

ROBOTIKA AZ OKTATÁSBAN – ROBOT PROGRAMOZÁS
(FELADATGYŰJTEMÉNY)
ROF 511

Abstrakt

Hoci využívanie robotov a robotiky vo vzdelávaní zatiaľ nemá svoje tradície, nemožno ignorovať fakt, že robotika sa neustále a rapidne rozvíja a v budúcnosti (možno už čoskoro) budú roboty súčasťou nášho každodenného života. Ich zaradenie do výučby je teda opodstatnené. Dôkazom dôležitosti robotov je aj fakt, že na najväčšom svetovom veľtrhu vzdelávacej techniky BETT v Londýne v roku 2010, hlavnú úlohu budú hrať programovateľné roboty. Naším cieľom je, aby sme pre začiatočníkov uľahčili vedu programovania, spríjemnili a urobili ju zábavnejšou. V tomto nám môže byť na veľkú pomoc robotika, pretože „oživené objekty“ sú nielen veľmi zábavné, ale majú aj zodpovedajúcu motivačnú silu. Ohľadom na to, sme vytvorili zbierku úloh, ktorá zobrazuje na základnej úrovni, čo robotika vlastne je. Obsahuje súčasne aj hotové úlohy, ktoré môže čitateľ individuálne rekonštruovať pomocou našich fotografií, videí a poznámok.

Kľúčové slová: robot, robotika, vzdelávanie, informatika, programovanie

Absztrakt

A robotika rohamosan fejlődik, egyelőre még csak találgatások vannak, hogy mit hozhat a jövő. Az oktatásban való megjelenése erősen indokolt. A 2010-es londoni BETT - a világ legnagyobb oktatástechnikai vásárán is a főszerepet többek között a programozható robotok tették ki. Célunk, hogy a programozás tudományát megkönnyítsük a kezdők számára, mint ahogy azt élvezetesebbé is tegyük. Ebben rendkívül nagy segítségünkre lehet a robotika, hisz az „életre kelő” objektumok nemcsak nagyon szórakoztatóak, de megfelelő motivációs erővel is bírnak. Ezt figyelembe véve megalkottunk egy feladatgyűjteményt, ami alapszinten vázolja a robot programozás területének lényegi mivoltát. Ugyanakkor kész feladatokat is tartalmaz, melyeket az olvasó individuálisan rekonstruálhat fényképeink, videóink és megjegyzéseink segítségével.

Kulcsszavak: robot, robotika, oktatás, informatika, programozás

Tartalomjegyzék

Bevezetés	8
1. Első lépések	10
1.1 Jármű robot összerakása.....	10
1.2 Szenzorok.....	16
1.3 A szoftver bemutatása	18
2. Programozáson belüli alapfogalmak szemléltetése robotok segítségével.....	21
2. 1 Mi az algoritmus?.....	21
2. 2 A „ciklus” fogalma.....	21
2. 3 A „ha” feltétel bemutatása.....	23
2. 4 A „for” ciklus és „ha” feltétel egymásba ágyazása	24
3 Első program	27
3.1 Vonalkövetés – fény- és színszenzor használata.....	29
3.2 Vonalkövetés kibővítése	33
3.3 Járőr Robot	35
3.4 A Járőr Robot program kibővítése	38
Befejezés	42
Felhasznált irodalom:	43

Bevezetés

A robotika az utóbbi időben nagyon elterjedt, ígéretes tudományág. Fejlődésére szükség is van; eme tényt több szempont is indokolja.

Mint tudjuk az informatika szakterületén belül az egyik legkérdésesebb témakör annak metodikája. Amint kidolgoznak egyet, mely megfelelőnek tűnik; mire nyomtatásba kerül az anyag, szinte aktualitását veszíti. Ebből a szempontból a robotika maradandónak bizonyulhat. Elsősorban a programozás szakterületén nyújthat komoly segítséget. Jelenleg ugyanis sem az általános iskolák, sem pedig a középiskolák és gimnáziumok nem rajonganak érte. A programozás ugyanis egy olyan „megváltozott” gondolkodásmódot igényel, melyet alapvetően elég nehéz elsajátítani. Nem beszélve arról, hogy bizonyos matematikai, logikai ismereteket is igényel, melyek komplexitása elég kevés tanulónak imponál, többek között a motiváció hiányában. A robotika itt lép a képbe. A megszokott és gyakran unalmas Pascal parancssorok helyett valami egészen újat kínál. Megadja a lehetőséget, hogy megírt parancssorainkat, s azok eredményét azon nyomban viszontlássuk a fizikai világ szintjén. Hogy adott esetben a programozandó objektumaink megépítése is élmény lehessen. Hogy képzeletünk szabadon szárnyalhasson, s kiélhessük kreativitásunkat is stb.

Ezek mind nagyon nagy motivációs erővel bírnak, mint ahogy azt nem rég tapasztaltuk is barátaimmal, kollégáimmal. Nem beszélve arról, hogy mennyivel nagyobb élményt jelenthetne ez fiataloknak, akik most találkoztak programozással, illetve most szeretnék elkezdni annak tanulmányozását. Ehhez kiváló segítséget nyújt a *LEGO Mindstorms NXT* szériája, mely egy profi szintű LEGO készlet kezdőknek és haladóknak egyaránt. A csomag 612 db. építőeszközt tartalmaz, 4 db. szenzort (2 db. érintő, fény és színérzékelő), 3 interaktív motort, egy 32 bites programozható mikroprocesszort, s megannyi segédeszközt. A robotok programozása NXT-G programozási nyelvben történik, mely grafikus építőkockákból áll, ezáltal jócskán felhasználóbarát, illetve a lépésről-lépésre leírt példaprogramok is segítséget nyújthatnak. És ez még nem minden. A teljesen kezdőtől a profi programozóig mindenki megtalálja a számára kihívást jelentő feladatot. Tehát mindig van továbblépési lehetőség, ami abban merül ki, hogy nagyon sok programozási nyelvvel kompatibilis a mi kis egységünk, mint amilyenek pl. az NBC, NXC, RobotC, LeJoS stb.

Láthatjuk hát, hogy a robotikában van fantázia, még hozzá nem is kevés. Nincs nagyobb élmény egy kezdő programozó számára annál, mikor egy *for ciklus* nem pusztán betűkből és teóriában áll, de hirtelen „átváltozik” egy forgó-pörgő, magából halálos muníciót

kilövellő rettegett harci járművé, amit ráadásul saját maga alkotott. Nagyon fontosnak tartjuk, hogy ezt az élményt minél többen átélhessék, ezáltal többen megkedveljék a programozást. Olyanok is, akik esetleg a későbbiekben nem fognak foglalkozni vele ugyanúgy, mint ahogy az a célcsoport is, akikből (talán pont a robotikának hála) kitűnő programozók válnak majd.

Célunk hát bemutatni egy olyan feladatgyűjteményt, ami bepillantást enged pár konkrét feladat kivitelezésébe, hogy elősegítsük a robotika köztudatba való elhintését, s annak népszerűsítését.

1. Első lépések

1.1 Jármű robot összerakása

Ebben a fejezetben megismerkedünk a LEGO Mindstorms NXT 2.0 alapkészlet azon elemeivel, melyek szükségesek egy jármű összerakásához, továbbá 7 lépéses leírás (tutorial) végrehajtásával megtanuljuk hogyan is kell megépíteni egy ilyen robotot. Akkor kezdjük a hardverrel, annak is a főbb elemeivel:

1. Vezérlőegység – 'Tégla'
2. 3 db szervo motor
3. A motorokat és a vezérlőegységet összekötő kábelek (3x db)
4. Kerekek (4x db), láncalpak (vagy gumik), továbbá a készlet lego elemei, melyekről a további képeken teszünk említést



1. ábra

1. lépés:

A három szervo motor egyikét olyan lego elemekkel látjuk el, melyek lehetővé teszik, hogy a másik két motort is hozzákapcsoljuk a járművünk konstrukciójához. Jogosan merülhet fel bennünk a kérdés, miszerint „Miért használjuk egy járműhöz mind a három szervo motort?”. A válasz nagyon egyszerű: a jármű alapkonsztukciójához azért kapcsoljuk hozzá a harmadik motort, mivel a későbbiekben, bizonyos

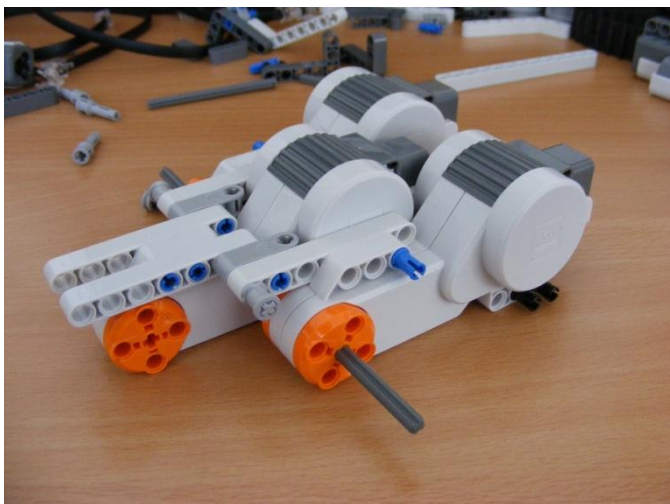


2. ábra

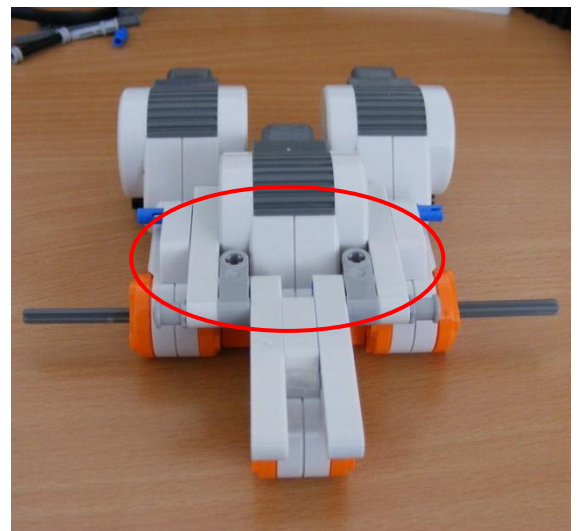
feladatoknál szükségünk lesz rá. Szintén felmerülhet a kérdés, hogy: „És mi van a kanyarodással?” .Természetesen itt meg kell említenünk, hogy a szervo motorokat külön-külön tudjuk irányítani, ezért a kanyarodáshoz nincs szükség feltétlenül egy harmadik motorra, viszont tény az is, hogy ha igazán szép, nem szaggatott, folyamatos kanyarodást szeretnénk azt legkönnyebben egy harmadik, a kanyarodásért felelős motor beiktatásával tudjuk megoldani. Miután összeraktuk a képen látható alkatrészt, folytathatjuk a második lépéssel. Az elemek mérete, alakja és színe alapján egyértelműsíthetjük, hogy hova melyik elemet kell illeszteni.

Tipp: Ha a készlethez kapott szoftverben található leírások alapján szeretnénk összerakni a robotunkat, általában problémát okoz az elemek hosszúságának ellenőrzése. Erre kétféle megoldás is létezik: 1. a leírásban az azonos alakú elemek hossza általában cm-ben van megadva. Elővehetünk egy vonalzót (vagy használhatjuk a készlethez kapott felhasználói kézikönyv egyik oldalára nyomtatott vonalzót) majd ennek segítségével meggyőződhetünk arról, hogy valóban a megfelelő elemet választottuk-e. 2. A „lyukas” elemeknél az x cm-hosszú elemet vehetjük x lyukú elemnek is, mivel a kettő megegyezik, tehát nincs más dolgunk, mint hogy megszámoljuk, hogy hány lyukú elemre is van szükségünk. Szóval ha a leírásban az áll, hogy egy 7 cm-es elemre van szükségünk, akkor egy 7 lyukú elemet keressünk.

2. lépés:



3. ábra



4. ábra

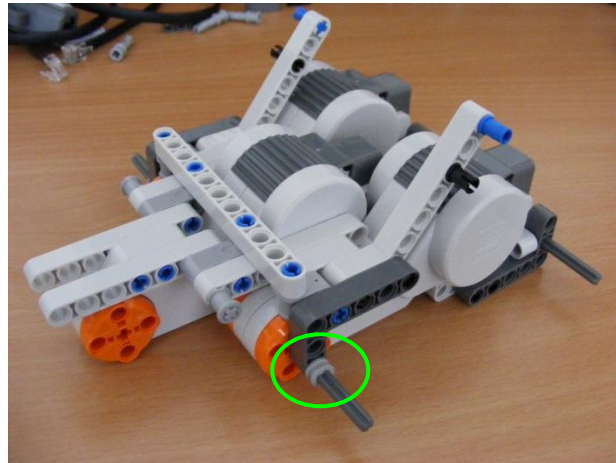
Az összerakást a 3-as és 4-es ábra alapján folytatjuk. Ügyeljünk, hogy a 4. ábrán a már előzőleg összerakott szervo motor és a két oldalsó motort összekapcsoló résznél (pirossal

jelölve) ne felejtjük ki a két szürke elemet, ugyanis a későbbiekben ezek (is) fogják tartani a vezérlőegységünket. Továbbá figyeljünk, hogy a két szervo motor alatti fekete elemeket a középső és a jobb szélső helyre tegyük (ha a motor meghajtó eleme balra mutat).

3. lépés:



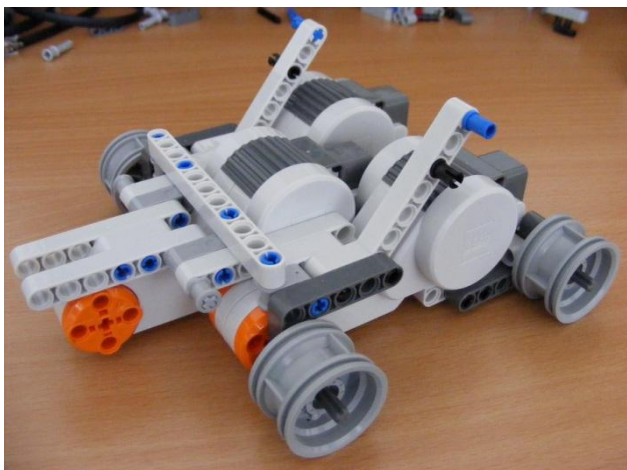
5. ábra



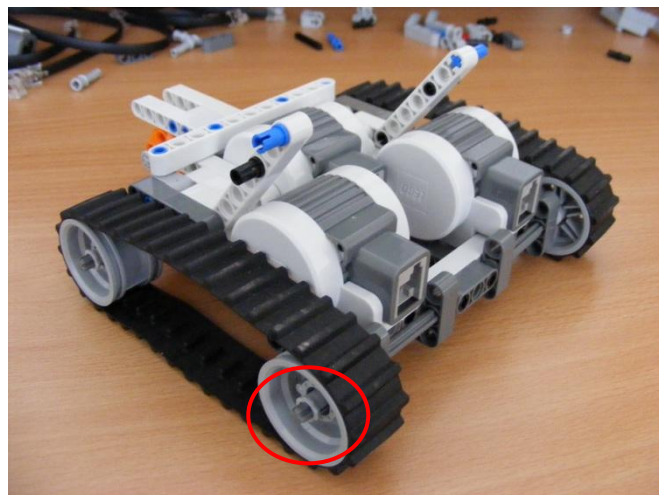
6. ábra

Ebben a lépésben biztosítjuk a kereket, valamint a vezérlőegység tartóelemeit. Ügyelünk arra, hogy az 5. ábrán pirossal jelölt helyre ne felejtjük el felhelyezni a 6. ábrán zölddel jelölt alkotóelemet, ugyanis ez akadályozza majd meg, hogy a hátsó két kerék gördülékenyen pörögjön, ne súrlódjon a közvetlenül mellettük lévő elemhez.

4. lépés:



7. ábra



8. ábra

Ebben a lépésben felkerül a négy kerék, valamint a lánc talpak, vagy a gumik. Ennél a lépésnél is ügyelnünk kell egy nagyon fontos dologra. Ez pedig a kerekek, a tengelyről való lecsúszását megelőző szorító. A 8. ábrán pirossal bekarikázott részben jól kivehető az elem

alakja. A kerekek stabilabbá tételéhez ezeket az elemeket úgy helyezzük fel, hogy a szélesebb fele legyen a keréktárcsák belső részénél. Ezután feltehetjük a láncalpakat, vagy a gumikat.

Tipp: Arról, hogy a láncalpakat vagy a gumikat tegyük-e fel, a körülmények megvizsgálása után már nem is olyan nehéz dönteni. Elsőként figyelembe kell vennünk a felületet, amelyen a jármű robotot tesztelni, használni szeretnénk. Ha ez a felület viszonylag sima, nem érdes (papírpálya, linóleum, úszóparketta...) akkor ajánlatos a láncalpakat használni, azért, mert kanyarodás esetében a gumik – a jármű nem túl nehéz súlya következtében – hajlamosak elcsúszni a felületen. Mikor okozhat ez problémát? Minden olyan esetben, amikor egy bizonyos mértékű kanyarodást állítunk majd be a robot programjában, legyen az kerékfordulat, szög, vagy akár másodperc alapján. Ennek elkerülésére használjuk a láncalpakat.

5. lépés:



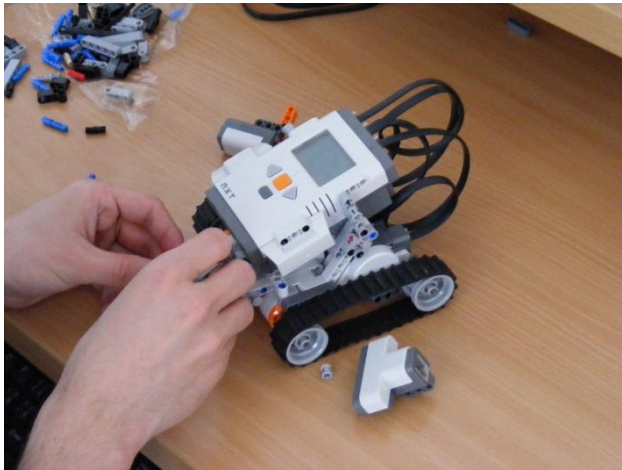
9. ábra



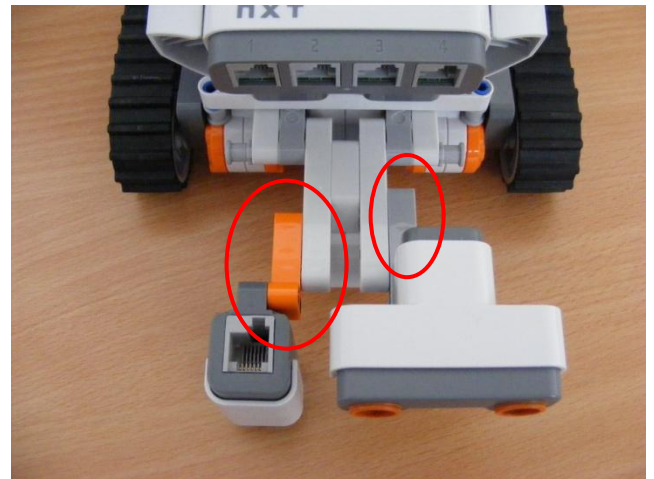
10. ábra

Ebben a lépésben a vezérlőegység kerül rá a konstrukciónkra. A 9. ábrán nagyon óvatosnak kell lennünk, ugyanis a téglát tartó elemnek nagyon stabilnak és erősnek kell lennie, ezért nézzük át alaposan a pirossal bekarikázott részt. Figyeljük meg, hogy a piros alkotórészt tartalmazó elem valójában két különálló elemből áll, melyeket maga a piros elem tart össze. Nagyon fontos, hogy ezeket a megfelelő irányban helyezzük fel. Ügyeljünk a tüköreffektusra.

6. lépés:



11. ábra



12. ábra

Miután ráerősítettük a vezérlőegységet, nincs más hátra, minthogy a jármű elejére felhelyezzük a megfelelő szenzorokat majd a kábelezéssel befejezzük az összerakást. Figyeljük meg, hogy a 12. ábrán milyen elemekkel kell felerősíteni a két szenzort. Mindkét elem más alakú, méretű és színű. A kisebbik elemet nagyon egyszerűen megtalálhatjuk a készletben, koncentráljunk a narancssárga színre és az L alakra. Ha ez megvan, a nagyobbik elem 1 cm-el nagyobb és I alakú. A tartóelemeket nem muszáj úgy felhelyeznünk, ahogy azt a 12. ábrán látjuk, ugyanis, ha mi a 'szem' (ultrasonic – ultrahang) szenzort a másik fény szenzor helyére szeretnénk tenni – mondjuk, mert nekünk az úgy esztétikailag jobban tetszik – akkor nincs más dolgunk, minthogy az imént megkeresett tartó elemeket felcseréljük. Ilyen módon megcserélődik a szenzorok helye is. Miért is van szükségünk a fény és a 'szem' szenzorra? Az előbbire azért, mert – mint azt már tudjuk – a fényérzékelő egyben színérzékelő is, és a jármű robotnak erre a funkciójára szükségünk lesz a követős feladatoknál, ahol is a jármű egy előre meghatározott pályát fog majd követni szín szerint. Az ultrahang szenzorra pedig azért van szükségünk, hogy az akadály kikerülős, tárgyfelismerős feladatokat végre tudjuk hajtani a robottal a programunk segítségével.

7. lépés:



13. ábra



14. ábra

Nincs más hátra, minthogy a motorokat és a szenzorokat összekössük a vezérlőegységgel, a 'téglaival'. Mi az, amire figyelünk kell:

1. az adatkábelek hossza
2. a tégl (szemből nézve) alján található négy port számozása
3. a tégl (szemből nézve) tetején található 3, betűvel jelölt portja

Kezdjük a motorok bekötésével. Figyeljük meg, hogy mindhárom szervo motor szélesebb, hátsó részén (a meghajtó rész ellenkező oldalán) található egy-egy port. Ezeket, és a tetején lévő 3, betűvel jelölt porttal kell összekötnünk. Nagyon fontos, hogy megjegyezzük, hogy melyik motorból jövő adatkábelt melyik a téglán lévő porttal kötjük össze, ugyanis a robot programozása során ez rendkívül fontos. Tétélezzük fel, hogy azt szeretnénk, hogy a robot haladjon előre, majd 5 másodperc után álljon meg. A program írásánál a motorok vezérlése úgy történik, hogy a megfelelő mozgáshoz egy A, B, vagy C betűvel jelölt portot kapcsolunk. Mivel a jármű robot meghajtása 2 motorral történik, ezért értelemszerűen két portot kell megjelölnünk, viszont nekünk 3 motor van a vezérlőegységhez csatlakoztatva. Ha egy meghajtó és a szabad motort választjuk ki az előre menéshez, akkor az fog történni, hogy a robot éles szögben kezd el majd fordulni. Tehát fontos megjegyeznünk, hogy a két meghajtó motort melyik betűvel jelölt portokhoz csatlakoztatjuk. Ugyanez vonatkozik a szenzorokra is, azzal a különbséggel, hogy míg a motorok esetében a portok jelölése betűvel történik és

számuk egy vezérlőegységen maximálisan 3, addig a szenzorok portjai számmal vannak megjelölve és maximális számuk egy vezérlőegységen 4 lehet.

Tipp: A portok és a hozzájuk tartozó motor/szenzorok jelölését a legegyszerűbben színes levonókkal tudjuk megoldani. Ilyen levonókat találunk a készletben is. Ragasszuk az egyik pár színt az adatkábel mindkét végére, ezáltal könnyen megkülönböztethetővé válnak a portok és a hozzájuk csatlakoztatott motor/szenzor.

1.2 Szenzorok

Robotunk életre keltéséhez a motorok és a vezérlőegység mellett szükség van úgy nevezett szenzorokra, melyek amolyan érzékszervként működnek. Az alapkészlet része 2 darab nyomás-, 1 szín- és fényérzékelő, valamint egy ultrahang érzékelő.(15. ábra) Ezek főként bemeneti eszközök, bár némelyikük kimenetként is szolgál. Lássuk tehát, miről is van szó.



15. ábra

Nyomásérzékelő



16. ábra

Az első szenzor, amivel a gyerekek megismerkedhetnek a nyomás-, vagy más néven ütközésérzékelő. Ez egy rugós mechanizmus, ami egyszerű kapcsolóként működik, 0 vagy 1 értéket közvetítve a robotnak. Lenyomáskor bekapcsolt állapotot érzékel, kiengedéskor fordítva (a programban beállítható a fordított viselkedés).

Jól használható olyan feladatokhoz, ahol a robottól egyéb utasítás végrehajtását várjuk, mikor akadályba ütközik, valamint robotunk távirányításához is alkalmazható.

Távolságérzékelő



17. ábra

Ahhoz hogy a robot tájékozódni tudjon a legfontosabb elem a távolságérzékelő, vagy más néven ultrasonic szenzor. Ez az eszköz a denevérekhez hasonlóan méri a távolságot. Vagyis ultrahangot bocsájt ki, ami a tárgyról visszaverődik, mint a visszhang. A szenzor méri az időt a hullám kibocsátásától kezdve annak visszatéréséig és ebből kiszámolja a távolságot. A maximum mérhető távolság 150 cm.¹

Tipp: Nagyon közeli, 5 cm alatti távolságnál az érzékelő veszít pontosságából. Funkciójából kifolyólag érdemes a szenzort a robot elejére szerelni, így a tájékozódás pontosabb.

¹ Róbert K., Badó Zs., Egyszerű robotika - A Mindstorms NXT robotok programozásának alapjai, 2010

Fényérzékelő, színérzékelő



18. ábra

Robotunk másik látásmódja a fény,- illetve színérzékelőkkel érhető el. A Fényérzékelő a világos és a sötét közti különbséget érzékeli és ezt egy 0-100-as skálán határozza meg. Az érzékelés a felület fényviszonyaitól is függ, mivel a szenzor vörös fényt kibocsátva a felületre, annak fényintenzitását határozza meg. Az újabb készletek alaptartozéka a

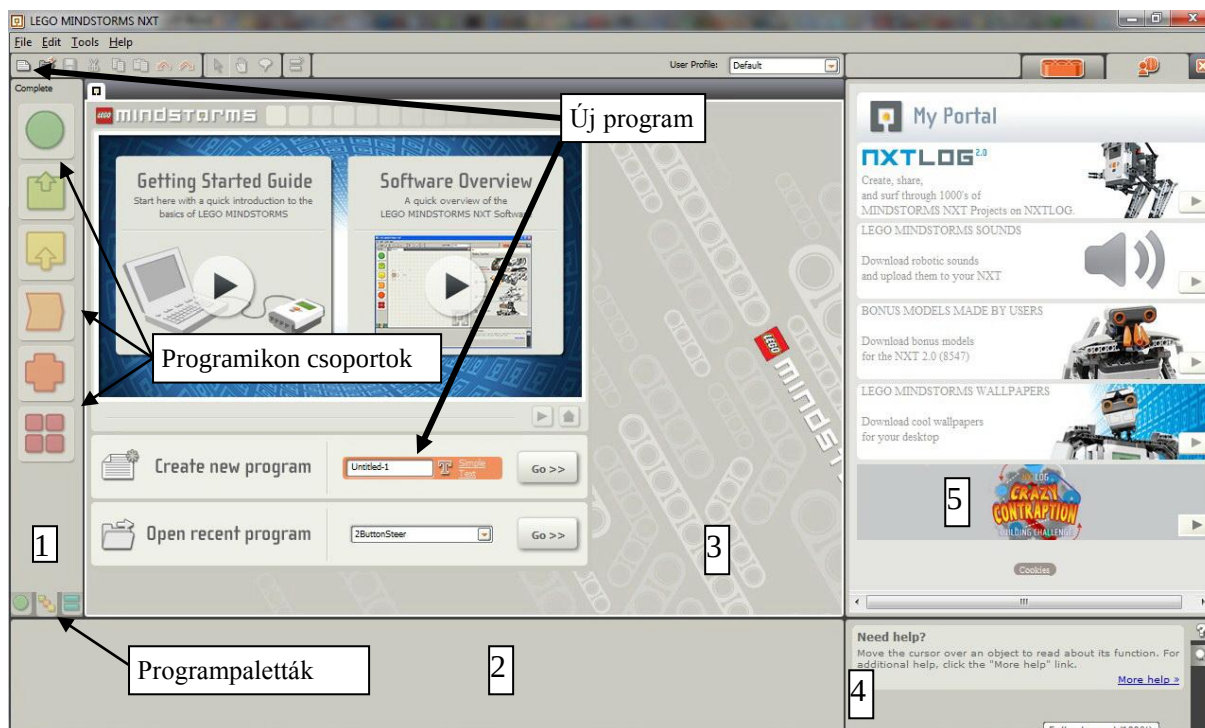
színérzékelő, mely már képes a három alapszín (zöld - piros - kék) segítségével a színeket is érzékelni. Ezen szenzorok nagy előnye, hogy nem csak input érzékelőként működnek, hanem képesek világítani is. Kiválóan felhasználhatóak például vonalkövetésre, világításra.²

Tipp: A távolságérzékelőhöz hasonlóan ajánlott a robot elejére szerelni őket. Vonalkövetéskor nagyon fontos, hogy a szenzor közelről érzékelje a vonalat (3-5 mm), máskülönben képtelen a pontos követésre. Bonyolultabb pályáknál, ahol több a szűk forduló szintén előnyös, ha a szenzor a lehető legközelebb kerül a robot autó fordulási tengelyéhez.

1.3 A szoftver bemutatása

Ebben a fejezetben megismerkedünk a programozási környezettel, annak alapfunkcióival és lehetőségeivel. Ezt a keretprogramot a készlethez kapjuk cd média formájában, melye telepítés után vehetünk igénybe. A szoftverhez tartozik egy grafikus felületű programozási környezet és egy illesztőprogram, ami a vezérlőegység és a számítógépünk USB porton keresztüli kommunikációját biztosítja. Nézzük tehát a programot, melyet az egyszerű telepítés (telepítő varázsló) után a képernyőn található LEGO Mindstorms NXT 2.0-ás ikonra kattintva érhetünk el:

² Róbert K., Badó Zs., Egyszerű robotika - A Mindstorms NXT robotok programozásának alapjai, 2010



19. ábra

A szoftver elindítása után a 15. ábrán látható képernyő fogad minket. Mielőtt bárhova, vagy bármire is kattintanánk, érdemes átböngészni az főablakon belüli ablakokat. A legfelső sávban találhatjuk a menüsört, ahol a fájlkezeléshez használatos menüpontokat találhatjuk, továbbá a Help menüpont alatt található egy részletes leírás a programról és az online támogatás, valamint a Tools menüpont alatt találunk olyan érdekességeket, mint a távirányítás és a firmware frissítés. A menüsor alatt, baloldalon láthatóak az egyes funkciók gyors eléréséért felelős kis ikonok. Ez a rész alatt a grafikus környezet 6 különböző részre oszlik:

1. Programpaletták – ezen menüpont segítségével illeszthetjük rá az elemekből felugró menüből az ikonokat (forráskód) a programkód területére, melyet a 3-as számú ablakban találhatunk meg, miután létrehozunk egy új (üres) programot.

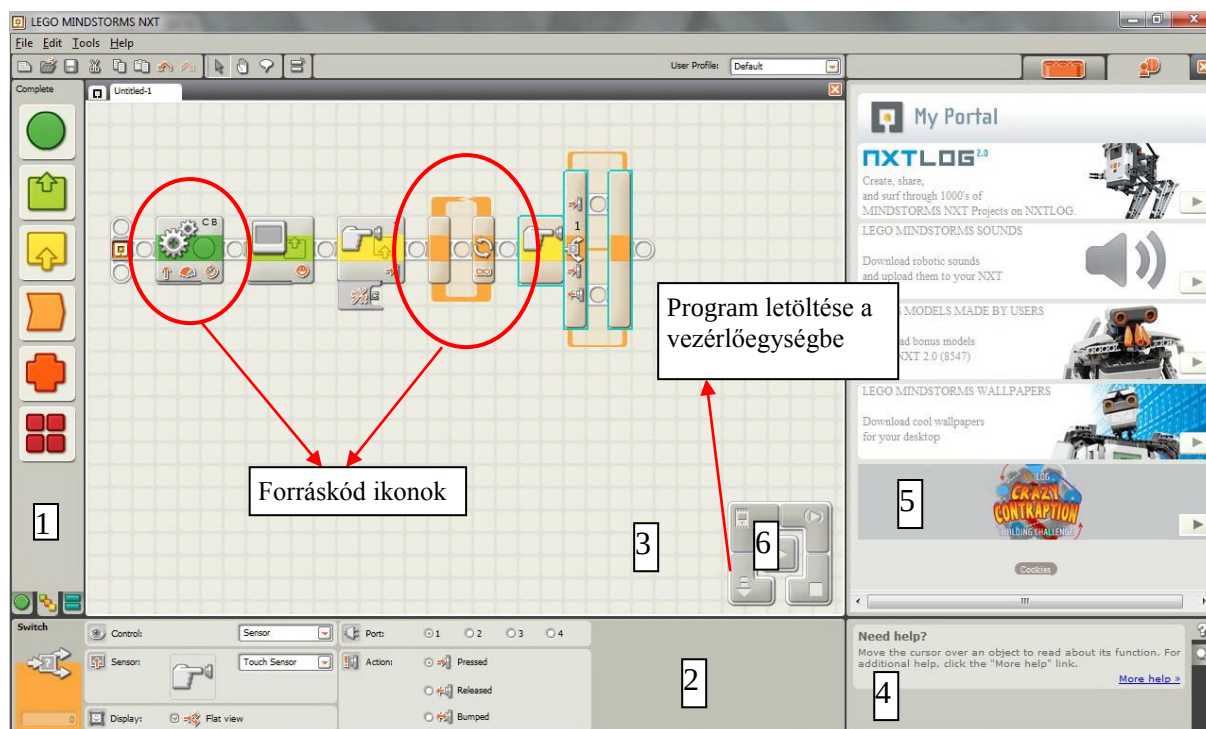
2. A 16. ábrán jól láthatjuk, hogy miután kijelölünk egy a programkód területére helyezett ikont, ezen a részen jelennek meg az ikon paraméterei.

3. Ez a rész a programkód terület. A programpalettáról erre a felületre tudjuk húzni az ikonokat, majd szerkeszteni magát a programot. Az ikonok helye a programkód területre helyezés után is változtatható, ciklusok, elágazások esetén akár egymásba is olvasztható.

4. Ez a rész a sűgó. A szerint változik, hogy a kurzorral épp melyik ablakra illetve ikonra kattintunk. Hasznos információkat tartalmaz, ha például egy olyan ikon funkciójára vagyunk kíváncsiak, amely esetében az ikonon lévő képről ez még nem egyértelmű, akkor az ikonra kattintva annak funkcióját a 4-es számú ablakban tudjuk elolvasni.

5. Az online profilunkat, valamint az egyes robotok összerakásához szükséges leírásokat tudjuk megnézni ebben az ablakban, valamint hivatkozásokat találhatunk itt felhasználók által készített robotok leírásáról, esetleg azok teszteléséről készült videókat.

6. Ezen a kis részen találhatunk olyan funkciókat, mint a robotkapcsolat beállításai, vagy az elkészült program a robot vezérlőegységének memóriájába való letöltése.



20. ábra

2. Programozáson belüli alapfogalmak szemléltetése robotok segítségével

A programozás alapfogalmait általában szokványos módokon szokták bemutatni (gondoljunk itt pl. a „Hello Világ” kiíratására), amivel feltétlen nem is lenne baj, ám miért ne tehetnénk ezt az egyszerű műveletsort is sokkal látványosabbá és élvezetesebbé? A következő fejezetben a feltétel és a ciklus alapfogalmaival ismerkedünk meg egy kicsit másképp.

2. 1 Mi az algoritmus?

Egy adott probléma megoldását jelentő, elemi lépések véges számú halmaza által alkotott műveletsor. Az algoritmusok különböző nyelveken - programozási vagy akár természetes, emberi nyelven is - készülhetnek, de mindenképpen egyértelműnek kell lenniük minden kérdés tekintetében, és jól meghatározott bemeneti valamint kimeneti pont(ok)kal kell rendelkezniük.

A számítógépes szoftverek mindegyike algoritmusok, valamint az ezekhez kapcsolódó adatok sokaságából áll. ³

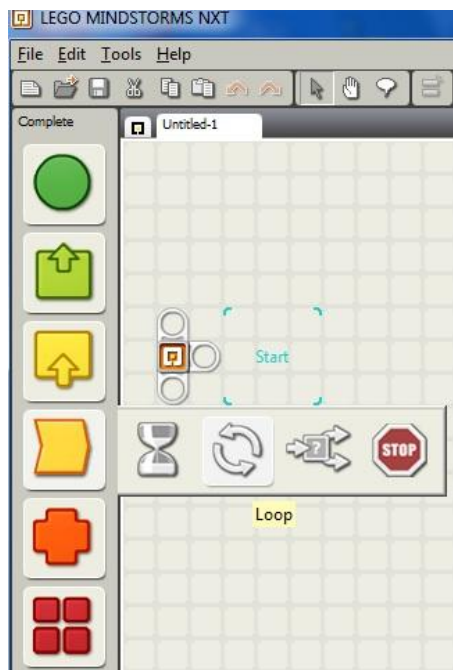
2. 2 A „ciklus” fogalma

A ciklus, vagy iteráció a számítógép-programozás és az algoritmusok egyik alapvető eszköze, amely az ismétlődő (azonos vagy hasonló) tevékenységek megvalósítására szolgál. A ciklus beépítését a programba ciklusszervezésnek is nevezik. A ciklust az egyes programozási nyelvek különböző kulcsszavakkal valósítják meg, de a működési módjukat tekintve három alaptípusba sorolhatók aszerint, hogy hányszor futnak le: ezek az előtesztelő, a hátultesztelő és a számlálós ciklus. ⁴

A LEGO Mindstorms NXT programjában a ciklusokat az angol „loop” kulcsszónál találjuk, mint ahogy azt a képen láthatjuk is.

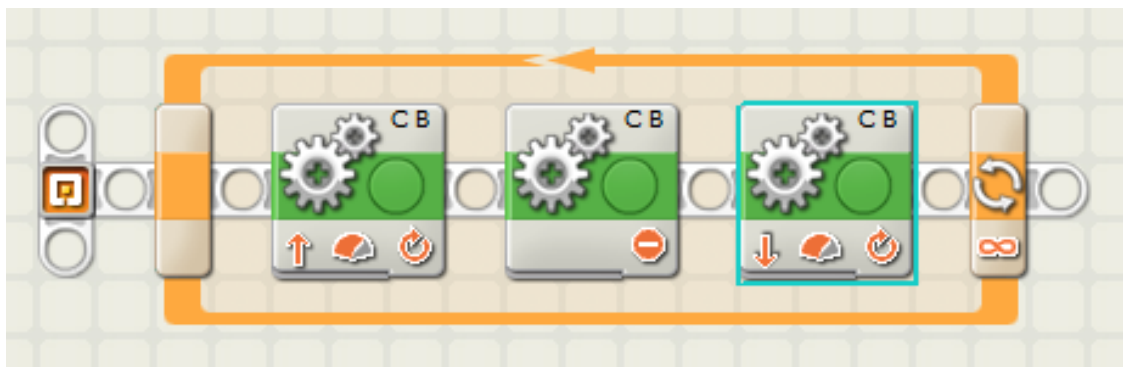
³ <http://pcforum.hu/szotar/algoritmus.html>

⁴ [http://hu.wikipedia.org/wiki/Ciklus_\(programozás\)](http://hu.wikipedia.org/wiki/Ciklus_(programozás))



Már a legalapvetőbb programjaink megírásánál is segítségül hívjuk. Például, ha azt szeretnénk elérni, hogy a robotunk haladjon előre, majd pedig adott idő múltán álljon meg, s kezdjen el tolatni – ezt egy egyszerű ciklussal megoldhatjuk, lásd köv. oldal.

21. ábra



22. ábra

Az itt látott ábra szemlélteti a ciklus (narancssárga vonal által magába foglalt rész) alkalmazását a gyakorlatban. Funkciós egységeink balról jobbra a következők: (1) a B és a C motorok előremozdítják adott objektumunkat előre meghatározott ideig, ezt követően (2) gépünk megáll, majd pedig (3) újraindul, mindössze ellenkező irányba, azaz tolatni fog.

A robotika előnyei közé sorolandó, hogy a kezdő programozók számára nem igazán szükséges előre meghatározni, hogy az adott algoritmusuknak milyen ciklusra is lesz szüksége, ugyanis nem kell parancsokban gondolkodniuk.

Az, hogy az adott algoritmus épp „for”, „while”, „repeat-until” stb. ciklust alkalmaz-e az határozza meg, hogy milyen feladattal rukkolunk elő. Ha az előbb feltüntetett példát, vesszük

alapul, *for* ciklusról fogunk beszélni. Ha viszont olyan robotot szeretnénk megépíteni, ami addig fog előre haladni, míg meg nem „pillant” valamit, már *while* ciklusról beszélünk.

2.3 A „ha” feltétel bemutatása

A „ha” feltétel az egyik legegyszerűbben és leghamarabban elsajátítandó függvények közé sorolható. Lényege: ha valamilyen feltétel teljesül, egy adott történés vagy cselekménysorozat végrehajtódik; ellenkező esetben nem történik semmi, vagy pedig az általunk megszabott másik parancs hajtódik végre.



23. ábra

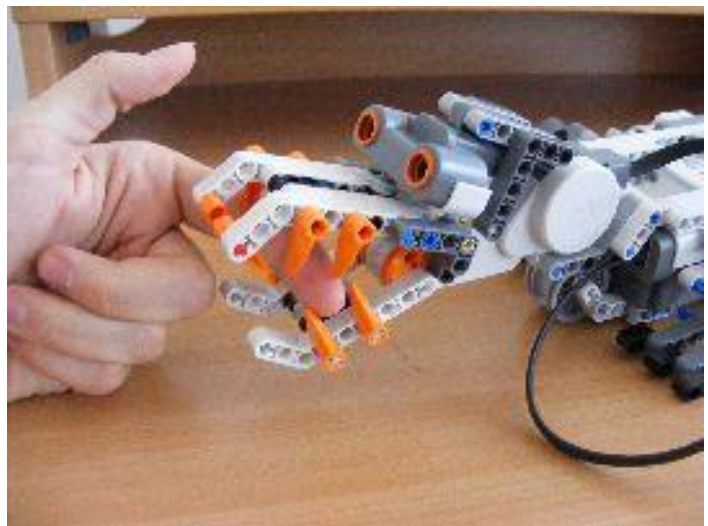
Szemléltessük ezt egy egyszerű példán:

Ha az éhes krokodil nem lát semmit, nem táog. Miért is tenné? A krokodil szája csukva (a feltétel nem teljesül).



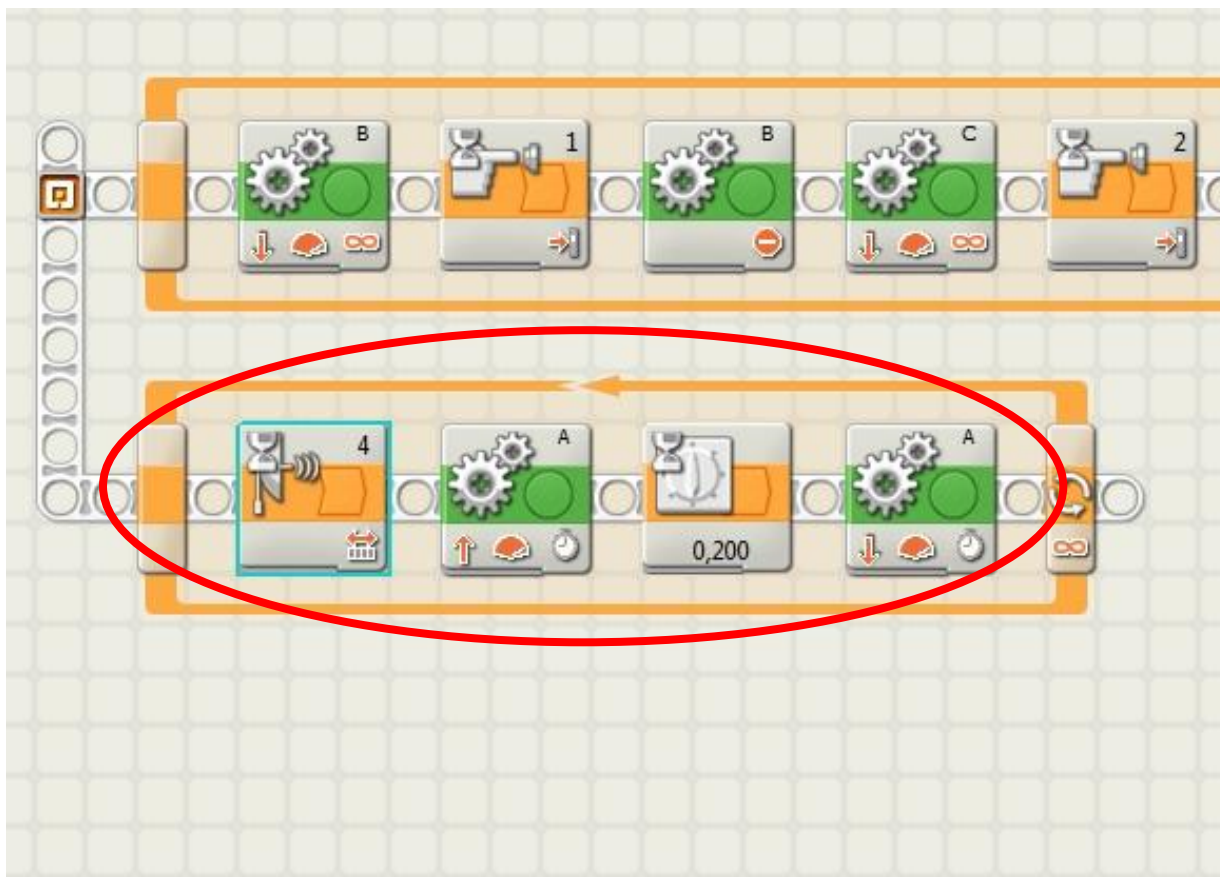
24. ábra

Ha viszont meglát egy gusztusos emberi ujját, akkor bizony ráharap! A krokodil harap (a feltétel teljesül, hiszen emberi ujját látott).



25. ábra

A példa megvalósításához nem kell több mint az itt látható algoritmus kiemelt részét bemásolnunk a programunkba.



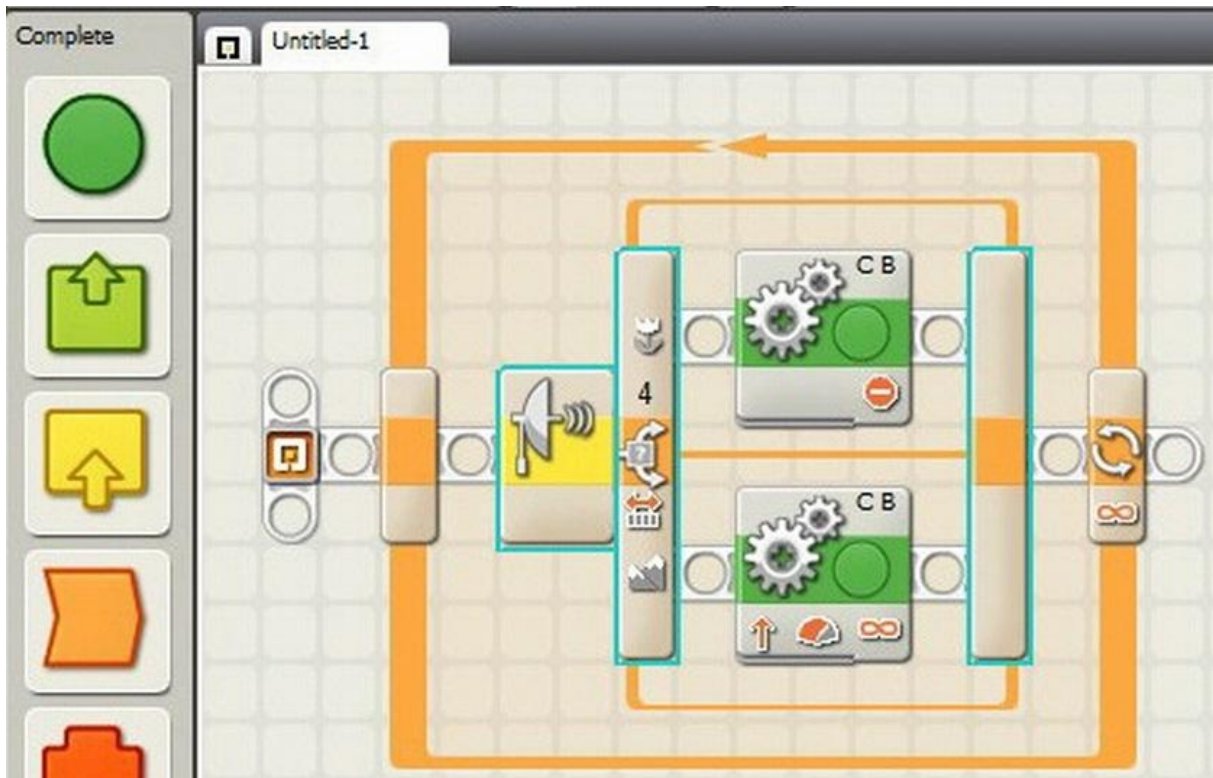
26. ábra

A kiemelt 4 funkciós rész értelmezése balról jobbra: (1) a 4-es portra kötött úgynevezett ultrazónikus szenzor érzékeli van-e előtte valami. Amennyiben van (2), az A portra kötött motor működésbe lép, ami azt fogja eredményezni, hogy a krokodil szája kitárul (azaz a motor az egyik irányba elkezd pörögni), majd (3) vár a másodperc tört részéig (0,2 sec), ezt követően (4) újra működésbe lép, csak az ellenkező irányba (ezzel azt a látszatot keltve, hogy a krokodil harap).

2. 4 A „for” ciklus és „ha” feltétel egymásba ágyazása

A ciklusok és a feltételek egymásba ágyazása szinte magától értetődő egy komolyabb program megírásakor, hiszen csak ezek együttes használhatóval írhatunk bonyolultabb algoritmusokat.

Szemléltessük hát, hogyan is néz ki egy ilyen eset a gyakorlatban:



27. ábra

Programunk *for* ciklussal (narancssárga keret) indít, melyben található egy *ha* feltétel (sárga alapon parabola), aminek felső ága akkor fog végrehajtódni, ha az a feltétel teljesül (a motor megáll); az alsó pedig akkor, ha a feltétel nem teljesül (a motor működésbe lép).

Ez a valóságban egy fényérzékelő szenzorral felszerelt gépjárműt fog jelenteni, mely addig fog előre haladni, míg meg nem lát valamit maga előtt. Abban a pillanatban, hogy kezünket megpillantja, a motor leáll.



28. ábra

Ezen feladat mintájára számos egyszerű algoritmust írhatunk, melyekkel tesztelhetjük robotjaink képességeit. Mint az látható is, egyáltalán nem nagy ördögösség megírni, hiszen a LEGO Mindstorms NXT felhasználóbarát felülettel rendelkezik. *Ciklus* megadásakor elég behúznunk egy *loop* egységet (lásd 22. oldal), *ha* feltétel megadásakor pedig attól függően, hogy milyen szenzorral



29. ábra

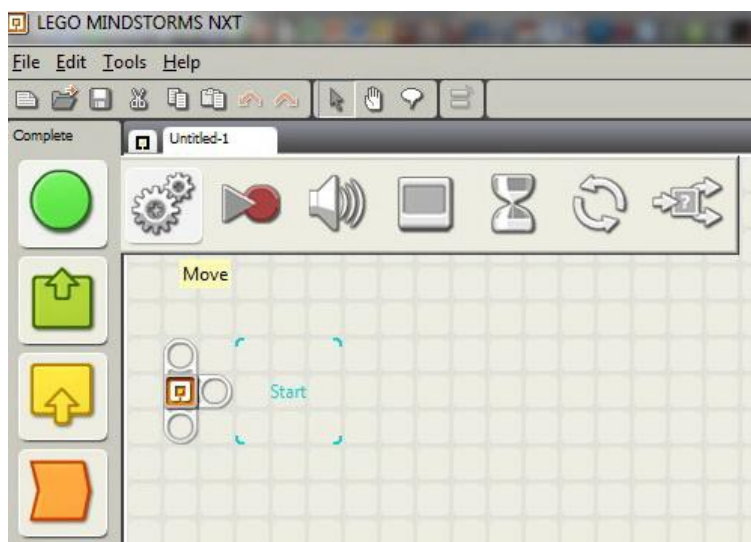
szeretnénk dolgozni, kiválasztjuk a számunkra megfelelő *switch*-et (lásd 23. oldal). A többit a program elvégzi helyettünk, mindössze attribútumokat fog kellenünk megadni, mint pl. milyen gyorsan -, előre vagy hátrafelé haladjon a járművünk, hány centiméterre kezdjen érzékelni a szenzor stb.

3 Első program

1. feladat: Készítsünk egy programot, melyet végrehajtva a robot megy előre, majd 5 másodperc után megáll!

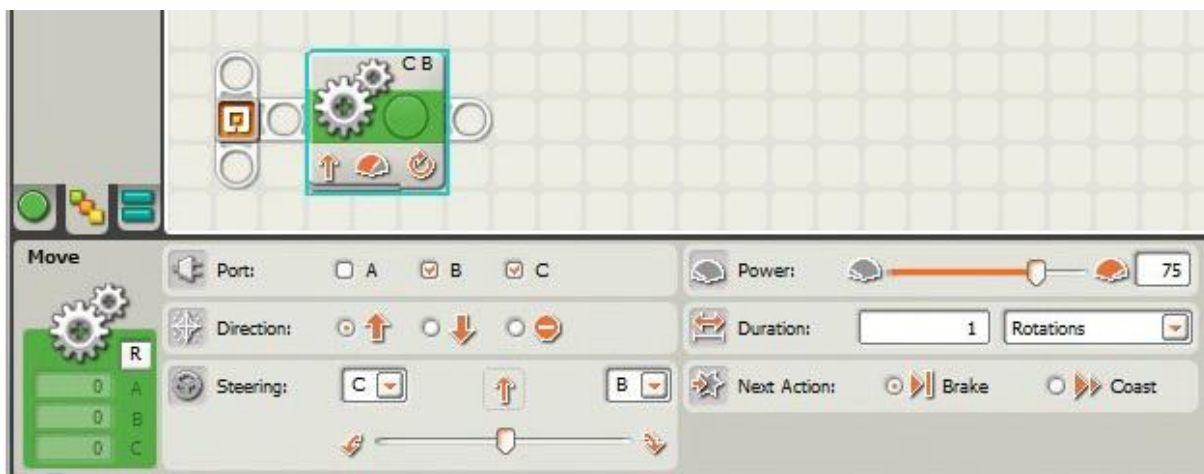
1. lépés

Miután megnyitottunk egy új programot, a programpalettán váltsunk 'Complete Palette' (Komplett paletta) fülre, majd a legfelső (Common – általános) ikon fölé húzva az egeret, az ikon jobb oldalán előugrik 7 új ikon. Ezek közül kattintsunk az elsőre (Move – mozgás), majd az egér bal egérgombját letartva húzzuk ezt az ikont a programkód területre.



30. ábra

2. lépés



31. ábra

Miután az első lépéssel megvagyunk, a programkód ablakban a fent látható (két fogaskerék) ikonnak kell megjelennie (a kép méretének csökkentése miatt az ikont a programkód ablak bal

alsó sarkába húztuk). Ha rákattintunk erre az ikonra, akkor a fő ablakunk bal alsó részén megjelennek az ikon paraméterei. Akkor nézzük mit is állíthatunk itt be:

1. Először is, itt állíthatjuk be, hogy melyik porton lévő motort szeretnénk irányítani (itt jön jól, ha az összerakás folyamata alatt megjegyeztük a bizonyos motorokhoz tartozó portokat).
2. Melyik irányba hajtson a motor. Az előre nyíl értelemszerűen az előre hajtás, a hátra nyíl a hátra hajtás, a behajtani piros lámpa pedig a megállást jelenti.
3. A Steering opció a kanyarodás paramétere. A legördülő menüben beállíthatjuk, hogy melyik két motor legyen meghajtva a kanyarodás során. A csúszkával pedig azt állíthatjuk be, hogy milyen legyen a kanyarodás mértéke. Ha például teljesen a jobb oldalra húzzuk, akkor az arra az oldalra beállított motort meghatja a program, a másik viszont nem mozdul.
4. A Power csúszkával a motor erősségét állíthatjuk be, tehát ezzel határozzuk meg, hogy milyen gyorsan mozogjon a robot.
5. Következő paraméter a Duration, vagyis időtartam. Itt egy beviteli és egy legördülő menüt találunk, nézzük mi az, amit ezek segítségével beállíthatunk. Rotation – Fordulat, tehát hányszor tegyenek meg a motorok 1 teljes fordulatot. Seconds – Másodpercek, tehát hány másodpercig menjenek a motorok. Degrees – Fok, hány fokot forduljanak a motrok; itt fontos megjegyezni, hogy nem kanyarodásról beszélünk, hanem hogy a motor meghajtó része hány fokot forduljon az adott irányba. Unlimited – Korlátlan ideig, tehát a motorok addig fog menni, amíg mi, vagy a programkód egy bizonyos része, feltétele le nem állítja azokat.
6. A Next Action paraméter esetében két rádiógomb közül választhatunk: Break – Fékel, tehát a motor megállítása fékezéssel történik, vagyis azonnal nyugalmi pozícióba kerül. Coast – csúsztatás, vagyis a motor leáll, de a robot még egy rövid ideig lassulva gurul tovább.

Ha a járművet meghajtó két motor nekünk is a C és B portra van kötve, akkor nincs más dolgunk, mint a Duration paraméternél beállítani az 5 másodpercet, és kész is a feladatunk. Ha viszont a két meghajtó motor más portra van állítva, ne felejtsük el ezeket átállítani a Port paraméternél.

3. lépés

Miután elkészültünk a 2. lépéssel, az USB adatkábel (33. ábra, készlet része) segítségével csatlakoztassuk a robot vezérlőegységét a számítógéphez. Ezután kattintsunk a programot a vezérlőegységbe letöltő gombra (32. ábra). A program



32. ábra



33. ábra

sikeres letöltését hangjelzés és egy felugró ablak jelzi. Előfordulhat, hogy a felugró ablakban valamilyen hibaüzenetet kapunk. A leggyakoribb ilyen üzenet a teli memóriára figyelmeztet minket. Ilyenkor ki kell hogy töröljünk a robot vezérlőegységén lévő programokból annyit, hogy elég hely szabaduljon fel az új programnak. Ezek a programok miatt nem kell aggódnunk, ugyanis a számítógépünkben úgyis elmentettük őket, bármikor visszatölthetők a robotba.

Tipp: A vezérlőegység memóriája gyorsan megtelik, ha a robotunkat több programban is beszéltetni szeretnénk, ugyanis a hangfájlok viszonylag nagyméretűek a programokhoz képest. Ezért ésszerű egyszerre csak egy-két hanganyaggal bővített programot töltenünk a robot memóriájába.

Jöhet a tesztelés. Válasszuk le a robotunkat a számítógépről. Tegyük biztonságos felületre, ügyelve arra, hogy legyen előtte legalább 1,5 m hosszú szabad pálya. A vezérlőegység narancssárga gombjának megnyomásával beléphetünk a letöltött programok mappájába. Az alapértelmezett program mindig az, amit legutoljára töltöttünk le a robotba. Ha mégsem az elmentett programunk neve áll a kijelzőn, akkor a jobb és bal navigációs gombok segítségével keressük azt ki. Ezután nyomjuk meg újra a narancssárga gombot. Ha mindent a robotnak 5 másodpercig előre irányba kéne mennie, majd ezután leállnia.

3.1 Vonalkövetés – fény- és színszenzor használata

Ebben a fejezetben már egy komplexebb feladattal folytatjuk, mely során már szükségünk lesz egy szenzorra is, konkrétan a fény- és színszenzorra.

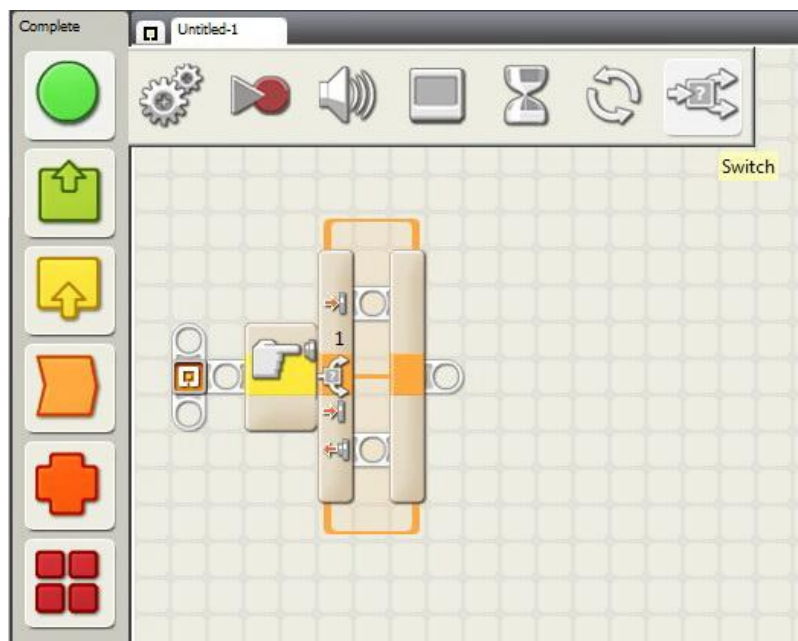
2. feladat: Készítsünk egy programot, mely segítségével a jármű robot vonalkövetést végez! Teszteléshez használjuk a készletben megtalálható, széthajtható pályát, melynek a fekete ovális alakú pályája a legmegfelelőbb erre a célra.

1. lépés

Tehát akkor mit is kell csinálnia robotnak? Ha rátesszük a fekete pályára, haladjon előre a fekete sávot követve, vagyis körözzön az ovális pályán. Felmerülhet a kérdés, hogy „Miért is van szükségünk a fény- és színszenzorra, hisz a pálya és a fekete sáv méretei alapján tudunk írni olyan programot, amelyet elindítva a robot körözni fog a pályán?”. A válasz nagyon egyszerű. Ezen a pályán csak tesztelni fogjuk a robotot, annak bármilyen irányú és hosszúságú fekete sávos pályán végig kell menni. Ezt pedig csakis a fény- és színszenzorral tudjuk megoldani.

Nyissunk egy üres programot!

Szükségünk lesz egy elágazásra. A Common – általános ikon fölé húzva az egerünket, hetedik lehetőségként előugrik a Switch – elágazás ikon. Ezt az ikon húzzuk rá a programkód részre. Amint az a 21. ábrán is láthatjuk, az elágazás ikon alapértelmezett beállítása a touch – nyomásérzékelő. Mivel nekünk most a színérzékelőre van



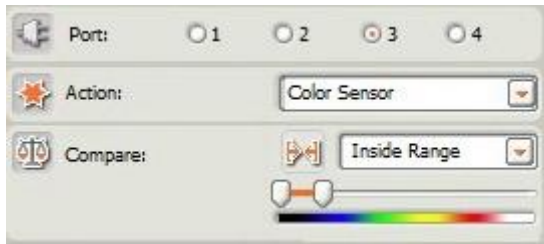
34. ábra

szükségünk – ahhoz, hogy a robot meg tudja különböztetni a pályán lévő fekete sávot a többi színtől – ezért az ikon paraméterei között ezt át kell állítanunk.

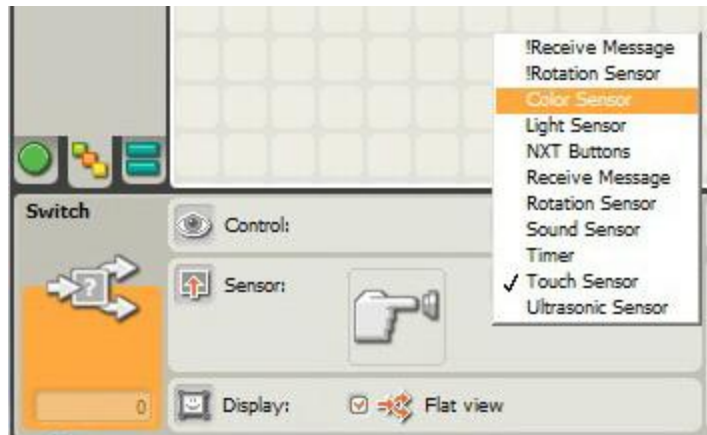
Tipp: Miért a színérzékelőt használjuk a fényérzékelő funkció helyett? Mivel ebben az esetben tudjuk, hogy a fekete színt szeretnénk követni, egyszerűbb a színérzékelő használata.

A szenzor fényérzékelő funkciója ugyanis egy számadatot ad vissza, amit előbb le kell tesztelnünk a fekete színre.

Kattintsunk az ikonra az egérrel, majd a 36. ábra szerint állítsuk be a színérzékelőt.



35. ábra



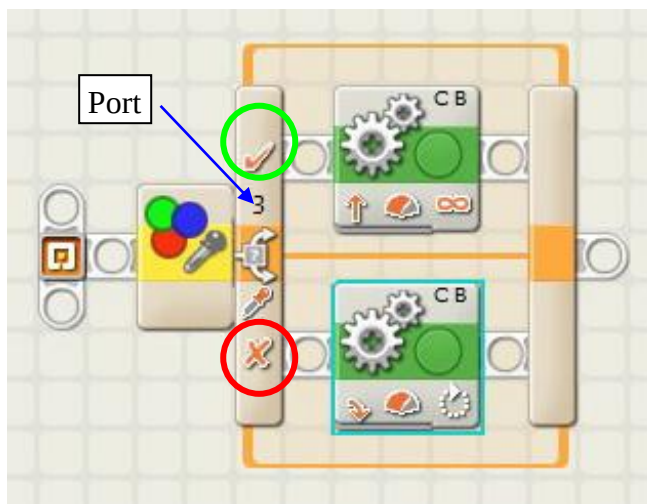
36. ábra

Ez után be kell állítanunk, hogy melyik színt szeretnénk figyeltetni a színérzékelővel. A 35. ábrán jól látható, hogy a Compare – összehasonlítás paramétert kell módosítanunk célunk érdekében. A csúszkát állítsuk a fekete színre. Figyeljük meg, hogy a csúszkának két része van. Ez azért van így, mivel egyszerre több színt is tudunk figyeltetni a színérzékelő segítségével. Mivel nekünk most csak a fekete színre van szükségünk, ezért mindkét csúszkát állítsuk a fekete színre. Ezen paraméter lehetőséget ad még arra, hogy a beállított szín belső (inside range) illetve külső (outside range) tartományát szeretnénk figyelni. Mivel mi most mindkét csúszkát a fekete színre húztuk, ezért a belső (inside range) tartományt kell figyelnünk, vagyis hagyhatjuk az alapbeállításon.

2. lépés

Ebben a lépésben húzzunk a programkód területére 2 db mozgás ikont. Az egyiket az elágazás igaz, felső sávjába illesszük, a másikat pedig a hamis, alsó sávba. Ha az állítás igaz – vagyis a robot színérzékelője fekete színt lát – menjen a robot korlátlan ideig előre. Ha viszont az állítás hamis – tehát a fekete színen kívül bármilyen más színt érzékel – akkor a robot meghajtó motorjainak paramétereit állítsuk be úgy, hogy forduljanak teljesen jobbra 20 fokos tengelyfordulással.

Nézzük mi az, amit eddig létrehoztunk. Miután elindítjuk a programunkat, a robot színérzékelője aktiválódik. Az elágazás segítségével figyeljük azt, hogy mikor látunk feketét és mikor látunk a feketén kívül az összes többi színt. Amint azt a 37. ábrán is láthatjuk, ha a feltétel teljesül (zöld jelölés) akkor a megyünk előre korlátlan ideig, ha pedig nem teljesül (piros jelölés), akkor fordulunk jobbra.



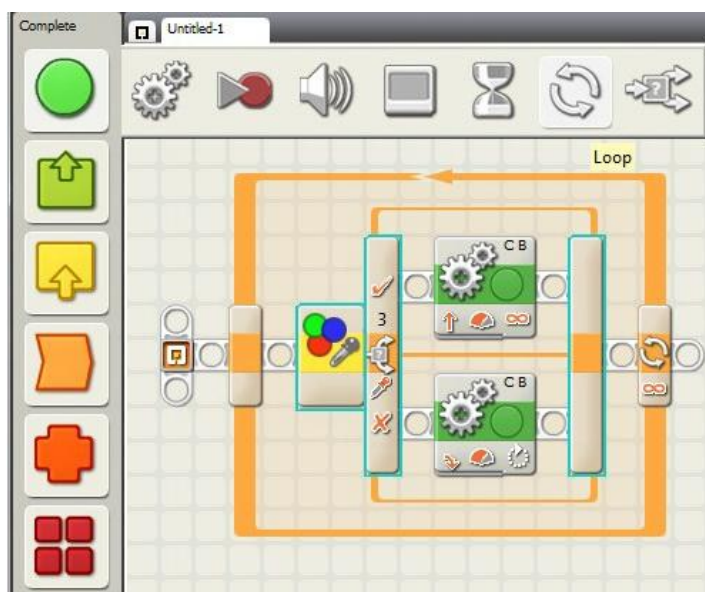
37. ábra

Még egy fontos dolgot meg kell, hogy

említsünk, ez pedig a színérzékelő portjának a száma. Figyeljük meg, hogy a robotunkon hányas számú portra kötöttük a színérzékelőt. Az elágazásban – ahogy a 23. ábrán a nyíl is mutatja – le tudjuk ellenőrizni, hogy a megfelelő port van-e beállítva. Ha nem, akkor az elágazás ikonjára kattintva a paraméterek között javítsuk azt ki a megfelelő portra.

3. lépés

A programunk még nem teljes, ugyanis ha leteszteljük, észrevesszük, hogy a robot a legelső döntés (elágazás) elvégzése után már le is áll. A probléma az, hogy a programunk még nincs ciklusban, tehát nem ismétli az adott programrészletet. Ahhoz, hogy ezt a hibát kijavítsuk, húzzuk az egerünket a Common – általános ikonra a programpalettán, majd a felugró menüből válasszuk ki a Loop – ciklus ikont. Miután ráhelyeztük a

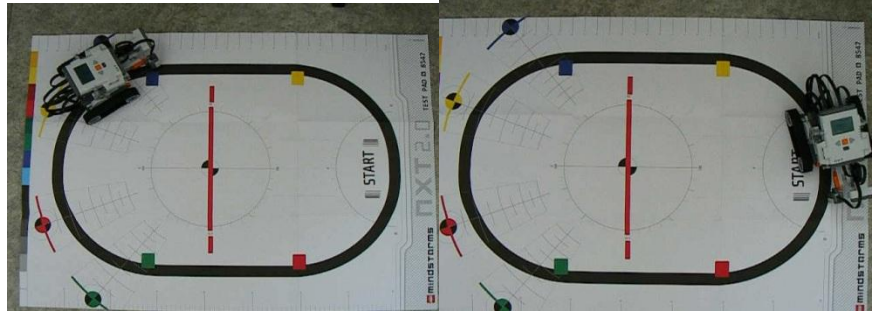


38. ábra

programkód területére, láthatjuk, hogy a már megírt programunk elé, vagy mögé tudjuk csak helyezni, ami még mindig nem megoldás a problémánkra. Az egér bal gombjával kattintsunk az elágazásunkra, majd húzzuk ezt a ciklus ikon középebe. Láthatjuk, hogy ekkor az ikon olyan mértékben kibővül, hogy a programrészletünket magába tudja foglalni. A ciklus ikonra kattintva megjelennek annak paraméterei, melyek közül a legfontosabb az, hogy hányszor

ismétlődjön. Alapbeállításon ez Forever – végtelen, az a beállítás, amire nekünk éppen szükségünk van, ugyanis azt szeretnénk, ha a robot folyamatosan haladjon a fekete sávot figyelve a pályán. Miután ezzel elkészültünk, nincs más dolgunk, minthogy az USB adatkábel segítségével letöltsük a programot a robot vezérlőegységébe, majd teszteljük a készlethez tartozó pályán.

Amint látjuk, a programunk az elvárásoknak megfelelően működik. Megfigyelhetjük, hogy ha a robotot a másik irányban indítjuk el, akkor, ha nem fekete színt észlel, akkor is jobbra fog fordulni, tesz



39. ábra

egy teljes kört, majd elindul az ellenkező irányba. Ha ezt el szeretnénk kerülni, több ciklus beiktatásával a program átírható úgy, hogy elkerüljük ezt az fölösleges fordulatot.

3.2 Vonalkövetés kibővítése

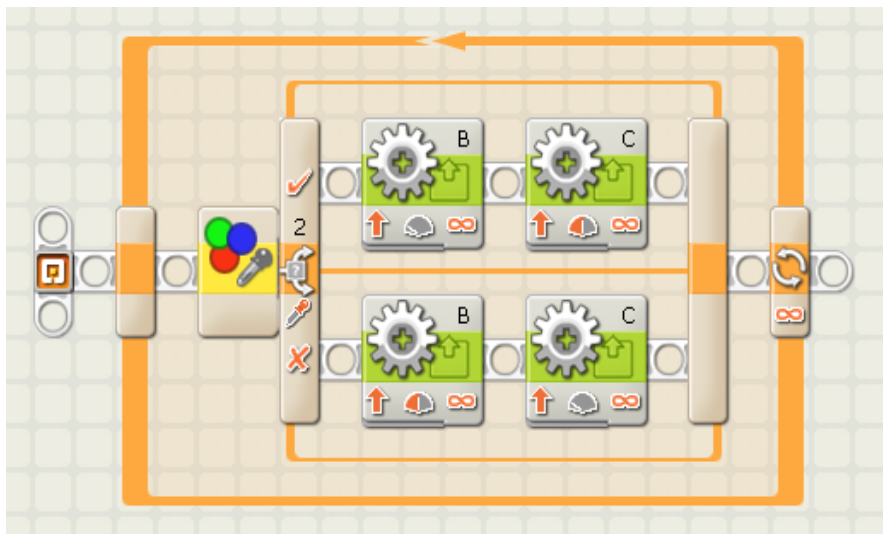
5. feladat: A 2. feladat alapján készítsetek egy programot, mely gyorsabb vonalkövetést tesz lehetővé. A programunk egyúttal küszöbölje ki azt a hibát, hogy robotunk megforduljon, és visszafelé kövesse a vonalat.

1. lépés

Miről is van szó? Amennyiben a 2. feladatot kipróbáltuk hamar észrevesszük, hogy robotunk csak egy irányban képes végighaladni a pályán. (Ha a fordulást jobbra állítottuk, akkor jobb kanyarok esetén, ha balra akkor bal kanyarok esetén képes végigcsinálni) Tovább nehezítené a feladatot, ha a pálya nem ovális, vagy kör alakú, ilyen esetben a robot képtelen teljesíteni a feladatot.

Ezen probléma megoldásához robotunkat nem kell átalakítani. A főszerep ismét a színszenzornak valamint a két motornak jut, csupán ezek irányítását állítjuk be pontosabban.

2. lépés:



Első ránézésre a program nagyon hasonló az előzőhöz képest. Ismét adott egy végtelen ciklus, ami segítségével a robot bármilyen hosszúságú útvonalon képes végigmenni.

40. ábra

Továbbá szintén adott egy elágazás, mely a színszenzortól függ. Az érzékelő beállításán nem kell változtatnunk. A trükk a motor ikonokban keresendő. Az előzőtől eltérően ahol a Mozgás ikonokkal dolgoztunk, most a motorokat külön fogjuk vezérelni. Ehhez válasszuk ki az Akció/Motor ikont, majd helyezzünk el négyet az elágazásba, ahogy a képen látható.

3. lépés



41. ábra

Az elágazás felső részében beállítjuk az egyik motort, hogy lassan forogjon előre.



42. ábra

Majd ugyanezt megcsináljuk a másik motorral, csak nagyobb erőt állítunk be.

Az elágazás alsó részében megismételjük ezt a lépést, csak fordítva (B forog gyorsan és a C lassan). Ügyeljünk arra, hogy az erő azonos legyen a gyors és lassú pároknál is.

Hogyan teljesíti ez a program a feladatot?

Az előző esetben robotunk egyenesen haladt előre, míg le nem tért a pályáról, mikor is egy irányú forgással keresni kezdte a kijelölt utat. Ha ez megvolt, ismét egyenes irányú mozgás következett. Ebben az esetben a robot nem végez teljesen egyenes mozgást, hisz az mindkét motorja forog, még ha az egyik kisebb mértékben is. Viszont mivel az útvonalról letérve is halad egy keveset előre, miközben keresi a kijelölt szakaszt, összességében gyorsabb.



43. ábra

Robotunk nem fordul meg, mivel mindkét oldalirányban végez kereső mozgást. A robot mindig a vonal szélén "lépked", amíg egyik illetve másik feltétel nem teljesül. Ez a program szinte bármilyen pályát képes teljesíteni.

Érdekességgént említhető meg, hogy a vonalkövetés népszerű versenyszámnak számít a Mindstorms robotok kedvelőinél, ahol a lehető leggyorsabb robot a győztes. Ehhez sokciklusú és elágazású programokat, valamint több fényérzékelőt használnak, miközben próbálják a lehető legfordulékonyabb és legkönnyebb robotot elkészíteni.

3.3 Járőr Robot

3. feladat: Készíts egy járőr robotot a következőképp:

Indítás után egyenesen előre halad és zölden világít, míg akadályt nem észlel. Ekkor kék színnel világítson. Robotunk várjon két másodpercet, majd ismét nézze meg van-e előtte akadály. Amennyiben a tárgy kitért, a szín váltson zöldre, robot pedig forduljon meg és járőrözzön visszafelé. Amennyiben a tárgy nem tért ki, a szín váltson pirosra, robotunk löje le az ellent, majd forduljon meg, a szín váltson zöldre és járőrözzön tovább.

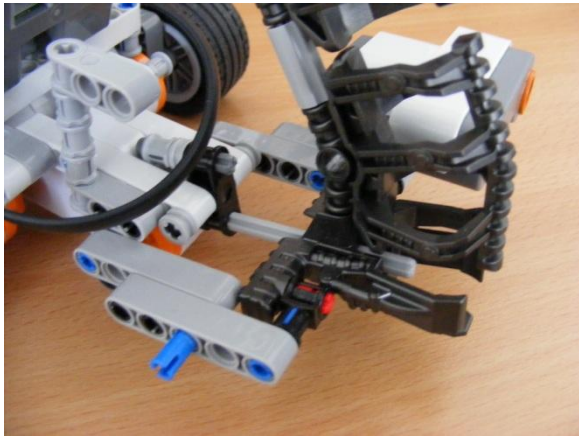
Az összeállításhoz szükségünk lesz tehát:

Távológ-, színérzékelőre, az előbbieken megismert robotautóra, valamint a robot fegyverére.

Ezt a fegyverzetet vizsgáljuk meg közelebbről!

1 lépés:

Az alapja egy motor, (a mi esetünkben a középső, mely az előzőekben már felkerült az autóra) mely elé egy hosszabb műanyag rudat erősítünk, valamint a lőtartály (az alapkészlet tartozéka, a golyók tárolására szolgál), amit viszont a motorunk fog előre-hátra mozgatni. A rúd vége a lőtartályban végződik (44., 45. ábra).



44. ábra

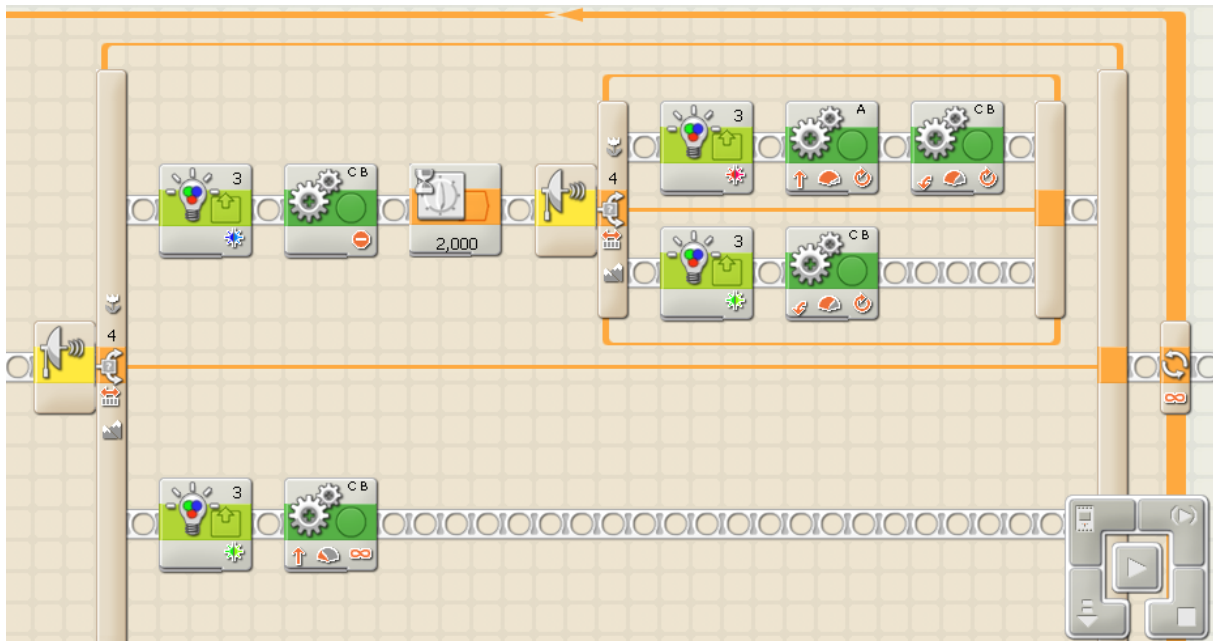


45. ábra

Programunkban egyszerűen beállítjuk, hogy a motor végezzen maximum sebességgel egy teljes fordulatot. Amikor ez megtörténik a lőtár közeledik a rúd felé, ami kipöcköli a tárban található legalsó golyót. A lőtár visszatér elülső állásba, így a soron következő golyó az előző helyére esik, a robotunk pedig újra tüzelésre kész.

2 lépés:

A program közelebbről:



46. ábra

A programban két elágazást alkalmazunk. Az első megnézi, hogy van-e előtte valami bizonyos távolságon belül (pl. 20 cm). Ha az állítás hamis a színerzékelő zölden világít és robotunk megindul előre. Ha az állítás igaz, kék színre vált és megáll. Vár két másodpercet, majd jön a második feltétel, ahol újra megnézi van-e előtte akadály. Ha nincs a ismét szín zöldre vált és elfordul. Amennyiben az akadály még mindig ott van, piros színnel világít, meghajtja az A-motort (vagyis kilövi a golyót), valamint elfordul. Végül az egész programot egy végtelen ciklusba illesztjük.



A járórobot felülnézetből (47. ábra). Itt jól látható a távolságérzékelő-szenzor és a golyótár a robot elején valamint a színszenzor a téglá felett.

47. ábra

Figyelmeztetés színjelzéssel



48. ábra

Na, itt már baj van...



49. ábra

3.4 A Járőr Robot program kibővítése

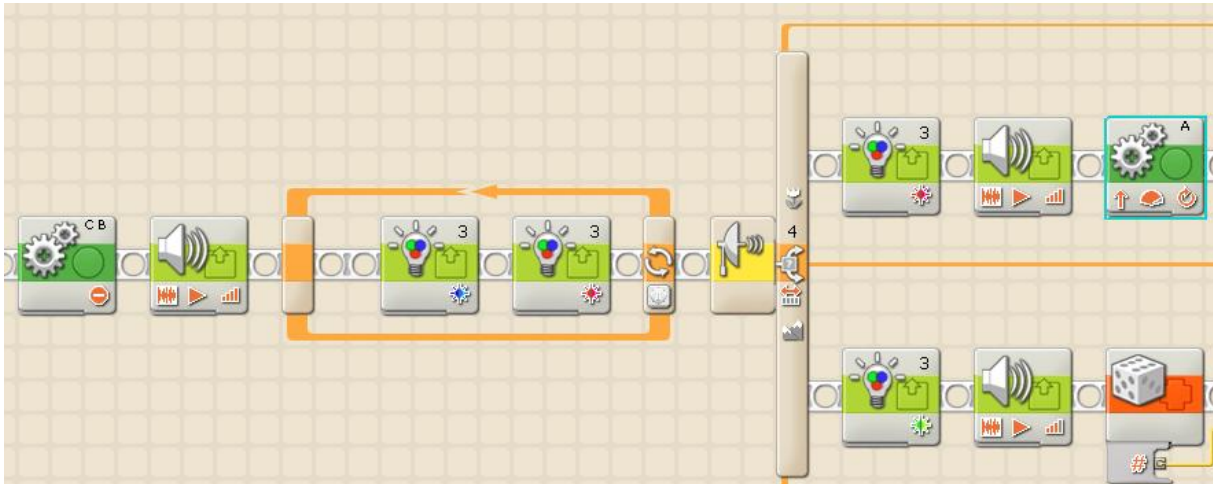
Robotunkat bizonyos hangjelzésekkel, változatosabb mozgással és egy villogó szimulálásával még élethűbbé tehetjük.

4.feladat: Egészítsd ki az előző programot a következőképpen:

- a,** hangjelzések: a robotunk szólítsa fel az akadályt, hogy térjen ki az útjából, majd ha az odébb áll, köszönjön, ha nem búcsúzzon el tőle, mielőtt lelövi (bármilyen hangjelzések lehetnek)
- b,** villogás: a felszólítás után, míg robotunk vár, a színszenzor villogó jelzést adjon akár egy rendőrautó (piros-kék szín váltakozása)
- c,** mozgás: a fordulásokat a robot többféleképp tegye meg (min 3) és véletlenszerűen válassza ki azokat (többféle irányban, illetve erővel forduljon)

4.a feladat

Ehhez a feladathoz nincs szükség másra, mint hang ikonok beillesztésére az adott cselekvések után. Mint az alant lévő ábrán látható az első feltételben miután a robot leállt, a második feltételben pedig a színváltások után.



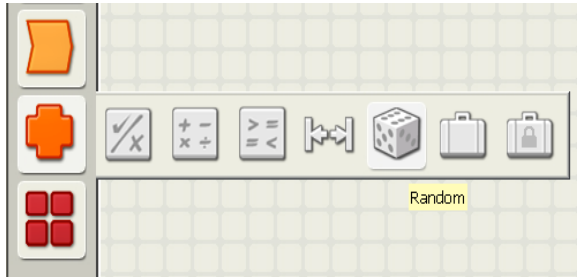
50. ábra

4.b feladat

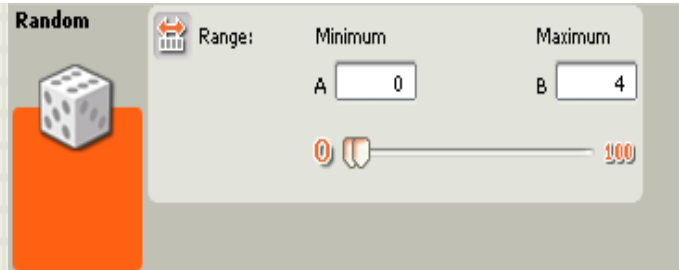
A villogó szimulálásához a színszenzor ad segítséget. Mint tudjuk lámpaként is szolgál. A programban két lámpaicont egymás mellé téve az egyiket pirosra, a másikat kékre állítjuk. Ha kipróbáljuk, felvillan a piros, utána a kék, majd a program folytatja a következő lépéssel. Ha villogót szeretnénk, akkor ismételni kell az előbbi folyamatot. Ahogy a fenti ábra mutatja, a két ikon egy ciklusban van. A feladat szerint robotunk vár két másodpercet és közben villog. Így hát állítsuk a ciklust két másodpercre és kész is.

4.c feladat

Ezen feladat megoldásához szükséges ismernünk a véletlen szám generátort. A Data fül alatt található Random ikonról van szó (51. ábra).



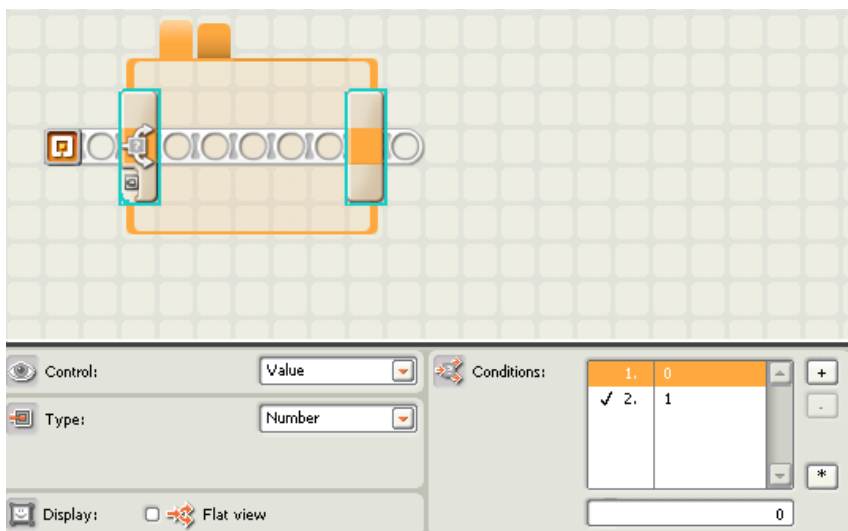
51. ábra



52. ábra

A generált szám alap esetben 0-100 terjed. Azt hogy mettől meddig generálja két csuszkával állítható be, vagy megadható számmal a Minimum A és Maximum B –ként (52. ábra).

A feladathoz használnunk kell többágú elágazást is. Az egyszerű elágazástól abban tér el, hogy itt egy bejövő értéktől függően több mint két lehetőség közül választ. Lássuk, hogy is néz ki! Kattintsunk az elágazás ikonra! A Control opciónál válasszuk a Sensor helyett a Value lehetőséget, majd a Type opciónál a Number-t és végül a Flat view- lehetőséget kapcsoljuk át, ha ez megvan valami hasonlót fogunk látni (53. ábra):



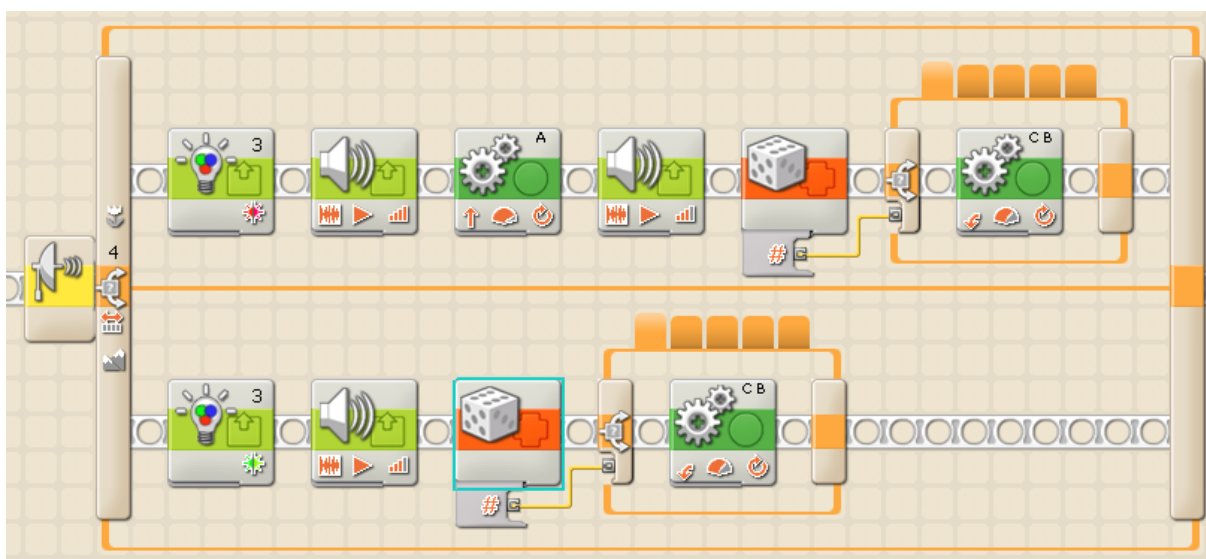
53. ábra

A Condition opciónál lévő + gombbal megadhatunk több lehetőséget. Adjunk meg 5-öt. A programban az ábrán látható fülek segítségével választhatunk közöttük.

A feladat szerint többféle fordulást kell robotunknak végeznie. Ehhez minden egyes fül alá helyezzünk be egy motor ikont, majd állítsuk be a paramétereiket különbözően (előre/hátra, jobbra/balra, erősség stb.).

Eddig van egy véletlen szám generátorunk, valamint egy több elágazású feltételünk, benne a különböző fordulási lehetőségekkel. Helyezzük a két ikont a programunkba (a Random ikont kövesse a Switch) a második feltétel alsó és felső lehetőségéhez (ahol eredetileg a forgást biztosító ikonok voltak).

Nincs más dolgunk, mint összekötni a kettőt és beállítani a Random értékeit. Az összekötéshez kattintsunk a random ikon alján látható kivezetéshez és húzzuk a vonalat az elágazás bevezetéséhez (54. ábra). Ha ezzel készek vagyunk, állítsuk be a véletlen szám generátort 0-4-ig terjedő skálán (mivel ugye 5 lehetőségünk van). Végül próbáljuk ki robotunkat!



54. ábra

Befejezés

Munkánk írása közben folyamatosan rájöttünk arra, hogy a robotok nyújtotta lehetőségek még nagyobbak, mint azt előre elgondoltuk.

Legnagyobb előnyük mindenképpen a fantáziadús kétértelmű megnyilvánulásuk. Ez óriási motivációs erőként hathat mindazon diákok számára, akik úgy érzik, hogy nem felel meg nekik többé az iskola által nyújtotta idejét múlt megoldások, vagy már belefáradtak az eddigi módszerekbe.

Feladatgyűjteményünk megalkotásával segíteni szerettünk volna mindazoknak, akik valami újra vágnak, akik a programozást egy más megközelítési módból szemlélnék szívesen. Kutatómunkánk során számos órát eltöltöttünk informatika tanszékünkön robotjainkkal. Ezen órák alatt próbáltuk ki a már eddig elsajátított programozási alapjainkat, majd ezután újabbak után kezdtünk kutatni, hiszen a végeredmény, melyet láttunk, s tapasztaltunk megérte a rá szánt időt és fáradozást. Örömmel mondhatom, elértük célunkat, és olyan információkkal illetve tapasztalatokkal gazdagodtunk, melyek előre nem voltak láthatóak. Ezeket szeretnénk megosztani másokkal. Új ismereteink fejében munkánk is bővült esetlegesen olyan dolgokkal, amivel még mi magunk sem számoltunk.

Annak ellenére, hogy a robotprogramozás mondhatni új szakterületnek számít az iskolákon belül, az internetnek hála mégis hozzáférhető számos tipp, tanács, algoritmus és egyéb segítség, mellyel könnyíthetünk a tanulási folyamaton. Ezen feladatgyűjtemény is egy ilyen forrás szeretne lenni.

Végezetül mondhatom, remek élményekben volt részünk. Reméljük, ez a szakterület megállja majd a helyét az oktatásban, s egy szép napon mindenki számára elérhető lesz a robotika.

Felhasznált irodalom:

1. KISS, R. & BADÓ, Zs. *Egyszerű robotika – A Mindstorms NXT robotok programozásának alapjai*, 2010.
2. KISS, R. & PÁSZTOR, A. *Mobil robotok programozása NXC és NXT-G nyelven*, Kecskeméti Főiskola, 2009.
3. KNUDSEN, J. B. *The Unofficial Guide to LEGO MINDSTORMS Robots*, USA: O'Reilly & Associates, Inc., 1999. ISBN: 9781565926929
4. VALK, L. *Lego Mindstorms NXT 2.0 Adventure Book*, USA: No Starch Press, 2010. ISBN: 9781593272111
5. MEADHRA, M. & STOUFFER, P. J. *Lego Mindstorms for Dummies*, USA: For Dummies, 2000. ISBN: 9780764507670
6. SATO, J. *Jin Sato's Lego Mindstorms: The Master's Technique*, USA: No Starch Press, 2002. ISBN: 9781886411562
7. BROWN, J. P. & ROLIGHED, S. & AQUILLO, M. & YABUKI H. *10 Cool Lego Mindstorm Dark Side Robots Transports and Creatures: Amazing Projects You Can Build in Under an Hour*, USA: Syngress Media Inc., 2002. ISBN: 9781931836593
8. ISOQAVA, Y. *Fantastic Contraptions*, USA: No Starch Press, 2010. ISBN: 9781593272791
9. BAUM, D. *Definitive Guide to Lego Mindstorms*, USA: Apress, 2002. ISBN: 9781590590638

Internetes források:

1. <http://pcforum.hu/szotar/algoritmus.html> (25.03.2012)
2. [http://hu.wikipedia.org/wiki/Ciklus_\(programozás\)](http://hu.wikipedia.org/wiki/Ciklus_(programozás)) (25.03.2012)