LabVIEW1. jegyzőkönyv

Pánicsics Hedvig Diána
Mérőpartner: Pap-Takács Noémi
Mérés ideje: 2019.02.27 14:15-17.00
Mérés helye: Pázmány Péter Katolikus Egyetem, Információs Technológiai és Bionikai Kar
1083 Budapest, Práter utca 50/a
pancsics.hedvig.diana@hallgato.ppke.hu

Mérőeszköz adatai: Számítógép, LabVIEW software

I. LabVIEW

A LabVIEW a National Instruments mérésautomatizáló programcsomagja, mely megkönnyíti a mérésadatgyűjtő hardverelemek rendszerbe integrálását és lerövidíti a programozási időt a grafikus programszerkesztővel. Segítségével gyorsan begyűjthetők és megjeleníthetők be/kimeneti eszközök adatai. A mérés során ezzel a software-vel ismerkedtünk meg, sajátítottuk el alapjait. Az előzetes és előadáson megismert esközökkel kiadott feladatokat késztettünk el.

II. A mérési feladatok (1-3)

1) Első feladat
(*Az előlapon elhelyezett tetszőleges típusú, nyomógombot bekapcsolva gyulladjon meg egy négyzet alakú kék LED. A m/s mértékben beadott sebességet írja ki és mutassa meg egy tetszőleges formájú kijelző km/h egységben.*)

Az első lépéskén a Front Panelben hozunk létre a controls palette-ből választva egy nyomógombot (push button) madj egy LED-et,ennek a színét megváltoztattuk a Tools palette-ben kékre. Következő lépésben a Block Diagramban létrehoztunk egy structure-t egy while-loopot ami a folyonos működését az izzónak tette lehetővé. A Loopban elsőként egy Stop vezérlőt hozunk létre hogy kiléphessünk a program futása közben, majd összekötöttük a LED-et és a kapcsolót. A programot futtatva minden az elvártak szerint működött.
A megadott sebességet, hogy m/s-ből km/h-ba váltsuk át az alábbi összefüggést kellett felhasználni:

1 méter/secundum = 3.6 kilométer/óra

Kiválasztottunk a Functionsból a bemeneti m/s értéknek Numeric box-ot továbbá a 3.6-os konstans átváltó értéknek és a kimeneti értéknek is. A bementi és a konstans értéket összekötve és egy szorzó vezértlőn átvive megkaptunk a kimeneti értéket.
Ebben a feladatban a két rész teljesen függetlenül működött, mivel semmifajta kapcsolat meg állt fent köztük. Az égő lámpa nem változtatott állapotán és a váltót sem befolyásolta az izzó helyzete.

2) Második feladat:
(*Alakítsa át és mentse el új néven az 1. pont feladatában a nyomógombot kapcsolóra, majd módosítsa a programot olyanra, hogy csak akkor történjen mértékegység átszámítás, ha a kapcsoló ki van kapcsolva. Tehát kikapcsolat kapcsoló esetén az eredményt km/h-ban bekapcsolt kapcsoló esetén m/s-ban kell kiírni. A mértékegység átszámítás be/kikapcsolt helyzetéről jelenjem meg információ az előlapon*)

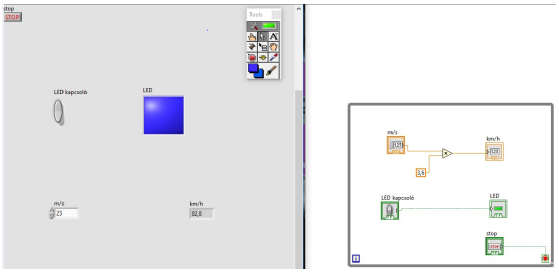

Fig. 1. Első feladat

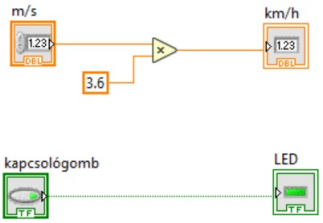

Fig. 2. Első feladat Block Diagram

Az első feladatot átszerkesztettük, hogy a nyomógomb helyett egy kapcsológombot alkalmaztunk melyet szintén a contorls paletteböl választottunk és a while loopon belül úrja kötöttük. A érétkegység átváltót egy subVI-ba mentettünk úgy,hogy a kívánt részletet kijelölve az EDIT fül alatt létrehoztuk az átváltó subVI-t. Ez egy ikont eredményezett a Block Diagrammban, amely ugyanolyan be és kimentettekkel rendelkezik mint az eredeti rendszer. Ahhoz, hogy a kapcsoló állása befolyásolja az átváltást egy Case structure-t kell létrehozni, mely bemenetéhez a kapcsológombot kötöttük. A kapcsoló gomb két fajta értéket ad true/false a case struct-nak is tehát kettő ága van. Az eredeti m/s érték is a C.S.-en kívül található.
Az elágazás igaz felében az értékkel nem történik semmi változás. Létrehoztunk még a belsejében egy string-et is mely a mértékegységet adja meg (ebben az ágban m/s).
A hamis felében található szintén egy string ami km/h-t ad mértékegységnek és ebben a félben található az átváltó subVi is.
A C.S.-en kívül található a kimeneti eredmény(összekötve az átváltóval/az eredeti

eredménnyel), a mértékegység helye(összekötve mértékegységgel) és a LED (összekötve a kapcsolóval amikor nem váltunk).

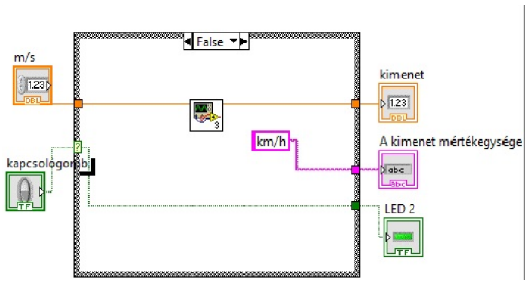

Fig. 3. második feladat

3) Harmadik feladat:
(Készítsen egy kockajáték szimulációt mely egy nyomógombot segítségével hozható működésbe. A nyomógomb megnyomására egyszer kell két független kockával dobni és az eredményeket kijelezni. Ha az eredmények összege 7 akkor gyulladjon ki egy kör alakú zöld LED.)

A harmadik mérés során létrehoztunk egy gombot (egy ok button) és bekötöttük egy case struct-ba, a structon belül: kettő subVI-t készítettünk amelyekben egy random numer generator van amely [0,1) intervallumban hoz létre valós számokat, létrehoztunk mellette egy numeric konstansot is melyebe a 6-os szorzót tároltuk. Ezt a két számot összeszorozva és felső egész részüket véve kaptunk meg az eredményünket. Így már az eredmény mindenképpen a [1,6] intervallumba esik és valószíműségük is megegyezik miven nem vész el számtartomány. Ezt a két subVI-t összeadva kapjuk meg a dobás összeredményét amit összehasonlítotunk az egyenlőségre nézve egy numeric konstansal (7). Ez az egyelőség a case structon belül van behuzalozva egy LEDhez így ha a dobásunk össszege 7 egy LED felgyullad amit előtte megváltoztattunk kerekre és zöld színűre.

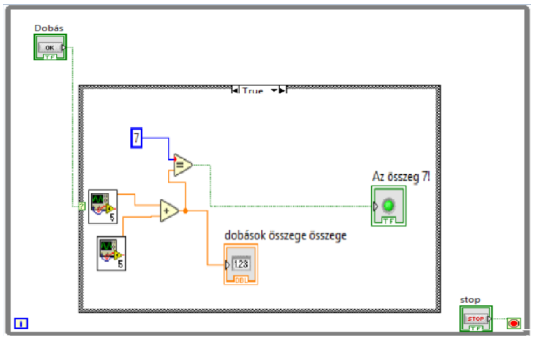

Fig. 4. Egy dobás

III. LabVIEW 2. - Második mérés(02.27), Feladatok(4-6)

4) Negyedik feladat:

(A 3. pontban elkészített dobókocka szimulációt subvi-ként felhasználva készítsen programot mely legalább 10000-szer dob a két kockával és megjeleníti az eredmények gyakoriságát, azaz, azt, hogy hány alkalommal lett az eredmény 2; 3; 4,... 12.)

Ezen feladat során a már meglévő dobás subVI-okat használtunk fel, melyeket egy 10.000 iteráció számú For loop-ba (ami a 10 000 dobást jelenti) helyezve összeadtunk kettesével (kettő dobás). A Loop-on kívülre tettünk egy numeric constansot ami a l0000 ciklusétr felel, továbbá a kimenethez huzalozunk egy histogramot.(A histogram a Funcions palette, mathematics: prob and stat füle alatt taláható.) Ezt a histogramot kötöttük össze egy 11 értékű konstansal ami a histogram beosztását adja meg. A kimenetéhez huzalozva van egy Histogram Graph mellyel az értékek eloszlását mutatjuk ki. A gráf az normál eloszlást mutat mivel a dobásoknál a közbülső értékeket(minél közelebb a 7-hez) egyre többféle módon állíthajuk elő.

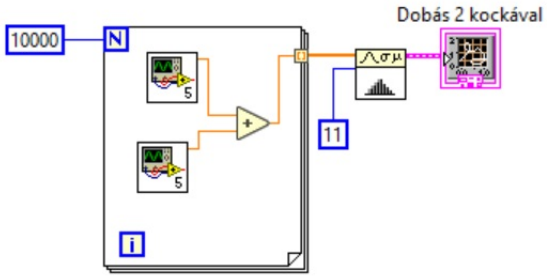

Fig. 5. Negyedik feladat: Block diagram

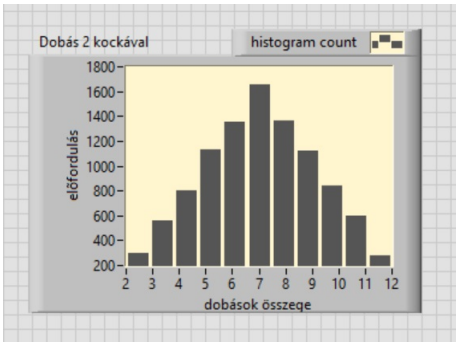

Fig. 6. Negyedik feladat: Histogram

5) Ötödik feladat:
(A 4.pont szerinti feladatot végezze el úgy, hogy a dobókocka 1..6 értéke helyett folytonos 0..1 intervallumot használ és a gyakoriságot legalább 6 független érték összegéből képezi.)

A feladat során a 4. feladat alapján készítettük el, ám itt egy folytonos (0,1) intervalumból vettünk véletlenszerűen számokat amiket hasonló módon oszlopdiagrammal ábrázoltunk. A front panelre helyeztünk két numeric control-t amelyekkel az eset lefutását és azt, mennyi random számot szeretnénk vizsgálni betudjuk állítani. Mindkét controlhoz létrehoztunk FOR loop-ot, a belsejébe egy random number generator-t. Ezt a generatort a masodik for loop-ba (ami a kísérlet számot adja meg) kötöttük be egy szummázó operátorba.

A loop-okon kívül huzaloztunk egy histogramot egy contol-lal ami a felbontástát adja meg. Ezt a histogramot ábrázoljuk az előre megadott felosztás szerint egy gráfon amin beállítottuk az autoscale tulajdonságot ami az automatikus igazításért felel (így lehet állítani a felbontáson és a grafikon követi.) Egy haranggörbe rajzolódik ki a lefutás után a grafikonra.

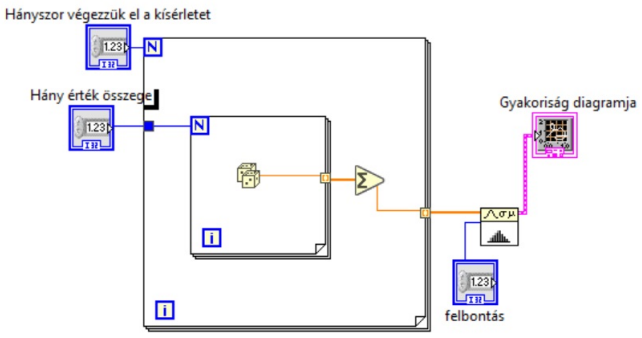

Fig. 7. Ötödik feladat: Block diagramm

6) Hatodik feladat:
(Szimuláció segítségével választható módon állítson elő szinusz, négyszög, fűrész és háromszög alakú jeleket. Az előállított jelek paraméterei (amplitudó, offszet, frekvencia) legyenek beállíthatók. Az előállított jeleket értelmezze úgy mintha egy feszültségforrás jele lenne. A számítás során a jelalakot mintánkét egy adott tömbben kell elhelyezni, és annak értékeit kell ábrázolni. A megjelenítés során ügyeljen arra, hogy futtatás közben sem ugrálhat össze vissza az ábra. A jegyzőkönyvben rögzítse, hogy milyen feltételek mellett milyen adatokkal határozta meg és ábrázolta a számítógépben a feszültségeket. A tömb felhasználásával kell kiszámítani az egyes jelalakokra jellemző effektív feszültséget. A kiszámított eredményt hasonlítsa össze az elmélet értékekkel és határozza meg az esetleges eltérés okát.)

A feladatban négy fajta jelet hoztunk létre változtatható paraméterekkel. Első ként combo egy boxot (control palette: string and path fül alatt) helyeztünk el, melyben kiválazthajuk egy legörbülő menüből milyen fajta jelet szeretnénk előállítani. Három numeric controllert létrehoztunk a paramétereknek (amplitudó, offset, frekvencia). A legördülő menühöz készítettünk egy case structot négy darab ággal mindegyik jelhez. Az ágakban elhelyeztünk egy simulate signal-t (waveform-¿analog waveform-¿generation), ahol minden ágban külön kiválasztottuk a kívánt jelet (sine,sawtooth, square, triangle), ebbe huzaloztuk be a paramért változókat és kivezettük egy dynamic data type to array váltón keresztül az értéket egy tömbbe. A case stucton kívülre helyeztünk egy waveform graphot a jeleket mutatja meg. Az alap egységeken kívül, hogy kíszámítsuk az effektív értékeket elhelyeztünk a caste struct ágakban különböző számításokat subVI-ként. A számított effektív értéket a tömbet tárolt adatok alapján számítottuk. A tömb elemeit négyzetre emeltük egy operátorral majd az eredmények összegét (szummája) elosztottuk a tömb elemeinek számával és a cast structon kívülre kikötöttük az így kapott értéket. Az elméleti értékeket külön kellett elkészíteni áganként, A szinuszos elméleti effektív érték kiszámításához az alábbi összefüggést használtuk:
$Umax = o + A$
$Ueff = \frac{Umax}{\sqrt{2}}$
ahol o- Offset, A-Amplitúdó, U-effektív érték. A fűrészfog és háromszöges jelekhez az összefüggés így változik:
$Ueff = \frac{Umax}{\sqrt{3}}$
A négyzettes jel effekív értéke pedig:
$Ueff = Umax$
Eltérés figyelhető meg a számított és eltérési effektív érték között ami magyarázható azzal, higy a számított érték csak közelít az elméleti ideálishoz, ami végtelen pontosságú és folytonos, amíg a számított csak bizonyos értékeketn méri a függvényt.

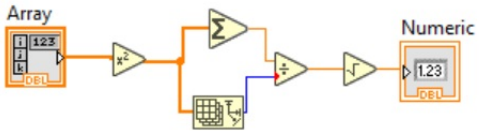

Fig. 8. Hatodik feladat: Számított effektív érték subVI

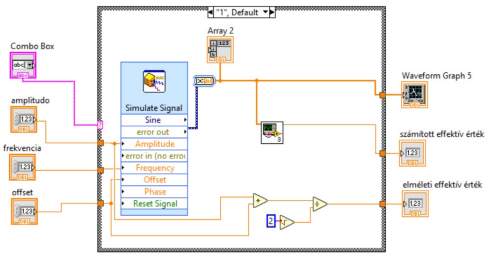

Fig. 9. Hatodik feladat: Szinusz Block Diagram

REFERENCES

[1] labview bemutatása: http://www.ni.com/hu-hu/shop/labview.html
[2] Feladatok és segédletek elérhetőek: http://digitus.itk.ppke.hu/ tihanyia/
[3] Egy hiba amit kiküszöböltünk: https://forums.ni.com/t5/LabVIEW/quot-Case-Structure-No-case-for-some-selector-values-quot-Error/td-p/2056768
[4] Képletek: https://www.codecogs.com/latex/eqneditor.php