# LabVIEW 1 jegyzőkönyv

Levente VAJNA
(Mérési partner: Válik Levente Ferenc)
(Gyakorlatvezető: Tihanyi Attila Kálmán)
Pázmány Péter Katolikus Egyetem, Információs Technológiai és Bionikai Kar
Magyarország, 1083 Budapest, Práter utca 50/a
vajna.levente@hallgato.ppke.hu

***Kivonat*—Megismerkedtünk a LabVIEW blockprogramozási, és szimulációskörnyezettel. Elkészítettük első programjainkat, mint egy LED villogtatás, dobókockával való dobás szimulációja, és még különböző típusú jeleket is mintavételeztünk, valamint azok effektív értékét számoltuk ki.**

***Keywords*-LabVIEW; átváltás; dobókocka; mintavétel; effektív feszültség**

Mérés ideje: 2023.03.16.

## I. feladat: LED használat, és m/s - km/h átváltás

Elsőként miután a LabVIEW megnyitási, telepítési, és BlankVI készítési nehézségeivel megbirkóztunk, mérőpartneremmel megcsináltuk az első kis programunkat. A nyomógombunkat, mint bemeneti eszköz konfigurálnunk kellett alapértelmezettről olyan állapotra, hogy amíg lenyomva tartjuk a gombot, addig égjen a LED. A Block Diagram oladalon pedig a gombhoz kellett kötni a kimeneti LED-et. Ezt is minimálisan át kellett állítani, hogy ovális legyen, és sárga, ne alapértelmezett zöld, kerek.

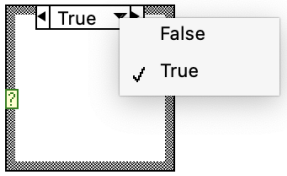
Továbbá egy átváltási feladatot is meg kellett oldjunk a programban, ehhez egy Numeric Contor-ra és egy Numeric Indicator-ra volt szükségünk, hiszen amit a bemeneten $m/s$ mértékegységben megadunk, azt egy kimeneten $km/h$ mértékegységben kell visszaadjuk. A Block Diagramon a bejövő értéket valamint a $3,6$ lebegőpontos konstanst vezetékkel összekötöttem egy szorzó műveleti egységgel, és az így kapott eredményt vezetékkel összekötöttem a kimeneti numerikus indikátorral.

Érdekes volt megtapasztalni, hogy miként előadáson Tanárúr elmondta, a parancsok nem irány szerint futnak le, hanem a műveletvégrehajtás szempontjából vett irányban, azaz ahogy bekábelezetük. És bár nem tudom leellenőrizni, hogy tényleg egyszerre futnak a külön lévő szálak, tekintve, hogy nincs két kurzorom, a működését megfigyelve azt tapasztaltam, hogy közel egyidőben biztosan tudja futtatni.

## II. feladat: Ki/Bekapcsolható átváltás, mértékegység kijelzése

Itt e feladat során az előző feladatot egészítettem ki olymódon, hogy a kapcsoló benyomott állapotában ne végezzen átváltás, ellenkező esetben pedig igen, és hogy még szebb legyen, a mértékegységet is kiírjuk mellé.

Ezt akképp tudtam megtenni, hogy behelyeztem a Block Diagram-ra egy Case struktúrát (1. ábra), mely a kapcsoló kiengedett állapotánál a bejövő számot megszorozza a megismert módon $3,6$-tal, és azt adja ki a numerikus indikátornak, továbbá egy string konstans értékét, ami pedig nem más, mint a ”km/h” -t is kiadja, amit egy string indikátoron jelenítünk meg, illetve a kpacsoló benyomott állapotánál csupán a bemeneti értéket viszi tovább a kimenetre, és a ”m/s” szöveg konstanst is kiadja a string indikátorra.




1. ábra. Case struktúra

Az 1. ábrán egy Case struktúra látható. Mivel a LabVIEW program nem tartalmaz külön ”if” elágazást, így mindennemű kódbeli elágazást ezzel lehet megvalósítani. Használható logikai kapuként, ez felel meg az ”if”-nek, ahol ha nincs kimeneti érték, amit visszaad az elágazás, elég csak az egyik eshetőségre elkészíteni a programsorozatot, azonban ha már kimeneti értékkel is rendelkezik a Case struktúránk, nem elhanyagolható mindkét ágnál valamilyen kimeneti értéket bekábelezni. Van lehetőségünk például string típusú elágazást is létrehozni, vagy integer, vagy double típusút is, mindez azon múlik, hogy milyen típusú vezeték kerül bekötésre a bemeneti kis ”[?]” kérdőjelbe. Kettő fontos dologra figyelni kell, hogy ezeknél ugyanúgy be kellenek kötve a kimenetek, ha vannak, illetve a lényeges, hogy legyen ”Default”, azaz alapértelmezett lehetőség is megadva, különben a VI addig nem is futtatható, amíg ez nem történt meg. Még egyféle lehetőségünk van, ez az úgynevezett ”Enumerate”, ami annyit takar, hogy előre megadott lehetőségeken kívül más biztosan nem lesz, ezért ennél a típusú elágazásnál nem szükséges az alapértelmezés beállítása. Ennél a módnál van még egy könnyedség, jobbegérrel a Case struktúrára kattintva kiválaszthatjuk azt a lehetőséget, hogy automartikusan legeneráljon minden lehetséges elágazási lehetőséghez egy ablaskocskát, ami azért is lehet segítség, mert így biztosra mehetünk azzal, hogy nem hagyunk ki semmit.

További nehézséget okoztott elsőre a SubVI elkészítése, de aztán megtaláltam azt a lehetőséget, mely a kijelölt objektumokat behelyezi egy SubVI-ba. (lásd: 4. ábra)

## III. feladat: Kockadobás 3 kockával

Először a partneremmel a dobókockát készítettük el. Egy random $I = [0; 1)$ intervallumbeli számot generáló objektum segítségét vettük igénybe. A generált számot felszoroztuk 6-tal, így $I_1 = [0; 6)$ lehetséges számot kapunk, de ehhez még hozzáadtunk egyet, így az intervallumunk $I_2 = [1; 7)$, vagyis 1 lehet még, de 7 nem. Ezt a számot még egy alsó egész számra kerekítő objektumhoz kábeleztük be, vagyis a valós szám egészrészét veszi, így bebiztosítva, hogy minden egyes számhoz azonos méretű intervallum tartozzon, vagyis azonos eséllyel ”dobódjon ki” mind a hatféle szám.

Például:

$$2 : [2; 3)$$

$$3 : [3; 4)$$

$$6 : [6; 7)$$

Ezt a objektumsorozatot SubVI-á alakítottam, majd hozzáadtam a VI-hoz ebből további kettőt. Ahhoz, hogy ezek gombnyomásra menjenek, megint a Case struktúrára esett a választásom, és ha a bemeneti érték True, vagyis a gomb le van nyomva rövid ideig, akkor a dobott kockák értékeit egyesével egy-egy Numeric Indicator-on megjelenítem, majd a három kocka eredményét összeadom, és az így kapott összeget és egy 18 konstanst egy $=$ operátorral összehasonlítom, és amennyiben ez a feltétel teljesül, a kis zöld LED kivillan.

Számításaim a következők:

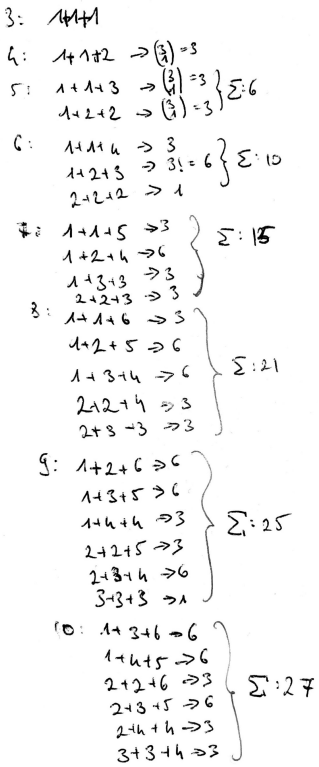
$$\begin{aligned}
P(3) &= \frac{1}{6^3} = \frac{1}{216} \\
P(4) &= \frac{3}{216} = \frac{1}{72} \\
P(5) &= \frac{6}{216} = \frac{1}{36} \\
P(6) &= \frac{10}{216} = \frac{5}{108} \\
P(7) &= \frac{15}{216} = \frac{5}{72} \\
P(8) &= \frac{21}{216} = \frac{7}{72} \\
P(9) &= \frac{25}{216} \\
P(10) &= \frac{27}{216} = \frac{1}{8} \\
P(11) &= \frac{27}{216} = \frac{1}{8} \\
P(12) &= \frac{25}{216} \\
P(13) &= \frac{21}{216} = \frac{7}{72} \\
P(14) &= \frac{15}{216} = \frac{5}{72} \\
P(15) &= \frac{10}{216} = \frac{5}{108} \\
P(16) &= \frac{6}{216} = \frac{1}{36} \\
P(17) &= \frac{3}{216} = \frac{1}{72} \\
P(18) &= \frac{1}{216}
\end{aligned} \tag{1}$$

Az egyes résszámításaimat tartalamazza a 2. ábrán látható levezetés. Elegendő volt csupán elmenni 10-ig a számításokban, hiszen egyrészt korábbi matematikai ismereteink is alátámasztják, illetve ugyanezek az eredmények lesznek visszafelé. Olyan módon, mint kombinatorikában is $n$-ből kiválasztani $k$ darabot ugyanannyiféleképpen lehet, mint $k$ darabot NEM kiválasztani, azaz $(n-k)$ darabot kiválasztani.

Az 1. egyenletben a valószínűségeket a 2. ábrán látható részszámításaim alapján határoztam meg, illetve azon eredményeket osztottam $6^3$-nal, hiszen az az összes lehetőség, tekintve, hogy 3 kockával dobunk, és mindegyik 6-féle eredményt vehet fel.

Szépen látszik, hogy a 10-nek és a 11-nek ugyanakkora a valószínűsége, ezek fordulnak elő legtöbbször. Így nem meglepő, hogy sokszori futtatás után az a kettő ugyanannyiszor fog kijönni nagyságrendileg, és az lesz a legtöbb. Tapasztalatszerzés során én is lejátszottam a szimulációt 20-30 alkalommal, és az is ezt az állítást bizonyította.

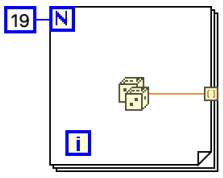
A kis led bekötése, ha 18 az összeg a korábbiakhoz hasonló módon történt, azonban nehéz volt tesztelni, hogy tényleg működik-e, hiszen matematikailag minden 216.-ra kell csak hogy kijöjjön, ezért ennek érdekében a tesztelés erejéig egy másik számra, 11-re írtam át, hogy szemléletese jobban a működést.



2. ábra. Számításaim az egyes valószínűségekre

## IV. Feladat: 10000 kockadobás 3 kockával

Ez a feladat okozta számomra a legnagyobb fejtörést, hiszen az úgynevezett array-ek működési elve kicsit nehezebben megérthető. Természetesen egy 10000-szer lefutó For Loop (3. ábra) használata elengedhetetlen ebben a feladatban, azonban az mindenképp tömbként adja vissza az eredményeket.




3. ábra. For Loop

A 3. ábrán keresztül mutatnám be a For Loop működési elvét. A C++ programozási nyelvből már jól ismert ciklust mind ismerjük, itt is hasonlóan végzi el a feladatot. A bal felső sarokban hozzá kell kötni egy integer konstanst, amit akár egy jobbegér kattintás/Create Constant opcióval még gyorsabban is elvégezhetünk. Ez az 'N' jelenti, hogy a ciklus hányszor fog lefutni. A kis 'i' betű is megjelenik, ami alatt a

ciklusváltozót értjük. Ez szolgál arra, hogy menet közben 0-tól indexelve le tudjuk kérni, hogy hányadik lefutásánál tarunk a For Loop tartalmazta kódrészletnek. Fontos tulajdonsága továbbá a ciklusnak, hogy a kimeneti eredményeket nem egyesével érhetjük el, hanem a teljes lefutás végén, array-ben tárolva.

A feladat elkészítése során, hogy jobban átlássam, a három kockával való dobást, amiben külön SubVI-ok vannak, szintén egy további SubVI-já alakítottam. Ezen tízezer eredményt egy tömbként adja vissza a For Loop, de csak akkor szeretnénk ezzel foglalkozni, ha lenyomtuk a gombot. Ezért egy beépített elágazást raktam bele, mely azért van, hogy ha lenyomom, továbbításra kerüljön a friss tömb, ha viszont nincs lenyomva, akkor az előző eredmények továbbra is kiírva maradjanak, ne csak egy villanásnyira lássuk az eredményeket.

4. ábra. SubVI

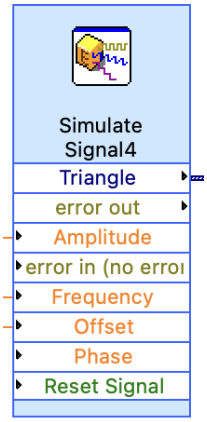
Egy LabVIEW VI-ba beágyazott SubVI látható 4. ábrán. Ez olyasformán működik, és úgy kell őket értelmezni, mint programozás során a függvényeket. Ha VI-ban egy bizonyos ”kódrészletet” az ember többször is fel szeretne használni, érdemes abból SubVI-t létrehozni, melyet úgy tehetünk meg, hogy a kívánt vezetékeket, paneleket, blokkokat és ki-, illetve bemeneteket kijelöljük a Block Diagramon, majd utána bal felül az Edit/Create SubVI opciót választjuk, hiszen onnantól kezdve könnyedén felhasználható bárhol a továbbiakban. Bármely már kész VI is beimportálható SubVI-ként egy másik tetszőleges VI-ba oly módon, hogy megnyitjuk mindkettőt, és a beágyazni kívánt VI jobb felső sarkában található 4. ábrabeli ikont Drag and Drop módon az új VI Block Diagramjára húzzuk. Innentől ugyanúgy működik kódon belül a maga be-, és kimeneteivel.

A következő For Loop (3. ábra) arra szolgál, hogy az összegeket 3-tól 18-ig megszámoljuk, olymódon, hogy ha egyenlő az elem a For Loop i ciklusváltozójával, akkor egy logikai igen helyett egy integer 1-est adjunk tovább, melyet egy előre beépített operátor tesz lehetővé. Ezen egyesek számát a szumma operátor megszámolja, array-ként továbbadja, és egy indikátor tömbben a Front Panelen megjeleníti. Ezt végigcsinálja mind a 16-féle dobott értékkel, és végül így külön-külön válogatva, de egy helyen egyszerre láthatjuk mindegyiknek az előfordulását.

Amiért tanulságos számunkra az eredmény, mert ekkora dobásszám már elég jól reprezentálja, hogy tényleg ezeknek az előfordulása a legnagyobb, és mint az előző bekezdésben kiszámoltam, $\frac{1}{8}$ valószínűséggel lesz 10, és ugyanekkora valószínűséggel, azaz minden nyolcadik lesz 11.

## V. feladat: Jelek

Egy beépített jelgenerátort (5. ábra) használtam munkám során, amit az effektív érték kiszámításához át kellett alakítsak egy tömbbé. Innen ki tudtam számítani a matematikai és fizikai ismereteim segítségével az effektív feszültséget, hiszen az nem lesz más, mint a vett minták négyzetes avagy kvadratikus közepe. Az egyik legkönnyebben ellenőrizhető módon tudtam tesztelni tudásomat, hiszen azt fizikán mind megtanultuk, hogy a hálózati effektív feszültség 230V, és kb. 325V ennek maximális értéke, hiszen a jel szinuszos, mivel váltakozó feszültségű áramról beszélünk. Ez a hálózati feszültség továbbá 50Hz-es, amit bár nem kell ismerni ennek kiszámításához, be lehet állítani a még életszerűbb szimuláció érdekében.




5. ábra. A mintavételezéshez jeleket szimuláló beépített objektum

Az 5. ábrán látható jelszimuláló egy hasznos beépített generátor, remekül alkalmazható többek közt ennél a szimulációnál is, de bárhol, ahol nem áll rendelkezésünkre éppen egy külső hardware. Sokféleképpen konfigurálható, ki tud bocsátani többféle jeltípust, mint szinuszost, háromszögalakút, négyzetest, vagy akár még fűrészest is. Be lehet állítani a mintaelőállítás számát, illetve, ami az egyik legfontosabb beállítás, hogy egész számú jelet állítson elő azonos időközönként, különben a frekvencia függvénye is lehetne az effektív érték, ami pedig helytelen lenne. További beállításra ad lehetőséget, hogy bemeneti numerikus értékkel szabályozható a kibocsátott jel amplitúdója, offszete vagy a frekvenciája is.

A beállított tekerők (”potméterek”) segítségével könnyedén beállítható mind a jel frekvenciája, mind a jel amplitudója, illetve még a kezdése is, vagyis az offszet is az imént leírtak alapján, de ennek a tesztelésére még külön, konstansokkal meghívott programot is készítettem. Szerencsémre stimmelt minden és az adatok megegyeztek az elvárt értékekkel, így kijött a 230V effektív feszültség, amit szerettem volna megkapni a számításokkal, de persze bármilyen más számmal is kijön, csak ez az az érték, amit középiskolából és a hétköznapokból mind jól ismerünk. Egyezést persze nem tudtam rá írni, mert szerintem a double nem látott tizedes számjegyei, amik a pontosság, illetve a kerekítés miatt eltűnhettek, de a numerikus kimeneten mégis jól látszik a 230V.

Ahhoz, hogy a jeltípust is ki lehessen választani, én egy legördülő szöveges bemenetet választottam ki céleszközként. Egy Case elágazásba behelyeztem minden ágához az adott típushoz tartozó jelgenerátort, mely annak kiválasztása esetén olyan típusú jelet fog adni, amilyet mi szerettünk volna. Ezt a jelet egy nagy ”oscilloscope” (Waveform Graph) bemenetére is bekötöttem, hogy a jel egy gráfon ábrázolva is szemlélhető legyen. A jelet egy speciális, úgynevezett dinamikus adat típus formában kapjuk meg a Signal Simulatortól. Annak érdekében, hogy ezzel számolni lehessen, egy beépített átalakító operátorral double típussá konvertálom, és azt adom át az effektív feszültség kiszámítására alkalmas

függvényemnek.

## VI. feladat: Mintavételezés és effektív érték számítás általánosítva

Tulajdonképpen én első esetben is ugyanazon képletet használtam, márpedig ugye a minták kvadratikus közepét vettem. Ezt az egyszerűség és a szépség kedvéért SubVI-já alakítottam, így mobilisabb is volt és nem okozott gondot a másik szimulációban is felhasználni. Ezt úgy tettem, hogy a beérkező tömb elemeit négyzetre emelem, majd azokat szummázom össze a korábban említett módon. Emellett a vezetéket korábban kettéoszottam, és a másik szálon a minták számát egy erre szolgáló beépített függvénnyel meghatározom. Ennek a kettőnek a hányadosát veszem, tehát elosztom a négyzetösszeget az összes mintával, majd végül az egészet négyzetgyök alá vonom, és így kapjuk meg a definíció szerinti effektív értéket: azt az egyenfeszültség-szintet vagy egyenáram-áramerősséget, amely átlagosan ugyanakkora Joule-hőt termel egy ellenálláson. [1]

A számítások itt is helytálltak:

$$U_{eff} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} U_i^2}$$

Ebben az esetben mivel minták feldolgozásával számolunk, használható ez a képlet, de nagyon fontos arra figyelni, hogy itt is csak egy periódusnyi mintára számítsuk, különben az effektív érték változni fog, és bár nem drasztikusan, de fog, annak ellenére, hogy nem lenne szabad, mivel az effektív értéknek nem szabad függnie attól, hogy hány van belőle másodpercenként, azaz mekkora a frekvenciája. Abban az esetben, amikor egy folytonos, periodikus függvény írja le a feszültség időbeni változását, akkor az integrálos formulát használjuk:

$$U_{RMS} = \sqrt{\frac{1}{T}\int_{t_0}^{t_0+T} u^2(t)dt}$$

ahol $t_0$ egy periódus kezdete, $T$ pedig a periódusidő.

## Lezárás

Összegzésképp, kipróbáltuk magunkat egy új, izgalmas környezetben, és bár a kezdeti nehézségek megvoltak, mégis sikerült mindent elvégeznünk. Ámbár a laborban elkezdtük a feladatok elvégzését, de csak a második feladatig jutottunk órán, a többit azon kívül készítettük el. Annak ellenére, hogy az Érzékelő robotika laborban találkoztam rendes, gyönyörűen jelmegjelenítő-, illetve mérőműszerrel, ez a program is egészen szemléletes, jól alkalmazható. És izgalmas, hogy itt is milyen jól visszaköszönnek a középiskolai fizika fakultáción tanultak. Kíváncsian várom amikor tényleges műszerekkel kötjük majd össze a programot, és ténylegesen mérjük azokat, nem csak szimuláljuk. Ott majd várhatóan jobban előjönnek a hibák, és akkor majd azokkal jobban lehet foglalkozni.

## Hivatkozások

[1] M. Zsolt, „Measuring ac voltage,” 04 2017. [Online]. Available: https://uni-obuda.hu/users/markellazs/mt/eng/MT1/UAC(ac%20feszultseg%20meres_angol).pdf