LabVIEW1 mérési jegyzőkönyv

Szauter Ajtony, Zsumbera Olivér
(Supervisor: Naszlady Márton Bese)
Pázmány Péter Katolikus Egyetem, Információs Technológiai és Boinikai Kar
1083, Budapest Práter utca 50/A
2024.03.06
szauter.ajtony@hallgato.ppke.hu
zsumbera.oliver@hallgato.ppke.hu

***Abstract*—A méréstechnika órán megismerkedtünk alaposabban a LabView program használatával. A jegyzőkönyvben kitérünk a megadott feladatok, mérések megvalósítására, körülményeire és elméleti megoldásaira .**

***Keywords*-LabView; Effektív érték**

I. Bevvezető Feladat

1. Az előlapon elhelyezett tetszőleges típusú, nyomógombot bekapcsolva gyulladjon meg egy ovális alakú piros LED, és ugyanabban a programban helyezzünk el egy számok bevitelére alkalmas controlt. A control m/s mértékben beadott sebességet fogadja és írja ki. Mutassa meg egy tetszőleges formájú kijelző km/h egységben. Figyelje meg a két rész egymástól független működését, és vizsgálja meg az egyes objektumok (control, indikátor) tulajdonságait! A jegyzőkönyvbe rögzítse, hogy milyen programozási elemeket mely tulajdonságokkal, mire és hogyan használt!

A LabView programban a Frontpanel-en elhelyeztünk a feladatban megadott elemeket mint - Round Led, Push Button, Numerical Control, Gauge. A Led színét a properties fülön át tudtuk állítani az alap zöldről a kívánt piros színre. A Round Led-et és a Push Buttont egy vezetékkel összekötöttük amivel tudtuk kapcsolni a Led világítását. A Numerical Controlt és a Gauge mutatót pedig egy külön vezetékkel kötöttük össze és megfigyelhettük, hogy a megadott érték megjelent a mutatón ugyan azzal az értékkel. Azt figyelhettük meg a program indításakor hogy a Led és gomb függetlenül működött a Numerical Control és a mutatótól, mert köztük nem hoztunk létre összeköttetést.

II. 2. Feladat

2. Alakítsa át és mentse el új néven az 1. pont feladatában. Helyezzen el a front panelen egy 3 állású kapcsolót, melynek a segítségével lehet szabályozni a kijelzett sebesség mértékegységét m/s; km/h; és mph mértékegységekben. A mértékegység átszámítás legyen subvi-ben elhelyezve. Vigyázzon arra, hogy hibás adat ne jelenjen meg a kijelzőn! A jegyzőkönyvbe rögzítse, hogy milyen új (az 1. ponttól eltérő) programozási elemeket mely tulajdonságokkal, mire és hogyan használt!

A második feladatban az első feladatot kellet tóvább fejleszteni, bővíteni. A Front panelre elhelyeztünk még egy Hozizontal Fill Slide-ot amit kicsit személyre szabtunk. A Hozizontal Fill Slide properties ablakában a Data Types fülön átállítottuk hogy csak Integer 32 bit számokat tudjunk megadni és az állásokat átírtuk 0-2-ig. Így sikerült megvalósítanunk egy három állású kapcsolót. A következő képés a mértékegység konvertálásának magvalósítása volt. Először is a Block Diagram ablakban létrehoztunk egy Case Structure-t ami lehetőséget bíztosított számunkra, hogy a három állású kapcsoló külömböző értékeire más és más funkciókat tudjunk létrehozni. Az első azaz a "0." állás a m/s átváltása volt m/s-ba azaz amikor nem volt szükség számításra így simán a Case Structure-on keresztül be tudtuk kötni a vezetéket egy Numeric Indicatorba, hogy a Front Panel felületen láthassuk az eredményt. A második azaz az "1" állás a m/s átváltása km/h-ba. Ezt a feladatot egyszerüen a Case Structure-on belül az 1-es állást kiválasztva a Numeric Control értékét összeszoroztuk, egy általunk létrehozott konstansal aminek 3,6 értéket adtunk. A konstans értéke úgy jött ki hogy az óra és a másodperc között 60*60 a vátószám azaz 3600, a kilóméter és a méter között pedig 0,001, így 3600 * 0,001 = 3,6. A szorzat eredményét pedig ugyan úgy ugyan abba a Numeric Indicator-ba kötöttük be egy vezetékkel. A Case Srtucture-ünket a "2" állásba állítottuk és elkészítettük a m/s-ból való mérföld per órába átváltást a következőképpen. A váltószám kiszámolásához szükségünk volt arra, hogy 1 méter hány mérföld ami 1m = 0.000621371192 mile és az előző számolásunkból tudjuk hogy 1 óra 3600 másodperc így a váltószám $0.000621371192 * 3600 = 2.23693629$ amit mi 2.23694-re kerekítettünk. A váltószámot és a Numeric Control értékét összeszoroztük és a szorzás értékét vezetékkel bekötöttük a Numeric Indicatorba. A program készítése közben észrevehettük, hogy a Numeric Indicatorba kötött vezeték a Case Srtucture határán az ugynevezett Input tunnel üres négyzete csak a mindhárom Case elkészítésekor és kivezetésekor vált telire és csak ekkor tudtuk lefuttatni a programot. A feladat végén pedig egy SubVI-t készitettünk a Case Srtucture-ből és a benne lévő számolásokból. A következő táblázatban láthatóak a mért eredmények:

m/s	km/h	mph
10	36	22.36
15	54	33.55
20	72	44,73
25	90	55,92
30	108	67,1

A. Mérési bizonytalanág

Egy alacsony mérési bizonytalanságnak vehetjük azt, hogy a m/s és a mérföld per óra közötti váltószámot kerekítettük de ennek az értéke minimális.

III. 3. Feladat

3. Készítsen egy kockajáték szimulációt mely egy nyomógomb segítségével hozható működésbe. A nyomógomb megnyomására egyszer kell három független kockával dobni

(ennek értéke 1...6 tartományon van) és az eredményeket külön-külön kijelezni. Számításokkal határozza meg, hogy mekkora összeg fog leggyakrabban előfordulni, és ellenőrizze, hogy ez tényleg így van. Ha az eredmények összege 18 akkor gyulladjon ki egy kör alakú zöld LED. Figyeljen arra, hogy ne legyen hamis a kocka, azaz minden értéke 1-és 6 között elméletileg is azonos valószínűséggel forduljon elő! Lehetőség szerint használjon subvi-t. A jegyzőkönyvben rögzítse azt a számítást is mely bemutatja az egyenletes valószínűségű előfordulást!

A harmadik feladatban az egy kockajátékot kellett készítenünk mellyel háromszor dobunk és a három eredményt kiírjuk a képernyőre. Ezt mi úgy oldottuk meg, hogy csináltunk egy for ciklust mely háromszor fut le. Beletettünk egy Random Number (Range)-et melynek a minimum értékét egyre és a maximum értékét hatra állítottuk. Ezt kivezettük a for ciklusból mely így három számot generált melyet egy tömbben adott vissza. Ezt a részt belehelyeztük egy SubVI-be így volt 3 kocka SubVI-nk. Erre két darab kimenetet kötöttünk egy Array, mely kiírja a képernyőre a három számot, amit generáltunk, a másik egy szumma melyhez kapcsoltunk egy Numeric Indicatort, így ez kiadja a három kocka összegét. A szumma és az Numeric Indicator közé bekötöttünk egy Equal? -t, aminek a másik végére egy Numeric Constants melybe beírtam egy tizennyolcas számot. Az Equal? -kimenetére rákötöttem egy Boolen zöld ledet. Ennek az a lényege, hogy ha tizenhatos számot kapunk akkor zölden világít, ami lehetséges legnagyobb szám, amit ki lehet dobni három hat oldalú szabályos dobókockával. A valószínűséget a középiskolában tanultak alapján így írunk fel:

$$V = \frac{kedvező}{összes} \tag{1}$$

V= Ez a valószínűség, hogy mennyi eséllyel jöhet ki az érték.
Kedvező= Az az érték, amiből kijöhet az érték. Ennyi féle képpen jöhet ki az érték.
Összes= Az összes variáció, ami kijöhet.
Itt az összes érték pontosan kiszámolható. 6^3, mert három darab dobókocka van.

Számok	Hányféleképpen jöhet ki?	*Valószínüsége*
3	1	0,00463
4	3	0,013889
5	6	0,027778
6	10	0,046296
7	15	0,069444
8	21	0,097222
9	25	0,115741
10	27	0,125
11	27	0,125
12	25	0,115741
13	21	0,097222
14	15	0,069444
15	10	0,046296
16	6	0,027778
17	3	0,013889
18	1	0,00463

IV. 4. Feladat

4. A 3. pontban elkészített dobókocka szimulációt subvi-ként felhasználva készítsen programot, mely egy gombnyomásra egyszer fut le és legalább 10000-szer dob a kockákkal. Megjeleníti az eredmények gyakoriságát, azaz, azt, hogy hány alkalommal lett az eredmény 3; 4, … 18. Gombnyomásra legyen ismételhető a program futása. Az eddig (bármely szinten) tanultak alapján magyarázza meg az eredményt.

Negyedik feladatban az volt a feladatunk, hogy a harmadik feladatban lévő dobókockás feladatot lefuttassuk 10000-szer. Ezt úgy oldottuk meg, hogy az egész hármas feladatot beleraktuk egy for ciklusba melyet 10000-szer futtattunk le. A for ciklusból az összegeket kivezettük egy General Histogramba melynek minimum értéke három volt és a maximum értéke 18. Ezeket Numeric Constantba helyeztük. A Bins-be tizenhatot vezettünk, mert tizenhat darab szám lehetséges a három dobókocka összegeként. Egy Histogram Graphba kivezettük az értékeket így kapunk a grafikonon egy harang alakot. Kivezettük még egy Histogramban, mely megmutatja a pontos számot, hogy hányszor kapjuk meg az adott összeget. A 10000 mérést háromszor lefuttattam és ezeket egy táblázatba kiírtam és vettem a darabszámok összegét. Ez után kiszámolható ezeknek a valószínűsége mely az előző feladatban kiszámoltól elhanyagolható különbséggel tér el.

Számok	1. mérés	*2. mérés*	3. mérés
3	51	48	54
4	141	133	141
5	268	277	293
6	460	436	431
7	716	749	709
8	981	970	932
9	1146	1175	1186
10	1242	1212	1229
11	1254	1205	1217
12	1190	1167	1204
13	989	1001	942
14	670	695	734
15	457	487	462
16	268	264	276
17	116	135	145
18	51	46	45

Számok	Átlag	*Valószínüség*
3	51	0,0051
4	138,3333	0,013833
5	279,3333	0,027933
6	442,3333	0,044233
7	724,6667	0,072467
8	961	0,0961
9	1169	0,1169
10	1227,667	0,122767
11	1225,333	0,122533
12	1187	0,1187
13	977,3333	0,097733
14	699,6667	0,069967
15	468,6667	0,046867
16	269,3333	0,026933
17	132	0,0132
18	47,33333	0,004733

Miért ilyen kevés az eltérés? Lehet, hogy a radnom szám nem is random szám?
A labview kettő féle random szám generátort használ a pszeudovéletlenszám-generátort és az az Linear Congruen-

tial Generatort. A pszeudovéletlenszám-generátorok (PRNG) olyan algoritmusok, amelyek látszólag véletlenszerű számokat hoznak létre, de valójában meghatározott kezdeti állapotból és előre meghatározott algoritmusok szerinti számításokkal jönnek létre. Az egyik gyakran használt PRNG algoritmus a Mersenne Twister (MT). Ez az algoritmus rendkívül hosszú periódust biztosít (2$\hat{1}$9937-1), így a generált számok hosszú sorozatban ismétlődés nélkül jönnek létre, ami ideális a véletlenszerűség szempontjából. A másik gyakran használt PRNG algoritmus az Linear Congruential Generator (LCG). Ez az algoritmus egyszerű és gyors, mivel egy egyszerű iteratív folyamatot alkalmaz, amely a kezdeti állapotot (seed) szorozza egy konstans értékkel, majd hozzáad egy másik konstanshoz, és a kapott eredményt használja az új szám generálásához. Habár az LCG gyors és egyszerű, a generált számok általában nem érik el ugyanazt a véletlenszerűséget és periódust, mint a Mersenne Twister által biztosítottak.

V. 5. Feladat

5. 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. Az eredményeket rögzítse a jegyzőkönyvben.

Az 5. feladatban elő kellett állítanunk szinusz, négyszög, fűrész és háromszög alakú jeleket Offszet Frekvencia és Amplitudó megadásával. Ennél a feladatnál 3 Numeric Controlt helyeztünk el amikkel az imént említett tulajdonságokat lehet beállítani. A különböző alakú jelek kiválasztásához pedig egy Horizontal Pointer Slide-ot használtunk amit beállítottunk a properties menüjében, csak Integer 32, azaz egész számú beviteleket fogadjon és 0-3 ig tartó értékeket adtunk meg amivel létrehoztuk a 4 állású kapcsolót. Egy Case Structuret használtunk a különböző kiválasztható kimenetekhez. Az első állás a négyzetes gráf amit a LabView beépített Waveform generátoraival hoztuk létre. Ezek a gráf generátorok a Functions ablak Signal Processing fülön a Waveform Generation részben találhatóak. Hasonlóan a szinus, fűrész és háromszög gráfokhoz ugyan innen a megfelelő generátort tettük be, a szinuszt a "1" helyre, a háromszöget a "2" helyre és a fűrészt a "3" helyre tettük a Case-ben. Ezeknek az állásoknak a végeit kivezettük a Case Structure-ből. A program készítése közben észrevehettük, hogy a Numeric Indicatorba kötött vezeték a Case Srtucture határán az ugynevezett Input tunnel üres négyzete csak a mindhárom Case elkészítésekor és kivezetésekor vált telire és csak ekkor tudtuk lefuttatni a programot. Ezt a vezetéket két helyre kötöttük be, az egyik Egy Waveform Graph gráf mutatóba ami egy grafikonon vizualizálja a függvényt. A másik végét pedig egy úgynevezett Get Waveform Components-be vezettük ami a Programming Waveform mnüpontban található. Ez a komponens megadja nekünk a koordinátarendszer szerinti Y tengelyen lévő értékeket és a hozzá tartozó x tengelyen lévő időt. Mivel fix pontokat kapunk vissza ezért a gráfoknak végtelen szakadási pontja van ezért a beépített integrálást nem tudtuk használni. Az integrálásra azért van szükség hogy a feladatban megadott effektív értéket ki tudjuk számolni. Az effektív érték kiszámolásának képlete a következő egy periódusra:

$$V_{eff} = \sqrt{\frac{1}{T} * \int_0^T v(t)^2 dt} \tag{2}$$

Az integrálszámítás alapját felhasználva egy körülbelüli értéket viszont ki tudtunk számolni a következőképpen: Az Y értékeket négyzetre emeljük majd a dt komponensünkel összeszorozzuk és egy beépített szumma matematikai komponensel összeadjuk ezeket az értékeket amivel megkapjuk a görbe alatti terület körülbelüli értéket azaz az integrálját. Majd eszt az eredményt összeszoroztuk a frekvenciával mert:

$$f = \frac{1}{T} \tag{3}$$

Majd ebből az eredményből négyzetgyököt vontunk és egy Numeric Indicator komponensre kiírtuk az eredmény amivel ki is számoltuk az effektív értéket.

VI. 6. Feladat

Az ötös feladatig jutottunk el így a hatos feladatra nem jutott idő. Ezt a következő órán pótoljuk.

A fázishasításos teljesítmény szabályozó (phase-shifted full-bridge converter) eszköze az, hogy a bemeneti váltakozó feszültséget egy félperóduson belül csak egy részében használja ki. Ennek az az oka, hogy a kimeneti feszültség és áram jelalakját befolyásolja a kapcsolási időpontok eltolása.

Az effektív érték (RMS) számítása esetén kiindulhatunk a definícióból, miszerint az effektív érték a jelalak négyzetének átlaga, majd négyzetgyök alatt véve. Tehát, ha egy periódus alatt a feszültség $V(t)$, akkor az effektív érték a következőképpen számolható:

$$V_{eff} = \sqrt{\frac{1}{T} \int_0^T V(t)^2 dt} \tag{4}$$

Ahol T a periodusidő. Ha az áram jelalakja ugyanolyan, mint a feszültség jelalakja, akkor a teljesítmény számítása a következőképpen alakul egy 1 Ω ellenálláson:

$$P = V_{eff}^2 \tag{5}$$

Most tekintsük a bekacsolási pillanat (a fáziseltolás) változtatásának hatását a kimeneti teljesítményre. Ha a bekacsolási pillanatot δ-val változtatjuk, akkor a feszültség jelalakja a következő lesz:

$$V(t) = V_m sin(\omega t - \delta) \tag{6}$$

ahol V_m a maximális feszültség, ω a szögsebesség. Az effektív feszültség kiszámítása ebben az esetben:

$$V_{eff} = \sqrt{\frac{1}{T} \int_0^T V(t)^2 dt} \tag{7}$$

A fentiek alapján ez egy összetett integrál, amelyet meg kell oldani az δ függvényében.