LabVIEW II. mérés

Mérést végezte: Radványi Zita
NEPTUN kód: F346YE
Mérőpár: Zahoray Anna
NEPTUN kód: EF2JUM
Mérés ideje: 2023. 03. 23. 8:00-9:30
Mérés helye: Pázmány Péter Katolikus Egyetem Információs Technológiai és Bionikai Kar
Magyarország, 1083, Budapest, Práter utca 50/a
radvanyi.zita@hallgato.ppke.hu

***Abstract*—LabVIEW programmal való ismerkedés, kiadott feladatok megoldása.**

I. LabVIEW program

A LabVIEW környezetben egy úgynevezett virtuális műszer megvalósítására van lehetőség grafikus programozási környezetben, de természetesen általános célú programok fejlesztésére is felhasználható. A grafikus programozási környezet annyit jelent, hogy a programozás során nem szöveges kód készül, hanem különböző függvényeket/utasításokat reprezentáló elemek összekapcsolásával épül fel a program. LabVIEW környezetben folyamatvezérelt, adatfolyam elvű programozásra van lehetőség, a program végrehajtási sorrendjét az utasítások kapcsolódási rendszere határozza meg.

II. Mérési feladatok

A. Első feladat

Az előlapon elhelyezett tetszőleges típusú, nyomógombot bekapcsolva gyulladjon meg egy ovális alakú sárga LED. A m/s mértékben beadott sebességet írja ki és mutassa meg egy tetszőleges formájú kijelző km/h egységben.
A LabVIEW progrsam elindítása után a Front Panelt megnyitottuk, majd kiválasztottunk a controls palette-ről egy megfelelő nyomógombot (push button), majd ezután egy LED-et (round LED) típusú visszajelző fényt választottunk. A LED színét a eszközmenün (tools bar-on) található színválasztó eszközzel (color brush) állítottuk át citromsárgára, valamint az egyik szerkesztő eszközzel (positioning tool) formáltuk ovális alakúvá.
Ennek befejeztével átváltottunk a Block Diagramra, ahol összekapcsoltuk a már létrehozott izzót, valamint a nyomógombot. Amikor rákattintunk a nyomógombra, akkor annak a logikai értéke false-ról true-ra vált, ezáltal felvillan a vele összekötött sárga, ovális alakú LED. Az izzó egészen addig fog világítani, amíg nyomógomb értéke újra false nem lesz, azaz, amíg újra rá nem kattintunk a gombra. Ezen kívül átneveztük az adott részeket a feladat szövegének megfelelően.
A feladat másik részében m/s-ban megadott értéket kellett átváltani km/h-ba. Ehhez fontos a matematikai képlet, amely biztosítja az átváltást az alábbi egységek közt. Ez alapján a képlet alapján tudjuk, hogy 1 m/s megegyezik 3.6 km/h-val. A Front Panel-en kiválasztottunk egy numeric control-t, amely az átváltandó érték bevitelére szolgált, valamint egy numeric indicator-t, amely a kiszámított értéket mutatta meg. Ezután a Block Diagram-on létrehoztunk egy numeric constant-ot, amelynek az értékének 3.6-ot adtuk, a későbbi számítások miatt. Ezután hozzáadtuk a szorzás matematikai műveletet, amelynek az egyik bemenete a numeric control, azaz a átváltani kívánt bemenet, a másik bemenet, pedig az imént létrehozott konstans érték volt. A kimeneti értékének pedig a numeric indicator-t állítottuk, ami ezáltal a kiszámított értéket mutatja futtatás során.
A feladat két része, a LED felkapcsolása, valamint a m/s km/h-ba való átváltása teljes mértékben elkülönül egymástól, nincsenek befolyással a másik feladat működésére. Ezáltal az egyik tud működni a másik nélkül is, valamint ugyanígy fordítva is.

B. 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.
A feladat leírása alapján az előbb elkészített feladatot fejlesztettük tovább, alakítottuk át az utasítások alapján. Elsősorban a nyomógombot alakítottuk át kapcsolóvá. Ez egy kétállású kapcsoló, ami szintén ture, vagy false (igaz, vagy hamis) értékeket vehet fel, ezáltal tökéletesen alkalmas a feladat megoldására. Ezt követően gondolkoztunk el azon, hogy miként tudjuk az izzó világításától függővé tenni a átszámítás kimenetét. Erre véleményünk szerint a legkézenfekvőbb módszer egy case structure alkalmazsa. Akárcsak különböző programozási nyelvek esetén, például a C++-ban egy if elágazás használata. Ez a struktúra rendelkezik egy igaz és egy hamis érték szerinti lefutási móddal, amelyek egymástól teljesen függetlenül működnek. Ezt a módot fogja befolyásolni a létrehozott kétállású kapcsoló. Mindkét módban összekötöttük a LED-et a kapcsolóval, hisz ez mindkét esetben fontos és megjelenítendő információ lesz (a LED világításán keresztül). A numeric control bemenetet szintén átadjuk a case struktúrának, de itt már különböző módon kezeljük az igaz, vagy hamis érték folyamán. Az első esetben, amikor a LED nem világít, azaz a kapcsoló hamis értékkel rendelkezik, az adott értéket megszorozzuk a 3.6-tal rendelkező konsttanssal, az előbbi számolásnak megfelelően. Ezt a kiszámított értéket adjuk át a kimenetnek, így megjelentetve azt a case struktúrából kilépve. A másik esetben, amikor a kapcsoló igaz értéket ad vissza, akkor a feladat szerint nem kell elvégezni az átváltást, ezáltal a konstanssal való szorzás nélkül adjuk át a kimenetnek a numeric control által érkező bemeneti értéket. Emellett a feladat kiírásánk megfelelően létrehoztunk egy string constant-ot is. Ennek az értéke szintén a kétállású

kapcsoló bemeneti információjától függ. Ennek a string konstansnak értéke szintén a case struktúra állapotával változik. Ha a hamis ág fut a programba, akkor az átszámításnak megfelelően km/h, ha igaz ágban van, akkor pedig m/s jelenik meg a string constantban.

C. Harmadik feladat

Készítsen egy kockajáték szimulációt mely egy nyomógombot segítségével hozható működésbe. A nyomógomb megnyomására egyszer kell három független kockával dobni (ennek értéke 1...6 tartományon van) és az eredményeket külön-külön kijelezni. Ha az eredmények összege 18 akkor gyulladjon ki egy kör alakú zöld LED.
Először a Front Panelen választottunk egy nyomógombot, egy push buttont a funkció elindításához, amely megfelel a feladat leításának. Létrehoztunk egy zöld színű LED-et is, amely kis zöld kör alakú. Ezáltal megkaptuk a LED-et, amely a feladat leírása szerint a 18-as dobásösszegnél felvillanhasson. Emellett hozzáadtunk 4 darab numeric indicator-t is, amely a kijelzőn jeleníti meg a feladatok eredményét. Ezzel nyomon tudjuk követni a 3 kockadobás számait, valamint, hogy meg tudjuk jeleníteni az összegüket is. Akárcsak az előző feladatban itt is egy case struct-ot hoztunk létre a Block Diagramban, amely két ágon tud futni, azaz egy igaz és egy hamis ágon, a bemenettől függően. Ebben az esetben az igaz ág során jönnek majd létre a kockadobások. Ehhez hozzáadtunk egy véletlen szám generátort a programunkhoz, amelyet egy dobókocka jelű szimbólum jelképez. Ám ez csak 0 és 1 közötti valós számokat generál, így meg kellett oldanunk, hogy 1 és 6 közötti eredményeket kapjunk, hiszen szabályos dobókockákkal kell szimulálnunk az adott feladatot. Ezt úgy küszöböltük ki, hogy megszoroztuk 6-tal a kapott eredményt, így kapva 0 és 6 közötti valós számokat. Ám ez nem teljesen oldotta meg a problémát, hiszen még mindig nem egész számokat generált a program. Így átkonvertáltuk az értéket int-é (egész számmá), amelyhez a numeric műveletek közül választottuk ki a megfelelőt. Ezután felfigyeltünk arra, hogy mivel 0-tól indulnak a számok, ezért hozzáadtunk még 1-et, így a generált számok tökéletesen megfeleltek a követelményeknek. Tekintve, hogy a véletlen szám generátor ugyanakkora eséllyel generál számokat 0 és 1 között és a konstanssal való szorzás nem változtat ezen az eloszláson, így továbbra is teljes mértékben véletlenszerű értékeket kapunk. Ezt követően ezt a metódust lemásoltuk két alkalommal, ezúton lérehozva az összesen három darab dobókockát, majd ezeket összekötöttük a korábban létrehozott három numeric indicator-ral. Ezek után a generált eredményeket összeadtuk, az összeadás művelet segítségével, de mivel annak csak két bemenete lehet, így kétszer egymás után alkalmaztuk a három eredményre. Ennek az eredménye került a negyedik numeric indicator, kijelzőpaneé által kiírásra, ezáltal megmutatva a három dobás összegét.
A feladat második részében a kapott eredményt vizsgáljuk, amelyet külön eltároltunk. Ha az pontosan 18, azaz mind a három dobás 6-os lett, akkor világítson a zöld színű LED. Ennek az egyik legkisebb az esélye, hiszen az összes kockának hatos számot kell mutatnia. A már létrehozott case struct-on belül hoztunk létre egy numeric constans-ot, melynek az értékét 18-ra állítottuk be. Emellett létrehoztunk egy egyenlőség függvényt annak az eldöntésére, a két bemeneti érték megegyezik-e. Ezt a vizsgálatot végző függvény bemenetéhez kötöttük a három dobás összegét, valamint a 18-as értéket viselő konstanst is, majd ennek a kimenetét hozzákapcsoltuk a LED-hez, ezzel elérve azt, hogy igaz érték esetén világítson az.
A case structure hamis ágát ebben a feladatban nem használtuk, hiszen csak akkor kell a dobásokat elvégezni, ha a nyomógomb lenyomásra kerül, azaz igaz igazságértéket vesz fel. Emellett a gomb beállításaiban meg kellett adni, hogy csak abban a pillanatban adjon igaz értéket, amikor le lett nyomva, hiszen az alap beállítással addig adna igaz értéket, amíg újbóli gomblenyomás nem történik.
A dobott összegek eloszlása egy haranggörbe formát vesz fel. A haranggörbe a változók normál valószínűségi eloszlása, amelyet a grafikon ábrázol, és olyan, mint egy harang alakja, ahol a görbe legmagasabb vagy legfelső pontja jelenti a legvalószínűbb eseményt a sorozat összes adata közül. Ez magában foglalja, hogy sokkal kisebb eséllyel dobunk 3-ast vagy éppen 18-ast, mint például a legnagyobb eséllyel dobható összeg a 11. Mivel a 11-hez közelítve pozitív és negatív irányból is egyre nő, azok száma, hogy hányféleképpen állíthatjuk elő az adott összeget, míg a hármat, vagy a tizennyolcat csak egyféleképpen lehetséges. Emellett vannak olyanok is, melyek szabályos dobókockával, három dobás esetén nem lehetetséges, mint a 3-nál kisebb, vagy 18-nál nagyobb számot dobni.
Ezt a programot egy subVI-ban kell elhelyezni. A subVI egy program csomag, vagy más néven egy modul. Ennek az a fő jelenfőssége és előnye, hogy az egész modul egyetlen ikonból áll, ezáltal nem fogjuk látni a szerkezetét, így könnyű és átlátható felhasználást biztosít.

D. Negyedik feladat

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.
Ebben a feladatban a 3. feladat subVI formátumát használtuk fel, ami egy dobást jelentett. A subVI forma előnyeinek köszönhetően egy sokkal átláthatóbb programot hoztunk létre, hiszen nem szerepel a szükségesnél több ciklus, ami nehezebbé tenné a kiigazodást a programon. Tekintettel arra, hogy itt megadott számú, ez esetben 10000 dobást kellett egymás után szimulálni, így a legkézenfekvőbb megoldás egy For Loop használata volt. Ez a ciklus pontosan annyiszor fog lefutni, amennyi értéket mi adunk neki, így esetünkben be is állítottuk az n=10000-es határt. Ennek a for loop belsejébe helyeztük el a korábban elkészített kockaszimulációt. Ennek a kimenetét csatoltuk hozzá a megfelelő gráfhoz, amelynek a bemenete az adott dobott kockák számainak összege. Ezáltal a gráfon nyomon tudjuk követni, hogy melyik összegből hány darab jött létre a szimuláció során. Ezzek bizonyítva az előző feladatban megfogalmazott eloszlást. Itt is jól megfigyelhető a haranggörbe, ahogy 11-et sokkal nagyobb eséllyel dobunk, mint az éppen nagyon kicsi, vagy éppen a nagyon nagy számot.

E. Ötödik 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. Első sorban néhány, a feladat megéréséhez alapvető fogalom definícióját ismertetném:

- Amplitúdó: Az amplitúdó időben változó mennyiségek legnagyobb eltérése az egyensúlyi állapottól. Jele: A Mindig pozitív szám. Harmonikus rezgőmozgás esetén az amplitúdó az egyensúlyi vagy nyugalmi helyzettől számított legnagyobb kitérést jelenti.
- Offszet: Az eltolásos görbék/felületek, más néven párhuzamos görbék/felületek, a generátorgörbéktől/felületektől a normálvektor mentén állandó távolságra lévő pontok helyeként határozhatók meg, ez az állítható adat az offszet.
- Frekvencia: A frekvencia egy periodikus jelenség (rezgés) ismétlődési gyakoriságát" jelenti: egy esemény hányszor ismétlődik meg egységnyi idő alatt (idő alatti periódussűrűség).

A feladat során létrehoztunk a Front Panelen egy Waveform Graph-ot, az előállított jelek megjelenítésére, egy Combo Box-ot amely segítségével a legördülő listából kiválaszthatjuk a megelelő alakot, valamint három darab Numeric Slidert, amelyen állíthatók a kért offszetet, amplitudót, frekvenciát. Létrehoztunk egy Case Structur-et, ahol négy állást hoztunk létre, amely megfelel a legördülő listán szereplő fogalmaknak. Az adott Case Structure bemenetéhez csatoltuk a combo boxot, valamint a három Numeric Slidert. A struktúra belsejében létrehoztunk egy-egy Simulate Signal-t, amelyen az előre elkészített függvényrajzokat használtuk, ezeket kötöttük össze a Case-en kívül elhelyekedő Waveform Graph-fal. Ezt megismételtük az összes Case ágban, így már mind a négy formát meg tudja jeleníteni a gráf a legördülő menü alapján választott alakzatot. Ezután már csak a folyamatos futtatást kellett megldanunk, amelyhez létrehoztunk a létrehozott objektumok köré egy While Loop-ot, amelynek a leállításához bekötöttük egy true/false értéket tartalamazó kapcsolót, amelynek az értékét mindig false-ra állítottuk.

F. Hatodik feladat

Középiskolai tanulmányai alapján tetszőleges periódikus jelalak alkalmazása esetén határozza meg annak effektív értékét. A meghatározása során induljon ki az effektív érték definícójából és mutassa meg, hogy hogyan is kell azt alkalmazni.

- Effektív érték: A villamos áram effektív értéke (vagy négyzetes középértéke) az áram hőhatására ad útmutatást. Az effektív érték annak az egyenáramnak az értékével egyenlő, amely azonos idő alatt ugyanakkora munkát végez (hőt termel), mint a vizsgált váltakozóáram.

Ezen feladat során az ötödik feldatban létrehozott programot fejlesztettük tovább. A Siulate Signalból kimenő értékhez kötöttük hozzá a Double to Ineger típusú konventátort. Ennek a két kimenete lesz az RMS, amelyet az egyik létrehozott Numeric Indicatorhoz kötünk hozzá, ez adja meg az effektív értéket. A konvertáló másik kimenete a két kimenetű Arrayhez kapcsoltuk hozzá. Ettől a ponttól a négy különböző alakhoz, négy különböző módszer tartozik. A sinus görbe esetén a tömb felső kimenetét elosztjuk a $\sqrt{2}$ -vel. Ennek az eredményét kötjük hozzá a másik létrehozott Numeric Indicatorhoz, ez tartalmazza az számított értéket. A fűrész és a háromszög esetén hasonló metódust hajtunk végre, annyi különbséggel, hogy $\sqrt{2}$ helyett $\sqrt{3}$ -mal osztjuk el az Array kimenő értékét, ezzel kiszámolva a keresett értéket. A négyzet esetén könnyebb volt a feladat, hiszen már nem kellett több számolást végezni, hiszen a kimenő Array értéket kötöttük össze a megjelenítő Numeric Indicatorral. Ezáltal a Front Panelen minden kiválasztott grafikon esetén, az állítható értékeket figyelembe véve, láthatjuk a kért kijelzőket, ezlátal remekül össze is tudjuk őket hasonlítani.

REFERENCES

[1] https://hu.wikipedia.org/wiki/Amplitúdó
[2] https://hu.wikipedia.org/wiki/Frekvencia
[3] https://wiki.ham.hu/index.php?title=Effektivérték
[4] https://www.hindawi.com/journals/jam/2014/124240/