Ismerkedés a LabVIEW programmal

Heiszman Henrik
Neptun kód: ENV2R9
Pázmány Péter Katolikus Egyetem, Információs Technológiai és Bionikai Kar
1083 Budapest, Práter utca 50/A
heiszman.henrik@hallgato.ppke.hu

***Téma*–A LabVIEW grafikus programozói környezet segítségével különböző matematikai és programozói feladatok illetve problémák megoldása olyan programokkal, melyeket a felhasználó egyszerűen, számára is érthető módon tud használni. Mérés során a National Instruments által fejlesztett LabVIEW programot használtam.**

I. A JEGYZŐKÖNYVBEN HASZNÁLT FOGALMAK

Front Panel: ez az általunk elkészített, virtuális műszer kezelőfelülete, ide helyezhetők el a különböző vezérlő és kijelző egységek, valamint különféle design elemek.

Block Diagram: itt az alkalmazás grafikus programozása folyik, ez a virtuális műszer belseje.

Controls Palette: itt találhatóak a Front Panel-en elhelyezhető vezérlő és kijelző elemeket (illetve szokás még kontrolloknak és indikátoroknak nevezni őket).

II. A LABVIEW ELŐKÉSZÍTÉSE

A LabVIEW ikonra kattintva elindítottam a programot, majd létrehoztam egy új „Blank VI” projectet. A tanultak alapján a „ctrl+T” billentyűkombinációval egymás mellé illesztettem a képernyőn a Front Panelt és a Block Diagramot. Ezzel a pár lépéssel elő is állítottam a két tiszta oldallal rendelkező munkafelületet.

III. SEBESSÉG ÁTVÁLTÁSA

Az általam elkészített programnak feladata lesz, hogy hiba nélkül, a felhasználó által meghatározott, $\frac{m}{s}$ mértékegységű sebességet átváltsa $\frac{km}{h}$ -ba, majd az így kiszámolt értéket jelenítse meg egy általam tetszőlegesen megválasztott kijelzőn. Valamint a program képes lesz egy, a felhasználó által üzemeltetett gomb hatására egy sárga színű, négyzet alakú LED-et felkapcsolni.

A program elkészítése során több fajta inputra (a felhasználó által kezelt bemenetre) is szükségem lesz. Kelleni fog egy nyomógomb, amely majd üzemelteti a LED, továbbá egy numerikus bemenet, amely segítségével az üzemeltető megadhatja az átváltani kívánt sebesség mértékét ($\frac{m}{s}$-ban).

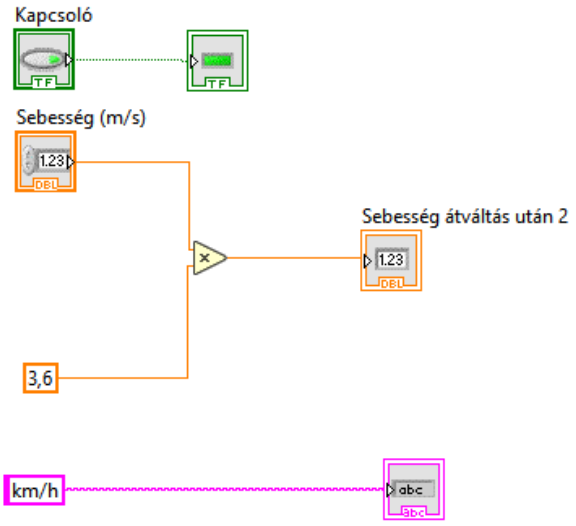
Ezeket a bemeneteket egyszerűen elhelyezem a Front Panel. A használni kívánt inputot a következőképpen hoztam létre: az egér jobb gombjával kattintottam a Front Panelen, amelynek köszönhetően megjelent Controls Palette. Ebből az ablakból kiválasztottam az általam használni kívánt inputokat: egy Boolean típusú nyomógombot és egy numerikus bemenetet. (Boolean típusú gombnak nevezzük azt a bemenet, amely igaz vagy hamis érték bevitelére szolgál.) Ezek után szinten a Controls Paletten „Boolean” menüpont alatt kiválasztottam a négyszögletes LED-et, majd elhelyeztem a Front Panelen. Elvárás, hogy az a dióda négyzet alakú legyen és sárgán világítson. A méretét egyszerűen a LED egyik szélére kattintva, húzással méretre lehet szabni. A dióda színének beállításához a jobb kattintással lenyíló menüsor „Properties” alpontját kell megnyitni. Itt az „Appearance” fülön a „Colors” alpontban beállítható az On/Off állásban lévő LED színe. Én az On állapot színét a feladatnak megfelelően a 255; 255; 100 és az Off állapot színét pedig a 100; 100; 0; RGB kódú sárga árnyalatra állítottam. Végül elhelyeztem egy numerikus és egy string típusú kimenetet is, amelyen keresztül a program képes lesz megjeleníteni az átváltott sebesség mértékét és mértékegységét a felhasználó számára.

Az vizuális kellékek elkészítése után elkezdtem a feladat software-es részét elkészíteni Block Diagramon.

Első lépésben a nyomógomb kimenetét összekötöttem a LED bemenetével, ezzel biztosítva, hogy a dióda kapcsolható legyen a felhasználó igénye szerint.

Második lépésben a mértékegységváltó helyes működését szolgáló programot készítettem el. A Block Diagramon létrehoztam egy numerikus konstanst, amelynek a 3,6-et adtam értékül, majd elhelyeztem a szorzás elvégzésére szolgáló függvényt. Ennek a függvénynek a bemenetéra rákötöttem a numerikus bementet és a konstans számot, majd a kimenetét pedig összekötöttem a numerikus kimenettel. Ezek után a Block Diagramon létrehoztam egy string típusú konstanst is, amelybe „$\frac{km}{h}$”-ot írtam. Ez az állandó felelős azért, hogy az üzemeltető lássa az átváltott szám mértékegységét. Végül összekötöttem a string állandót a string kimenettel.

E két lépés elvégzésével a program elkészült és az ellenőrző futtatás során hibátlannak bizonyult: a nyomógomb kapcsolja a lámpát és a bemenetre érkező az adatott átváltás során helyesen meg is jeleníti. (*1. ábra*)



1. ábra
Elkészült átváltó

Meg kell még említeni a program matematikáját is. Fizikából tanultak alapján tudjuk, hogy $1\frac{m}{s} = 3{,}6\frac{km}{h}$. Ebből adódóan volt szükségem egy konstans számmal (3,6) való szorzásra.

Mivel a program készítése során egymástól független működésre kalibráltam az mértékegységváltót és a LED kapcsolóját, így az applikáció használata során ez a két funkció egymást semmiben sem befolyásolja.

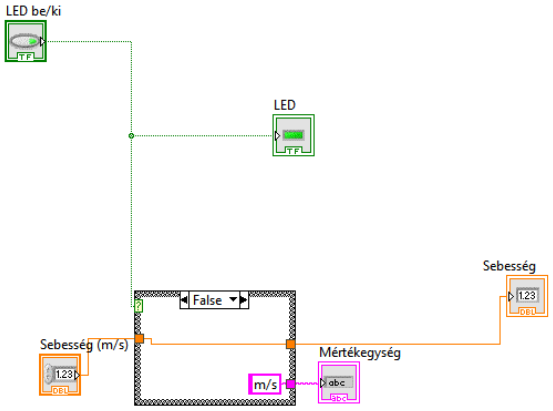
IV. VÁLASZTHATÓ ÁTVÁLTÁS

Ebben a részben átalakítottam a már meglévő (1. feladatban létrehozott) programomat úgy, hogy a felhasználó saját maga tudja meghatározni, hogy át szeretné-e váltani a megadott sebességet vagy sem.

Annak érdekében, hogy a program el tudja végezni a feladatát, két esetet kell elkülöníteni egymástól, és mind a két szcenáriót egymástól különböző módon kell kezelni. Az egyik eset az, ha a felhasználó nem szeretne átváltást, tehát a kapcsoló az „OFF” állásban van. Ekkor a programtól azt várjuk el, hogy egyszerűen kiírja a bemeneti értéket és a mértékegység kijelzőn a „$\frac{m}{s}$” jelenjen meg. A másik esetben a felhasználó bekapcsolja az átváltó funkciót, tehát a kapcsoló az „ON” állásban van. Ekkor azt várom el a programtól, hogy helyesen átváltsa a bemeneti adatot és a mértékegység kijelzőn a „$\frac{km}{h}$” szerepeljen.

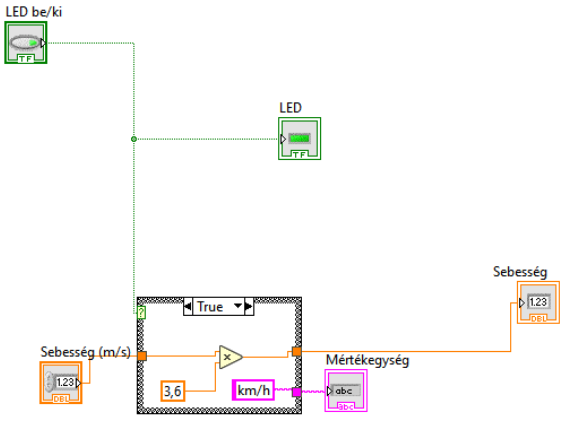
Első lépésben a Block Diagramon létre kell hozni egy „Case Structure ” nevű függvényt, amely arra szolgál, hogy különböző eseteket lehessen megkülönbözetni (jelen esetben ez az „OFF” / „ON” (igaz/hamis) állása a kapcsolónak.), és össze kell kötni a kapcsoló kimenetével.

Második lépésben meg kell tervezni azt az esetet, amely során a kapcsoló a hamis, tehát kikapcsolt állásban van. Ekkor egyszerűen a „Case Structure” False állásánál át kell kötnünk a sebesség bemenetet a kimenettel és a string konstanst $\frac{m}{s}$ állapotban be kell kötni a string kimenetre. Ezt a következő ábrán szemléltetem. (*2. ábra*)



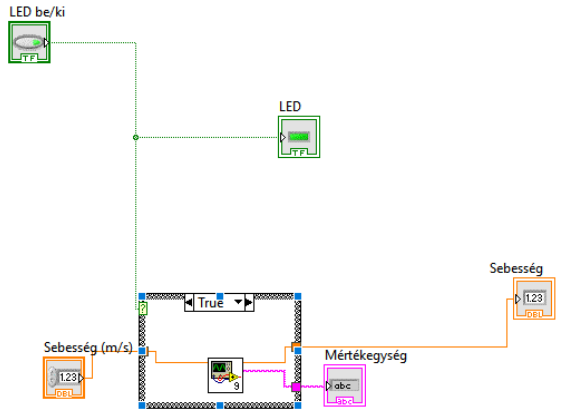
2. ábra
False ág

Harmadik lépésben megterveztem azt az esetet, amikor a kapcsoló az igaz, tehát a bekapcsolt állapotban van. Ekkor a „Case Structure” True állásában be kell húzni az első feladat során elkészített átváltó program szorzó és mértékegységet kiíró részét. Ezt szemlélteti a következő ábra. (3. *ábra*)



3. ábra
True ág

Feladat része volt továbbá az, hogy használjak úgynevezett SubVI-t a program elkészítése során. A SubVI arra szolgál, hogy a programon belül egy-egy részletet össze lehet sűríteni, így az elkészült program jobban átláthatóvá és könnyebben kezelhetővé válik. SubVI létrehozásához ki kell jelölni a tömörítendő elemeket a Block Diagramon és az „Edit” menüpont alatt a „Create SubVI” alpontot kell kiválasztani. A következő ábrán látható, hogy hogyan jelzi a LabVIEW az elkészült SubVI-t. (*4. ábra*)



4. ábra
SubVI használata

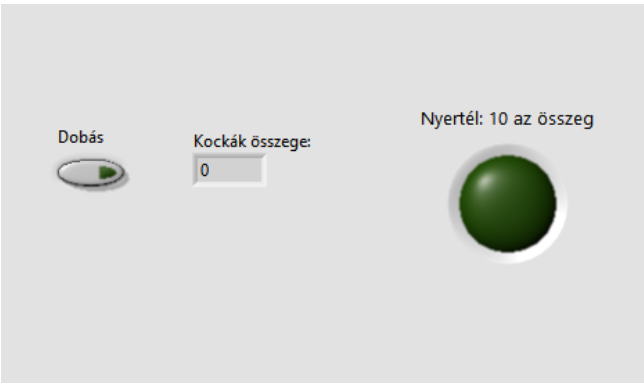
A programban továbbra is él az a funkció, hogy a kapcsoló segítségével vezérelni lehet a LED-et. Itt ez hasznos volt mert, így tudja a felhasználó a dióda állapota alapján, hogy be van- e kapcsolva az átváltás.

Az előbb ismertetett lépések elvégzése során a program elkészült. Az ellenőrző futtatás során hibátlannak bizonyult.

V. KOCKAJÁTÉK

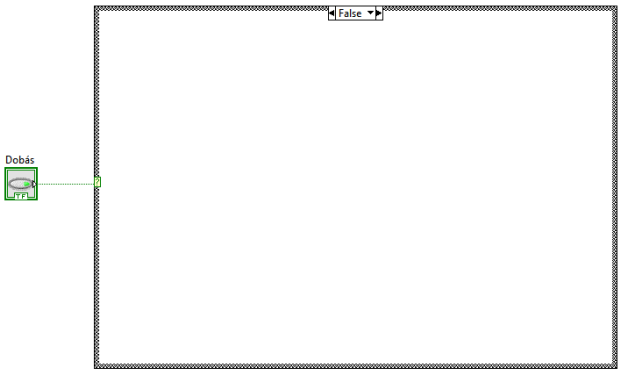
Ez a program egy kockajátékot fog szimulálni, amely egy nyomógomb segítségével üzemeltethető. A gomb megnyomása után a program egyszerre fog „dobni" három hatoldalú dobókockával és abban az esetbe, ha a dobókockákon szereplő számok összege pontosan megegyezik tízzel, akkor elkezd világítani zölden egy LED. Fontos részlet, hogy ne legyenek hamisak a dobókockák, tehát minden dobás során az egyes kockák értéke 1 és 6 között azonos valószínűséggel forduljanak elő.

Első lépésben, a már megszokott módon, elhelyeztem a Front Panelen az üzemeltetéshez elengedhetetlen eszközöket: a nyomógombot, a zöld színű kerek LED-et és egy numerikus kijelzőt, hogy a felhasználó lássa, hogy mennyi az adott dobás során, a kockákon szereplő értékek összege. (*5. ábra*)



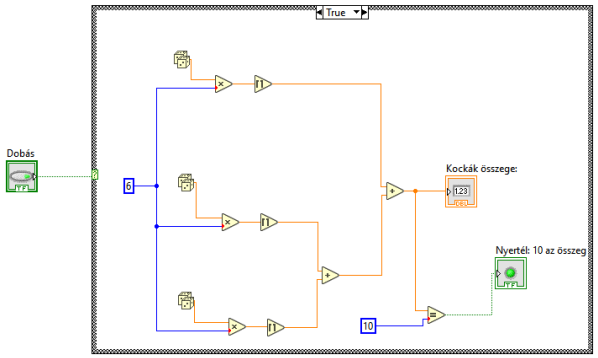
5. ábra
Kockajáték Front Panel

Második lépésben nekiláttam elkészíteni a program sofware-es hátterét. A korábbi példához hasonló, itt is két esetet kell megkülönböztetni. Egyik eset, amikor a gombot nem nyomták le, a másik esetben pedig, amikor a felhasználó elindította az adott kört, tehát lenyomta a gombot. Az első esetben nem kell semmit csinálnia a programnak, hiszen ilyenkor a játékos még nem indította el a játékot. Ez számomra azt jelenti, hogy a Block Diagramon létrehozott, a nyomógomb által vezérelt esetvizsgáló függvény „false" állásában nem szabad, hogy bármi is szerepeljen. *(6. ábra)*



6. ábra
False állás

Harmadik lépésben leprogramoztam annak az esetnek a kezelését, amikor a felhasználó megnyomja a gombot. Ekkor le kell generálni három, egymástól független, 1-6 közötti, egész számot, majd ezeknek az összegét ki kell írni a kijelzőre, valamint be kell kapcsolni az izzót abban az esetben, ha ez az összeg megegyezik tízzel. Ez az alábbi módon néz ki a LabVIEW-ban. *(7. ábra)*



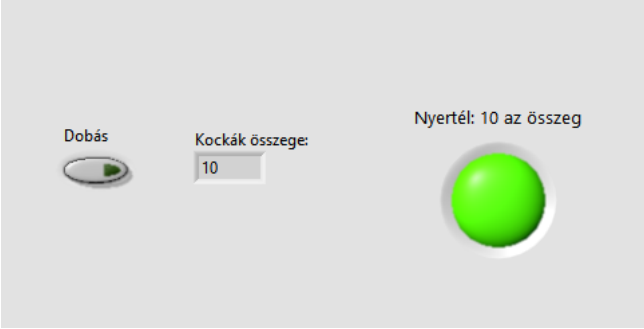
7. ábra
True állás

LabVIEW-ban a random szám generálására létrehoztam az esetvizsgálón belül három „Random Number (0-1)" nevű függvényt (ezek reprezentálják a három dobókockát). Ezek a függvények legenerálnak egy 0 és 1 között lévő számot. Nyilván való, hogy ez számomra csak részben jó, ezzel valamilyen matematikai műveletet kell végezni, hogy 1-től 6-g valamilyen egész számot kapjak. Erre tökéletes megoldás, hogy a generált számot megszorzom hattal, amely így 0 és 6 közötti tört szám lesz, majd ezt egy újabb függvény segítségével felfelé kerekítem. Ezzel a módszerrel a program

legenerál egytől hatig egy random számot, amely során az egyes számok az valószínűséggel fordulnak elő.

Ezt a random szám generálást a program háromszor végzi el egymás után és az így kapott három számot összegzi. Az összeadás elvégzése során az eredmény megjeleníti a kijelzőn. A programba fel kellett használ egy egyenlőséget vizsgáló függvényt, amely két bemenetére a kockák összegét és egy konstanst kötöttem. Ennek az állandónak az értéke 10, hiszen azt szeretnénk vizsgálni, hogy az összeg értéke megegyezik-e tízzel. A függvény kimenetét, amelyen vagy igaz, vagy hamis érték szerepel a működés során, a zöld LED-re kötöttem, amellyel azt értem el, hogy az egyenlőséget vizsgáló függvény értéke határozza meg az állapotát.

A fent leírt lépések elvégzésével a program elkészült és a futtatások során hibátlannak bizonyult. *(8. ábra)*



8. ábra
A program futás közben

A mérési utasításban szereplő feladatokból hármat tudtam elvégezni labor végéig.

Felhasznált források

Online LabVIEW Help Resources

LabVIEW mérési utasítás

LabVIEW segédlet