# LEGO robotok

XVI. rész

### III.3.2. A Bricx CC környezet

A Bricx Command Center (BricxCC) a LEGO MINDSTORMS, a CyberMaster és a Spybot robotrendszerekkel való könnyebb munkavégzésre készült. A Dave Baum által kigondolt, jelenleg John Hansen által fejlesztett NQC (Not Quite C Compiler) köré épül, amely lehetővé teszi az EV3, RCX, a Scout, a Cybermaster és a Spybot téglák programozását egy C-hez hasonló nyelven.

A BricxCC környezetet eredetileg Mark Overmars alkotta meg, ma pedig szintén John Hansen fejleszti.

Az RCX-re fejlesztett NQC nyelv hamarosan kibővült az NXT-re, így született meg az NXT (Not eXactly C).

A programozás elsajátítása előtt ismerkedjünk meg a Bricx CC környezettel!

Napjaink tendenciája, hogy a fordítóprogramokat *környezettel* lássuk el, mely integrálja a különböző elemeket. Legfontosabb kritérium, hogy a környezet egy szövegszerkesztővel rendelkezzen, amelyben meg tudjuk írni a forráskódot, közvetlenül lehessen hívni a fordítóprogramot vagy a szerkesztőt, a környezet tartalmazzon egy jól megírt kontextusfüggő súgórendszert is (*help*), amely a nyelvleírást és az egyes modulok, eljárások, függvények stb. bemutatását tartalmazza lehetőleg sok példaprogrammal.

Ezeket a környezeteket *IDE*-nek (*Integrated Development Environment*), *beágyazott fejlesztési környezetek*nek nevezzük.

Egy modern fordítóprogram környezete a következő elemeket tartalmazza:

- szövegszerkesztő,
- fordítórendszer,
- szerkesztőrendszer (linker),
- futtatórendszer,
- súgó,
- kódkiegészítők, sablonok,
- varázslók, kódgenerátorok,
- tervezőfelület (vizuális tervezés elősegítése: folyamatábrák, UML tervezési lehetőségek stb.),
- projekt kezelése, egyszerre több forráskód-állomány szerkesztése,
- debugger, nyomkövető (töréspontok definiálása, részletes futtatás, változók értékeinek nyomon követése, kifejezések kiértékelése stb.),
- szimbólumkövető,
- verem, regiszterek tartalmának kijelzése, gépi kód,
- adatbázis-tervező (relációk megadása),

- csoport- és nemzetközi programozás támogatása,
- automatikus dokumentációkészítő,
- tennivalók listája (ToDo),
- más környezeti eszközök, beágyazott lehetőségek (pl. ikon rajzolóprogramok stb.).

A 145. ábrán látható Bricx CC környezet a következő főmenü-pontokkal rendelkezik:

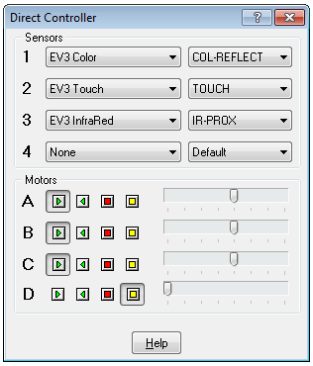
- *Fájl menü (File):* innen létrehozhatjuk, megnyithatjuk, elmenthetjük, bezárhatjuk a forráskódokat tartalmazó állományokat. A BricxCC lehetővé teszi a forráskódok nyomtatását. A menü alján található egy hasznos lista a nemrég megnyitott állományokról.
- *Edit menü (Edit):* Mint bármely szövegszerkesztőben, itt megtaláljuk a visszavonási (Undo) és visszaállítási (Redo) funkciókat, valamint a vágást (Cut), másolást (Copy), beillesztést (Paste) és törlést (Delete). Itt van a mindent kijelöl (Select All) funkció is. A Speciális beillesztés lehetővé teszi, hogy HRML vagy RTF formátumban szúrjunk be szöveget. A Következő mező (Next Field) funkció (F10) kiemeli a következő idézőjelek közé tett szöveget. A sablonokkal (F9) együtt használva, ez a funkció felgyorsítja a programírást. Például, ha behozunk egy `for ("init"; "condition"; "increment") { "body" }` sablont, akkor F10-et nyomva először az `"init"`-re ugrik a kurzor, majd a `"condition"`-ra, az `"increment"`-re és végül a `"body"`-ra. A menü utolsó pontja a Beállítások (Preferences…), ahol a környezetet, a fordítókat, sablonokat, makrókat szabhatjuk testre.
- *Keresés menü (Search):* innen megtalálhatunk és helyettesíthetünk egy szöveget a programban (mint bármely más szövegszerkesztőben). Egy pontos sorszámra léphetünk vagy megnyithatjuk az eljárások listáját. A kereséshez a *grep* linuxból jól ismert segédprogramot is felhasználhatjuk.
- *Nézet menü (View):* Ezzel a menüvel váltogathatjuk az összes panel és eszköztár láthatóságát. Hasznos ablak a Kód / Hiba lista (F12). Itt láthatjuk a fordított program kódját (ha sikeresen fordítják) vagy a hibás sorokat.
- *Fordítás menü (Compile):* Innen lehet lefordítani, futtatni, letölteni, elindítani, leállítani a programot.
- *Eszközök menü (Tools):* A környezet egyik leghasznosabb menüje, a következő fejezetben részletesen foglalkozunk vele.
- *Ablak menü (Window):* Itt beállíthatjuk a gyerekablakok elhelyezkedését, a pozíciójukat akár le is menthetjük, majd betölthetjük.
- *Súgó menü (Help):* Innen megnyithatjuk az online útmutatót, a régi NQC útmutatót, a Névjegy doboz (About), és a hivatalos BricxCC weboldalt is, ahol frissítéseket, mintákat és dokumentumokat érhetünk el.

### III.3.3. A Bricx CC eszközei és segédprogramjai

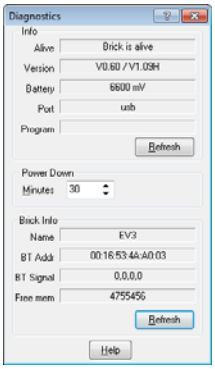
A John Hansen készítette eszközök nagyon hasznosak a téglával való kommunikálás, az információnyerés, adatfolyam szempontjából.

*Közvetlen vezérlés (Direct Control):* Innen kapcsolhatjuk be és ki a motorokat mindkét irányban, bármely sebességgel, beállíthatjuk az érzékelők típusait és módjait. Ezt a segédprogramot hibakeresési célokra fejlesztették.

*Diagnosztika (Diagnostics):* Ez a segédprogram kiírja az összes rendelkezésre álló információt a csatlakoztatott tégláról: a firmware verzióját, az akkumulátor feszültségét, ahogyan a tégla csatlakozik a számítógéphez (USB vagy Bluetooth), annak nevét és Bluetooth címét, a szabad memória mennyiségét stb.



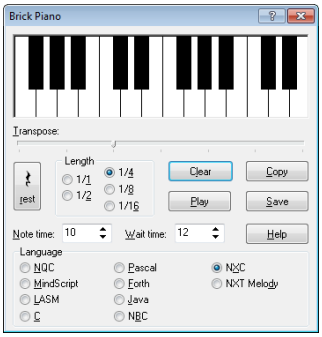

147. ábra: *A közvetlen vezérlés*




148. ábra: *A diagnosztika*

*A tégla követése (Watching the Brick):* Ez a segédprogram egy teljesen átfogó párbeszédablak, amelyben információkat kaphatunk a tégla érzékelőiről, a szervomotorok paramétereiről, az üzenetekről stb. A megfigyelt esemény grafikonjait is nyomon követhetjük. Sajnos, az EV3 tégla esetében ez a funkció az is ismertetett verzióban nem működik!

*Zongora (Piano):* Ezt az eszközt a zenészek használhatják, hogy dallamokat írjanak bármely programozható tégla számára. A legenerált kódot különböző programozási nyelvekbe exportáljuk.




149. ábra: *A zongora*

```
#include <config.h>
#include <dsound.h>
#include <tm.h>

static const note_t music[] = {
  { PITCH_E4, 40 },
  { PITCH_E4, 40 },
  { PITCH_F4, 40 },
  { PITCH_G4, 40 },
  { PITCH_G4, 40 },
  { PITCH_F4, 40 },
  { PITCH_E4, 40 },
  { PITCH_D4, 40 },
  { PITCH_C4, 40 },
  { PITCH_C4, 40 },
  { PITCH_D4, 40 },
  { PITCH_E4, 40 },
  { PITCH_D4, 40 },
  { PITCH_C4, 40 },
  { PITCH_C4, 40 },
  { PITCH_END, 0 }
};

int main(int argc,char *argv[]) {

  dsound_set_duration(10);
  dsound_set_internote(0);
  dsound_play(music);
  wait_event(dsound_finished, 0);
  dsound_set_duration(
  DSOUND_DEFAULT_16th_ms);
  dsound_set_internote(
  DSOUND_DEFAULT_internote_ms);
  return 0;
}
```

*A C/C++ kód*

```
task main() {
  PlayTone(330,400);
  Wait(480);
  PlayTone(330,400);
  Wait(480);
  PlayTone(349,400);
  Wait(480);
  PlayTone(392,400);
  Wait(480);
  PlayTone(392,400);
  Wait(480);
  PlayTone(349,400);
  Wait(480);
  PlayTone(330,400);
  Wait(480);
  PlayTone(294,400);
  Wait(480);
  PlayTone(262,400);
  Wait(480);
  PlayTone(262,400);
  Wait(480);
  PlayTone(294,400);
  Wait(480);
  PlayTone(330,400);
  Wait(480);
  PlayTone(294,400);
  Wait(480);
  PlayTone(262,400);
  Wait(480);
  PlayTone(262,400);
  Wait(480);
}
```

*Az NXC kód*

25. táblázat: *A zongora generált kódjai*

*Joystick:* Ezzel az eszközzel szabályozhatjuk a különböző hajtásrobotokat (például Tribot, JohnNXT vagy Turtle). A kormányozás vagy tank üzemmódban is vezethetjük a robotot, az egyik motorral vezetve a kerekeket, a másikkal pedig kormányozhatunk.

*Távirányító (Remote):* Mind az RCX, mind a Scout a Mindstorms távirányítóval vezérelhető. Ebben az ablakban emulálhatjuk a távoli parancsot azzal, hogy a téglával egyenértékű parancsokat küldünk.

*Konfigurálható követés (Configurable Watch):* Ez az eszköz hasonló a *tégla követése* ablakhoz, kivéve, hogy kézzel kell hozzáadnunk a kiválasztott források monitorjait.

*Értékek beállítása (Set Values):* Ezzel a párbeszédablakkal bármely írható forrás/érték kombinációt bármilyen olvasható forrás/érték kombináció értékére állíthatunk be. A források és az értéktartományok listája attól függ, hogy melyik téglát választottuk ki.

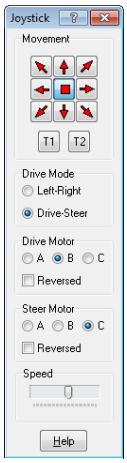
*Spybot EEPROM:* Ezt az eszközt az EPROM értékek követésére használhatjuk.

*Intéző (Explorer):* Ez az eszköz a tégla flash memória állományböngészője. Segítségével állományokat másolhatunk, törölhetünk, indíthatunk. A tégla teljes linux-os állományrendszere látszik.

*Képernyőmentő (Screen Capture):* Ez a segédprogram lehetővé teszi a téglaképernyő tartalmának megjelenítését és lementését képként a számítógépen. Hasznos egy nem elérhető tégla képernyőjének megtekintéséhez vagy a tégla távoli vezérléséhez is. Segítségével a tégla nevét is beállíthatjuk. JPEG, PNG, BMP és GIF formátumban tudjuk lementeni a képernyőt, de AVI-ban mozgóképekben is tárolhatjuk ezt. Akár 4-szeres nagyításban is lementhetjük a képeket. Ez az eszköz akkor is nagyon hasznos, ha ki akarjuk vetíteni a tégla képernyőjét.

*Követendők listája (Watch List):* a debugolás során követendő elemeket és ezek értékeit tartalmazza.

*Üzenetküldés (Send Messages):* A téglák képesek reagálni az üzenetekre, egymásnak üzeneteket küldhetnek, vagy a számítógéppel is kommunikálhatnak így. Az üzenetküldés azonban eléggé lassú művelet, körülbelül fél másodpercig tart, míg a robot reagál.




150. ábra
*A joystick*

*Datalog*: Az RCX az adatokat egy belső adatnaplóba is írhatja, amelyet fel lehet tölteni a számítógépre.

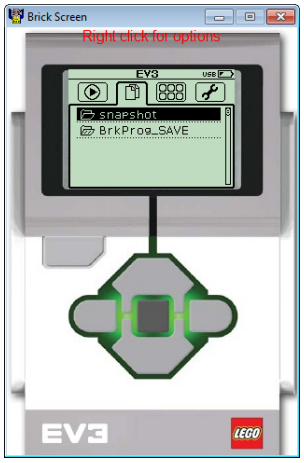
*Memóriatérkép (Memory Map):* Itt a tégla memóriájával kapcsolatos információkat kaphatjuk meg. Akkor hasznos, ha hibát keresünk, vagy a téglában memóriaproblémák merültek fel. Az eszköz különösen akkor hasznos, ha gyanítjuk, hogy nincs elegendő memória a programjainkhoz.

*Memóriatisztítás (Clear Memory):* Használjuk ezt a parancsot, ha ki akarjuk törölni a tégla memóriáját! Ez a parancs minden programban eltávolítja az összes feladatot és alprogramot, és kitörli az RCX adatnaplót is. A memória törlése fontos a nagy programok betöltése esetén, mivel a tégla megtartja az összes programot és feladatot akkor is, ha kikapcsoljuk.

*MIDI átalakítás (MIDI Conversion) és Hang átalakítás (Sound Conversion):* Ezek az eszközök egy MIDI vagy WAV hangállomány konvertálására szolgálnak egy kódba vagy egy RSO állományba. Mielőtt átalakítanánk egy WAV állományt, konvertálnunk kell mono (csak egy csatorna), 8 bites 8 kHz-es formátumra. Az RSO állományt tömöríthetjük, hogy helyet takarítson meg a téglán.

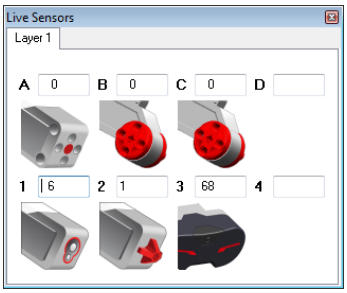
*Egyszerű terminál (Simple Terminal):* Ez az eszköz egy egyszerű kommunikációs terminálablak.

*Élő érzékelők (Live Sensors):* Az összes ki és bemeneti – A, B, C, D és 1, 2, 3, 4 – portot tudjuk követni a segédprogram segítségével. Megjelennek a csatlakoztatott eszközök, le tudjuk olvasni ezek értékeit.

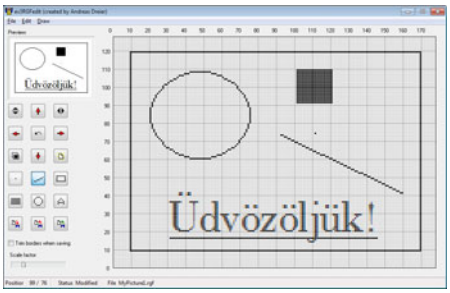



151. ábra: *A képernyőmentő*

A *Képszerkesztő* (Image Editor) segítségével képeket, grafikákat tervezhetünk az EV3-tégla kijelzője számára. A képet .RGF (Robot Graphics File) formátumba menthetjük le. Lehetőség van a kép eltolására, tükrözésére, különböző betűtípusú szövegek írására stb.




152. ábra: *Élő érzékelők*




153. ábra: *Képszerkesztő*

*Téglakereső (Find Brick), Tégla lezáró (Turn Brick Off) és Kapcsolatlezáró (Close Communication):* Ezen segédprogramok funkcionalitása magától érthető. A téglakereső működését már bemutattuk, a tégla lezáró lekapcsolja, lezárja a beindított téglát, a kapcsolatlezáró pedig bontja az aktuális élő kapcsolatot.

*Firmware letöltő (Download Firmware):* amint már láttuk, ez az eszköz alkalmas arra, hogy frissítsük a tégla firmearé-t.

*Firmware kinyitása (Unlock Firmware):* A firmware biztonsági beállításait adhatjuk itt meg.

*Eszközök beállítása (Configure tools):* Itt állíthatjuk be a BricxCC-t makrók és külső programok futtatásához.

### III.3.4. Az EV3-as tégla programozása Bricx CC környezetben

III.3.4.1. A „Helló, világ!” program

A „Helló, világ!” programok olyan számítógépes programok, melyek egyszerűen kiírják a megjelenítő eszközre: „Helló, világ!” (angolul: „Hello world!”). Mivel ez a program többnyire a legegyszerűbbek közé tartozik, gyakran használjuk arra, hogy kezdő programozókat megismertessük a nyelv alapvető szintaxisával, illetve arra, hogy teszteljük a fejlesztői környezet helyes telepítését.

Itt is ezzel fogunk kezdeni a környezettel való ismerkedés után.

A File (Állományok) menüből válasszuk a New (Új) menüpontot.

Ekkor egy *Untitled1 (Névtelen1)* fülecske jelenik meg a szövegszerkesztőben, ide már beírhatjuk a programot.

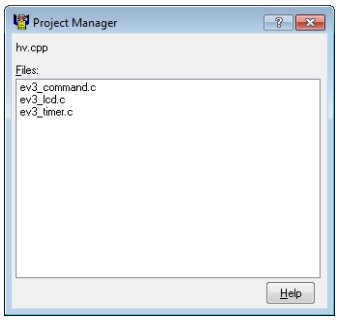
Mentsük le az üres programot, hogy az állománytípusnak megfelelően beinduljon a környezet szintaxis kiemelője (Syntax highlighting).

A File / Save menüpont segítségével adjunk egy nevet a forráskódot tartalmazó állományunknak, legyen ez pl. *hv.c*, válasszuk ki a C++ Files (*.c, *.cpp, *.hpp) opciót, és mentsük el a forrásszövegünket.

Írjuk be a következő forráskódot:

```
1.  #include <stdio.h>
2.  #include <unistd.h>
3.  #include "C:\Apps\Bricx\API\ev3_lcd.h"
4.  #include "C:\Apps\Bricx\API\ev3_command.h"
5.
6.  int main()
7.  {
8.    LcdInit();
9.    LcdText(1, 0, 0, "Hello, vilag!");
10.   Wait(SEC_1);
11.   LcdExit();
12.   return 0;
13. }
```

A View / Project Manager menüpont segítségével hívjuk elő a projektmenedzser ablakot. Kattintsunk a jobb egérgombbal, majd válasszuk az Add... (Hozzáadás) lehetőséget. A megjelenő párbeszédablakban navigáljunk a Bricx CC telepítési mappájában az API mappára, majd innen válasszuk ki az *ev3_command.c*, *ev3_lcd.c*, *ev3_timer.c* állományokat, majd zárjuk be a projekt menedzsert. A lementett forrásszöveget tartalmazó mappában így létrejön egy *hv.prj* nevű szöveges állomány.



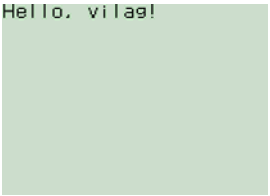
154. ábra: *A projektmenedzser*

Sajnos az így létrejött állományba be kell írnunk az elérési útvonalakat is, így a File / Open menüpont segítségével nyissuk meg a *hv.prj* nevű állományt, így ez az IDE egy új fülecskéjében fog megjelenni.

Minden sor elé írjuk be az elérési útvonalat is – ahova az API mappát létrehoztuk (*C:\Apps\Bricx\API*), majd mentsük le az állományt:

```
1. C:\Apps\Bricx\API\ev3_timer.c
2. C:\Apps\Bricx\API\ev3_command.c
3. C:\Apps\Bricx\API\ev3_lcd.c
```

Így megvan a fordításhoz szükséges két állomány, a forráskód és a projekt. Vigyázzunk, hogy a fordítás előtt mindig mentsük le a forráskódot, különben a régit fordítja le!



155. ábra: *Helló, világ!*

A Compile / Compile (F5) menüpont segítségével fordítsuk le a programunkat. Ez megtörténik hiba nélkül. Ha hibaüzenet jelenne meg, akkor a View / Show Code/Error|Warning Listing (F12) segítségével megnézhetjük a pontos hibaüzenetet és azt a sort, amelyik a hibát okozta.

A fordítás után a Compile / Download and Run (Ctrl+F5) menüpont segítségével tölthetjük le a lefordított programot a téglára, és futtathatjuk ott. A Compile / Download (F6) segítségével csak letölthetjük, a Compile / Run (F7) segítségével pedig futtathatjuk a programot, amelynek az eredménye a 155. ábrán látszik.

**Kovács Lehel István**