LabView 2. mérési jegyzőkönyv

Mátyás Antal
(Supervisor:Attila Tihanyi)
Pázmány Péter Catholic University, Faculty of Information Technology and Bionics
50/a Práter street, 1083 Budapest, Hungary
antal.matyas.gergely@hallgato.ppke.hu

***Abstract*—A mérés célja volt ismerkedni a LabView programmal. A feladatok megoldásával közelebb kerültünk a program tulajdonságaihoz, lehetőségeihez és használatához.**

I. Feladat

A mérési utasításnak megfelelően a Front Panelen elhelyeztem egy gombot, majd pedig egy LED-et, melynek properties fülén a színét átállítottam sárgára, majd a Block Diagram felületén a gomb kimenetét összekötöttem a LED bemenetével, ezzel elérve, hogy mikor a gombot megnyomjuk, a LED kigyullad. A feladat második részéhez elhelyeztem továbbá egy numeric control valamint egy numeric indicator elemet, majd a Block Panelen ezeket úgy kötöttem össze, hogy a Numeric Control kimeneti értékét elosztottam egy 3,6 értékű numerikus konstanssal, majd hozzákötöttem a Numeric Indicator bemenetéhez, így elérve, hogy a beállított km/h sebességet m/s-ban jelezze ki. Megfigyelhető, hogy a gomb és LED kapcsolat függetlenül működik a sebesség átváltó funkciótól.

II. Feladat

Az előző feladatot átdolgozva hoztam létre a következőt. A gomb Properties fülén az Operation fülön a Button behavior értékét Switch when pressed-re állítottam, ezzel elérve, hogy ki-be kapcsolható gombot kapjak. A mértékegység átváltó funkciót kijelölve az Edit fül Create SubVI parancsával SubVI-t hoztam létre, a későbbiekben ezzel dolgoztam. Annak érdekében, hogy az átváltás csak akkor következzen be, ha a kapcsoló kikapcsolt állású, egy Case struktúrát hoztam létre, melynek False ágába elhelyeztem a korábban létrehozott SubVI-t. Mivel a feladat kikötötte, hogy hibás adat nem jelenhet meg a kijelzőn, a kimeneti Numeric Indicatorhoz létrehoztam egy Property Node-ot, mellyel a Visible tulajdonságát változtathatom a program futása közben. Ezt egy NOT funkció közbeékelésével összekötve a kapcsolóval elértem, hogy az átváltás eredményét kijelző Numeric Indicator csak a kapcsoló kikapcsolt állapotában legyen látható, amikor az átváltás is történik, ellenkező esetben láthatatlan legyen. A feladat utasítása szerint hozzákötöttem továbbá egy LED-et is a kapcsolóhoz, hogy annak jelenlegi állását a Front Panelen is ellenőrizni tudjam.

III. Feladat

A feladat egy kockajáték szimulásása volt. Mivel a játék kezdetét egy gomb megnyomása kell, hogy jelezze, az egész program egy Case struktúra belsejében helyezkedik el, melynek feltétele egy a Front Panelen elhelyezett, Latch when pressed tulajdonságú gomb. A gomb megnyomásakor elindul a Case struktúra True ága, és két Random Number generátorral létrehozzuk a kockadobást. Mivel a random szám generátor 0 és 1 közötti értékeket ad, ezt egy 6-os Numeric Constant-tal megszorozva, majd az értéket felfelé kerekítve értem el, hogy a valós dobókockához hasonlóan az érték 1 és 6 közé essen. Így az egyes dobott értékek valószínűsége azonos, hiszen a random generátor 0 és 1 között azonos valószínűséggel ad számot, ezt 6-tal szorozva a valószínűség továbbra is azonos, a kapott értékek 0 és 6 közé esnek. Mivel mindegyik értéket felfelé kerekítünk, minden kapott értékre ugyanakkorra eséllyel számíthatunk. Miután megkaptuk a két dobókocka értékét, továbbra is a Case struktúrán belül dolgozva a két értéket egy ADD függvénnyel összeadjuk, majd egy Equal? függvény és egy 7 értékű numerikus konstans segítségével ellenőrizzük, hogy valóban egyenlő-e 7-tel. Az egyenlőség vizsgálat kimenetét kötjük ezután össze egy LED-del, hiszen a boolean típusból kifolyólag az egyenlőség esetén True kimenetet ad, aktiválva ezze a LED-et.

IV. Feladat

A feladat a korábbi feladatban használt sebesség átváltó SubVI futási sebességének mérése volt. Az időméréshez Tick Countereket használtunk, értékük különbségeivel adtuk meg az eltelt időt. A mérés pontosságának növelése érdekében az átváltást sokszor kellett elvégeznünk egymás után, így ezt egy for ciklusba helyezve 100000-szer futtattuk le, majd ennek futási idejét osztottuk el ugyanezzel a számmal, hogy megkapjuk egy darab átváltás idejét. Mivel jelen számításban nem vettük figyelembe, hogy a for ciklusnak is van saját futási ideje, ezekután megmértem 1000000 darab üres for ciklus idejét, majd a végeredményhez ezt kivontam az előző mérés idejéből. Gyakorlatban ahhoz, hogy az időmérés alkalmazható legyen, a bizonyos feladatoknak egymás után kellett végrehajtódniuk. Ezt egy Flat Sequence struktúra használatával értem el, melynek első frame-jében az első Tick Counter, második frame-jében a for ciklusban található átváltás, harmadik frame-jében pedig a másik Tick Counter helyezkedett el. Miután az első mérést elvégeztem, a következő framekben végrehajtotam ugyanezt a számítást üres for ciklussal, majd az utolsó frame-ben a két mérés eredményét kivontam egymásból, ezzel megkapva az átváltás futási idejét.

V. Feladat

A feladat a felhasználó reakcióidejének mérése volt. A Front Panelen létrehozott LED felvillanása után az ugyanitt elhelyezett kapcsolót kellett megnyomni, a program pedig kiszámolta a két esemény közötti idő különbséget. A feladat megoldásához itt is Flat Sequence struktúrát használtam. Annak elérésére, hogy random időpontban villanjon fel a LED, az első frame-ben elindítok egy Tick Countert, majd a második frameben egy while ciklusban elhelyezett második Tick Counternek, és a kettő különbségének felhasználásával minden ciklusban megvizsgálom, hogy az így megkapott jelenlegi eltelt idő, megegyezik e az ugyanebben a frameben generált random

számmal. A random generátor kimeneti értékét itt megszorzom 20000-el, így érem el, hogy az időintervallum milliszekundumban mérve megfelelő tartomány legyen. A while ciklusból a program akkor lép ki, ha az eltelt idő megegyezik, vagy nagyobb a random generált időponttal. Ekkor a következő frame-ban egy True Constant segítségével felgyújtom a LED-et, valamint egy újabb Tick Counterrel megnézem az aktuális időpontot. Az ezt követő frame-ben szintén egy while ciklus helyezkedik el, melyben minden ciklusban megvizsgálom, hogy a Front Panel-en elhelyezett gombot megnyomták-e. Amennyiben a gombot megnyomják, a while ciklus végetér, a program pedig átlép a követzető framebe, ahol egy utolsó Tick Counter időpontjából kivonom a LED felgyújtásakor ellenőrzött Tick Counter értékét, ezzel megkapva a LED felgyulladása óta eltelt időt, azaz a felhasználó reakcióidejét.

VI. Feladat

A feladat egy szinusz, egy négyszög, egy fűrész és egy háromszög függvény szimulációja volt, majd a függvények effektív értékének kiszámítása. A megoldáshoz először az Express VIs menüből kiválasztott Simulate Signal jelgenerátort hoztunk létre, a Properties ablakában pedig beállítjuk a jel típusát. A jel megjelenítéséhez a Simulate Signal elemből a jelet egy létrehozott Waveform Graphba vezetjük, ezzel is tudjuk majd ellenőrizni a Front Panelen, hogy a megfelelő jellel dolgozunk, illetve annak frekvenciáját módosítani tudjuk. Ugyanezt a kivezetett jelet ezután tömbbé alakítjuk. Ezen tömb felhasználásával számoljuk ki ezután a jel effektív értékét egy hozzáadott RMS VI segítségével, melynek kimenetét egy Numeric Indicatorhoz kapcsoljuk, hogy az érték leolvasható legyen. Ellenőrzés végett a frekvenciát állíthatóvá tesszük, egy Numeric Control-t bekötve a Simulate Signal VI Frequency pontjához, ezzel vizsgálva, hogy az effektív érték a frekvencia változása ellenére állandó marad-e. Amennyiben a program folyamatos futtatása közben a frekvencia változtatásával az érték kis mértékben különbözik, a Simulate Signal VI Properties fülén a Number of samples opcióval megnöveljük a jelgenerátor jelének mintavételezési mennyiségét, így pontosabb kimenetet kapva. Mind a négy jelgenerátort hasonló módon hozzuk létre, a Simulate Signal VI-ben módosítva a jel típusát Square, Triangle, majd pedig Sawtooth értékre.