LabVIEW Jegyzőkönyv

<u>A mérés célja</u>: Ismerkedés a LabVIEW program lehetőségeivel és felhasználhatóságával.

Név: Kincs Boglárka Bianka, Ekart Csaba

Mérés időpontja: 2017. 02. 23. 12:15-15:00, 2017. 03. 02. 12:15-15:00

Mérés helye: PPKE ITK, 420-as mérőlabor

Mérendő objektum: nincs

Méréshez felhasznált műszerek, eszközök: LabVIEW grafikus programrendszer

1) **Az előlapon elhelyezett tetszőleges típusú kapcsolót bekapcsolva gyulladjon meg egy tetszőleges sárga LED. A km/ó mértékben beadott sebességet írja ki és mutassa meg egy tetszés szerinti formájú kijelző m/s egységben.**

Minden feladatot azzal kezdtük el, hogy megjelenítettük a controls palette-t és a tools palette-t (a view menüből), emellett a jobb átláthatóság érdekében osztott képernyőn jelenítettük meg a block diagramot és a front panelt (Ctrl+T billenytűkombináció segítségével).

LED

A listából kiválasztottunk egy szögletes LED-et, amelynek átállítottuk a színét sárgára, az utasításnak megfelelően, majd választottunk egy kapcsolót is. Ezek után a block diagram ablakban összehuzaloztuk a kapcsoló kimenetét a LED bemenetével. Majd futtattuk a programot a kapcsoló lekapcsolt és felkapcsolt állapotában is, utóbbinál tapasztalhattuk, hogy a LED a front panelben világított.

Mértékegység átváltó

Kiválasztottunk az adott listából egy numerikus kontrollt és egy numerikus indikátort. Ezután a kontroll tulajdonságainál a display format fül alatt, a format string mezőnél átállítottuk a mértékegységet km/h-ra (nagyon fontos a „#_” jel meghagyása az átállítás során, a felesleges tizedes jegyek elhagyása végett). Ugyanezt elvégeztük az indikátorra is, m/s-al. Ezután beillesztettünk a block diagramba a function menüből egy szorzás műveletet, valamint egy valós numerikus állandót (DBL numeric constant), melynek értékét 3,6-ra állítottuk, hiszen ez a váltószám a két mértékegység között (1 m/s = 3,6 km/h). Végül pedig a megfelelő módon összehuzaloztuk a numerikus kontrollt és az állandót a szorzással, majd ezt az indikátorral, ezzel kialakítva a megfelelő műveleti sorrendet.

2) **Alakítsa át az 1. pont feladatát olyanra, hogy csak akkor történjen mértékegység átszámítás, ha a kapcsoló ki van kapcsolva, és a mértékegység átszámítás legyen subvi - ben elhelyezve.**

Az „Mértékegység átváltó” résznél leírt rendszert egy case struktúrába helyeztük el (functions/structures/case structure), amelynek logikai értékét igazra (true) állítottuk. Ezután a LED kapcsolóját összekötöttük a case struktúrával. Így akármikor felkapcsoltuk a kapcsolót, a mértékegységátváltás megtörtént. A subvi megvalósításhoz kijelöltük a block diagram megfelelő szekcióját, majd létrehoztunk belőle egy subvi-t (edit menü/create subvi). Természetesen ezután el is mentettük ezt.

3) **Egy nyomógombot egyszer megnyomva induljon el egy kockadobás és numerikus kijelzőn és mutatós műszeren jelezze ki a dobás értékét. Ha az eredmény 6-os akkor gyulladjon ki egy kör alakú zöld LED.**

Kiválasztottunk egy OK-gombot a kapcsolók listájából, melyet átneveztünk „dob”-ra, hogy jobban illeszkedjen a feladat kontextusába . Az OK-gombot összekötöttük egy case struktúrával (true-ra állítva), ezután minden további elemet már ebbe struktúrába helyeztünk. Egy random number generatort helyeztünk ki, valamint egy 5 értékűre állított (long integer típusú) állandót, majd ezeket összeszoroztuk. Ezután ezt, és egy 1 értékű (long integer) állandót összeadtunk. Erre azért volt szükség, mert a véletlenszám-generátor 0 és 1 közötti értékeket képes csak generálni, viszont nekünk 1 és 6 közöttiekre van szükségünk.

Mivel egész számokra van szükségünk kimenetként, a kapott értékeket átkonvertáljuk byte integerre (ezen konvertáló elemmel kötjük össze az összeadást). Ezt az elemet végül még összehuzalozzuk az indikátorral és a front panelben kiválasztott mutatós műszerrel is, így elméletileg megfelelő eredményeket produkáló programrészhez jutottunk.

Végül megvizsgáltuk ennek az értéknek az esetleges egyenlőségét egy 6-ra beállított értékű numerikus állandóval mégpedig úgy, hogy összekötöttük a konvertáló elemet és a 6-állandót az „equal?” elemmel. Ezt az egyenlőségvizsgáló elemet (comparison/equal?) pedig összekötöttük egy kör alakú zöld LED-del. Amennyiben 6-ost „dobtunk”, azaz az indikátor és a mutatós műszer szerint egyaránt 6 volt a kimeneti érték, a zöld LED kigyulladt.

4) **Mérje meg és jelezze ki a mértékegység átszámító subvi futási sebességét! Alakítsa át a 2. pont programjait úgy, hogy azok többször fussanak le. A mért adatokból határozza meg az egyes programok egyszeri lefutása által felhasznált futási időt!**

A 2. feladat eredményeit átmásolva, új néven lementve tovább szerkesztettük, ügyelve arra, hogy a SubVI is átmentésre kerüljön. A feladat megoldásához szükségünk volt egy 3 db frame-ből álló szekvenciára, mely segítségével az általunk megszabott sorrendben hajtódnak végre a megfelelő parancsok.

Az első frame-be beillesztettünk egy Tick Count elnevezésű időmérő függvényt, mely még a mértékegység átváltás megkezdése előtt elindít egy időmérést. A második frame-be helyeztünk egy For ciklust, melyben elhelyeztük a mértékegység átváltást végző feladatrészeket. A ciklus „loop count”-ját egy 10000-re állított konstanssal összekötve erre a számra állítottuk át, így ennyiszer futtattuk le a loop-ot. A harmadik frame-be egy újabb időmérő Tick Count függvényt raktunk, melynek mért értékéből kivontuk az első frame időmérője által mért értéket, majd ezt 10000-el osztottuk, hiszen ennyiszer futtattuk le a programrészt a For ciklusban, így az egyszeri futás idejét csak „visszaosztással” kaphatjuk meg. Végül az így megkapott értéket egy parcellákon kívüli numerikus indikátorral írattuk ki. Így megkaptuk az egyszeri futás időtartamát.

5) **Az előlapon helyezzen el egy kapcsolót és egy numerikus kijelzőt. Mérje meg az emberek időérzékelő képességének pontosságát. A feladat az, hogy lehetőleg egy másodpercig tartsa bekapcsolva a kapcsolót. A visszakapcsolás után írja ki a kijelzett tényleges időtartamot. Ha az eltérés 10%-ot nem halad meg, akkor gyújtson ki egy zöld LED-et.**

A feladat megoldását a szükséges elemek elhelyezésével kezdtük a Front Panelen. Elhelyeztünk egy kapcsolót, melynek a viselkedését átállítottuk „Switch until released”-re, így a program addig fogja mérni az időt, amíg a gomb le van nyomva. A kapcsolón kívül - az eredmény kijelzésének céljából - egy numeric indicator kapott helyet, illetve a zöld LED, mely felvillanással jelzi, hogy sikerült-e a 10%-on belül teljesíteni.

A program helyes működésének érdekében létrehoztunk egy 4 frame-ből álló flat sequence-t, melynek első frame-jében egy while ciklus található, mely addig fut, ameddig a kapcsoló értéke hamis. Erre azért volt szükség, hogy a program működése folyamatos legyen. A kapcsolóból létre hoztunk egy lokális változót, melyet jobb klikkel olvasás típusra állítottunk, és ezt elhelyeztük egy while ciklusban a 3. frame-ben, mely egészen addig fut, míg a kapcsoló igaz értékre van állítva. A ciklust a 2. és a 4. frame-ben található két Tick Count függvény fog közre, melyeket az eltelt idő meghatározására használtunk. A két függvény által mért eredményt kivontuk egymásból, majd a mértékegység átváltása céljából 1000-rel elosztottuk. Az osztás eredményét egyik ágon összehuzaloztuk az idő kiíratásának céljából létrehozott numeric indicatorral, a másik ágon pedig logikai Less or Equal, illetve Greater or Equal logikai műveletek segítségével meghatároztuk, hogy 0,9 és 1,1 közé esik e. A mért logikai eredményt összehuzaloztuk a mérés pontosságát jelző LED-del.

6) **Készítsen egy szinusz, háromszög négyszög -generátorból és voltmérőkből álló műszer-együttest. Szabályozható legyen a generátor kimenő és offszet feszültsége, valamint frekvenciája. Az egyik voltmérő mutassa a mért jel effektív értékét, a másik pedig a csúcsértékét.**

A három különböző műszert egymás alatt helyeztük el. A front panelen található elemeik minden esetben megegyeznek. Mindegyik esetben található 3 knob – egy a kimeneti

feszültség, egy az offszet feszültség, illetve egy a frekvencia szabályozásának céljából, 2 Meter típusú mutató az effektív érték és a csúcsérték kijelzésére, illetve egy darab Waveform Graph, a görbék megjelenítésének érdekében.
A programozás a három műszer esetében hasonlóan történt. Először a Block Diagramon a függvények közül, a „Signal Processing” menü alatt a „Wfm Generation” alpontból kiválasztottuk a „Simulate Signal” függvényt Sinus típussal, majd a „Wfm Measurements” menüpontból kiválasztottuk az „Amplitude and Level Measurements” opciót. Létrehozásánál figyelve, hogy az „RMS”, valamint a „Positive Peak” legyenek bepipálva, ezek segítségével tudjuk meghatározni az effektív- és a csúcsértéket.
A Simulate Signal bemeneti értékeit összehuzaloztuk az azoknak megfelelő Knobokkal, majd a kimenetét egy Waveform Graph-fal, illetve a korábban létrehozott „Amplitude and Level Measurements”-szel. Utóbbi RMS kimenetét az effektív értéknek, Positive Peak kimenetét pedig a csúcsértéknek megfelelő mutatóval kapcsoltuk össze.
A háromszög és négyszög generátornál minden hasonlóan történt, csak a Simulate Signal létrehozásánál a típust Triangleként, illetve Squareként határoztuk meg.