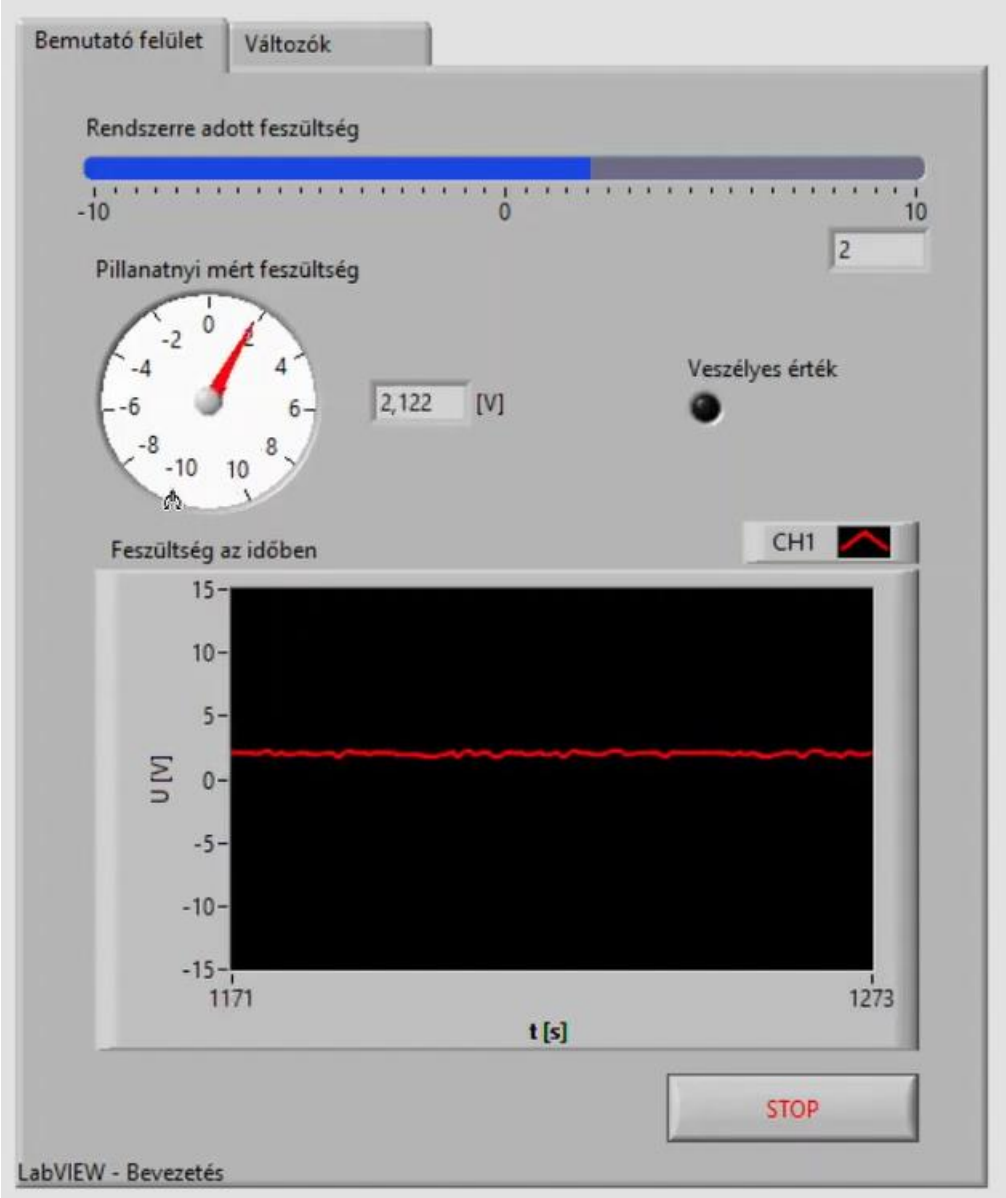
Mi a LabVIEW?

/Laboratory Virtual Instrument Engineering Workbench/
A National Instruments (NI) terméke, 1986-os megjelenését követően hamar a világ vezető mérésautomatizálási szoftverévé vált.
A LabVIEW egy grafikus programozói környezet. Napjainkban népszerűbbek és közismertebbek a szöveges programozási nyelvek, mint akár a C vagy a Python. A múlt század végén azonban a mérnökök körében nagyobb népszerűségnek örvendtek a grafikus nyelvek. Ezeknek talán legjelentősebb képviselője az itt bemutatott LabVIEW. A környezet virtuális mérőműszerek létrehozására hivatott, ezért is hívjuk az itt készített programokat vi-nak (Virtual Instrument). Ettől függetlenül természetesen bármilyen általános célú program is elkészíthető benne.



1. ábra – Egy egyszerű virtuális műszer

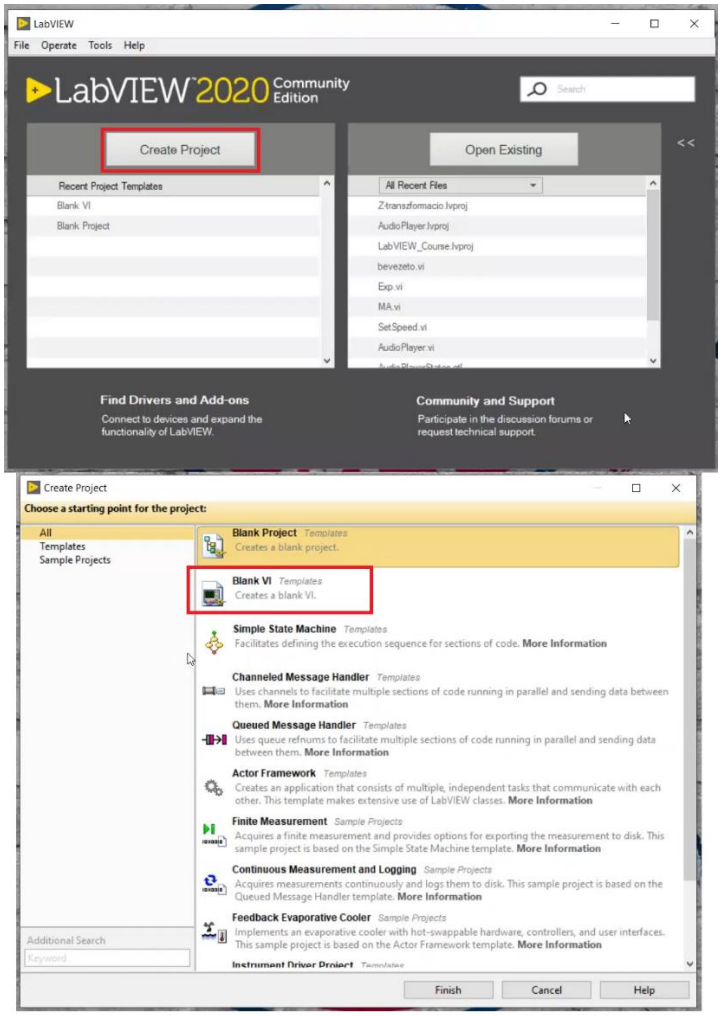
A grafikus programozásról néhány szóban

A grafikus programozás mögöttes logikája lényegileg megegyezik a szöveges formájúéval. Nagyon hasonló gondolkodásmódot igényel, de a megvalósítás módjában vannak eltérések. A nagy és szembetűnő különbséget az képezi, hogy az egyes változókat, műveleteket, függvényeket, stb. grafikus elemek, blokkok („dobozok") reprezentálják. Közöttük a kapcsolatot vezetékek biztosítják.

2. ábra – Példa LabVIEW-ban írt programra

A változókra, a vezetékezés mikéntjére és sok minden másra kitérünk még a későbbiekben

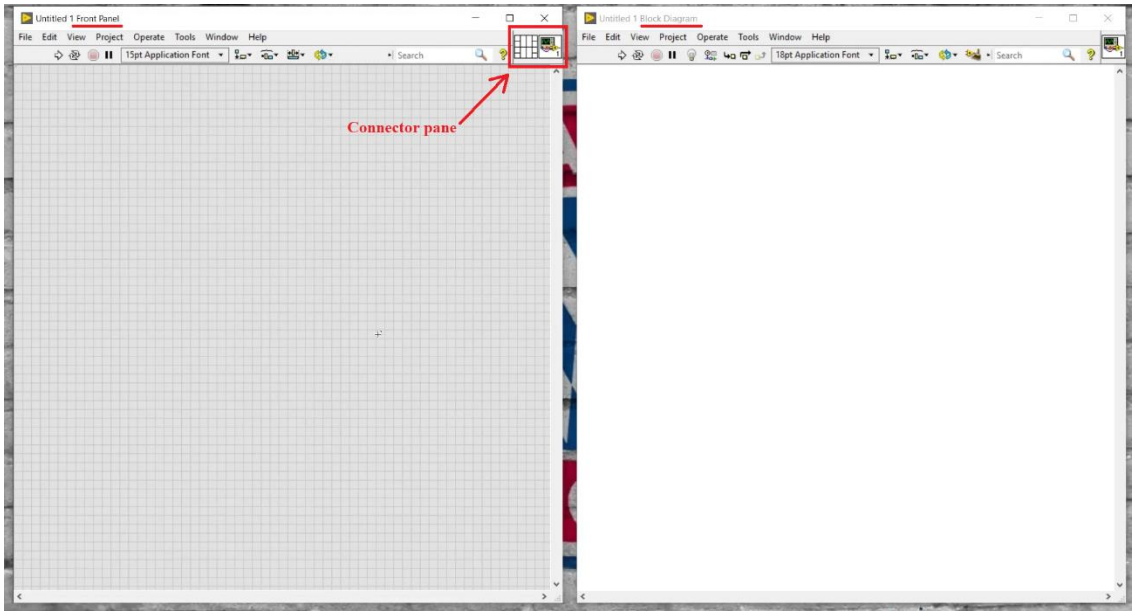
Új VI létrehozása



3. ábra – Create Blank VI

Egy VI részei

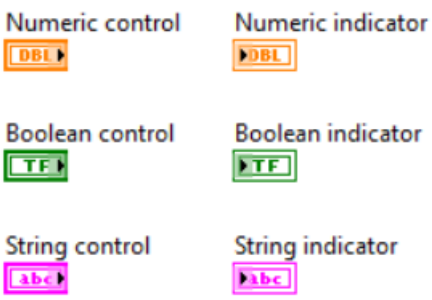
- Icon / Connector Pane – icon: a VI szimbóluma; connector pane: ki- és bemenetek megjelenítése
- Block Diagram – a grafikus kód helye, itt hozzuk létre a programot
- Front Panel – user interface, amin a controlok és az indicatorok vannak



4. ábra – A VI három fő része

A LabVIEW változóiról általánosságban

Más programnyelvek esetében egy változó rendszerint rendelkezik névvel, memóriacímmel, változótípussal és értékkel. A LabVIEW változói két további meghatározó tulajdonsággal is bírnak.
Ami az értéküket illeti, megkülönböztetünk Default és Current value-t. Előbbi az az érték, amelyre egy reset után visszaáll a változó, utóbbi pedig az, amelyet futás közben aktuálisan, az adott időpillanatban tárol. Egy változóhoz tehát mindig két érték tartozik, a kezdeti és a pillanatnyi.
Egy további jellemző, hogy az adott változó Control vagy Indicator. Azaz hogy a mérési folyamat kontrollálására vagy indikálására hivatott, tehát input vagy output adatokhoz rendelt.



5. ábra – Controlok és indicatorok

Numerikus típusok

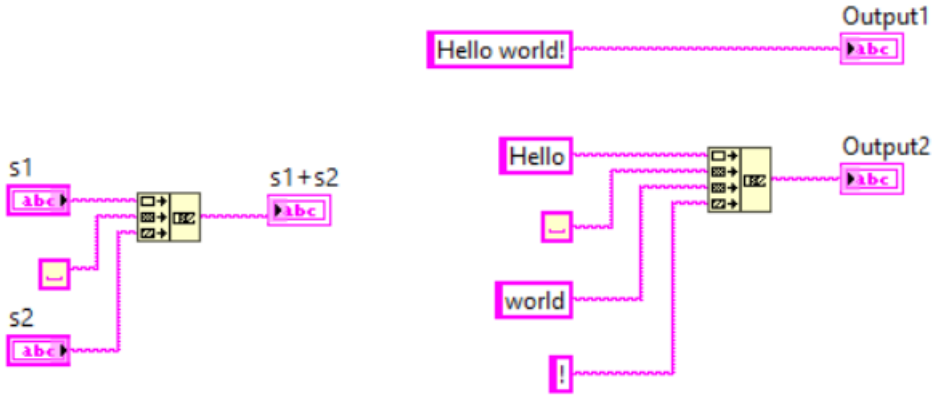
6. ábra – Numerikus változótípusok a LabVIEW-ban

Három fő típust különböztethetünk meg:
- valós számok (felső sor)
- egész számok (középső két sor)
- komplex számok (alsó sor)

Az egész számok esetében a jelölés jelentése:
- I = integer (előjeles), U = unsigned integer (előjel nélküli)
- A szám a biteket jelöli, így pl. U8 egy bites, előjel nélküli egész (0 – 255)

Stringek

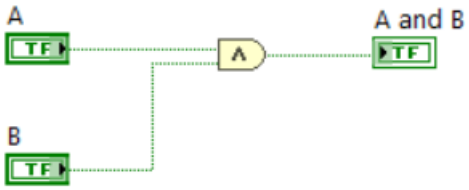
Más nyelvekhez hasonlóan a LabVIEW is ismeri a karakterláncokat.



6. ábra – Stringek kezelése, a LabVIEW „Hello world!”-je

Az ábrákon a control és indicator mellett megjelent a konstans is. A többi adattípusnál is létrehozhatunk ilyeneket (pl. numerikus: konstans 1; logikai: konstans True). Ezek olyan egyszerű fix értékek, amiket a program írásakor létrehozunk, futás közben a front panelről azonban nem tudunk hatni rájuk. Ha példának veszünk egy programot, ami a sugárból kiszámolja a kör területét, akkor egyértelmű, hogy csak „r”-hez kell controlnak tartoznia, hiszen a π értékét nem akarjuk változtatni, az állandó.

Logikai változók



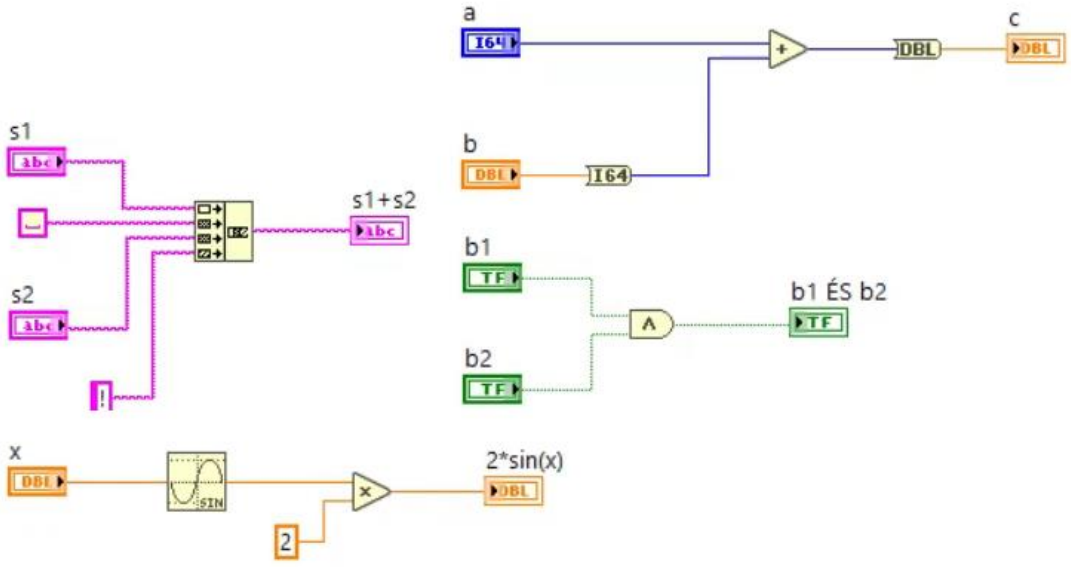
7. ábra – Bool típus a LabVIEW-ban

Létrehozhatunk olyan változókat, amik a logikai 1-et vagy 0-t, vagyis ahogy itt hívjuk, a True vagy a False értékek valamelyikét vehetik fel. Control esetben ezek valamilyen gombok vagy kapcsolók (meghatározó tulajdonságuk a mechanical action – azaz az, hogy mikor vált értéket a változó, pl. a gomb megnyomásakor vagy felengedésekor), indicatorokra jellemzően LED-ek. Hiszen, ne feledjük, virtuális műszerek kezelő felületéről beszélünk.
Ahogyan a numerikus típusokon elvégezhetünk egy összeadást, úgy a bool típusú változókon eszközölhetünk logikai műveleteket, például egy ÉS-t.

Színek

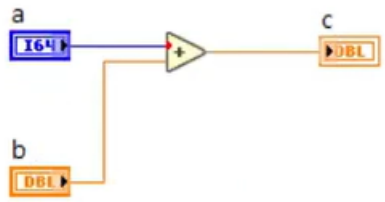
Láthattuk, hogy az egyes színek rendre valamilyen változótípust reprezentáltak:
- kék – egész számok
- narancssárga – valós számok
- pink – stringek
- zöld – logikai típus

A LabVIEW más színeket is megkülönböztet, de azokkal egyelőre nem találkozunk.



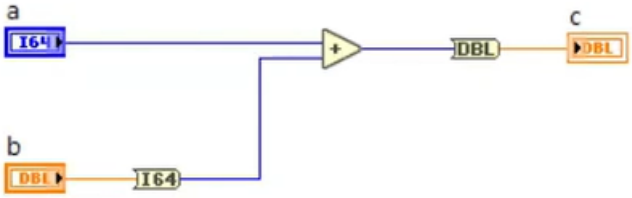
8. ábra – Színek megjelenése a G kódban

A Coercion dot



12. ábra – Coercion dot

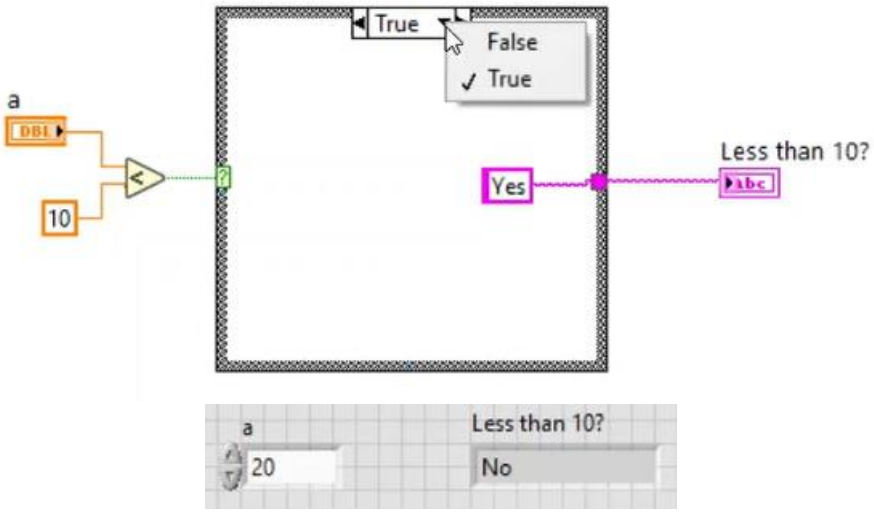
Az összeadó blokk felső bemeneti terminálján megjelent az úgynevezett Coercion dot, egy piros pötty, ami arra hívja fel a figyelmet, hogy automatikus típuskonverzió megy végbe. Ennek oka, hogy eltérő változótípusokkal végzünk műveletet. Ez potenciálisan az átalakítás adatvesztéssel járhat. Kerüljük ilyen Coercion dotok megjelenését, hozzunk létre helyette típuskonverziót megvalósító blokkokat.



13. ábra – Példa az automatikus konverzió elkerülésére

Struktúrák

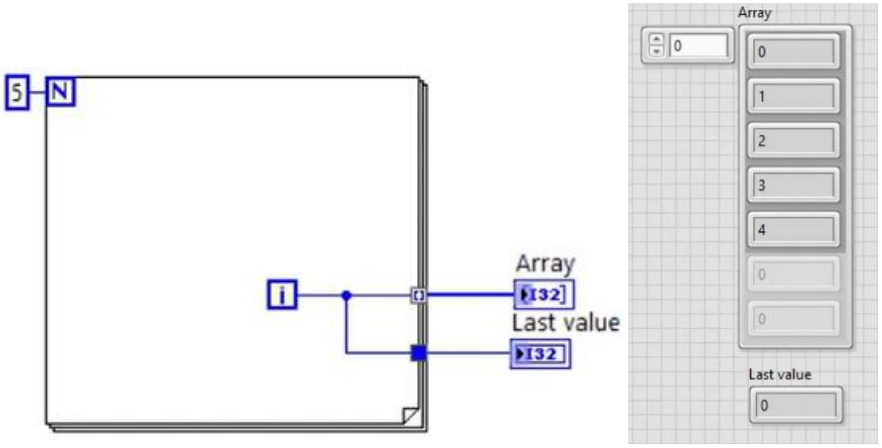
Ahogy például a Pythonban, itt is találkozhatunk feltételes utasítással, for ciklussal, valamint while ciklussal.
Az if neve itt Case structure lesz, de a logika hasonló. A struktúra bal oldalán található Case selector fogad valamilyen bemenetet (a legegyszerűbb esetben boolt), és annak értékétől függően hajtja végre valamely esethez tartozó utasításokat.
A struktúrákban található kódrészleteket összeköthetjük a külvilággal. Ezt a kapcsolatot az úgynevezett tunnel biztosítja, mely az ábrán egy – a típusnak megfelelően pink – kitöltött négyzet formájában jelenik meg. (Amíg nem kezeltük minden esetben a tunnel bemenetét, kitöltetlen marad.) Nemcsak kimenetet köthetünk ki a struktúrából, bemenő tunnel is létrehozható, ezen meg fog jelenni az oda kötött érték (ciklus esetén minden futás esetén elérhető módon).



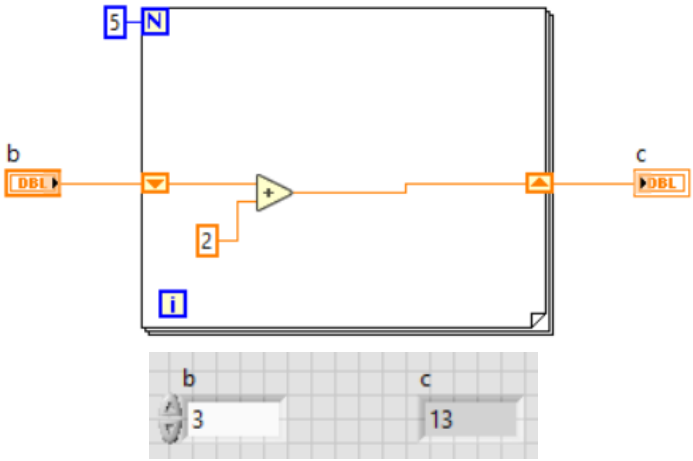
9. ábra – Példa Case structure-re

A for ciklus arra hivatott, hogy egy benne elhelyezett kódrészletet valahányszor lefuttassunk egymás után. Szükséges ehhez megadnunk egy N-nel jelölt egész számot, ami a futások számát jelenti. Található benne egy i-vel jelölt iterátor (más néven ciklusváltozó) is, ez a LabVIEW ciklusok esetében rendre 0-ról indul. Beláthatjuk így, hogy maximális értéke N-1 lesz.
A for cikluson belüli magot is összeköttetésbe hozhatjuk a struktúrán kívüli kódrészekkel egy tunnelen keresztül. Azt vehetjük észre, hogy ez alapértelmezetten nem kitöltött négyzet, benne szögletes zárójeleket láthatunk. Ez arra utal, hogy kimenetén nem egy egyszerű változót, hanem egy tömböt ad. Természetesen megváltoztathatjuk a tunnel módját olyanra, hogy csupán az utolsó futáskor kapott érték jelenjen meg a kimenetén. (Ennek módja: jobb klikk a tunnelre; tunnel mode; indexing helyett last value.)
A tunnelt helyettesíthetjük egy memória jellegű elemmel, egy úgynevezett Shift registerrel is (jobb klikk; replace with shift register). A ciklus mindkét függőleges szélére kerül egy-egy belőle. A tunnel átalakításakor ez azzal jár, hogy a létrehozás helyével ellentétes szélére is rá kell kattintanunk a bal egérgombbal. Próbáljuk ki, egyértelmű lesz! A Shift register egyébként arra szolgál, hogy valamilyen értéket átmentsünk a ciklus futásának egyes iterációi között. Nézzük meg, hogyan történik ez. Mind a bal-, mind a jobboldali rendelkezik be- és kimeneti terminállal is. A baloldali register bemenetére valamilyen kezdőértéket köthetünk. Az első

futásnál ez kerül a baloldali Shift register kimenetére. Ezután a működés nagyon egyszerű. Az érték, amit a ciklus k. futásakor a jobboldali Shift register bemenetére adunk, a k+1. lépésben meg fog jelenni a baloldali Shift register kimenetén.
Érdemes megjegyezni még a for ciklusról, hogy abban az esetben, ha N-re 0-t kötünk, a ciklusmag egyszer sem fut le.

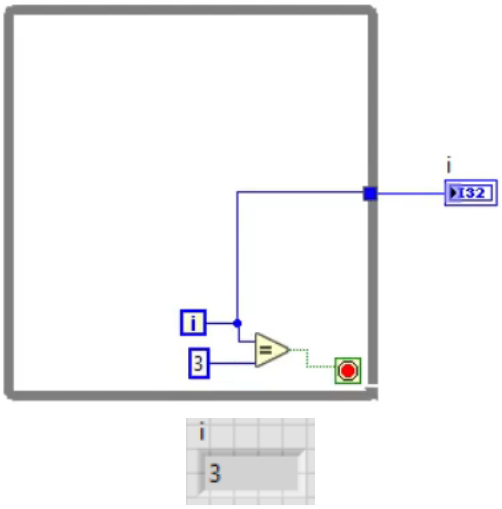


10. ábra – Példa for ciklusra és két különböző tunnel mode-ra



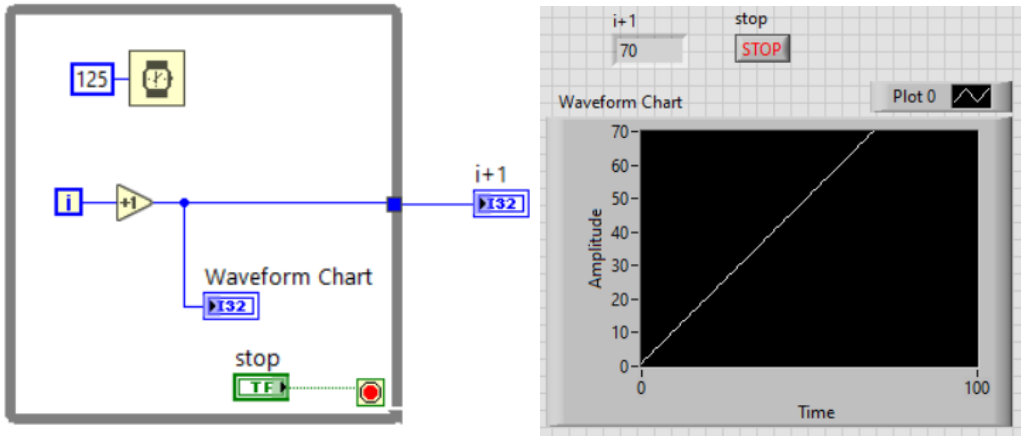
11. ábra – Példa Shift register alkalmazására for ciklusban

A másik ciklus a while. Itt nem a futások számát adjuk meg, hanem megfogalmazunk valamilyen logikai leállás feltételt. Ez lehet akár valamilyen logikai művelet eredménye, akár egy Stop gomb megnyomása. A ciklus addig fut, míg a feltétel igaz nem lesz. A while-ban is megtalálható az iterátor, valamint a leállási feltétel teljesülését figyelő Loop condition. Működésében – a futások számán, illetve a leálláson kívül – nagyon hasonló hasonlít a for ciklusra, van azonban egy szembetűnő különbség. Ha létrehozunk egy kimenő tunnelt, az alapértelmezetten nem tömböt ad vissza, hanem egy egyszerű értéket, méghozzá azt, amit az utolsó futáskor megkapott. Egy további különbség, hogy while ciklus magja (azaz a benne található kódrészlet) legalább egyszer egészen biztosan le fog futni.



12. ábra – Egyszerű while ciklus (az indicatoron látható, hogy i=3 esetén még lefutott a ciklusmag)

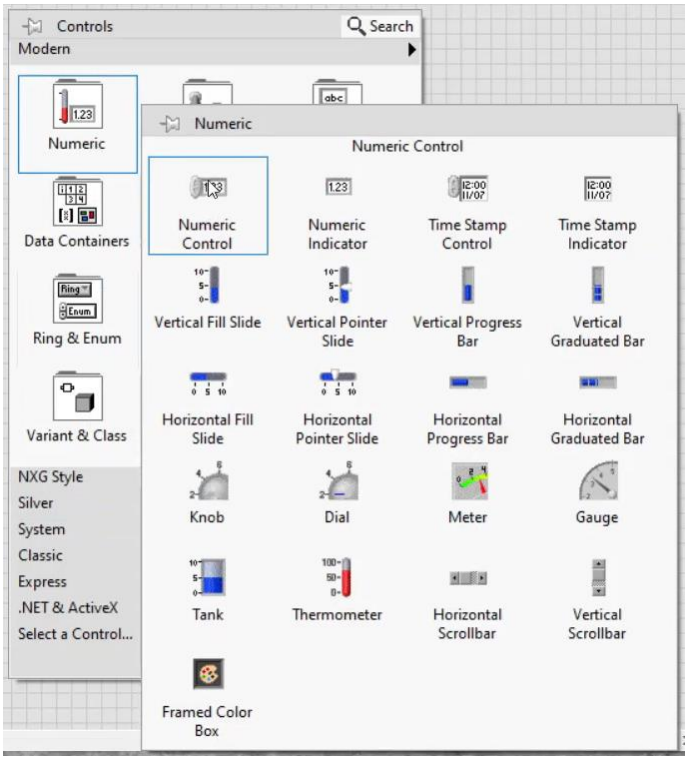
A ciklusban elhelyezhetünk egy időzítőt, ami az egyes iterációk között eltelt időt (ms) fogja szabályozni. Ha ez egy műszer lenne, ezt hívhatnánk mintavételezési időnek. Az ehhez tartozó blokkot a Timing alatt találjuk a Block Diagram palettáján.
A Front Panel palettáján a Graph gyűjteményben találhatunk különböző grafikus kijelzőket, a szokott módon tehetjük le őket, a vezetékezésük is megegyezik a már látottakkal – jelen esetben legalábbis. A többi grafikonnál vannak eltérések, ezt a következő órán látni fogjuk.



13. ábra – Grafikon és időzítővel ellátott while ciklus

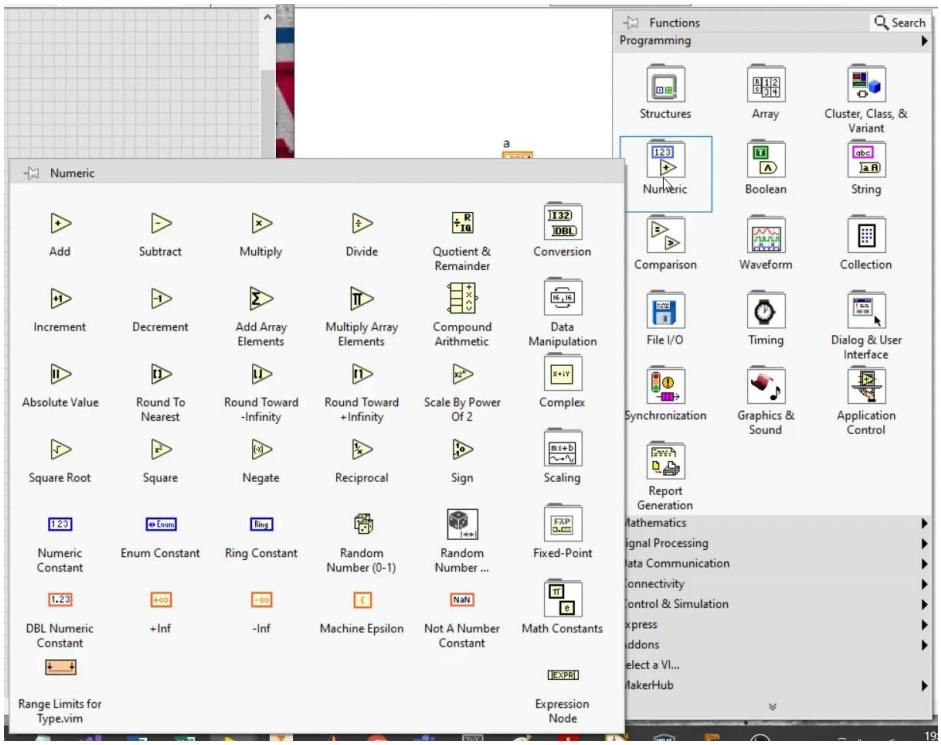
Egy VI elkészítése a gyakorlatban

Controlok, indicatorok, grafikus megjelenítő felületek létrehozásához klikkeljünk a jobb egérgombbal a Front Panelon. Ekkor az alább látható ábrán szemléltetett panel ugrik fel. Innen válasszuk ki a kívánt változót, egyszerűen klikkeljünk rá, majd húzzuk oda a Front Panelon, ahol el kívánjuk helyezni. Láthatjuk, hogy a Block Diagramon is megjelent egy a változónkhoz tartozó „doboz”.
Ha változtatni szeretnénk a változó nevén, jelöljük ki, írjuk át és klikkeljünk ki, ne pedig entert nyomjunk. Ha egy numerikus változó reprezentációját akarjuk módosítani, jobb egérgombbal klikkeljünk a változó blokkjára. Hasonlóan kell eljárnunk egy bool, illetve az ahhoz rendelt mechanical action megváltoztatásának esetében.

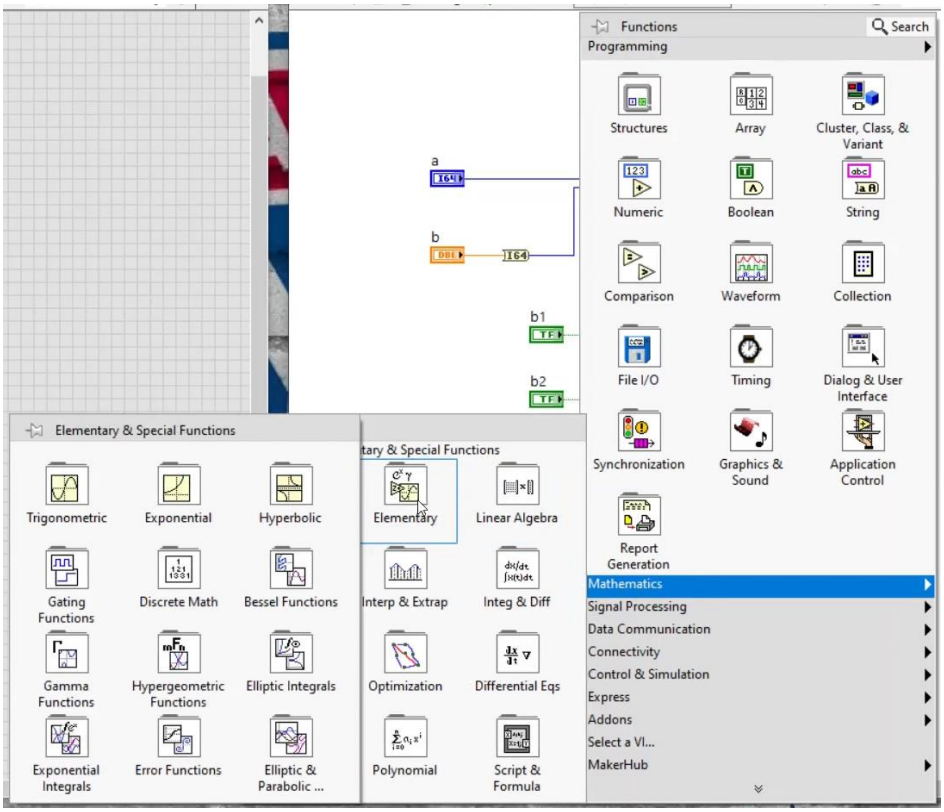


9. ábra – Egy Numeric Control kiválasztása a Front Panelon

Térjünk át a Block Diagram vizsgálatára. A Front Panelon elhelyezett változók rendelkezésre állnak, szükségünk van még az egyes utasításokat (műveletek, függvények stb.) reprezentáló blokkok elhelyezésére, valamint a vezetékezés létrehozására.
A Block Diagram „dobozait” ugyanúgy tehetjük le, mint a Front Panel változóit. Jobb klikk után megjelenik egy paletta, ahonnan kiválaszthatjuk a kívánt elemet. Húzzuk a kívánt helyre és engedjük el, ezzel létre is hoztuk.



10. ábra – A Block Diagram palettája



11. ábra – Trigonometrikus (és más összetettebb matematikai kifejezések) elérése

Ha minden szükséges blokk a rendelkezésünkre áll, elkezdhetjük a vezetékezést. Ha a controlokra és indicatorokra tekintünk, láthatjuk, hogy előbbiek jobb- és utóbbiak baloldalán van egy fekete nyíl. Ez mutatja, hol és milyen irányultsággal csatlakoztatható vezeték a változót reprezentáló szimbólumhoz. Az egyéb blokkokon is jól láthatóak a terminálok, azaz a be- és kimeneti csatlakozási pontok. Ha vezetéket akarunk létrehozni, nincs más dolgunk, mint hogy egy kívánt kimeneti terminálra klikkelünk, majd a megfelelő bemenetre. Láthatjuk, hogy az egér mozgatása közben egy szaggatott vonal jelenik meg, ez lesz majd a vezetékünk. Ha törést akarunk elhelyezni a vezetékben, kattintsunk ott, ahol létre akarjuk hozni, majd mozgassuk tovább az egeret a választott bemeneti terminálig, ahol újra klikkelve befejezhetjük a vezeték húzását.
Vezetékdarabot egy klikkel, egy ágat dupla, a teljes összefüggő vezetékrendszert tripla kattintással jelölhetjük ki. Egy vezetékbe beszúrhatunk valamilyen blokkot is egy jobb klikk után az insertet választva.

Egy struktúra létrehozása némiképp eltér más elemekétől. A megfelelő palettáról kiválasztjuk, letenni pedig úgy tudjuk, hogy kattintunk a Block Diagramon, majd elmozgatjuk az egeret (közben láthatjuk, hogy megjelenik a struktúra sziluettje), és ismét kattintunk.
A tunnelekre vonatkozó tudnivalókat tartalmazza a korábban tárgyalt „Struktúrák” pont.

Megjegyzések

Hasznos billentyűkombinációk:

- Ctrl-B - törött drótok törlése
- Ctrl-E - váltás a Block Diagram és a Front Panel között (ezt nem használtuk)
- Ctrl-H - Context Help ablak megnyitása
- Ctrl-T - a Block Diagram és a Front Panel megjelenítése egymás mellett a képernyőn
- Ctrl-Z - visszavonás
- Ctrl-Space - Quick Drop ablak megnyitása (ezt szintén nem használtuk az órán)

(+ Ctrl-t nyomva tartva vonszolhatunk kijelölt kódrészleteket, ezzel másolatot készítve róluk)