LabVIEW I. jegyzőkönyv

A mérést végző személyek: Ekart Csaba [ZWPMKP]

A mérés időpontja és helyszíne: PPKE-ITK 420-as mérőlabor, 2017.03.16. 15:15-18:00

A mérés célja: A LabVIEW programcsomaggal való ismerkedés, a kijelölt feladatok megoldása

Tapsztalatok a LabVIEW programrendszer kapcsán

A LabVIEW a National Instruments mérésautomatizáló programcsomagja, melynek első verziója 1986-ban jelent meg. A környezetben különböző méréseket, kísérleteket végezhetünk virtuális műszerek segítségével, melyeket magunk programozhatunk. A LabVIEW a saját grafikus programozási nyelvével működik, tehát a programozás során – más nyelvekkel ellentétben – nem szöveges kód készül, hanem a különböző elemeket, függvényeket, operátorokat megfelelő módon összekapcsolva építhetjük fel a programunkat.

Habár a grafikus programozási nyelv mivolta elsőre néhányaknak nehézséget jelenthet, a módszer jóval intuitívabb, és számos esetben könnyebben átlátható és használható, mint a szöveges társai.

Mérési feladatok megoldása

1. Feladat

Az előlapon elhelyezett tetszőleges típusú kapcsolót bekapcsolva gyulladjon meg egy szögletes sárga LED. A km/h mértékben beadott sebességet írja ki és mutassa meg egy tetszőleges formájú kijelző m/s egységben. Figyelje meg a két rész egymástól független működését!

A feladat megoldása során első lépésben elhelyeztem a *Front Panelen* a szükséges elemeket – így egy *Slide Switchet*, egy *Numeric Controlt*, egy *Numeric Indicatort*, egy *Square LED-et*, illetve egy *Meter* típusú kijelzőt. Az elemeket az átláthatóság érdekében átrendeztem, átméreteztem, és funkcióját tükrözően elneveztem, továbba a LED beállításait úgy modosítottam, hogy kikapcsolt állapotban narancssárga, bekapcsolt állapotban sárga színben világítson. A *Block Diagramon* a kapcsolót összehuzaloztam a szögletes LEDdel.

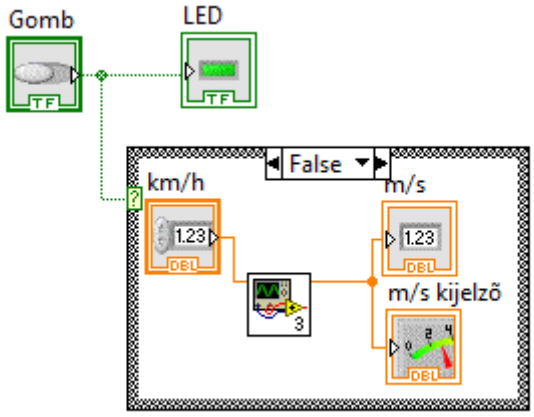
A mértékegység átalakítás során figyelembe vettem, hogy a két mértékegység közötti váltószám 3,6. Az átalakítás érdekében létrehoztam egy osztás függvényt, illetve egy 3,6 értéket felvevő *Numeric Constantt*. A *Numeric Control* kimenetét összehuzaloztam a *Divide* függvény x, míg a létrehozott 3,6-os konstanst az y bemeneti értékével. A függvény eredményét a *Numeric Indicatorhoz* kapcsoltam.

2. Feladat

Alakítsa át az 1. pont feladatát olyanra, hogy csak akkor történjen mértékegység átszámítás, ha a kapcsoló ki van kapcsolba, és a mértékegység átszámítás legyen subvi-ban elhelyezve. Vigyázzon arra, hogy hibás adat ne jelenjen meg a kijelzőn!

A feladat átalakítása során első lépésben elhelyeztem egy *Case Structuret*, melynek bemenetét a korábban elhelyezett *Slide Switchhez* huzaloztam, és értékét *False*-ra állítottam. A *Case Structurebe* áthelyeztem a mértékegység átalakításához tartozó programrészletet.

A *Block Diagramon* kijelöltem a konstanst, és az osztás műveletet, majd az *Edit* menüsorban a *Create SubVI* opciót választottam. Így a mértékegység átalakításának művelete egy subvi fájlba került, melyet *subvi.vi* néven elmentettem.

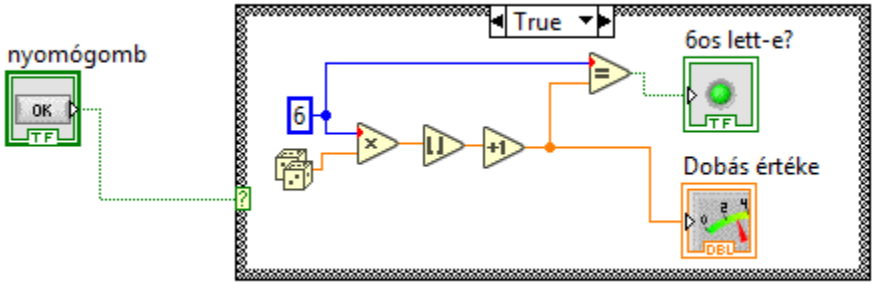


3. Feladat

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. Figyeljen arra, hogy ne legyen hamis a kocka, azaz minden értéke 1 és 6 között azonos valószínűséggel forduljon elő! A jegyzőkönyvben rögzítse azt a számítást, mely bemutatja az egyenletes valószínűségű előfordulást!

A feladat elkészítéséhez elhelyeztem a *Front Panelen* a méréshez szükséges elemeket, így egy *OK buttont*, egy *Round LED-et*, és egy *Meter* kijelzőt. Az elhelyezett elemeket átneveztem és átméreteztem az átláthatóbb megjelenés céljából. A *Meter* kijelzőn a kezdeti értéket 1-re, a végső értéket 6-ra az elemek közti különbséget pedig 1-re állítottam, a dobókocka lehetséges eredményeinek megfelelően.

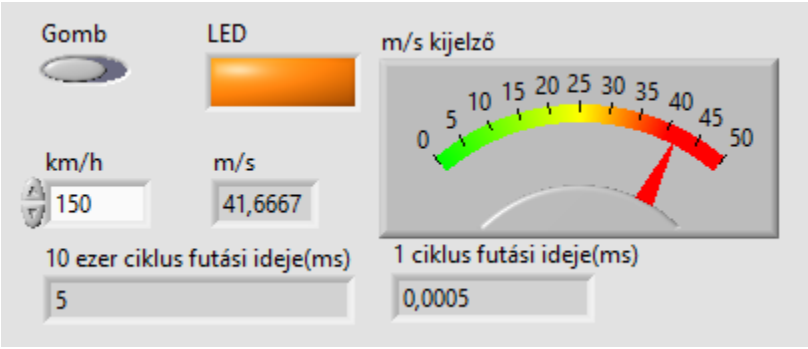
Első lépésként létrehoztam egy *Case Structuret True* értékkel, melynek az *OK button* kimeneti értéket adtam meg bemeneti feltételnek. A létrehozott *Structurbe* beillesztettem egy *Random Number (0-1)* függvényt, illetve egy 6-os értékre állított *Numeric Constantot*, s beállítottam őket egy *Multiply* függvény két bemeneti értékének. Ez a számítás egy valós számot fog adni 0 és 6 között, azonban mivel dobókockával csak egész szám dobható kerekítésre lesz szükségünk. A számítások helyességének okán a szorzás kimenetét tovább kötöttem egy *Round Toward -Infinity* függvénybe, mely elvégzi a szám lefelé kerekítését. A szám tehát amit kapunk 0 és 5 közé esik, így még az eredményt eggyel növelni kell, melyhez az *Increment* függvényt használtam. A függvény kimenetelét összehuzaloztam a *Meter* kijelzővel, illetve a korábban létrehozott 6-os konstanssal egyetemben egy *Equal?* függvénnyel kapcsoltam össze, melynek kimeneti értékét a *Round LED-hez* csatlakoztattam. Így a LED felvillan, ha a dobás értéke 6-ost vesz fel.



4. Feladat

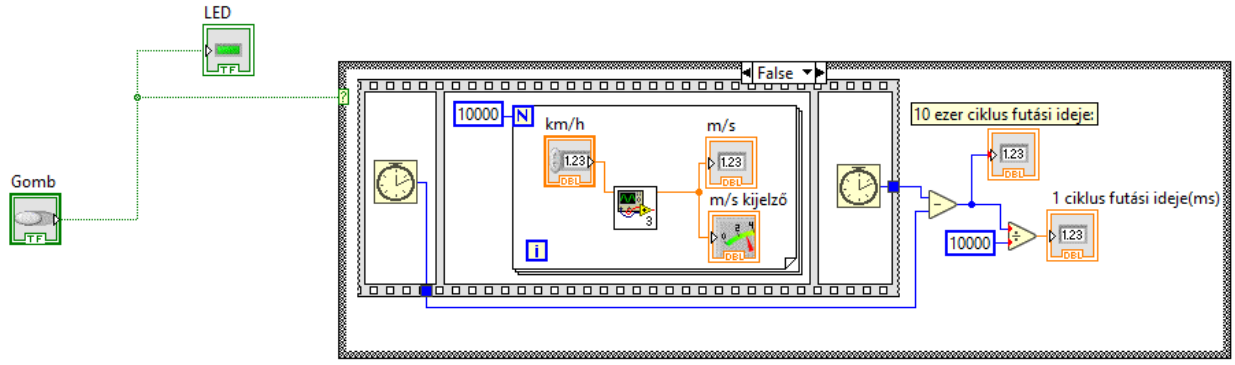
Mérje meg és jelezze ki a mértékegység átszámító subvi futási sebességét! Alakítsa át a 2. pont programjait úgy, hogy azok többször fussanak le. A mért adatokból határozza meg az egyes programok egyszeri lefutása által felhasznált futási időt! A program tervezése során figyeljen arra, hogy a mért eredmény pontossága legalább 1% legyen!

Az első két feladatnál használt elemeken kívül további két *Numeric Indicatort* helyeztem el a *Front Panelen*, a ciklus futási ideinek megjelenítésének céljából.



Az időmérés megvalósításához létrehoztam egy *Flat Sequence* struktúrát a korábbi *Case Structure* belsejébe, melyhez további két *framet* csatoltam így összesen 3-at kapva. A két szélső *framebe* egy-egy *Tick Count (ms)* függvényt helyeztem el, míg a középső *frameben* egy *For Loopot* hoztam létre, melynek küszöbindexét 10 ezerre állítottam. A *For Loopon* belül helyeztem el az 1-es és 2-es feladat során már ismertetett sebesség mértékegység átalakító részprogramot. A *Flat Sequencen* kívűl, de továbbra is a *Case Structuren* belül egy *Substract* függvénnyel kapcsoltam össze a két *Tick Count (ms)* értékét a megfelelő

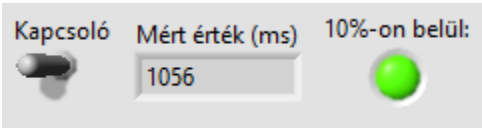
sorrendben. A kapott értéket az egyik *Numeric Indicatorhoz* huzaloztam. Ez megjeleníti 10 ezer ciklus futásának sebességét. Ennek értékéből egy ciklus futás ideje úgy határozható meg, hogy a kapott eredményt elosztjuk 10 ezerrel. Az eredményt tehát ennek megfelelően a logikus sorrendben összehuzaloztam egy *Divide* függvénnyel, melynek kimeneti értékét összekötöttem a másik *Numeric Indicatorral.*



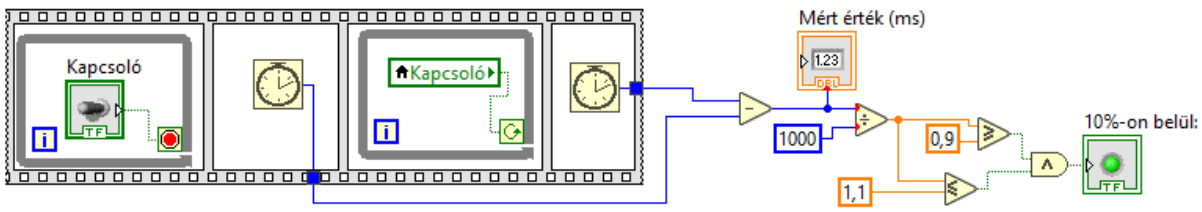
5. Feladat

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

Első lépésként elhelyeztem a *Front Panelen* a szükséges elemeket, így egy *Horizontal Toggle Switch* kapcsolót, egy *Numeric Indicatort*, és egy *Round LEDet*.



Az előző feladat tapasztalatait felhasználva az időméréshez létrehoztam egy *Flat Sequencet*, azonban most az előző feladattal ellentétben összesen 4 *frameből* állt. Az első *frameben* egy *While Loopot* hoztam létre melybe beillesztettem a *Front Panelen* létrehozott *Horizontal Toggle Switch* kapcsolót. A ciklus feltételét hamisra állítottam, majd összehuzaloztam a kapcsoló kimeneti értékével. A második és negyedik *frameben* egy-egy *Tick Count (ms)* függvényt helyeztem el, melyek közre fogják a 3. *frameben* található második *While Loopot*. A ciklus belsejében egy *Local Variablet* helyeztem, el melyet az eredeti kapcsolóból hoztam létre, és értékét olvasásra állítottam a *Change to Read* opció segítségével. A változó kimenetét a ciklus feltételéhez kapcsoltam, melyet *true* értéken hagytam. A két *Tick Count (ms)* függvényt az előzőekhez hasonlóan egy *Substract* függvénnyel huzaloztam össze, mely megadja a mért értéket milliszekundum mértékegységben. A kimeneti értéket tehát a feladat elején létrehozott *Numeric Indicatorhoz* kapcsoltam.



A 6-os dobás esetén felvillanó LED miatt létrehoztam egy *Divide* függvényt, melyet a logikus sorrendben összekötöttem az előző részfeladat kimeneti értékével, illetve egy újonnan létrehozott 1000-es *Numeric Constanttal*, így elérve, hogy az értéket másodpercben kapjam meg. Ennek kimeneti értékéről kell megállapítani, hogy 0,9 és 1,1 közé esik e. Ennek meghatározásához egy *Greater or Equal?,* illetve egy *Less or Equal?* függvényt hoztam létre, melyeknek x bemenetét az 1000-rel való osztás eredményéhez kapcsoltam, kimenetét pedig rendre egy 0,9-es és egy 1.1-es *Numeric Constanthoz.* Ezek kimeneti értékeit egy *And* függvénnyel kapcsoltam össze, melynek eredményét a létrehozott *Round LED-hez* kapcsoltam. Így a LED felvillan, amennyiben a mért idő és a tényleges 1 másodperc eltérése kisebb vagy egyenlő mint 10%.

Megjegyzés: A kapcsoló beállításainál az *Operation* fül alatt *Button behavior* résznél választhatjuk *Switch until released* opciót, ekkor az idő mérés addig tart, amíg a felhasználó nyomva tartja a gombot. A 10%-os eltérésen való belül esés kiszámításhoz nem feltétlen szükséges a rész eredmények átváltása másodperce, de számomra átláthatóbbá tette a programot.