzor în funcţie de distanţa de sursă.

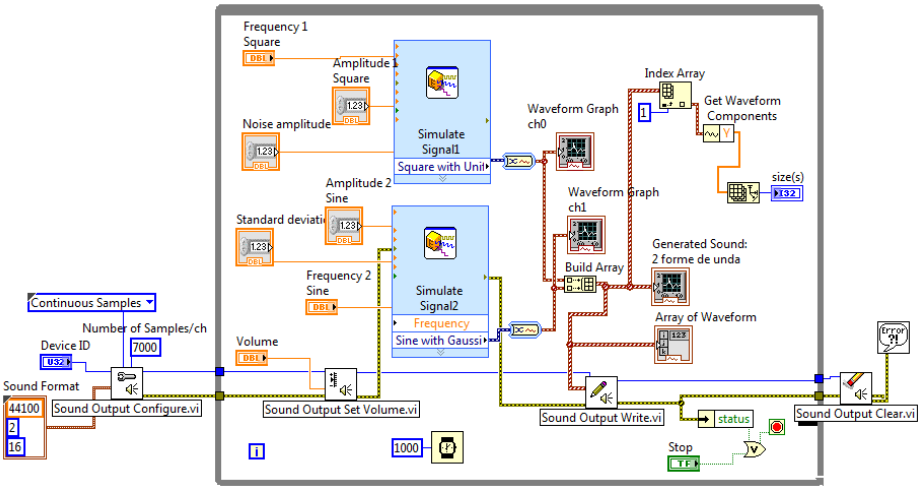


Fig. 23

Dacă se reduce Sampling Rate de la 44100 la 22050 sunetul generat va dura mai mult şi se va auzi un ton mai jos deoarece acelaşi semnal (şi număr de eşantioane) este trimis la placa de sunet cu o rată mai mică. Frecvenţa generată se schimbă deoarece nu este corelare între SO Config.vi/Sound Format şi generarea semnalelor (acestea sunt setate pe 44100 eş/sec şi number of samples #s = 4410; rezultă 10 cicluri pe secundă).

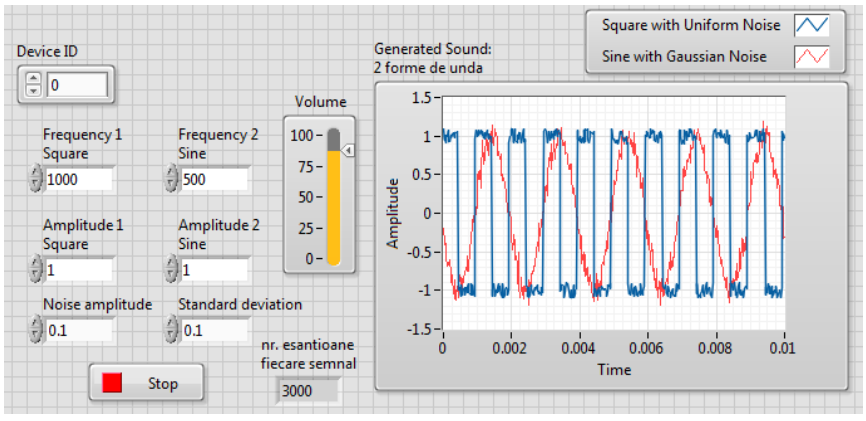


Fig. 24

În fereastra *Generated Sound 2 forme de unda* se afişează aceleaşi semnale fiindcă semnalele generate de funcţiile Simulate signal.vi sunt aceleaşi. Semnalul dreptunghiular se aude în difuzorul de pe primul canal, iar semnalul sinusoidal la difuzorul conectat canalului al doilea. Volumul la difuzoare se reglează prin funcţia SO Set Volume.vi (în limitele 0...100) şi prin amplitudinea funcţiilor periodice în intervalul 0 şi 1 (dacă amplitudinea semnalului sinusoidal creşte peste 1 volumul nu creşte).

7. Simularea sunetului asociat tastelor telefonului

Aplicaţia (preluată din Help şi modificată) generează câte un sunet la apăsarea unei taste din matricea de 4x3 butoane de tip logic (Fig. 25). Se rămâne în bucla While până la apăsarea butonului STOP (Fig. 26).

Fig. 25

La fiecare iteraţie, dacă se intră pe cazul True, registrul de transfer este iniţializat cu structura de valori logice 4x3 False. La apăsarea unei taste structura keys va conţine un câmp True, iar restul False; rezultă o diferenţă între controlul din PF şi constanta structură (cluster) de 12 valori False. Se execută prin urmare cazul True al structurii Case. Funcţia *Search 1D Array.vi* identifică numărul de ordine al tastei apăsate (T) în tabloul boolean (după conversia de la structură la tablou de valori logice).

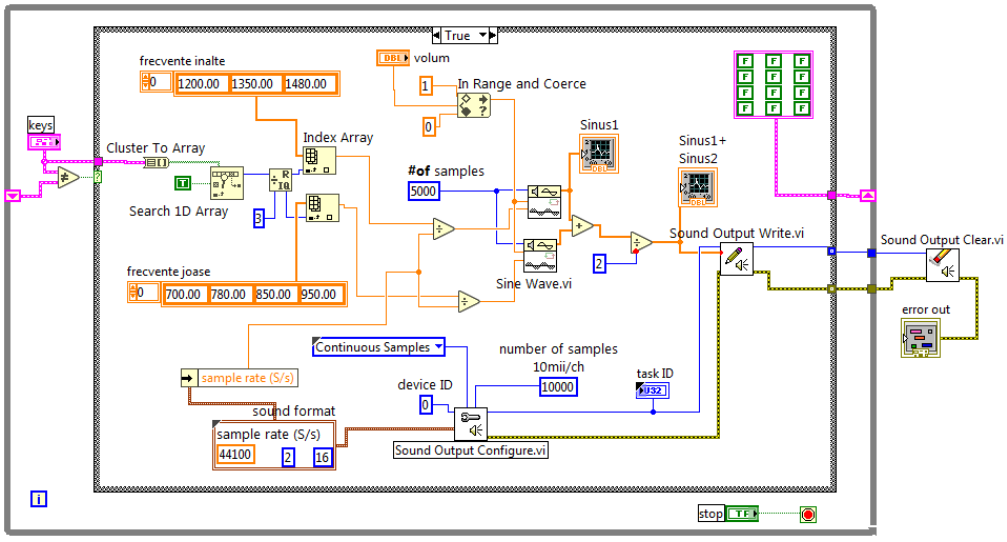


Fig. 26

Impărţirea cu rest (aplicată numărului de ordine) determină linia şi coloana tastei apăsate. Prin funcţia *Index Array.vi* se extrage o valoare din constanta tablou *frecvenţe înalte* şi altă valoare din constanta tablou *frecvenţe joase*. Cu aceste

valori se pregătesc intrările *frequency* ale semnalelor sinusoidale generate de funcţiile *Sine Wave.vi*. Semnalele adunate vor genera sunetul asociat fiecărei taste compus din două tonuri.

Sound Output Write.vi generează sunetul dat de suma celor două tonuri, sunet specific tastei. Slide-ul *volume* modifică amplitudinea ambelor sinusoide. Funcţia *In range and Coerce.vi* limitează volumul între 0 şi 1**.**

Operatorul *diferit* (care alimentează selectorul instrucţiunii Case) este setat să compare (Fig. 27) cele două structuri ca ansamblu (rezultă o singură valoare T sau F) şi nu pe componente. La prima iteraţie este selectat cazul False fiindcă cele două structuri sunt egale. Se poate folosi *Use Default if Unwired* pentru ambele tuneluri (task ID şi Error out) în cazul False (Fig. 28). Cazul False este de asemenea traversat de firul structurii (cluster) cu 12 taste.

Fig. 27

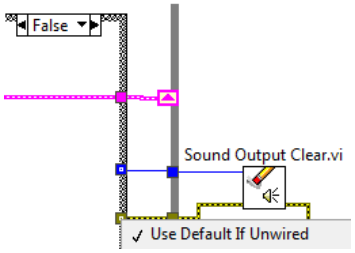


Fig. 28

Se poate modifica aplicaţia cu scoaterea funcţiei SO Configure.vi din ciclul While, astfel va fi o singură configurare a plăcii de sunet şi un singur Task ID.

XV. INTERFAŢA LabVIEW - ARDUINO

LIFA (LabVIEW Interface for Arduino) şi LINX (LabVIEW INterface for X, by LabVIEW MakerHub) sunt produse soft (firmware) care asigură controlul 'low-level' a unor componente hardware. Acestea asigură interfaţa şi controlul din LabVIEW a platformelor Arduino (plăci cu microcontroler construite pe un singur PCB - printed circuit board, având la bază un procesor din familia Atmel) şi shield-uri (module ataşate plăcilor pentru extinderea caracteristicilor). Interfaţa LIFA nu se mai dezvoltă, iar LINX permite interfaţare în sens mai larg cu Arduino, chipKIT, NI myRIO. Platformele Arduino se conectează la calculator (laptop, desktop) şi sunt programate prin interfaţa serială USB (Universal Serial Bus).

1. Instalare LIFA şi Arduino IDE

Se lansează Tools/*VI Package Manager* (Fig. 1)

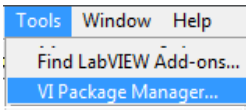


Fig. 1

Se instalează *LabVIEW Interface for Arduino* (LIFA) toolkit din Tools/*VI Package Manager* (Fig. 2) prin download de pe www.ni.com

Se observă/instalează toolkit Arduino (Fig. 3).

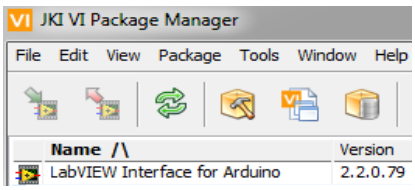


Fig. 2

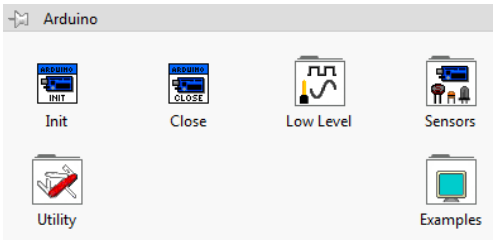


Fig. 3

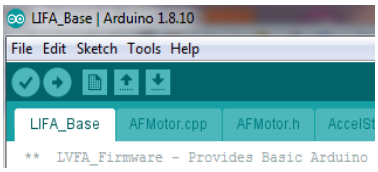
Se instalează Arduino IDE (Integrated Development Environment) prin download de pe www.arduino.cc.

Lansarea în execuţie se realizează folosind pictograma din Figura 4.

Fig. 4

Se conectează Arduino Mega 2560 prin USB la laptop.

Din mediul Arduino se *încarcă LIFA_Base*, C:\Program Files\National Instruments\LabVIEW Instruments\...\vi.lib\ LabVIEW Interface for Arduino\Firmware\LIFA_Base (Fig. 5). (pentru unele versiuni în fişierul Labviewinterface.cpp se pune: unsigned char checksum=0).



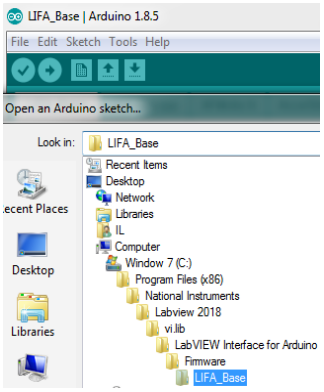


Fig. 5

Alegem placa conectată: Tools\Board\Arduino*Genuino Mega or Mega 2560* şi portul disponibil, de exemplu: COM5 (Fig. 6). COM1, COM2 sunt deja alocate pentru hard serial.

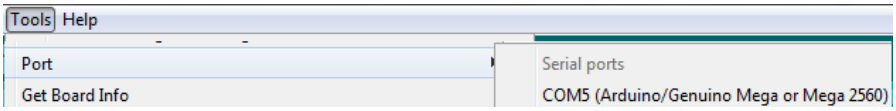


Fig. 6

Se acţionează (click mouse) butonul *Upload* pentru încărcare *firmware* în Arduino Mega (Uno). Se observă ledurile RX şi TX de pe placă pâlpâind (blinking). Arduino IDE raportează *Done Uploading* după încărcare în Arduino cu succes. Se poate acum folosi Arduino Mega 2560 cu toolkitul LabVIEW Interface for Arduino. Dacă este conectată altă placă (Uno) decât cea selectată apare eroare (Fig. 7). *sketch* este denumirea programelor dezvoltate în Arduino IDE.

Fig. 7

2. Comandă LED de pe placa Arduino Uno/Mega

2.1. Dioda LED folosită este lipită pe cablajul platformei Arduino Uno/Mega şi este conectată la pinul digital D13 prin intermediul unei rezistenţe de 220 ohmi. Aprinderea LED-ului se realizează prin aplicarea unui nivel logic de 5 V la pinul D13. Se construieşte Diagrama şi Panoul Frontal din Figura 8, folosind paleta Arduino (LabVIEW). Sunt apelate repetitiv funcţiile Set Digital Pin Mode.vi şi Digital Write Pin.vi, prima pentru setarea pinului D13 în mod Output şi a doua pentru scrierea valorii 0 sau 1 la pinul selectat. Observăm conversia de la valoarea logică furnizată de Control Boolean la valoarea întreagă acceptată de Digital Write Pin.vi. Se rulează aplicaţia LabVIEW şi se aprinde sau stinge dioda LED pe placă prin Control Boolean de pe PF.

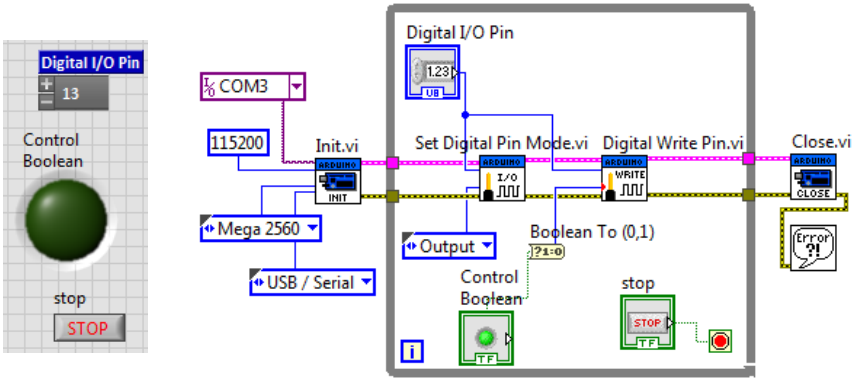


Fig. 8

Funcţia *Init.vi* (Fig. 9) iniţializează legătura cu placa Arduino aflată sub interfaţa *LabVIEW interface for Arduino*.

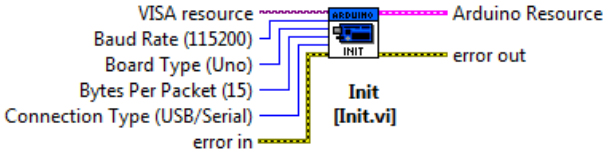


Fig. 9

Funcţia *Set Digital Pin Mode.vi* (Fig. 10) setează pinul specificat din plaja D0→D13 (a plăcii Arduino Uno) pentru *input* sau *output*.

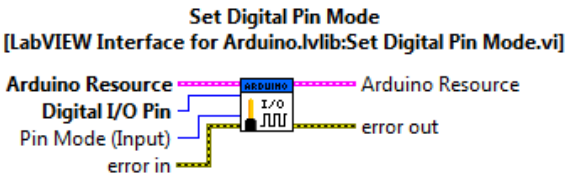


Fig. 10

Funcţia *Digital Write Pin.vi* (Fig. 11) scrie valoarea specificată prin intrarea Value, la pinul indicat din plaja D0→D13 a plăcii Arduino Uno. În prealabil pinul trebuie setat pe output cu funcţia Set Digital Pin Mode.vi.

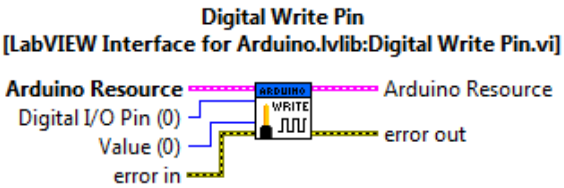


Fig. 11

2.2. Comandaţi intensitatea luminoasă a LED-ului prin apelul funcţiei PWM Write Pin.vi precedat de apelul funcţiei Set Digital Pine Mode.vi.

2.3. Modificaţi aplicaţia considerând evenimente schimbarea valorii controalelor logice *Control Boolean* şi *stop* folosind Event Structure.

3. Comandă buzzer piezoelectric

3.1. Aplicaţia generează un ton, de durată o secundă, la frecvenţa impusă prin control în Panoul Frontal, în mod ciclic. Este folosit KEPO KPR-G3010-6250 Piezo Transducer 1.3 kHz, 35 mm (Fig. 12) conectat la pinul GND digital cu fir negru şi la pin digital 40 (sau alt pin/conector) cu fir roşu.

Fig. 12

Funcţia Tone.vi, în exemplul din Figura 13, generează un semnal dreptunghiular (un ton la frecvenţa dorită plus armonice) cu factor de umplere 50% şi de frecvenţa comandată. Oprirea ciclului While se realizează prin butonul de Stop sau la apariţia unei erori prin câmpul *status* (al structurii *error out)*, caz în care valoarea din câmp devine True.

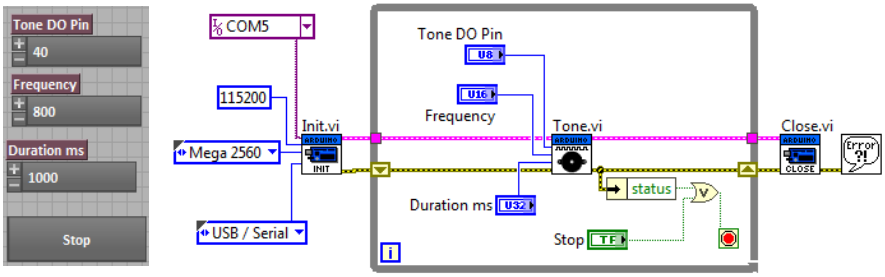


Fig. 13

3.2. Adăugaţi cod pentru a genera gama Do major (Fig. 14) parcursă în sens crescător şi descrescător. De asemenea tabelar (Fig. 15) sunt date frecvenţele notelor din gama (octava) C4 în scala muzicală egal temperată.

Note	Frequency
C4	261.63
D4	293.66
E4	329.63
F4	349.23
G4	392
A4	440
B4	493.88

Fig. 14

Numărul	Nota (solfegiu)	Frecvența (in Hertz)
1	Do	$440/(1.0595)^{9} \approx 261.6$
2	Do ♯	$440/(1.0595)^{8} \approx 277.2$
3	Re	$440/(1.0595)^{7} \approx 293.7$
4	Re ♯	$440/(1.0595)^{6} \approx 311.1$
5	Mi	$440/(1.0595)^{5} \approx 329.6$
6	Fa	$440/(1.0595)^{4} \approx 349.2$
7	Fa ♯	$440/(1.0595)^{3} \approx 370.0$
8	Sol	$440/(1.0595)^{2} \approx 392.0$
9	Sol ♯	$440/(1.0595) \approx 415.3$
10	La	440
11	La ♯	$440 \times 1.0595 \approx 466.2$
12	Si	$440 \times (1.0595)^{2} \approx 493.9$
1	Do	$440 \times (1.0595)^{3} \approx 523.2$

Fig. 15

3.3. Selecţie set tonuri cu Enum

Se completează aplicaţia precedentă cu un selector Enum (Fig. 16) având articolele Vioara, O frecventa şi Octava. Asociat în diagramă se introduce un Case cu trei cazuri (Fig. 17).

Dacă selectaţi *Vioara* se aud repetitiv cele patru note asociate corzilor viorii. La fiecare iteraţie se selectează (prin Index Array) o frecvenţă din tabloul de valori întregi. Când indicele frecvenţei ajunge la valoarea 4 se setează indicele din nou la valoarea 0 (prin Select) şi trece în registrul de transfer.

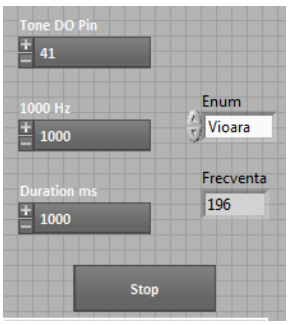


Fig. 16

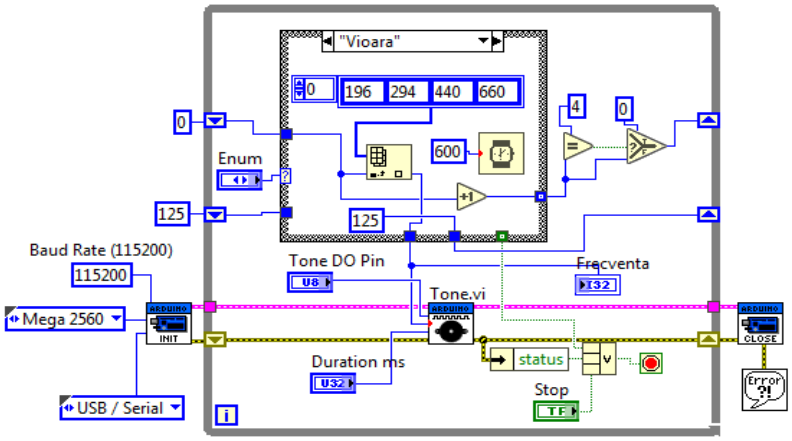


Fig. 17

Dacă este selectat articolul *O frecventa*: se aude un ton la 1000 Hz (plus armonice). Pentru selecţia articolului *Octave*, se aud sunete care îşi dublează frecvenţa pornind de la 125 Hz până la 8000 Hz inclusiv (Fig. 18).

Oprirea ciclului se realizează cu butonul Stop, la apariţia unei erori în funcţiile Tone.vi sau cea de iniţializare dar şi din cazul *Octave* când frecvenţa atinge 8000 Hz.

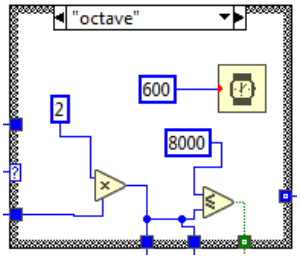


Fig. 18

Măsuraţi frecvenţele generate (2 kHz, 3 kHz,...) cu aplicaţia Advanced Spectrum PRO de pe telefonul mobil (Android) setând reţinerea vârfurilor de frecvenţă prin Peak Hold şi scară liniară a axei absciselor (frecvenţele) în locul celei logaritmice.

3.4. Aplicaţia următoare (Fig. 19) generează un ton la frecvenţa impusă în PF de durată 0.2 secunde. Funcţia Tone.vi este apelată o singură dată.

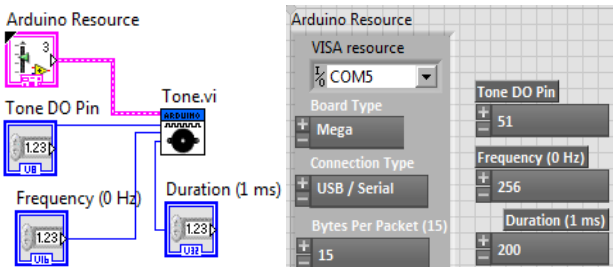


Fig. 19

4. Citire pin Analog In - senzori de umiditate şi de lumină

4.1. Senzorul de lumină tip brick SEN-LUM-03 (Fig. 20) sesizează nivelul de iluminare al mediului. Valoarea iluminării variază liniar între 0 şi 1023. Senzorul nu este calibrat, deci nu se citeşte o mărime corespunzătoare unei unităţi de măsură.

Fig. 20

Pinul (conectorul) de semnal (OUT) al senzorului se conectează la un pin analogic al plăcii Arduino. Pinul de alimentare (VCC) se conectează la pinul 5V al plăcii Arduino. Pinul de masă (GND) se conectează la pinul GND al plăcii Arduino. Se poate măsura voltajul citit pe pin de la senzor la 23 °C şi la 30 °C şi scalat corespunzător.

4.2. Senzorul de umiditate - brick SEN-VRM-08 sesizează nivelul de umiditate al mediului. Într-o cameră obişnuită, valoarea citită pe portul analogic variaza între 900 (mediu saturat cu vapori de apă) şi 300 (foarte uscat). Senzorul nu este calibrat.

Fig. 21

Senzorul prezintă 3 pini: VCC, GND şi OUT. Pinul de semnal OUT se conectează la un pin analogic al plăcii Arduino.

4.3 Pentru montaj se foloseşte un mini breadboard (mBB) pentru conectarea senzorilor. Cinci intrări (orificii) aşezate pe o linie sunt conectate între ele electric. Sunt astfel 17 linii pe o parte şi 17 linii pe cealaltă parte a canalului de răcire a mBB-ului şi perpendiculare pe canal (Fig. 22).

O linie a mBB se conectează la sursa de cinci volţi (5 V) a plăcii Arduino (zona Power). La această linie se conectează pinii VCC (pin poziţie centrală) a celor doi senzori (Fig. 23).

O altă linie a mBB se conectează la un GND a plăcii Arduino (zona Power). De pe aceeaşi linie se conectează cei doi senzori cu pinii GND (pin poziţie marginală).

Fig. 22

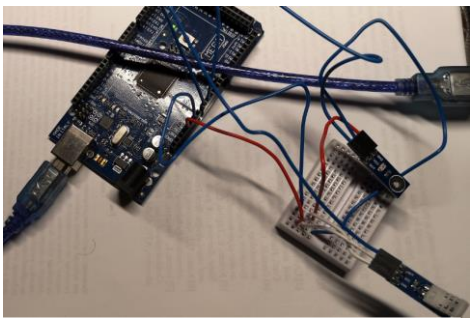


Fig. 23

Pinii de semnal OUT ai senzorilor se conectează la pinii A0 şi A1 ai plăcii Arduino, zona ANALOG IN. În diagrama din Figura 24, se citesc în aceeaşi buclă pinii OUT a doi senzori, cel de umiditate şi cel de lumină.

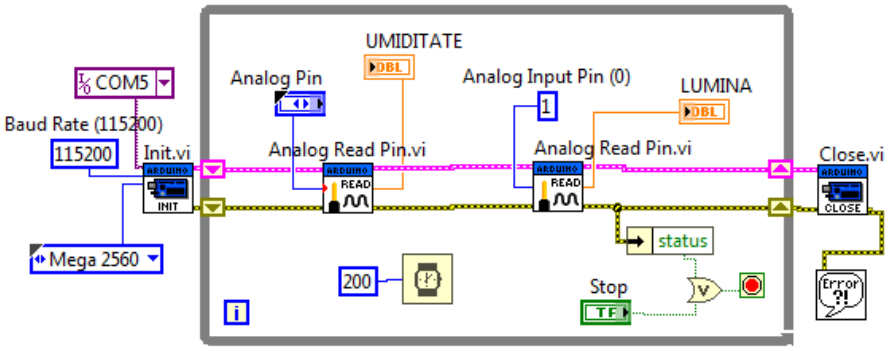


Fig. 24

4.4. Citirea simultană a mai multor senzori analogici, a unui potenţiometru şi un senzor digital.

Se completează aplicaţia precedentă (Fig. 24) astfel încât să conţină achiziţie de la patru senzori analogici: Umiditate - pin A0, Lumină pin - A1, Potenţiometru - pin A2, Temperatură - pin A3 şi unul digital de tip Magnetic Hall.

Potenţiometrul axial sau divizorul de tensiune cu trei terminale (Fig. 25) se conectează astfel:

1. terminalul (picior) de mijloc conectat la un pin din plaja A0,...,A5 (Arduino Uno) din grupul de pini ANALOG IN (fir albastru);
2. pinul lateral al potenţiometrului este conectat la masă GND (fir negru);
3. pinul de la celălalt capăt (lateral) al potenţiometrului se conectează la pin 5V (Arduino). Rotind ştiftul potenţiometrului se modifică tensiunea citită de pe potenţiometru prin pinul Analog In.

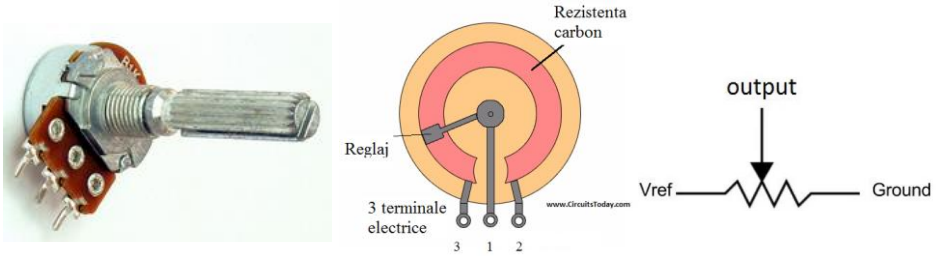


Fig. 25

5. Citirea unui senzor piezoelectric care sesizează vibraţii mecanice - pin Analog In

Senzorul piezoelectric (disc negru, Fig. 26, cod produs: COMP-GEN-16) se poate fixa cu două şuruburi sau prin lipire pe suprafaţa vibrantă (verso disc: Loudity LD BZPN-2312). În paralel cu senzorul piezoelectric se află o rezistenţă de 1.MOhm (Fig. 26) care-l face nepotrivit pentru a funcţiona ca buzzer (pentru buzzer se poate considera COMP-GEN-10, mini difuzor brick Robofun).

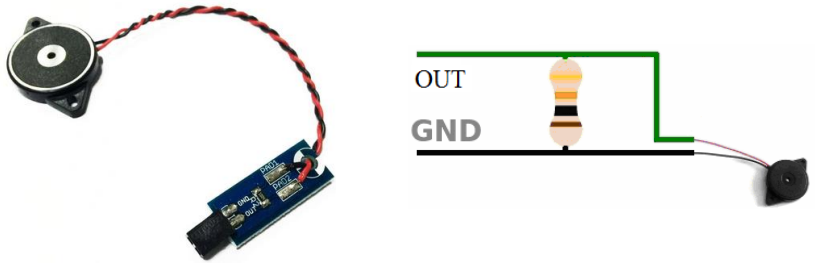


Fig. 26

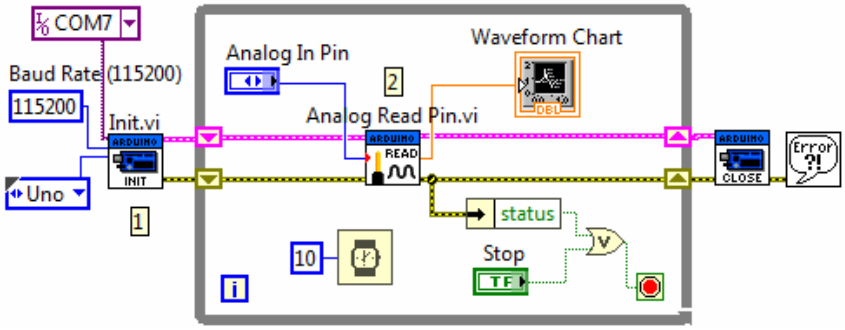


Fig. 27

5.1. Conectarea senzorului la platforma Arduino Uno sau Mega se realizează astfel: terminalul GND–senzor se conectează la GND Arduino Uno/Mega şi terminalul OUT–senzor se conectează la un pin Analog IN (A0, A1,...) al plăcii Arduino.

Prin funcţia Init.vi este iniţializat dialogul cu placa Arduino la rata de transfer specificată, iar valoarea de tensiune este citită de la pinul analogic (Fig. 27).

Rulăm aplicaţia LabVIEW. Se loveşte repetat discul negru al senzorului de vibraţii (brick) cu unghia sau cu un obiect. Materialul piezoelectric din senzor vibrează, iar deformaţia asociată generează tensiune electrică foarte slabă. Tensiunea este citită prin pinul Analog In. În indicatorul grafic Waveform Chart se observă vârfurile de tensiune colectate, câte un vârf la fiecare şoc. Se obţin 100 cicluri sau citiri de tensiune pe secundă prin plasarea funcţiei Wait.vi cu intrarea 10 milisecunde.

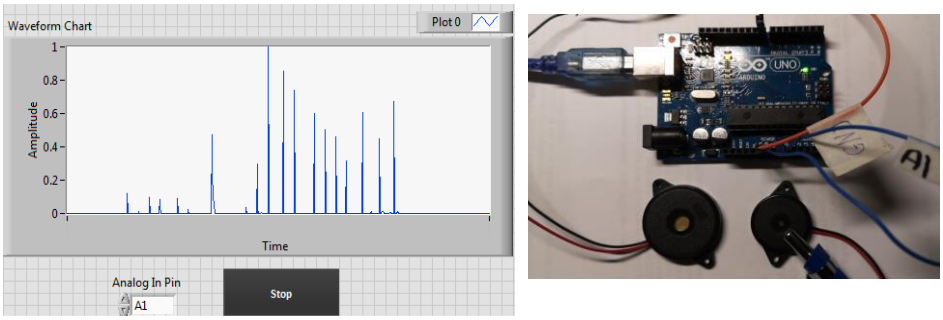


Fig. 28

5.2. Generaţi cu buzzer-ul sunete la frecvenţe tot mai înalte, proporţional cu amplitudinea vârfului de tensiune obţinut la lovirea senzorului piezo de vibraţii (Fig. 28). La fiecare iteraţie vom avea o citire pe pin Analogic de la senzorul de vibraţii şi un apel de Tone.vi cu scriere pe pin digital. Frecvenţa furnizată funcţiei Tone.vi va fi dependentă de tăria impactului, aspect care se poate programa cu un Case (Fig. 29).

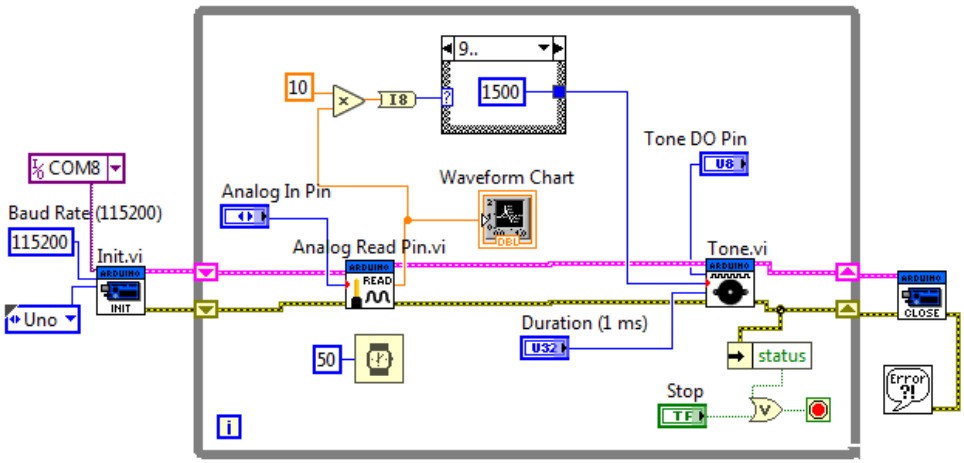


Fig. 29

6. Comandă sens şi turaţie motor cc. cu IC motor driver tip L293D

Se va comanda un motor de curent continuu (de gabarit mic) prin Panoul Frontal LabVIEW din Figura 30, pe baza diagramei din Figura 31. LED-urile 1 şi 2 sunt folosite pentru schimbarea sensului de rotaţie al motorului, iar controlul Slide orizontal pentru variaţia turaţiei.

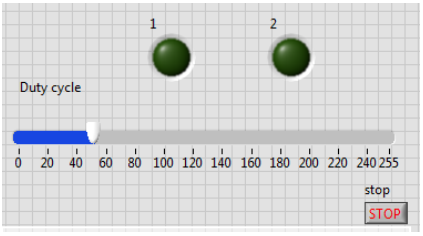


Fig. 30

6.1. Pinii digitali 8 şi 9 ai plăcii Arduino sunt setaţi în mod output. Prin aceşti pini este comandat sensul de rotaţie al motorului, prin valorile numerice 0 sau 1, de la LED-urile control boolean 1 şi 2. Se observă conversia de la valoare logică (T, F) la valoare numerică (1 sau 0). Pinul 45 al plăcii este digital PWM, setat în mod output pentru comanda turaţiei motorului.

Sens 1 rotaţie motor se realizează prin combinaţia: LED1=On, LED2=Off

Sens 2 rotaţie motor se realizează prin combinaţia: LED1=Off, LED2=On

Pentru oprirea motorului ambele LED-uri sunt pe Off.

Reglajul turaţiei motorului se impune prin pinul 45 (digital PWM), prin care se modifică parametrul *Duty cycle* al semnalului dreptunghiular între valorile 0 şi 255. Aceasta corespunde cu variaţia factorului de umplere al semnalului dreptunghiular între 0% şi 100% (Fig. 32).

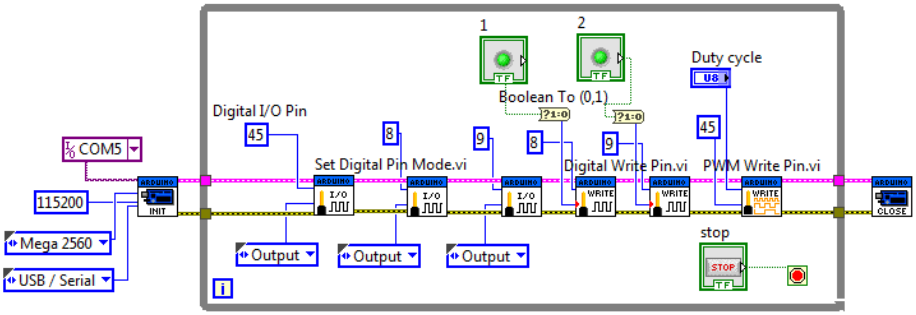


Fig. 31

6.2. Conectare Arduino MEGA şi circuitul integrat L293D

Driverul L293D prezintă 16 pini (Fig. 33) şi poate comanda simultan sensul de rotire şi turaţia pentru două motoraşe. Circuitul realizează amplificarea semnalului redus de control, la un curent mai mare pentru acţionarea motorului.

Pinii 8 şi 9 se conectează la intrările (input) IN1(2) respectiv IN2(7).

Pinul 45 se conectează la intrarea (Enable) ENA(1) (turaţie).

Pinul GND Arduino (Power) conectat la pinii 4 şi 5 (GND1 şi GND2 ale L293D).

Pinul 5V Arduino (Power) se conectează la VCC1 (16) (pentru alimentare L293D) şi de asemenea la VCC2 (8) L293D pentru alimentare motor.

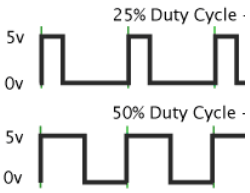


Fig. 32

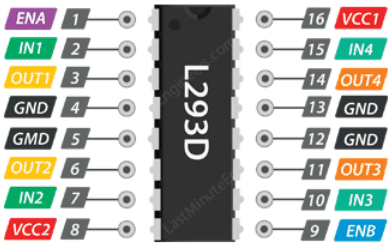


Fig. 33

6.3. Conectare L293D (IC) şi Motor

Pinii OUT1(3) şi OUT2(6) sunt conectaţi la bornele motorului de curent continuu.

6.4. Comandaţi sensul şi turaţia pentru două motoraşe cc (DC), în mod independent, folosind ambele laturi ale circuitului L293D.

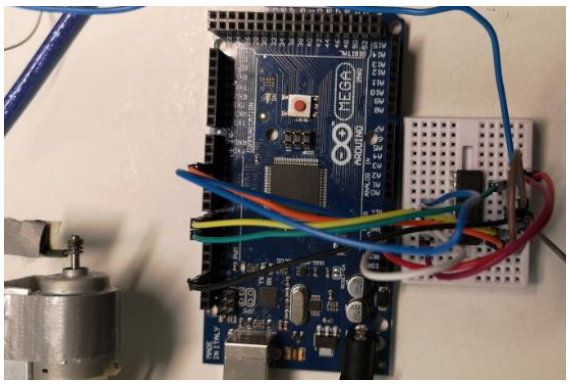


Fig. 34

6.5. Comanda turaţiei unui ventilator cu alimentare de la o baterie 9V

Ventilatorul (cooler) are nevoie de 12 Volţi la alimentare şi nu se poate schimba sensul de rotaţie. Se va alimenta de la baterie 9 V. Dacă L293D se alimentează cu 5 V de la placa Arduino turaţia obţinută la ventilator este mică. În loc de alimentare cu 5 V (de la pinul 5 V secţiunea Power Arduino) la pin VCC2 (controler motor) se va folosi alimentare de la baterie 9 V, astfel (Fig. 35):

borna − a bateriei se conectează la GND controler L293D pin 5 (sau 4) partea motor#1 sau pinul 12 (sau 13) partea cealaltă pentru motor#2.

borna + a bateriei se conectează la pinul VCC2 controler (L293D).

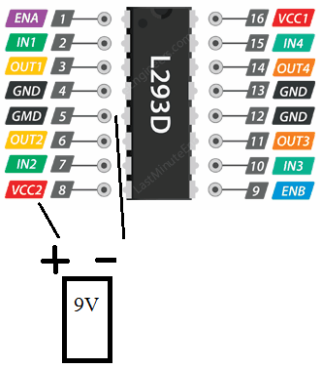


Fig. 35

7. Comandă LED - Arduino Uno - programare pe evenimente

Se transformă programul din Figura 8 pentru comanda iluminării LED-ului de pe placa Arduino Uno din varianta bazată pe ciclare continuă, în varianta de programare pe evenimente. Evenimentul principal este schimbarea valorii controlului *Control LED Boolean* (Fig. 36). Oprirea aplicaţiei se face la evenimentul de tip Value Change şi anume schimbarea valorii butonului de Stop. Ambele evenimente sunt tratate în acelaşi caz al structurii Event (Fig. 37). Contorul ciclului incrementează numai la apariţia unui eveniment.

Fig. 36

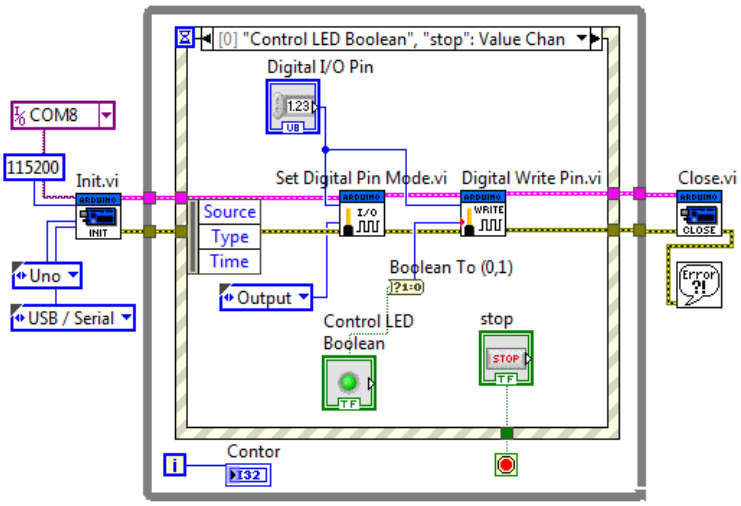


Fig. 37

8. LINX - interfaţă alternativă pentru LIFA

Pachetul LINX (firmware) pune la dispoziţie instrumente virtuale (VIs) LabVIEW pentru programarea platformei Arduino, dar şi pentru alte platforme cum sunt chipKIT, NI myRIO etc. Prezintă funcţii *built in* 'sensor VIs' pentru citirea datelor de la senzori şi funcţii 'peripheral VIs' pentru utilizarea dispozitivelor periferice: digital I/O, analog I/O, SPI, I2C, UART, PWM.

8.1. Prin interfaţa VI Package Manager (paleta Tools) se descarcă şi se gestionează instalarea modulului MakerHub Toolbox (Fig. 38):

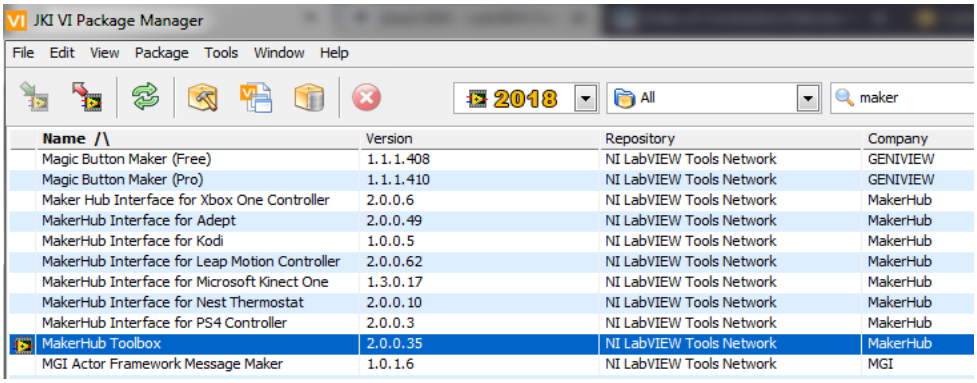


Fig. 38

8.2. Se instalează (Fig. 39) Digilent LINX

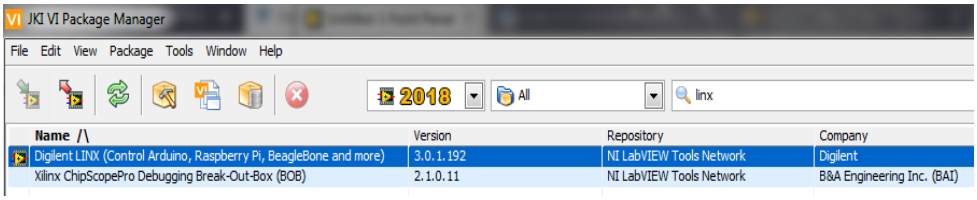


Fig. 39

Sarcina LINX firmware este de a face posibil ca placa cu microcontroler să primească comenzi din LabVIEW, să le interpreteze şi să răspundă la aceste comenzi.

8.3. Din paleta MakerHub pot fi accesate subpaletele de funcţii prezentate în Figura 40, pentru a fi incluse în diagrama bloc a aplicaţiei.

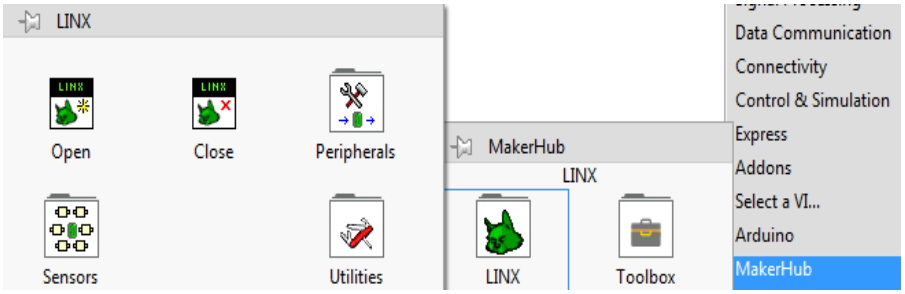


Fig. 40

9. Comanda LED prin pinul digital 13 - Arduino Uno, LINX

Aplicaţia pentru iluminarea şi stingerea LED-ului conectat la pinul digital 13 este reluată în Figura 41, apelând la funcţii din interfaţa LINX.

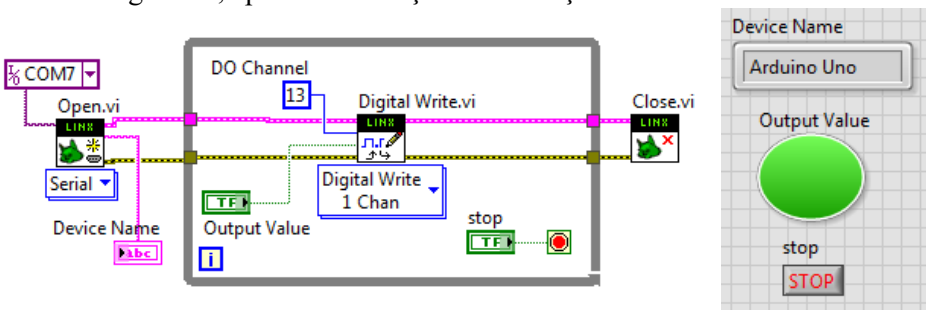


Fig. 41

10. Măsurarea acceleraţiei cu accelerometrul digital triaxial ADXL345

10.1. Conectare accelerometru 3 axe ADXL345 la Arduino Uno, LINX

ADXL345 măsoară acceleraţia (statică) gravitaţională şi acceleraţia dinamică generată la variaţia de viteză (pe trei axe ortogonale x, y, z, indicate pe modul), cu posibilitate de selecţie a domeniului de măsură +/-2g, +/-4g, +/-8g şi +/-16g, unde g este acceleraţia gravitaţională. Acest senzor este întâlnit în diverse dispozitive mobile (telefon mobil, jocuri, instrumente medicale şi industriale etc.). Rezoluţia semnalului de acceleraţie la ieşire este implicit pe 10 biţi, dar se poate seta diferenţiat pentru fiecare domeniu plecând de la 10 biţi pentru +/-2g şi ajungând la 13 biţi pentru domeniul cel mai larg de măsură +/- 16g. Transferul datelor între senzor şi microcontrolerul Arduino UNO se realizează prin interfaţa de comunicaţie serială sincronă I2C (sau prin interfaţa SPI). Interfaţa I2C se realizează hard prin două fire de semnal: SDA (Serial DAta - linia pentru trasmisia serială a datelor) şi SCL (Serial CLock - linia pentru sincronizarea trasmisiei de date).

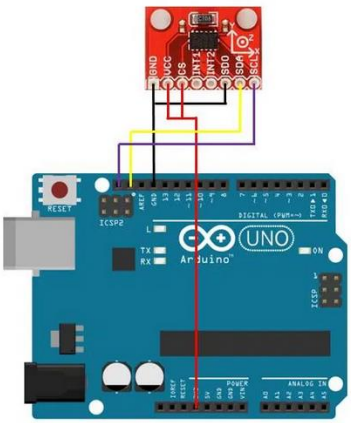


Fig. 42

Pentru placa Arduino Uno (sau 100% compatibile) se vor folosi pinii: A4 pentru SDA (data) şi A5 pentru SCL (clock). Se pot folosi şi pinii poziţionaţi după AREF (spre butonul de Reset) în ordine SDA şi SCL.

Pentru placa Arduino Mega: SDA este pinul 20 şi SCL este pinul 21.

Se vor conecta pinii SDA ai plăcii şi ai ADXL345 între ei, şi pinii SCL ai plăcii şi ai ADXL345 între ei. De asemenea, se va realiza legarea pinilor GND de pe accelerometru şi de pe placă, şi se va alimenta accelerometrul (pinul VCC) cu

5 V de la placă (pinul 5 V). Pinii CS şi VCC ai accelerometrului se conectează între ei. Cei opt pini sunt in ordine: GND, VCC – Supply Voltage, CS – Chip Select, INT1 – Interrupt 1 Output, INT2 – Interrupt 2 Output, SDO – Serial Data Output (SPI)/I2C Address Select, SDA – Serial Data Input (SPI)/I2C Serial Data şi SCL – Serial Communications Clock.

10.2. Lansare LINX firmware Wizard

Se va lansa LINX firmware Wizard din Tools/ MakerHUB/LINX (Fig. 43) în loc de load LIFA Base din ARDUINO IDE;

(ARDUINO IDE nu se lansează). Firmware este un pachet de programe conţinute permanent într-un dispozitiv hard (exemplu, EPROM) cu scop de susţinere a sarcinilor prestate de un aparat, cum este în acest caz placa Arduino, iar în general robot de bucătărie, ABS auto etc.

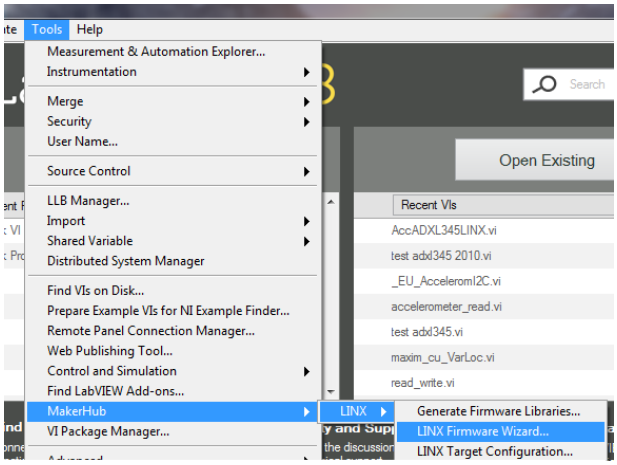


Fig. 43

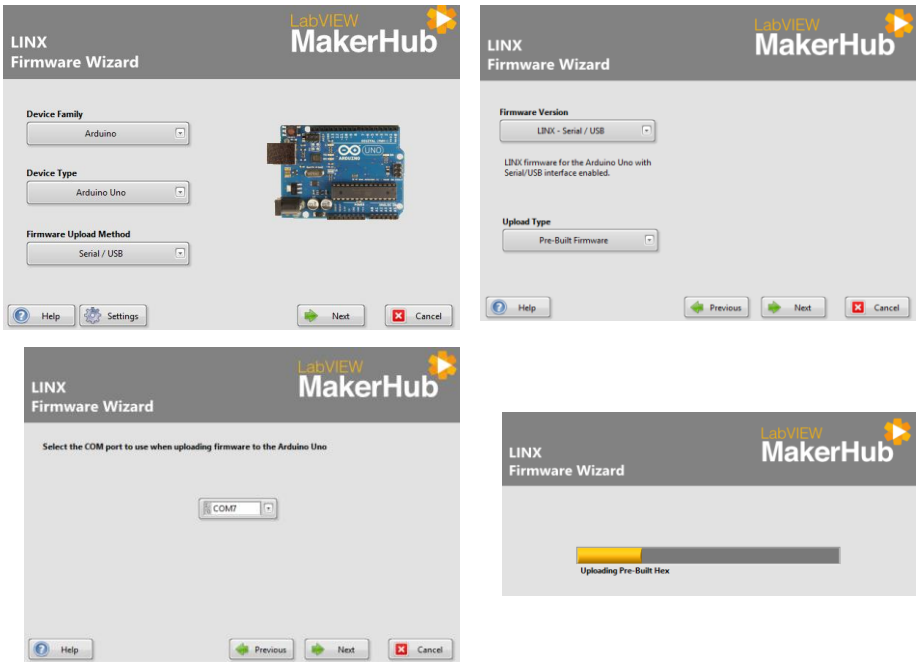


Fig. 44

Lansarea şi instalarea pe placă a interfeţei LINX este prezentată în Figura 44 prin imagini. Se conectează placa Arduino Uno la calculator prin USB, este selectată placa şi portul (exemplu COM7) → Pre-Built Firmware → Uploading...

10.3. Aplicaţia LabVIEW pentru măsurarea acceleraţiei cu ADXL345

Diagrama (Fig. 45) este o înlănţuire de funcţii, iar funcţia Read.vi de citire a acceleraţiei pe cele trei axe este apelată repetitiv. Înainte de ciclu se iniţializează conexiunea LINX cu placa şi se activează accelerometrul (adresa 0x53), iar după închiderea ciclului se închide sarcina de lucru cu accelerometrul şi interfaţa LINX.

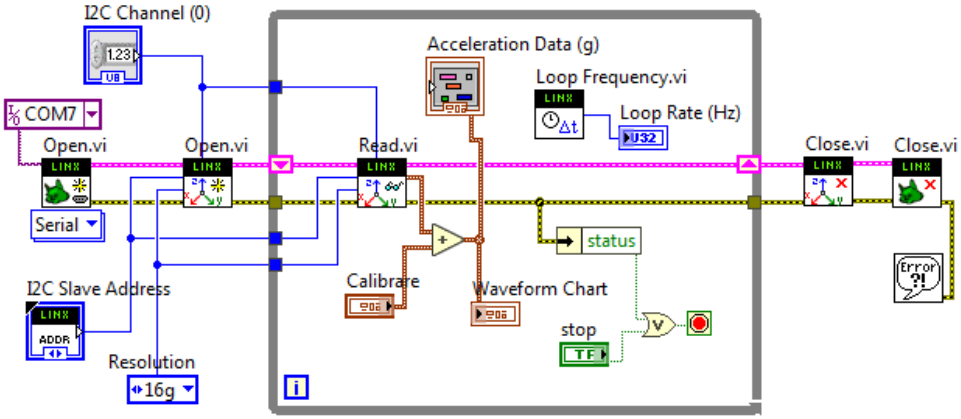


Fig. 45

Funcţia *Read.vi* citeşte acceleraţia pe axele X, Y, Z de la ADXL345 în unităţi g (multipli de g). Se obţine 1g pe axa verticală, în cazul prezent axa Y, având în vedere poziţionarea senzorului pe mini breadboard. În Figura 46 observăm efectul deplasării periodice a accelerometrului pe rând după cele trei axe ortogonale. Se înmulţeşte cu 9.81 pentru a obţine acceleraţia în m/s^2 dacă se doreşte. Se pot introduce corecţii prin structura Calibrare (conţine trei controale sau câmpuri numerice) pentru a obţine amplitudine 0 pe două axe şi 1 pe axa verticală când nu sunt vibraţii la nivelul plăcuţei mBB pe care este poziţionat ADXL345.

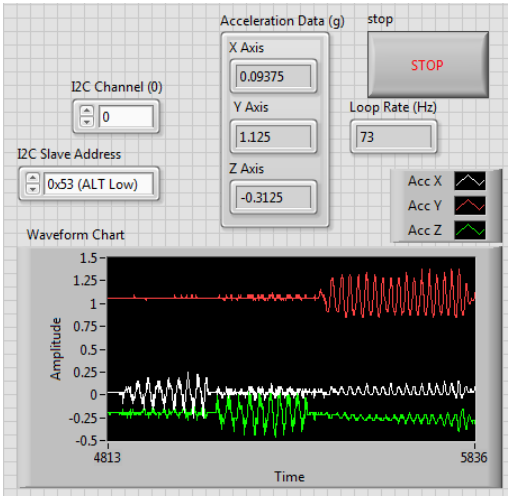


Fig. 46

Funcţia *Read.vi* (Fig. 47) prezintă următoarele intrări:

I2C Channel (0) specifică canalul I2C la care este conectat senzorul ADXL345;
I2C Slave Address specifică adresa slave I2C a senzorului ADXL345;
Resolution specifică rezoluţia curentă (+/-2g, +/-4g, +/-8g şi +/-16g) a senzorului ADXL345. Această valoare foloseşte la convertirea în multipli ai acceleraţiei gravitaţionale (g) a valorii de ieşire din ADXL345;
Error In este linia de erori, fiind o structură cu trei câmpuri (status, code, source) prin care se descrie o eventuală eroare dacă apare.

Funcţia *Read.vi* returnează prin *Acceleration Data (g)* un triplet (cluster) ce conţine acceleraţia pe fiecare din cele trei axe X, Y şi Z.

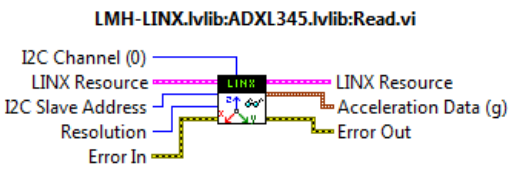


Fig. 47

În corpul ciclului While este apelată repetitiv funcţia *Loop Frequency.vi* pentru a calcula durata fiecărei ciclări, iar prin inversarea duratei rezultă frecvenţa. În diagrama funcţiei (Fig. 48) observăm operatorul Select comandat de *First Call?* şi medierea *punct cu punct* a duratelor cu funcţia *Mean PtByPt.vi*. Registrul de transfer memorează media de la o ciclare la următoarea iteraţie a ciclului exterior (cel în care se apelează funcţia Read.vi). Ciclul While din diagrama funcţiei *Loop Frequency.vi* se (Fig. 48) repetă o singură dată.

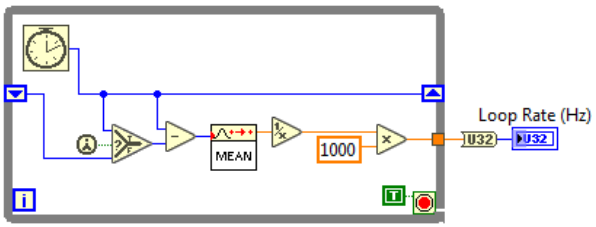


Fig. 48

11. Măsurarea acceleraţiei şi a vitezei unghiulare cu senzorul triaxial MPU6050

11.1. Se conectează Arduino Uno prin cablul USB la laptop

Se realizează interfaţa I2C de comunicare între MPU 6050 (accelerometru + giroscop) şi Arduino UNO. Conectarea pinilor este umătoarea:

Analog In A4 la SDA senzor,
Analog In A5 la SCL senzor,
5V (Power) la VCC senzor,
GND (Power) la GND senzor,
Digital pin 2 la INT senzor.

MPU6050	Arduino	Mega
VCC	5V	5V
GND	GND	GND
SCL	A5	21
SDA	A4	20
INT	2	2

11.2. Se încarcă aplicaţia LabVIEW

Din diagrama bloc se execută Refresh intrare Serial Port (Fig. 49) a funcţiei Open Serial.vi (Fig. 50). Este configurat şi selectat portul COM7 pentru a realiza legătura cu dispozitivul Arduino condus prin firmware LINX.

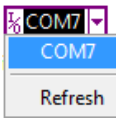


Fig. 49

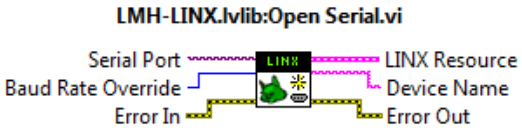


Fig. 50

Funcţia *Open.vi* (Fig. 51) iniţializează senzorul MPU 6050 (adresa hexazecimală a dispozitivului este 0x68), setează sursa clock şi domeniul (full) pentru accelerometru şi gyro.

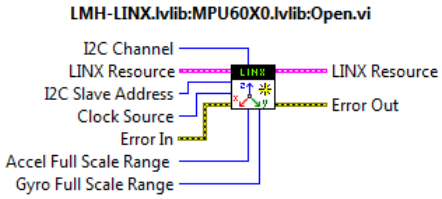


Fig. 51

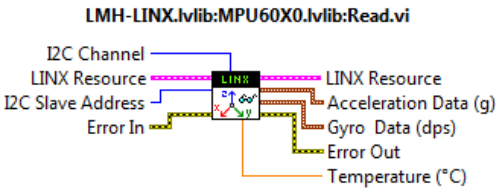


Fig. 52

Funcţia *Read.vi* (Fig. 52) citeşte de la MPU6050 acceleraţia pe axele X, Y, Z în multipli de *g*, rotaţia în jurul axelor X, Y, Z în grade pe secundă şi temperatura de calibrare (°C).

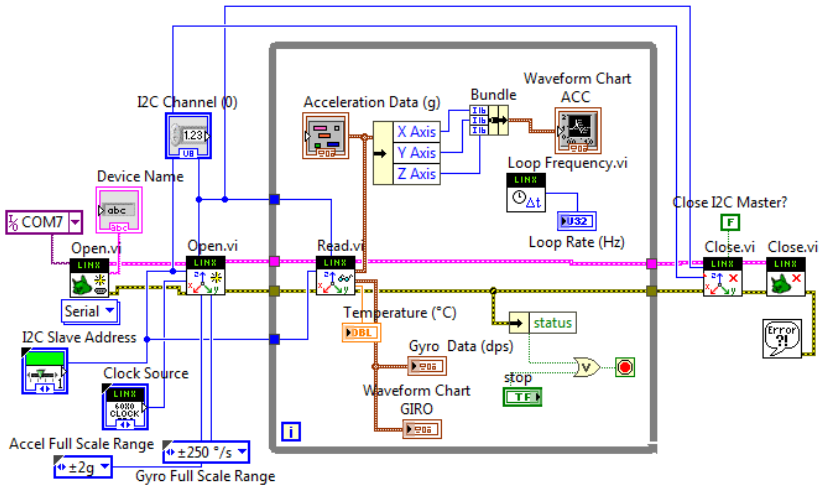


Fig. 53

Observăm în Panoul Frontal: I2C Slave Address 0x68, I2C Channel =0;

În Panoul Frontal din Figura 54 se observă efectul deplasării (vibraţiei) aplicate senzorului, succesiv după cele trei axe (Waveform Chart ACC) şi a rotirii pe rând a senzorului în jurul celor trei axe ortogonale (Waveform Chart GIRO).

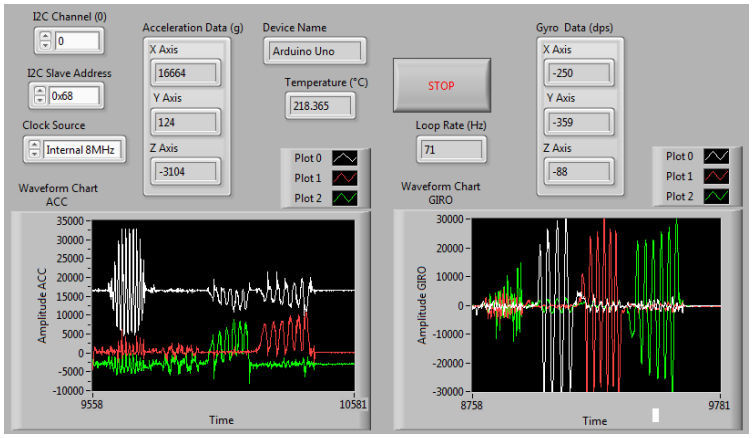


Fig. 54

12. Comanda orientării unui obiect paralelipipedic

12.1. Generarea obiectelor 3D şi sistemul de axe

În porţiunea de diagramă din Figura 55 (diagrama se continuă în dreapta) sunt generate două obiecte grafice de tip paralelipiped (alăturate) şi este ataşat un sistem ortogonal de axe.

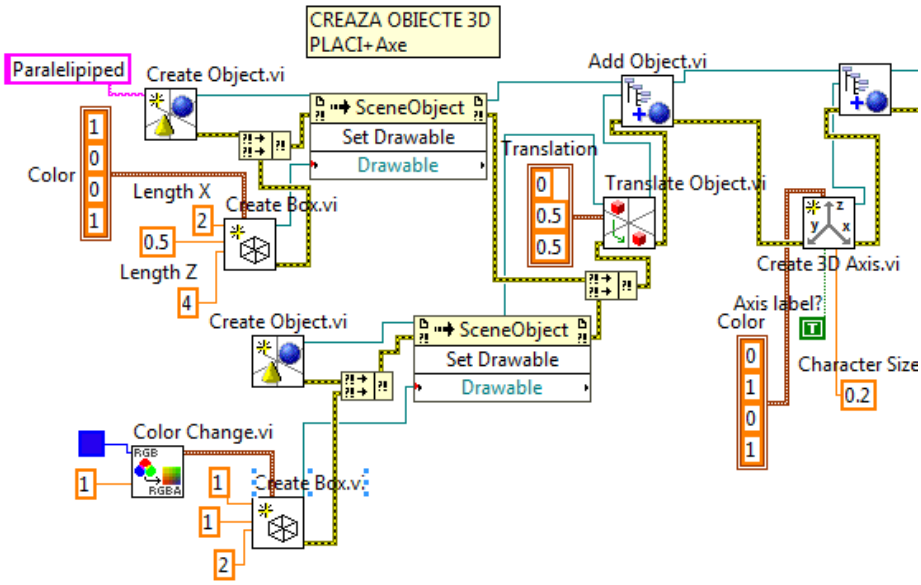


Fig. 55

Obiectele sunt vizualizate în fereastra de tip *3D Picture* din paleta Graph asociată Panoului Frontal (Fig. 56). Se apelează de două ori Create Object.vi pentru a se genera două obiecte grafice. Se particularizează obiectul grafic ca fiind obiect paralelipipedic prin apelul funcţiei Create Box.vi, prin care sunt impuse cele trei cote: (2, 0.5, 4) şi respectiv (1, 1, 2) pentru al doilea paralelipiped. Primul obiect este implicit de culoare roşie, iar al doilea, prin apelul Color Change.vi, va fi albastru. Al doilea obiect este translatat folosind Translate Object.vi astfel încât să nu se intersecteze volumele lor, apoi este adăugat la scenă cu Add Object.vi. Prin apelul Create 3D Axis.vi şi Add Object.vi se ataşează (obiectelor existente) un obiect de tip sistem de coordonate ortogonal. Prin activarea Camera Controller/Spherical (click dreapta pe suprafaţa ferestrei grafice) se oferă posibilitatea de a roti obiectele după dorinţa operatorului.

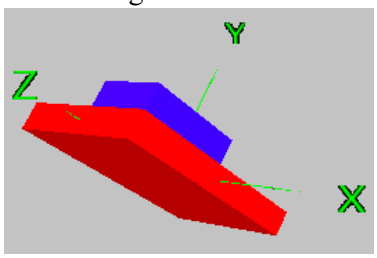


Fig. 56

12.2. Realizarea de rotaţii şi translaţii comandate

Prin funcţiile apelate în zona de diagramă din Figura 57 (continuarea diagramei din Figura 55) se impun valorile vitezei unghiulare citite de la senzorul MPU6050 obiectelor grafice din 3D Picture. Astfel, se poate observa rotirea celor două obiecte simulate în jurul fiecărei axe după cum operatorul uman roteşte în realitate suportul (mBB) pe care este montat senzorul MPU6050. Semnalul preluat de la senzor trebuie filtrat înainte de a fi impus în simulare prin cele trei funcţii Rotate Object.vi. Fiecare funcţie Rotate Object.vi realizează rotirea în jurul axei selectată prin valoarea 1 în constanta cluster *Axis*.

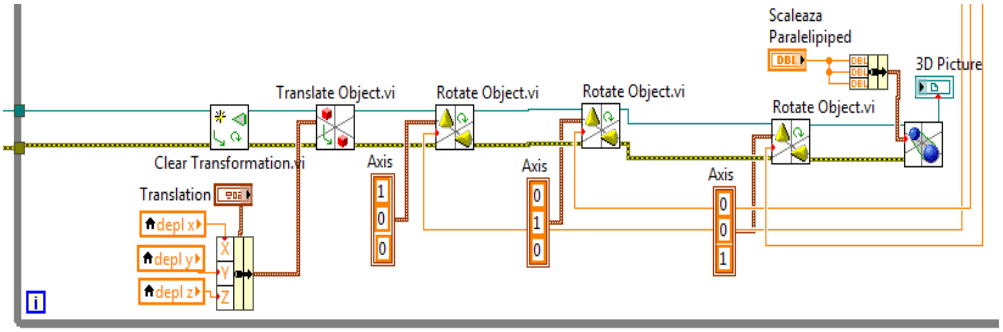


Fig. 57

Valorile unghiurilor de rotaţie provin dintr-o zona a diagramei care nu este vizibilă în diagrama din Figura 57. Acea zonă este similară cu diagrama din Figura 53, unde sunt preluate valorile de la componenta GIRO a senzorului MPU6050 urmând ca semnalele să fie filtrate (trece jos) pentru a elimina variaţia la frecvenţe înalte şi zgomotul din semnalele senzorilor. Prin controlul *Scalează Paralelipiped*

este mărită, respectiv micşorată imaginea din 3D Picture. Se poate adăuga şi translaţie corpului în urma unei duble integrări a semnalului de acceleraţie pe fiecare axă.

12.3. Filtrarea semnalului achiziţionat - filtrul exponenţial

Filtrul exponenţial propus este simplu fiind de tip trece jos (low pass). Este folosit pentru filtrarea semnalului contaminat cu zgomot achiziţionat de la un senzor. Filtrul permite trecerea frecvenţelor joase şi atenuează frecvenţele înalte din componenţa semnalului. Filtrul prezintă un parametru de reglaj *alfa*. La calculul eşantionului curent y(k) (din semnalul filtrat) participă eşantionul precedent y_prec (semnal filtrat) şi eşantionul curent din semnalul nefiltrat (x) conform relaţiei:

$$y(k)=alfa*y(k-1)+(1-alfa)*x(k)$$

Valori mici pentru coeficientul de netezire *alfa* în intervalul [0.1 – 0.2] favorizează ponderarea lui x(k) cel curent, observându-se schimbări rapide ale lui y(k) în semnalul filtrat, şi defavorizează valoarea veche.

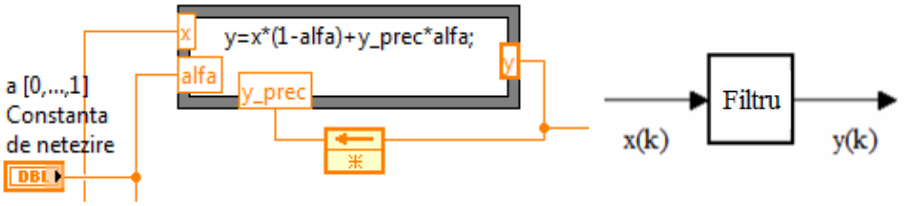


Fig. 58

Valori mari pentru *alfa* (0.8-0.99) favorizează valoarea veche în defavoarea schimbărilor rapide proprii lui x(k) (eşantionul curent nefiltrat).

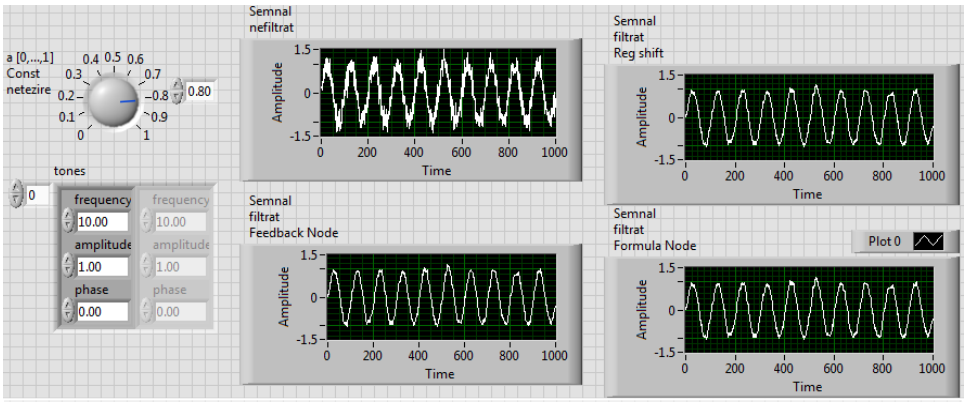


Fig. 59

Efectul aplicării filtrului, din Figura 58, se poate observa pentru semnalul simulat sinusoidal la care se adăugă zgomot şi pentru semnalul achiziţionat de la senzorul MPU6050, componenta de rotaţie.

12.4. Să se genereze cod grafic în diagrame pentru Panourile Frontale din Figurile 59 şi 60.

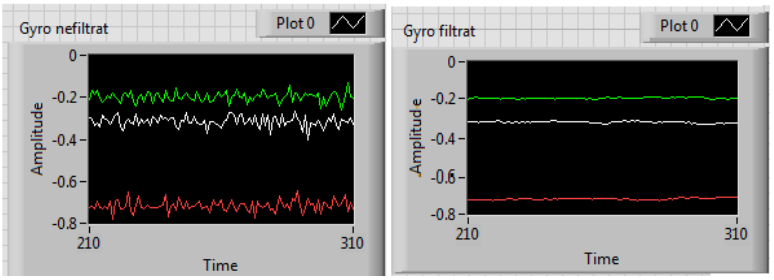


Fig. 60

13. Comandă servomotor - poziţionarea unghiulară a axului

În următorul experiment este folosit un servomotor tip Tower Pro micro Servo 9g (SG90 servo) foarte popular (hobby servo), de mici dimensiuni 23 × 12.2 × 29 mm, paralelipipedic, masa motorului fiind 9 grame, iar preţul servomotorului foarte convenabil (Fig. 61). Se urmăreşte comanda poziţiei unghiulare a axului motorului cu servomecanism între limitele de operare ale servomotorului, asemănător unui volan de automobil, cu posibilităţi de schimbare a sensului de rotire. În compoziţia servomotorului intră un motoraş de curent continuu, un controler şi un senzor de poziţie/ potenţiometru ataşat la axul de ieşire pentru a simţi poziţia, pentru feedback. La alimentarea motorului cu 4.8 V se obţine o viteză de rotire de 60° în 0.12 secunde.

Fig. 61

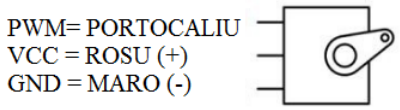


Fig. 62

Conectarea servomotorului la placa Arduino se face prin trei fire (Fig. 62), astfel: firul portocaliu se conectează la un pin digital PWM Arduino, fiind firul de control, firul roşu se conectează la pinul sursă 5V a plăcii, iar firul maro la un pin de masă (GND) al plăcii.

Semnalul PWM dreptunghiular de comandă are o frecvenţă de 50 Hz rezultând o perioadă (T) de 20 milisecunde (T=1/f). Lăţimea pulsului de tensiune de 600 µS (0.6 mS) poziţionează axul la un unghi de

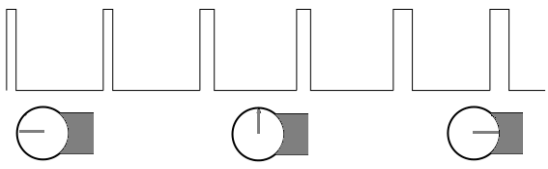


Fig. 63

început de cursă, la zero grade. Cu creşterea lăţimii pulsului semnalului dreptunghiular (Fig. 63), axul se roteşte ajungând la mijlocul cursei unghiulare (90°) la o lăţime a pulsului de 1500 microsecunde (1.5 mS). La lăţimea pulsului de 2.4 mS axul este rotit la un unghi maxim, aceasta fiind cealală limită a cursei axului motorului, la aproximativ 180°. Se poate privi şi poziţionarea la mijlocul cursei ca fiind unghiul de zero (1.5 mS), iar de o parte şi de alta a poziţiei de zero poziţionarea se face la lăţimile de 0.6 mS, respectiv 2.4 mS. Astfel, variind durata în timp a pulsului de tensiune în limitele 0.6 mS,..., 2.4 mS poziţionăm axul în plaja 0,...,180°. Adesea cursa unghiulară a axului motorului este mai mică de 180°, influenţată fiind de roţile dinţate din plastic şi potenţiometrul existente în montaj.

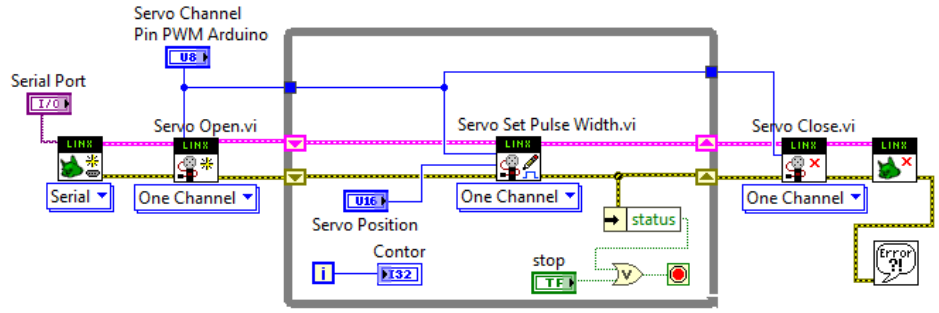


Fig. 64

Curentul absorbit pentru alimentarea la 5 V este dependent de sarcină. Acesta variază de la 10 mA la mersul în gol, la 100 mA până la 250 mA în funcţie de sarcină (maxim 360 mA la ax blocat). Folosirea unui breadboard între placa Arduino şi servomotor asigură o mai bună izolare electrică a plăcii faţă de motor.

Se poate folosi o sursă externă pentru alimentarea plăcii, aceasta fiind necesară la comanda simultană a mai mult de două servomotoare de tipul menţionat. Putem astfel conecta o baterie de 9V la mufa jack a plăcii Arduino.

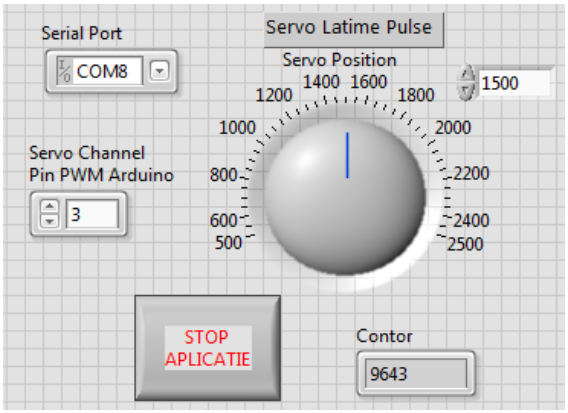


Fig. 65

La poziţionarea butonului rotativ din PF (Fig. 65) pe 600, 1500 sau 2400 se va poziţiona axul pe unghiul -90, 0 şi respectiv +90 (sau 0, 90, 180).

Sunt servomotoare digitale pentru rotaţie continuă (Parallax Feedback 360° High-Speed Servo, Feetech FT90R etc., www.pololu.com).

Diagrama aplicaţiei LabVIEW este în Figura 64, iar Panoul Frontal în Figura 65. Funcţiile Open.vi şi Close.vi deschid, respectiv închid canalul de comunicaţie serială cu dispozitivul extern, gestionat prin LINX. Funcţiile Servo Open.vi şi

Servo Close.vi deschid, respectiv închid canalul servo pentru control, unde prin intrarea *Servo Channel* se specifică numărul pinului PWM al microcontrolerului Arduino Uno (pinul ~3 în aplicaţia curentă).

Funcţia Servo Set Pulse Width.vi (Fig. 66) este apelată repetitiv în cadrul ciclului While pentru a urmări modificările aduse de către operator intrării *Pulse Width (µS)* prin care este comandată poziţia unghiulară a axului motorului.

În cazul în care se doreşte comanda mai multor servomotoare, de exemplu patru, se vor folosi patru funcţii Servo Open.vi conectate între ele prin firele LINX Resource şi Error In/Out, urmate în corpul ciclului While de patru funcţii Servo Set Pulse Width.vi, conectate toate în linie şi patru funcţii Servo Close.vi după ieşirea din ciclul While. Pentru fiecare triplet de funcţii Servo Open.vi, Servo Set Pulse Width.vi şi Servo Close.vi se comandă poziţia unui servomotor prin pinul PWM al plăcii şi *Pulse Width (µS)* specific. Fiecare servomotor este astfel comandat independent prin controlul propriu în Panoul Frontal (având terminalul asociat în corpul ciclului), conform cu poziţia unghiulară dorită.

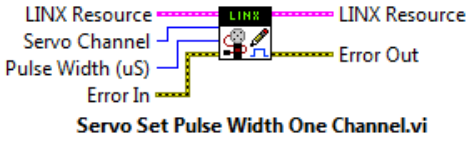


Fig. 66

14. Noţiuni despre platforme Arduino Uno/Mega, intrări/ieşiri semnale

Arduino MEGA 2560 este o placă cu microcontroler construită pe baza procesorului ATmega2560, pe 8-bit cu arhitectură RISC (Reduced Instruction Set Computer), microcontroler *single-chip*. Prezintă 54 pini digital input/output, din care 15 pot fi folosiţi ca ieşiri PWM (Pulse Width Modulation) şi anume pinii 2,3,...,13 (pinii 0 şi 1 sunt pentru tx şi rx) şi pinii 44, 45, 46 sunt, de asemenea, pini output 8-bit PWM comandaţi (cu funcţia analogWrite).

Sunt disponibile, de asemenea, 16 intrări analogice A0,...,A15 şi 4 UARTs (hardware serial ports), un cristal oscilator (clock speed) la 16 MHz, un conector USB (acesta permite conectarea la calculator prin cablu USB), un power jack (permite alimentarea de la un adaptor AC to DC sau de la baterie), un ICSP header, un buton de reset, un LED Built-in: conectat la pinul 13, 5 pini Ground (GND) şi altele (Fig. 67).

14.1. Semnalul digital

Placa cu microcontroler Arduino prezintă intrări şi ieşiri pentru semnal digital. Pinul digital poate avea numai două stări logice True şi False sau valori electrice de tensiune High (5 V) şi Low (0 V) măsurate între GND (ground) şi pinul observat. Primii doi pini (0 şi 1) din totalul de 14 pini digitali (0,...,13) ai variantei Uno, sunt folosiţi şi pentru comunicaţia (printr-un protocol serial) cu plăcile de extensie (shields). Cei doi pini au asociate două LED-uri (RX-recepţie şi TX-transmisie) pe placă, LED-uri care pâlpâie când se face transmitere de date. Placa Arduino prezintă numai intrări pentru semnal analogic. Semnalele din lumea

reală sunt analogice şi în general amplitudinea poate lua orice valoare într-un interval.

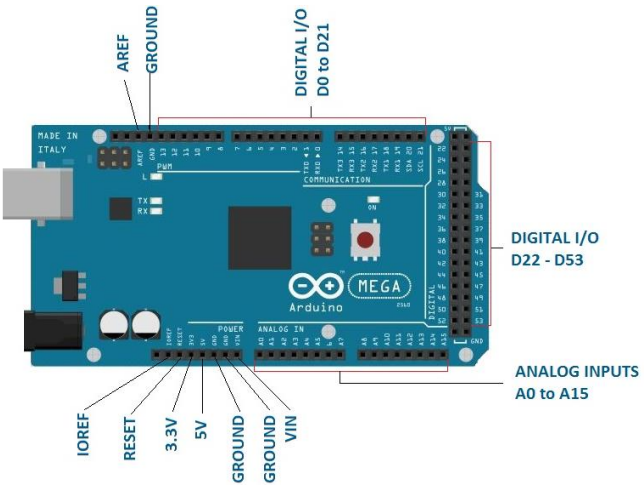


Fig. 67

14.2. Măsurare semnal analogic

Pentru a măsura semnale electrice analogice (tensiunea de intrare) platforma Arduino are un convertor analog→digital ADC (Analog Digital Converter inclus în microcontrolerul ATmega) cu care converteşte semnalul analogic în valori (de amplitudine) discrete (digitale). Pentru aceasta se foloseşte funcţia Arduino IDE: *analogRead*(pin) sau funcţia LabVIEW: *Analog Read Pin.vi*. Aceste funcţii convertesc valoarea de tensiune generată de senzor disponibilă la pinul analogic (între valorile 0 şi 5 V sau 0 şi 3.3 V în funcţie de placa Arduino) în valoare digitală întreagă între 0 şi 1023. Astfel, convertorul A→D este pe 10 biţi (2^10=1024). Rezoluţia de citire a semnalului analogic pentru intervalul de tensiune 0-5 V este 5V/1024 unităţi, rezultând 0.0049 V (4.9 mV) pe unitate. Pentru plăcile cu procesor ATmega (Uno, Nano, Mini, Mega), o citire din semnalul analogic se face în 100 microsecunde (0.0001 s) astfel rata de citire este de maxim 10,000 eşantioane pe secundă.

14.3**.** Semnal PWM, dreptunghiular digital de ieşire

Plăcile Arduino nu au convertor Digital→Analog DAC, dar pot parţial suplini neajunsul prin semnal PWM (Pulse-Width Modulation) care este un semnal digital dreptunghiular (formă de undă) de o anumită frecvenţă (490 Hz, 500 Hz, 980 Hz etc., în funcţie de placă). Astfel, este folosită funcţia Arduino IDE *analogWrite*(pin, value) (*PWM Write Pin.vi* în LabVIEW) unde *pin* este numărul pinului folosit pentru ieşirea semnalului PWM; *value* este număr proporţional cu *duty cycle* a semnalului dreptunghiular (value=0, semnal Low tot timpul; value=255, semnalul este High tot timpul). Astfel, se obţine un semnal modulat în lăţimea pulsurilor de comandă. Pentru Arduino Uno (ATmega328P), pinii PWM (marcaţi cu semnul ~)

sunt: 3, 5, 6, 9, 10 şi 11. Pentru Arduino Mega, pinii PWM sunt: 2,...,13 şi 44, 45 şi 46.

14.4. Tensiune de alimentare placă şi de ieşire

Tensiunea de alimentare şi funcţionare a plăcii Arduino se poate introduce prin portul USB la tensiune de 5 volţi prin care placa se leagă la calculator. Prin mufa Power Jack se poate obţine o tensiune în plaja 7-12 V sau prin pinii Vin şi GND (din zona Power a plăcii) de la o baterie de 9 V.

Microcontrolerul Arduino prezintă două ieşiri (zona Power a plăcii), una cu tensiune de 5 V şi alta cu tensiune de 3.3 V (curentul maxim oferit fiind de 50 mA) pentru alimentarea unor senzori.

BIBLIOGRAFIE

1. Anghel, T., Programarea plăcii Arduino, Editura Paralela 45, 2020.
2. Bishop, R., Learning with LabVIEW 6i, Prentice Hall, 2001.
3. Blokdyk, G., LabVIEW A Complete Guide - 2021 Edition Kindle Edition, 2021.
4. Bress, T., Effective LabVIEW Programming, NTS Press, 2013.
4. Cottet, F., Ciobanu, O., Bazele programării în LabVIEW, Editura Matrix Rom, Bucureşti, 1998.
5. Fadhil, A., Lecture Notes in LabVIEW and Data Acquisition, Kindle Edition, 2021.
6. Hedeşiu H., Munteanu R. Jr., Introducere în programare grafică instrumentală, Editura Mediamira, Cluj-Napoca, 2003.
7. Isen, F., DSP for MATLAB and LabVIEW II: Discrete Frequency Transforms (Synthesis Lectures on Signal Processing), 2008.
8. Johnson, G., LabVIEW Power Programming, McGraw-Hill, NY, 1998.
9. Larsen, R.W., LabVIEW for engineers, Prentice Hall, 2011.
10. Lupea I., Roboţi şi vibraţii, Editura Dacia, Cluj-Napoca, 1996.
11. Lupea I., Lupea, M., Limbajul C, teorie şi aplicaţii, Casa Cărţii de Stiinţă, Cluj-Napoca, 1998.
12. Lupea I., Măsurători de vibraţii şi zgomote prin programare cu LabVIEW, Casa Cărţii de Stiinţă, Cluj-Napoca, 2005.
13. Lupea I., LabVIEW - Programare Grafică, Editura Risoprint, Cluj-Napoca, 2008.
14. Lupea, I., The length of flexible coiled tubes by observing the sound propagation time, Acta Technica Napocensis, AMME, Vol. 62, Issue IV, 2019.
15. Lupea, I., The length of flexible coiled tubes measured by using a sound card and the FRF, Acta Technica Napocensis, Series: AMME, Vol. 62, Issue III, 2019.
16. Lupea, I., The Length of Flexible Coiled Tubes Measured by Using Standing Waves, Acta Technica Napocensis, Series: Applied Mathematics, Mechanics, and Engineering Vol. 61, Issue III, September, 2018, pages 297-300.
17. Lupea, I., Sound absorption coefficient measurement set-up by using LabVIEW and a simple impedance tube, Acta Technica Napocensis, Series: Applied Mathematics, Mechanics, and Engineering, Vol. 62, Issue IV, 2019.
18. Maier, V., Maier, C.D., LabVIEW în Calitatea Energiei Electrice, Editura Albastra, Cluj-Napoca, 2000.

19. Morris, A., Langari, R., Measurement and instrumentation: theory and application, 2012.
20. Paton, B., Sensors, transducers, and LabVIEW, Prentice Hall, 1999.
21. Ricardo A., LabVIEW: A Flexible Environment for Modeling and Daily Laboratory Use, 2021.
22. Sabou, S.P., Îndrumător laborator microcontrolere, U.T.Press, Cluj-Napoca, 2018.
23. Schwartz, M., Manickum, O., Programing Arduino with Labview, Packt Publishing, 2015.
24. Travis, J., LabVIEW for everyone: graphical programming made easy and fun, Prentice Hall, 2007.
25. ** www.arduino.cc
26. ** www.fritzing.org
27. ** www.hackster.io
28. ** www.intorobotics.com
29. ** www.instructables.com
30. ** www.mschoeffler.com
31. ** www.makerguides.com
32. ** www.ni.com
33. ** www.ni.com, Data Acquisition Basics Manual, 1998
34. ** www.ni.com/academic
35. ** www.ni, LabVIEW Analysis Concepts, 2004
36. ** www.ni.com, Getting Started with LabVIEW, 2013
37. ** www.ni.com, LabVIEW User manual, 2003
38. ** www.ni.com, LabVIEW Advanced Signal Processing Toolkit, 2005
39. ** www.ni.com, LabVIEW Signal Processing Course Manual, 1997
40. ** www.ni.com, LabVIEW Order Analysis Toolset User Manual, 2005
41. ** www.ni.com, LabVIEW Fundamentals, 2005
42. ** www.ni.com/docs/en-US/bundle/labview/page/lvconcepts/front_oolv.html LabVIEW Object-Oriented Programming, 2022
43. ** www.the-diy-life.com