LabVIEW I. mérés

Mérést végezte: Paulicsek Ádám Imre
Mérés dátuma: 2021.02.18. 11:15-14:00
Mérés helye: Budapest
paulicsek.adam.imre@hallgato.ppke.hu

Mérőeszköz adatai: LabVIEW software

I. LabVIEW

A LabView egy grafikus programfejlesztő, amely elsősorban méréstechnikai és a hozzákapcsolódó jelfeldolgozási feladatok megoldására szolgál, de alkalmas más, pl. szimulációs munkákra is. A grafikus programozás egy látványos, könnyen követhető programozási módot jelent, amelyet könnyen sajátíthatnak el a hagyományos programnyelveket nem ismerő szakemberek is. Mindazonáltal meg kell jegyezni, hogy a programozás alapvető jellemzői (változók deklarálása, ciklusok) teljes mértékben a C programnyelvhez hasonlóan, illetve azzal analóg módon történnek, ezért bármilyen hagyományos programnyelv alapszintű ismerete sokat segít a kezdeti lépésekben.

II. Mérési feladatok

A. *Első feladat*

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

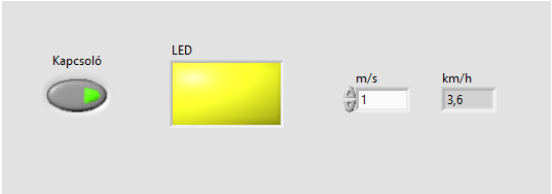

Fig. 1. Első feladat (Front Panel)

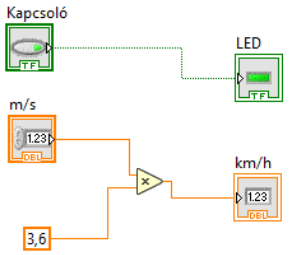

Fig. 2. Első feladat (Block Diagram)

A Front Panel megnyitása után, választottam egy nyomógombot (push button) a controls palette-ből, majd ezután egy LED-et (square LED). A LED-nek a színét, a tulajdonságainál (properties) változtattam meg egy sárga árnyalatú színre. Ennek befejeztével, átváltottam a Block Diagramhoz, ahol összekapcsoltam a két azonos boolean típussal rendelkező kapcsolót, és LED-et. Az izzó, akkor villan fel, amikor a bekapcsológombra rákattintok, ezzel a kapcsoló értéke megváltozik false-ról, és true értékkel fog rendelkezni, amit a vezeték tárol el, és adja tovább az izzónak. Mivel igaz értéket kap az izzó, ezért felvillan, és addig fog világítani, amíg a nyomógomb értékét meg nem változtatjuk hamisra.

A feladat másik részében, hogy egy m/s értéket kell átváltani km/h. Ehhez használnunk kell a két érték között egy matematikai összefüggést, hogy 1 méter per szekundum = 3,6 kilométer per órával. A Fornt Panel-en kiválasztottam egy numeric control-t, és egy numeric indicatort a controls palette-ből. Következőben a Block Diagramban, a function palette segítségével hozzáadtam egy numeric control-lal azonos numeric constant-ot, aminek az értékét 3,6-ra állítottam, a matematikai össuefüggés felhasználásával. Ezek után egy szorzás függvényt adtam hozzá, aminek a két bemenete a constant, és a control, a kimenete pedig az indikátor.

Ez a két része a feladatnak, teljesen egymástól függetlenül működnek, nem befolyásolják egymás működését, mivel nincsenek összekapcsolva. Nem változik meg az átváltás kimenő értéke azzal, hogy fel van-e kapcsolva az izzó, vagy nincsen. És ugyanez fordítva.

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. 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.)

Az előző feladatot fogom tovább módosítani, és alakítgatni a feladat leírása szerint. Először a nyomógombra jobb egérklikk után, a replaceban kiválasztottam egy kapcsolót (slide switch). A következő lépésben arra kell megoldást találni, hogy hogyan tudnám, csak akkor átalakítani a mértékegységet, amikor a kapcsoló ki van kapcsolva. Abban tér el az előző feladattól, hogy itt összekötjük ezt a két különböző eseményt. Erre szerintem a legjobb módszer, ha egy case structure-t alkalmazok. Ezt a C++ programozásban elágazásnak nevezzük , vagyis van egy igaz, és van egy hamis ága. A struktúra két bemenettel fog rendelkezni, de hogy melyik ága fog ”lefutni”, a kapcsoló értéke fogja eldönteni. Mindkét esetben összekötöm a kapcsolót az izzóval, hogy kapjak valami információt arról, hogy mikor milyen állapottal rendelkezik a kapcsoló. A másik numeric control bemenetett, pedig már másképpen fogom kezelni a két különböző esetben. Az első esetben, amikor hamis, akkor a bemenetet megszorzom egy 3,6 numeric constant-al, és az eredményt összekötöm a kimenettel. A másik esetben, amikor igaz, akkor csak simán összekötöm a bemenetet, a kimenettel.

Ekkor, ha kikapcsolt állapotban van, akkor a m/s bemenetet, átváltja km/h-ba, viszont ha bekapcsolt állapotban van, akkor a m/s bemenet, egyben a kimenet is lesz.

Emellett a feladat kéri, hogy jelenjen meg valami információ az átváltással kapcsolatban, ezért mindkét esetben létrehoztam egy string constant-ot, amit összekötöttem egy string indicator-ral. Ez jelzi, hogy éppen a kimeneti érték, pontosan milyen mértékegységgel rendelkezik.

A mértékegység átváltást egy subVI-ben kell elhelyezni. A subVI egy program csomag, vagy modulnak is szoktuk nevezni. Ennek azzal a jelentőséggel bír, hogy az egész modul egyetlen egy ikonbol fog állni, és nem fogom látni a szerkezetét. Vagyis egyszerűen, amit szeretnék egy modulba tenni, csak kijelöljöm egy négyzettel, aztán az edit fülben kíválasztom, hogy csináljon belőle egy subVI.

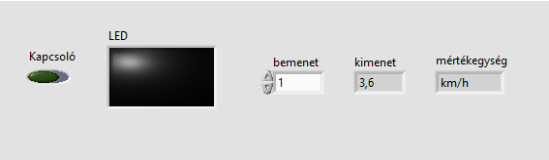

Fig. 3. Második feladat (Front Panel)

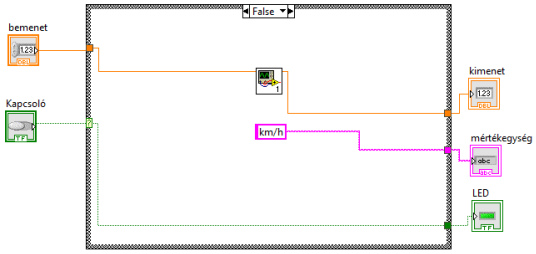

Fig. 4. Második feladat (Block Diagram)

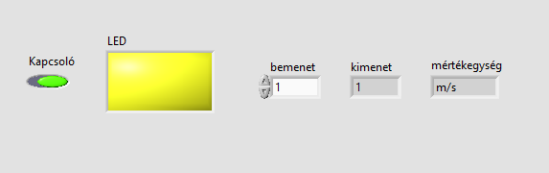

Fig. 5. Második feladat (Front Panel)

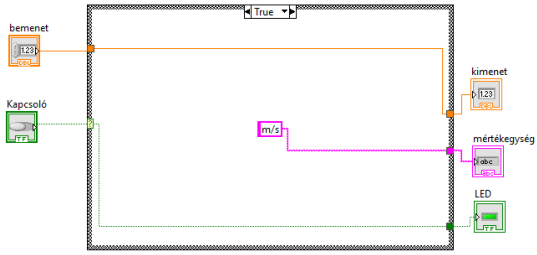

Fig. 6. Második feladat (Block Diagram)

C. Harmadik feladat

(Készítsen egy kockajáték szimulációt mely egy nyomógombot segítségével hozható működésbe. A nyomógomb megnyomására egyszer kell három független kockával dobni és az eredményeket kijelezni. Ha az eredmények összege 10 akkor gyulladjon ki egy kör alakú zöld LED.)

Először a Front Panel-en választok egy nyomógombot (OK button), ebben az esetben egy olyat választottam, amelyre tudok szöveget írni a könnyebb kezelhetőség érdekében. Emellett még egy numeric indicator-t is hozzáadtam, hogy lássam, hogy a dobásoknak az összege pontosan mennyi is lesz. Ez a megjelenítés, nem csak esztétikai, hanem ellenőrzés szempontjából is nagy jelentősége lesz, hogy jól működik-e a mérőeszközöm. Mint az előző feladatban, itt is egy case structure-ot fogok létrehozni a Blokk Diagram-ban. Ebben az esetben nem a hamis ág, hanem az igaz ág fog nagyobb szerepet betölteni. Ezekután a dobókockát kezdtem el elkészíteni. Hozzáadtam egy véletlenszám generátort, de ez csak 0 és 1 közötti számokat generál. Erre a problémára, olyan megoldást találtam ki, hogy az eredmény megszorzom hatszor, de ez még nem teljesen oldotta meg a problémámat, mert nem egész számok jöttek létre így. Ezt a problémát úgy orvosoltam, hogy a kapott eredmény felfelé kerekítettem. Ezzel egy 1 és 6 közötti egész számokat tudok generálni, ami pont egy nem hamis dobókockát tudok szimulálni, vagyis minden egyes lehetőség, ugyanakkora valószínűsége van. Mivel a véletlenszám generátor, a 0 és az 1 közötti számokat, ugyanakkora eséllyel generálja, és ezt egy konstans tag beszorzásával nem változik meg a valószínűsége. Ezek után ezt a methodot lemásoltam, és ekkor létrejött három különálló dobókocka, amiket utána egy-egy külön subVI-ként tudok kezelni. Ezek után az eredményeket összeadtam, az összeadás függvény segítségével, ami csak két bemenete lehet, ezért kétszer kellett alkalmaznom. A végeredményt összekötöttem az elején létrehozott numeric indicator-ral, ami kiírja a három dobás összegét.

A feladat második része az, hogy az eredményt vizsgájuk meg, és ha az pontosan 10, akkor gyulladjon ki egy kör alakú zöld LED. Az előzőleg elkészült case structure-ot fogom tovább módosítani úgy, hogy létrehozok a casen belül egy numeric constans-ot, aminek az értéke 10 lesz. Emellé létrehozok egy egyenlőség függvényt, aminek a két bemenetéről eldönti, hogy egyenlők-e az értékek. Ezt összekötöm a konstanssal és az eredménnyel, majd ennek a kimenetére hozzákötöm a LED-et, amit elöbb hozzáadok a Front Panel-ben. A case structure másik elágazását nem fogom használni, mert ha nem dobok a kockákkal, akkor nem történik semmi. A program lefuttatása során észrevettem, hogy nem egyszer, fut le, hanem végtelenszer. Aztán ezt a problémát úgy küszöböltem ki, hogy a gombnak a mechanikáját átallítottam, mert előtte olyan mechanikára volt rakva, hogy a gomb megnyomása után, a gomb addig fog igaz értéket adni, amíg nem nyomjuk meg mégegyszer. Ezt állítottam át arra a mechanikára, hogy a gomb megnyomása után le is kapcsolja saját magát.

Fig. 7. Harmadik feladat (Front Panel)

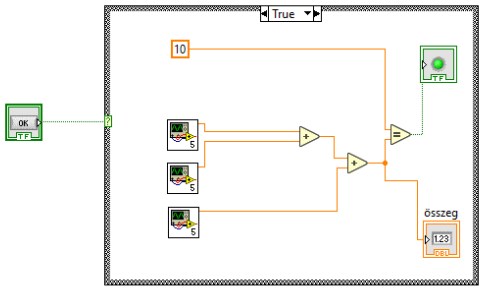

Fig. 8. Harmadik feladat (Block Diagram)

D. Negyedik feladat

(A 3. pontban elkészített dobókocka szimulációt subVI-ként felhasználva készítsen programot mely legalább 10000-szer dob a három kockával és megjeleníti az eredmények gyakoriságát, azaz, azt, hogy hány alkalommal lett az eredmény 3, 4, ... 18.)

Az előző feladatban elkészült dobókocka szimulációt fogom felhasználni ebben a feladatban subVI-ként, ami egy-egy kocka dobását fogja szimulálja. Ebben az esetben egy meghatározott számszor kell megismételni a szimulációt, és ennek az adatait utána eltárolni, hogy később fel tudjam majd használni. Egy ismert alkalomszor kell ismételni valamit, arra a legjobb megoldás, ha a For Loop-ot használom. Ezt másképpen nevezhetjük egy for ciklusnak is, és ami ennek a belsejében van, az annyiszor fog lefutni, ahányszor megadjuk hogy lefusson, ami ebben az esetben 10000-szer kell. Ennek a belsejébe fogjom a dobókocka szimulációkat tenni, amiket összeadok két darab összeadó függvénnyel, aztán ezeket eltárolom. Ekkor ütköztem egy problémába, hogy hogyan lehetne a legjobban ezeket az adatokat eltárolni. Végül az internet segítségével megtaláltam, hogy van hisztogram beépített modul, ezért nagyon könnyű dolgom volt, mert nem nekem kellett elkészíteni ezt a modult. Ennek a hisztogramnak két bemenete van, az egyik a dobások összege, a másik pedig, hogy mennyi legyen a hisztogramnak a beosztása. Ezt egy numeric constant segítségével adtam meg, aminek az értékét 16 állítottam be, mert a legmagasabb összege a három kockának 18 lehet, és a legkisebb pedig 3, és egymásból kivonva őket, megkapjuk a szükséges beosztást. Végül ennek a hisztogramnak a kimenetére, összekapcsolom egy előtte a Front Panel-en létrehozott diagrammal. Mint az Excel-nél, itt is szabad kezet kapunk a diagram szerkeztésével szemben. Én egy X-Y diagrammot alakítottam át úgy, hogy az egy hisztogram diagrammnak nézzen ki. Sok minden kellett megváltoztatnom, például a X-Y tengely beosztásait eltűntetni, és más kisebb szépség beállításokat végezni.

Lefutottva a programot, láthatjuk az elkészült ábrán, hogy a három dobókocka összege, milyen gyakorisággal dobtuk. Ezeknél a diagramoknál azt vehetjük észre, hogy úgy nézz ki, mint egy piramis. És egyben azt jelenti, hogy a szimuláció, tényleg sikeres volt, mert nagyobb eséllyel dobunk kilencet, vagy tizet, mint hármat.

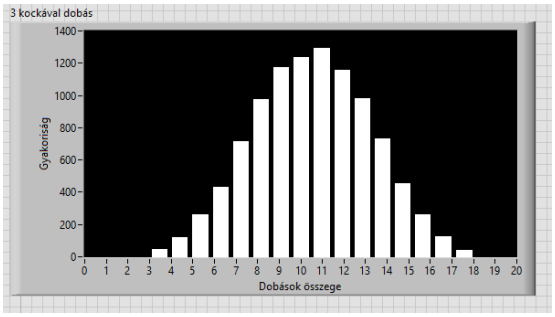

Fig. 9. Negyedik feladat (Front Panel)

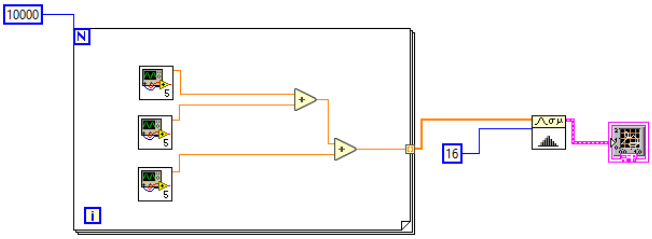

Fig. 10. Negyedik feladat (Block Diagram)

REFERENCES

[1] Labview, http://www.electro.uni-miskolc.hu/~elkszabo/Oktatas/LabView_bevezeto.pdf