LabVIEW1 mérési jegyzőkönyv

Szauter Ajtony, Zsumbera Olivér
(Supervisor: Naszlady Márton Bese)
Pázmány Péter Katolikus Egyetem, Információs Technológiai és Boinikai Kar
1083, Budapest Práter utca 50/A
2024.03.20
szauter.ajtony@hallgato.ppke.hu
zsumbera.oliver@hallgato.ppke.hu

***Kivonat*—A méréstechnika órán megismerkedtünk alaposabban a LabView program használatával. A jegyzőkönyvben kitérünk a megadott feladatok, mérések megvalósítására, körülményeire és elméleti megoldásaira .**

***Keywords*-LabView; Effektív érték,Fázishasítás**

I. Bevvezető Feladat

A LabView programban a Frontpanel-en elhelyeztünk a feladatban megadott elemeket mint - Round Led, Push Button, Numerical Control, Gauge. A Led színét a properties fülön át tudtuk állítani az alap zöldről a kívánt piros színre. A Round Led-et és a Push Buttont egy vezetékkel összekötöttük amivel tudtuk kapcsolni a Led világítását. A Numerical Controlt és a Gauge mutatót pedig egy külön vezetékkel kötöttük össze és megfigyelhettük, hogy a megadott érték megjelent a mutatón ugyan azzal az értékkel. Azt figyelhettük meg a program indításakor hogy a Led és gomb függetlenül működött a Numerical Control és a mutatótól, mert köztük nem hoztunk létre összeköttetést.

II. 2. Feladat

A második feladatban az első feladatot kellet tóvább fejleszteni, bővíteni. A Front panelre elhelyeztünk még egy Hozizontal Fill Slide-ot amit kicsit személyre szabtunk. A Hozizontal Fill Slide properties ablakában a Data Types fülön átállítottuk hogy csak Integer 32 bit számokat tudjunk megadni és az állásokat átírtuk 0-2-ig. Így sikerült megvalósítanunk egy három állású kapcsolót. A következő képés a mértékegység konvertálásának magvalósítása volt. Először is a Block Diagram ablakban létrehoztunk egy Case Structuret ami lehetőséget bíztosított számunkra, hogy a három állású kapcsoló külömböző értékeire más és más funkciókat tudjunk létrehozni. Az első azaz a "0." állás a m/s átváltása volt m/s-ba azaz amikor nem volt szükség számításra így simán a Case Structure-on keresztül be tudtuk kötni a vezetéket egy Numeric Indicatorba, hogy a Front Panel felületen láthassuk az eredményt. A második azaz az "1" állás a m/s átváltása km/h-ba. Ezt a feladatot egyszerüen a Case Structure-on belül az 1-es állást kiválasztva a Numeric Control értékét összeszoroztuk, egy általunk létrehozott konstansal aminek 3,6 értéket adtunk. A konstans értéke úgy jött ki hogy az óra és a másodperc között 60*60 a vátószám azaz 3600, a kilóméter és a méter között pedig 0,001, így 3600 * 0,001 = 3,6. A szorzat eredményét pedig ugyan úgy ugyan abba a Numeric Indicator-ba kötöttük be egy vezetékkel. A Case Srtucture-ünket a "2" állásba állítottuk és elkészítettük a m/s-ból való mérföld per órába átváltást a következőképpen. A váltószám kiszámolásához szükségünk volt arra, hogy 1 méter hány mérföld ami 1m = 0.000621371192 mile és az előző számolásunkból tudjuk hogy 1 óra 3600 másodperc így a váltószám $0.000621371192 * 3600 = 2.23693629$ amit mi 2.23694-re kerekítettünk. A váltószámot és a Numeric Control értékét összeszoroztük és a szorzás értékét vezetékkel bekötöttük a Numeric Indicatorba. A program készítése közben észrevehettük, hogy a Numeric Indicatorba kötött vezeték a Case Srtucture határán az ugynevezett Input tunnel üres négyzete csak a mindhárom Case elkészítésekor és kivezetésekor vált telire és csak ekkor tudtuk lefuttatni a programot. A feladat végén pedig egy SubVI-t készitettünk a Case Srtucture-ből és a benne lévő számolásokból. A következő táblázatban láthatóak a mért eredmények:

m/s	km/h	mph
10	36	22.36
15	54	33.55
20	72	44,73
25	90	55,92
30	108	67,1

II-A. Mérési bizonytalanág

Egy alacsony mérési bizonytalanságnak vehetjük azt, hogy a m/s és a mérföld per óra közötti váltószámot kerekítettük de ennek az értéke minimális.

III. 3. Feladat

A harmadik feladatban az egy kockajátékot kellett készítenünk mellyel háromszor dobunk és a három eredményt kiírjuk a képernyőre. Ezt mi úgy oldottuk meg, hogy csináltunk egy for ciklust mely háromszor fut le. Beletettünk egy Random Number (Range)-et melynek a minimum értékét egyre és a maximum értékét hatra állítottuk. Ezt kivezettük a for ciklusból mely így három számot generált melyet egy tömbben adott vissza. Ezt a részt belehelyeztük egy SubVI-be így volt 3 kocka SubVI-nk. Erre két darab kimenetet kötöttünk egy Array, mely kiírja a képernyőre a három számot, amit generáltunk, a másik egy szumma melyhez kapcsoltunk egy Numeric Indicatort, így ez kiadja a három kocka összegét. A szumma és az Numeric Indicator közé bekötöttünk egy Equal? -t, aminek a másik végére egy Numeric Constants melybe beírtam egy tizennyolcas számot. Az Equal? - kimenetére rákötöttem egy Boolen zöld ledet. Ennek az a lényege, hogy ha tizenhatos számot kapunk akkor zölden világít, ami lehetséges legnagyobb szám, amit ki lehet dobni három hat oldalú szabályos dobókockával. A valószínűséget a középiskolában tanultak alapján így írunk fel:

$$V = \frac{kedvező}{összes} \tag{1}$$

V= Ez a valószínűség, hogy mennyi eséllyel jöhet ki az érték.
Kedvező= Az az érték, amiből kijöhet az érték. Ennyi féle képpen jöhet ki az érték.
Összes= Az összes variáció, ami kijöhet.
Itt az összes érték pontosan kiszámolható. 6^3, mert három darab dobókocka van.

Számok	Hányféleképpen jöhet ki?	*Valószínüsége*
3	1	0,00463
4	3	0,013889
5	6	0,027778
6	10	0,046296
7	15	0,069444
8	21	0,097222
9	25	0,115741
10	27	0,125
11	27	0,125
12	25	0,115741
13	21	0,097222
14	15	0,069444
15	10	0,046296
16	6	0,027778
17	3	0,013889
18	1	0,00463

IV. 4. Feladat

Negyedik feladatban az volt a feladatunk, hogy a harmadik feladatban lévő dobókockás feladatot lefuttassuk 10000-szer. Ezt úgy oldottuk meg, hogy az egész hármas feladatot beleraktuk egy for ciklusba melyet 10000-szer futtattunk le. A for ciklusból az összegeket kivezettük egy General Histogramba melynek minimum értéke három volt és a maximum értéke 18. Ezeket Numeric Constantba helyeztük. A Bins-be tizenhatot vezettünk, mert tizenhat darab szám lehetséges a három dobókocka összegeként. Egy Histogram Graphba kivezettük az értékeket így kapunk a grafikonon egy harang alakot. Kivezettük még egy Histogramban, mely megmutatja a pontos számot, hogy hányszor kapjuk meg az adott összeget. A 10000 mérést háromszor lefuttattam és ezeket egy táblázatba kiírtam és vettem a darabszámok összegét. Ez után kiszámolható ezeknek a valószínűsége mely az előző feladatban kiszámoltól elhanyagolható különbséggel tér el.

Számok	1. mérés	*2. mérés*	3. mérés
3	51	48	54
4	141	133	141
5	268	277	293
6	460	436	431
7	716	749	709
8	981	970	932
9	1146	1175	1186
10	1242	1212	1229
11	1254	1205	1217
12	1190	1167	1204
13	989	1001	942
14	670	695	734
15	457	487	462
16	268	264	276
17	116	135	145
18	51	46	45

Számok	Átlag	*Valószínüség*
3	51	0,0051
4	138,3333	0,013833
5	279,3333	0,027933
6	442,3333	0,044233
7	724,6667	0,072467
8	961	0,0961
9	1169	0,1169
10	1227,667	0,122767
11	1225,333	0,122533
12	1187	0,1187
13	977,3333	0,097733
14	699,6667	0,069967
15	468,6667	0,046867
16	269,3333	0,026933
17	132	0,0132
18	47,33333	0,004733

Miért ilyen kevés az eltérés? Lehet, hogy a radnom szám nem is random szám?
A labview kettő féle random szám generátort használ a pszeudovéletlenszám-generátort és az az Linear Congruential Generatort. A pszeudovéletlenszám-generátorok (PRNG) olyan algoritmusok, amelyek látszólag véletlenszerű számokat hoznak létre, de valójában meghatározott kezdeti állapotból és előre meghatározott algoritmusok szerinti számításokkal jönnek létre. Az egyik gyakran használt PRNG algoritmus a Mersenne Twister (MT). Ez az algoritmus rendkívül hosszú periódust biztosít (2Î9937-1), így a generált számok hosszú sorozatban ismétlődés nélkül jönnek létre, ami ideális a véletlenszerűség szempontjából. A másik gyakran használt PRNG algoritmus az Linear Congruential Generator (LCG). Ez az algoritmus egyszerű és gyors, mivel egy egyszerű iteratív folyamatot alkalmaz, amely a kezdeti állapotot (seed) szorozza egy konstans értékkel, majd hozzáad egy másik konstanshoz, és a kapott eredményt használja az új szám generálásához. Habár az LCG gyors és egyszerű, a generált számok általában nem érik el ugyanazt a véletlenszerűséget és periódust, mint a Mersenne Twister által biztosítottak.

V. 5. Feladat

Az 5. feladatban elő kellett állítanunk szinusz, négyszög, fűrész és háromszög alakú jeleket Offszet Frekvencia és Amplitudó megadásával. Ennél a feladatnál 3 Numeric Controlt helyeztünk el amikkel az imént említett tulajdonságokat lehet beállítani. A különböző alakú jelek kiválasztásához pedig egy Horizontal Pointer Slide-ot használtunk amit beállítottunk a properties menüjében, csak Integer 32, azaz egész számú beviteleket fogadjon és 0-3 ig tartó értékeket adtunk meg amivel létrehoztuk a 4 állású kapcsolót. Egy Case Structuret használtunk a különböző kiválasztható kimenetekhez. Az első állás a négyzetes gráf amit a LabView beépített Waveform generátoraival hoztuk létre. Ezek a gráf generátorok a Functions ablak Signal Processing fülön a Waveform Generation részben találhatóak. Hasonlóan a szinus, fűrész és háromszög gráfokhoz ugyan innen a megfelelő generátort tettük be, a szinuszt a ”1” helyre, a háromszöget a ”2” helyre és a fűrészt a ”3” helyre tettük a Case-ben. Ezeknek az állásoknak a végeit kivezettük a Case Structure-ből. A program készítése közben észrevehettük, hogy a Numeric Indicatorba kötött vezeték a Case Srtucture határán az ugynevezett Input tunnel üres négyzete csak a mindhárom Case elkészítésekor

és kivezetésekor vált telire és csak ekkor tudtuk lefuttatni a programot. Ezt a vezetéket két helyre kötöttük be, az egyik Egy Waveform Graph gráf mutatóba ami egy grafikonon vizualizálja a függvényt. A másik végét pedig egy úgynevezett Get Waveform Components-be vezettük ami a Programming Waveform mnüpontban található. Ez a komponens megadja nekünk a koordinátarendszer szerinti Y tengelyen lévő értékeket és a hozzá tartozó x tengelyen lévő időt. Mivel fix pontokat kapunk vissza ezért a gráfoknak végtelen szakadási pontja van ezért a beépített integrálást nem tudtuk használni. Az integrálásra azért van szükség hogy a feladatban megadott effektív értéket ki tudjuk számolni. Az effektív érték kiszámolásának képlete a következő egy periódusra:

$$V_{eff} = \sqrt{\frac{1}{T} * \int_0^T v(t)^2 dt} \tag{2}$$

Az integrálszámítás alapját felhasználva egy körülbelüli értéket viszont ki tudtunk számolni a következőképpen: Az Y értékeket négyzetre emeljük majd a dt komponensünkel összeszorozzuk és egy beépített szumma matematikai komponensel összeadjuk ezeket az értékeket amivel megkapjuk a görbe alatti terület körülbelüli értéket azaz az integrálját. Majd eszt az eredményt összeszoroztuk a frekvenciával mert:

$$f = \frac{1}{T} \tag{3}$$

Majd ebből az eredményből négyzetgyököt vontunk és egy Numeric Indicator komponensre kiírtuk az eredmény amivel ki is számoltuk az effektív értéket.

Az effektív érték (RMS) egy matematikai fogalom, amellyel egy váltakozó áramú (AC) jel amplitúdójának egyenértékű egyenáramú (DC) értékét jellemezhetjük. Az effektív érték fontos szerepet játszik a villamosmérnöki tudományban és a fizikában, és számos területen alkalmazzák:

1) Villamosenergia-mérés: Az elektromos hálózatokban az effektív értéket használják a villamos energia mérésére. A fogyasztók által felhasznált energia az effektív áram és a feszültség szorzata alapján kerül kiszámításra.
2) Elektronika: Az elektronikában az effektív értéket használják az alkatrészek, például ellenállások és kondenzátorok méretezésénél. Ez fontos az áramkörök hatékonyságának és megbízhatóságának biztosításához.
3) Méréstechnika: A méréstechnikában az effektív értéket használják a váltakozó feszültségek és áramok mérésére. Ez fontos a fizikai jelenségek pontos méréséhez.
4) Hangtechnika: A hangtechnikában az effektív értéket használják a hangnyomás szintjének (SPL) mérésére. Az SPL egy logaritmikus skála, amely a hangnyomás effektív értékét méri decibelben (dB) kifejezve.

VI. 6. Feladat

A fázishasítás egy olyan jelenség, amely hullámok, például hang- vagy fényhullámok terjedése során figyelhető meg. Amikor a hullámok egy akadályba ütköznek, vagy áthaladnak egy nyíláson, a hullámok fázisa megváltozik. Ez a változás a hullámhossz egy részének eltolódását eredményezi.

A fázishasításnak számos hasznos alkalmazása van a tudomány és a technológia területén. Néhány példa a fázishasítás felhasználásáról:

1) Hologáfia:A holográfia egy 3D-s képalkotó eljárás, amely a fázishasítást
2) Interferométria: Az interferométria egy olyan technika, amely a fázishasítást használja a távolság, a vastagság és a törésmutató pontos mérésére.
3) Optikai kommunikáció: Az optikai kommunikációban a fázishasítást használják az információ kódolására a fényhullámok fázisában.
4) Agykutatás: A fázishasítást az agyi aktivitás mérésére is használják. Az EEG (elektroencefalográfia) és MEG (magnetoencefalográfia) elektródák segítségével méri az agy elektromos és mágneses aktivitását, és a fázishasítást használja a jelek elemzésére.

A fázishasításos teljesítmény szabályozó (phase-shifted full-bridge converter) eszköze az, hogy a bemeneti váltakozó feszültséget egy félperóduson belül csak egy részében használja ki. Ennek az az oka, hogy a kimeneti feszültség és áram jelalakját befolyásolja a kapcsolási időpontok eltolása.

Az effektív érték (RMS) számítása esetén kiindulhatunk a definícióból, miszerint az effektív érték a jelalak négyzetének átlaga, majd négyzetgyök alatt véve. Tehát, ha egy periódus alatt a feszültség $V(t)$, akkor az effektív érték a következőképpen számolható:

$$V_{eff} = \sqrt{\frac{1}{T} \int_0^T V(t)^2 dt} \tag{4}$$

Ahol T a periodusidő. Ha az áram jelalakja ugyanolyan, mint a feszültség jelalakja, akkor a teljesítmény számítása a következőképpen alakul egy 1 Ω ellenálláson:

$$P = \frac{V_{eff}^2}{R} \tag{5}$$

Most tekintsük a bekacsolási pillanat (a fáziseltolás) változtatásának hatását a kimeneti teljesítményre. Ha a bekacsolási pillanatot δ-val változtatjuk, akkor a feszültség jelalakja a következő lesz:

$$V(t) = V_m sin(\omega t - \delta) \tag{6}$$

ahol V_m a maximális feszültség, ω a szögsebesség. Az effektív feszültség kiszámítása ebben az esetben:

$$V_{eff} = \sqrt{\frac{1}{T} \int_0^T V(t)^2 dt} \tag{7}$$

A fentiek alapján ez egy összetett integrál, amelyet meg kell oldani az δ függvényében.

A LabView programban úgy oldottuk meg, hogy vettük az előző feladatban létrehozott programot és átalakítottuk. A fázishasítást úgy tudtuk megvalósítani, hogy fogtuk a szinusz függvényünket és egy négyzetes függvényt, amiket összeszoroztunk. A négyzetes függvénynek pár paramétert meg kellett adni. A frekvenciát felhasználtuk ugyan úgy, ami a szinusz függvényünkbe ment azonban az megszoroztuk -2-vel hogy a függvényünk először 0-ról kezdődjön és a szinusz jelet csak a periódus egy részében vegye ne teljesen az elejétől. Amivel megtudtuk adni, hogy honnan is kezdje a hasítást azt a duty cycle paraméterrel tudjuk megadni. A duty cycle egy százalékban értendő paraméter, ami megadja, hogy a félperiódus hányad részénél hasítsa el a függvényünket. Amplitúdónak 1-et adtunk meg hogy meglegyen az 1 vagy 0 érték a függvényünkön, hogy a szorzáskor vagy vegyük a szinusz

görbéjét vagy 0 konstans legyen a gráfunkon. Az effektív érték kiszámolásához felhasználtuk a következő képletet:

$$P = U * I \tag{8}$$

Amiben az áramerősségünk nem ismert ezért a következő képletből következik az egyenletünk:

$$R = \frac{U}{I} \tag{9}$$

Ezt az egyenletet átrendezve megkapjuk hogy:

$$I = \frac{U}{R} \tag{10}$$

Majd a teljesítmény képletében fel tudjuk használni az alábbi képletek átalakításait a következő képpen:

$$P_{eff} = U * \frac{U}{R} \tag{11}$$

Amiből láthatjuk hogy az effektív feszültségünk egymással szorzódik így:

$$P_eff = \frac{U_{eff}^2}{R} \tag{12}$$

Képlettel meg tudjuk adni az effektív teljesítményt.

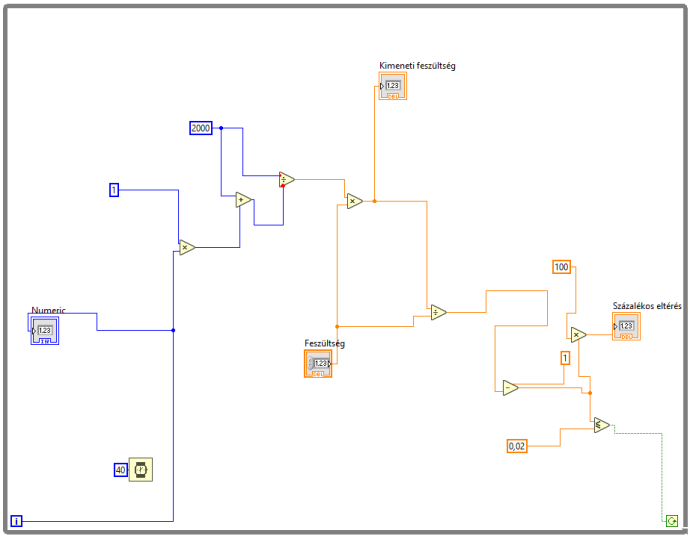
VII. 7. Feladat

A hetedik feladatban egy feszültségmérőt kell készítenünk Labview-ban melynek több adata már meg van adva. Az órán nem jutottunk el a hetedik feladat végéig így azt írjuk le ameddig eljutottunk. A feladat egy feszültségmérőnek a mérési bizonytalanságát megjeleníteni. A feszültségmérő vagy voltmérő egy olyan eszköz, amelyet elektromos feszültség mérésére használnak egy áramkörben vagy elektromos rendszerben. A működése az alapvető fizikai elveken alapul.

Az analóg feszültségmérők működése alapján a mérésbizonytalanságukat általában a mérőműszer skálájának pontossága határozza meg. Az analóg mérőműszer egy mutatóval vagy nyíllal rendelkezik, amely a mérőpanelen lévő skálán mozog, és a mutató helyzete mutatja a mért feszültség értékét. A skála felosztásai azonban csak meghatározott pontossággal rendelkeznek, ami határozza meg a mérési pontosságot vagy bizonytalanságot. Például, ha egy analóg feszültségmérőnek a skálája csak 1 V-os felbontással rendelkezik, akkor egy ilyen műszerrel mért feszültség értéke akár plusz vagy mínusz fél volt is lehet a mért értékhez képest. Ez a fél volt a mérés bizonytalansága.

A digitális feszültségmérők mérési bizonytalansága leginkább az áramkörben használt ADC (analog-digitális átalakító) felbontásától és a kijelző pontosságától függ. Az ADC felbontása meghatározza, hogy milyen finom felbontással képes az eszköz az analóg jel digitális formába alakítani. Minél magasabb a felbontás, annál kisebb a mérési bizonytalanság. Emellett a kijelző pontossága is fontos szerepet játszik a mérési eredmények megjelenítésében. A digitális feszültségmérők általában magasabb pontosságot és kisebb mérési bizonytalanságot kínálnak, mint az analóg mérőműszerek.

A multimeter, mint egy sokoldalú mérőeszköz, különböző mérési funkciókat kínál, beleértve a feszültségmérést, az ellenállásmérést, az árammérést stb. A mérési bizonytalanság ebben az eszközben minden mérési funkciótól függ, és az adott mérési beállítástól. Például, ha egy multimeterrel feszültséget mérünk, akkor a fent említett módon a feszültségmérési bizonytalanság az alkalmazott feszültségmérési módszertől és



1. ábra. Labview számolás

az eszköz beállításaitól függ. Az ellenállásmérésnél pedig más paraméterek okoznak mérési bizonytalanságot, és így tovább minden egyes mérési funkció esetében. Általánosságban elmondható, hogy minél magasabb a műszerek felbontása és pontossága, annál kisebb a mérési bizonytalanság. Mi úgy álltunk neki a feladatnak, hogy Labviewban készítettünk egy for ciklust mely addig fut, ameddig el nem éri a két százalékost feszültség eltérést így meg kaptuk azt az ellánálás értéket mellyel lehetett volna folytatni a feladatot, ha lett volna időnk.

A for ciklusban van egy növekvő érték mely folyamatosan növekszik egytől kezdve 1. Ebbe belekötöttünk egy numericet amivel látjuk, hogy mennyi éppen az ellenállás száma. Utána ezt a képletet használtuk, hogy kiszámoljuk a kimeneti feszültséget:

$$U_{ki} = U_{be} * \frac{R_{be}}{R_e} \tag{13}$$

Ebből ezzel a képlettel megkapjuk a százalékos eltérést:

$$Eltérés = 100 - ((\frac{U_{ki}}{U_{ki}}) * 100) \tag{14}$$

Nekünk a százalékos eltérés minden feszültségre 40 ohm jött ki. Ha ezenek van eltérése akkor maximum ennyi lehetne az adott ellenállásoknál a maximum ohm szám:

0,1%	39,96004
0,2%	39,92016
0,5%	39,801
1%	39,60396
2%	39,21569
5%	38,09524

Méretezett osztósor működése: Az előosztó működése során a bemenő feszültséget csökkentjük annak érdekében, hogy a műszer által könnyen kezelhető legyen a mért tartomány. Az előosztó során a különböző ellenállásokkal történő osztásokat alkalmazzuk annak érdekében, hogy az elvárt tartományban a kívánt felbontás és érzékenység érhető legyen el. Eredő bizonytalanság (hiba) számszerű értéke: Az eredő bizonytalanságot a műszer mérési bizonytalansága és az előosztó által bevezetett hiba határozza meg. Ezeket az értékeket összeadva megkapjuk az összesített bizonytalanságot. Nagyobb bizonytalanságú ellenállások alkalmazása esetén: Ha nagyobb

```cpp
#include <iostream>


double calculateUncertainty(double Hmuszer, double Hoszto) {
    return Hmuszer + Hoszto;
}

int main() {

    double Hmuszer = 0.02;
    double Hoszto = 0.0;

    double tolerances[] = {0.001, 0.002, 0.005, 0.01, 0.02, 0.05};
    int numTolerances = sizeof(tolerances) / sizeof(tolerances[0]);

    for (int i = 0; i < numTolerances; i++) {
        Hoszto = tolerances[i];
        double totalUncertainty = calculateUncertainty(Hmuszer, Hoszto);
        std::cout << "Resistor tolerance: " << tolerances[i] * 100 << "%" << std::endl;
        std::cout << "Total uncertainty: " << totalUncertainty * 100 << "%" << std::endl;
        std::cout << std::endl;
    }

    return 0;
}
```

2. ábra. c++ program

bizonytalanságú ellenállásokat alkalmazunk az előosztóban, akkor az előosztó által bevezetett hiba növekedhet. Ezáltal az összesített mérési bizonytalanság is növekedhet. Ennek a hatásnak a mértéke azonban függ az előosztóban alkalmazott ellenállások relatív pontosságától és az osztások számától. Ha az előosztóban nagyobb tűrésű ellenállásokat alkalmazunk, ez növelheti az összesített mérési bizonytalanságot. Ez a program egy olyan eszköz, amely segít megérteni, hogyan változik egy mérési eredmény bizonytalansága attól függően, hogy milyen pontosságú ellenállásokat használunk egy áramkörben.

Először is, van egy műszerünk, amelyet használunk a mérésekhez. Ez a műszer bizonyos mértékű pontatlansággal rendelkezik, mint például minden mérőeszköz. Ezt a pontatlanságot hívjuk ”műszerbizonytalanságnak”. Ebben a programban ezt a műszerbizonytalanságot Hmuszer néven jelöljük, és egy előre definiált értéket adunk meg neki.

A másik tényező, amit figyelembe kell vennünk, az az ellenállások pontatlansága. Az ellenállások gyártási folyamata nem tökéletes, és kis eltérések lehetnek az elvárt értékektől. Ezeket az eltéréseket hívjuk ”toleranciának”. A tolerancia azt mondja meg, hogy az adott ellenállásnak mennyivel térhet el az elvárt értéktől. Ezt a toleranciát jelöljük Hoszto néven a programban.

2A program fő része egy ciklus, amely végigiterál különböző ellenállás-toleranciákon. Minden egyes alkalommal a program kiszámolja és kiírja az összesített bizonytalanságot az aktuális tolerancia alapján. Tehát gyakorlatilag azt nézzük, hogy milyen hatással van az ellenállások változó pontossága a mérési eredményünk bizonytalanságára.

A program kimenete azt mutatja meg, hogy a mérési bizonytalanság mennyire változik a különböző ellenállás-toleranciák alapján, ami segíthet abban, hogy megbecsüljük, mennyire megbízhatóak az általunk használt alkatrészek az adott mérési feladatban.

I. táblázat. Ellenállás tolerancia és teljes bizonytalanság

Ellenállás tolerancia (%)	Teljes bizonytalanság (%)
0.1	2.1
0.2	2.2
0.5	2.5
1	3
2	4
5	7