

Lego robot programozása

LEGO MINDSTORMS NXT

Készítette: Holló Tamás
Batsányi János Gimnázium
Tapolca

Tartalomjegyzék:

1. Lecke	3
HELLÓ VILÁG!.....	3
Gyakorló feladatok:.....	9
2. Lecke	10
MOZGASD MEG A ROBOTOT!.....	10
Gyakorló feladatok:.....	11
3. Lecke	14
CSINÁLJ ZAJT!.....	14
Gyakorló feladatok:.....	16
4. Lecke	17
ÉRZÉKELD A KÖRNYZETED!.....	17
Gyakorló feladatok.....	23
5. Lecke	24
Most már bekövetkezett!.....	24
Gyakorló feladatok.....	28
6. Lecke	29
ISMÉTELD AZ UTASÍTÁST!	29
Általában a ciklusokról.....	32
Gyakorló feladatok.....	36
7. Lecke	38
Merre induljak? Melyiket válasszam?.....	38
Általában az elágazásokról.....	42
Gyakorló feladatok.....	42
8. Lecke	43
Tárolj adatokat!.....	43
Lámpa blokk	46
Véletlen szám előállítása	47
Saját blokk készítése	50
Gyakorló feladatok.....	56
9. Összetett feladatok	57
Bibliográfia.....	60

Lego robot programozása

Jelmagyarázat:

☞ Megjegyzés

☞ Feladat

1. Lecke

Ez egy hagyományos első feladat, a világ összes programnyelveinek első feladata. A visszajelzés, vagyis a program futás eredményének kiírása a legfontosabb. Az angol programozási környezetekben (angol a programozás nemzetközi nyelve) ez a „Hello World” szöveg kiírásával indul.

HELLÓ VILÁG!

DISPLAY BLOKK HASZNÁLATA

1.1. ☞ Írnod ki a NXT téglán LCD kijelzőjére a „Helló világ!” szöveget.

☞ Az idéző jelek a későbbiekben azt jelzik, hogy egy olyan szövegről van szó a feladatban, amit változtatás nélkül kell kiírni, átadni vagy a feladatban szerepeltetni. Az idézőjeleket nem kell a szöveg részének tekinteni. Ez az adattípusok közül a szöveges adat.

Indítsd el a LEGO MINDSTORMS robot programozásához használható egyik lehetséges programot.

☞ Ez egy magas szintű, a LabView által fejlesztett, grafikus felületű objektumorientált programozási felület, amelyet elsősorban gyerekek számára fejlesztettek ki.



Add a „hello” nevet az új programnak!

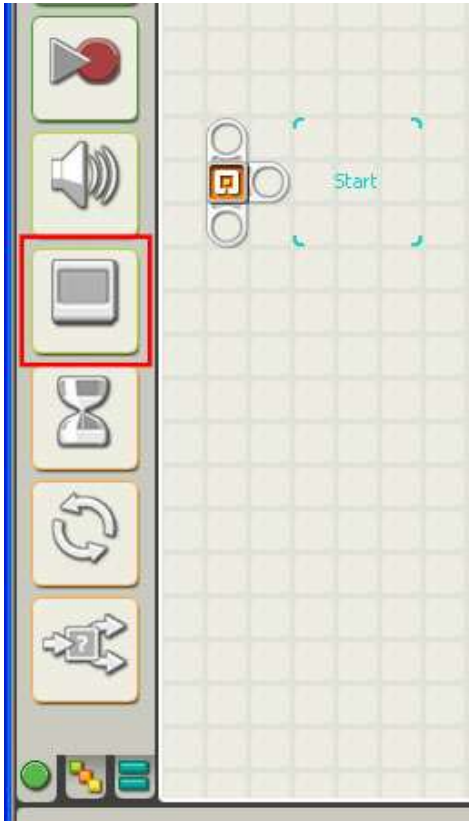


Mentsd el a forrás programot.

A File → Save As (Mentés másként) menüt használva mentsd el az egyelőre látszólag üres forrásprogramot. A Browse (Tallózás) gomb használatával keresd meg a HOME könyvtáradat és a Save (Mentés) gomb segítségével pedig mentsd el hello.rbt néven a forrásprogramot.



A Common paletta kimeneti utasításblokkjai közül válaszd a Display utasításblokkot.



☞ A Common paletta ikonjai fölé mozgatott egérkurzor segítségével választhatsz blokkot (utasítást). Ha egy ikon blokkcsoportot jelöl, akkor a csoport kinyílik, és egy bal egérgombba kattintással kiválaszthatod a blokkot.

Helyezd a Display eszköz ikonját a Start pozícióra.



☞ Mozgasd a Start felirat fölé és kattints az egér bal gombjával.

Változtasd meg a Display blokk paramétereit a feladatnak megfelelően.



☞ A szerkesztő felületen a Display blokknak kijelöltnek kell lennie, hogy lásd a tulajdonság panelt.

Válaszd ki a működési lehetőségek (Action) közül Text működést.



Írd a „Helló Világ!” szöveget a Text beviteli mezőbe!



☞ Remélem feltűnt, hogy az ékezetes betűk nem jelentek meg a minta kijelzőn. Erre két megoldás is van! 😊

Ha meggondoltad a megoldást, akkor folytasd a feladatot.

Változtasd meg a kiírt szöveg helyzetét az X: 12 Y: 8.



☞ A szöveg, kép, vonal pozíciójának kezdőpozíciója (origó) a X:0 Y:0 a bal alsó sarok.

Mentsd el az elkészített forrásprogramot.

File → Save menü (CTRL + S)

☞ Ez elegendő, hiszen a nevét és mentés helyét a már meghatároztad a feladat megoldásának legegyszerűsége.

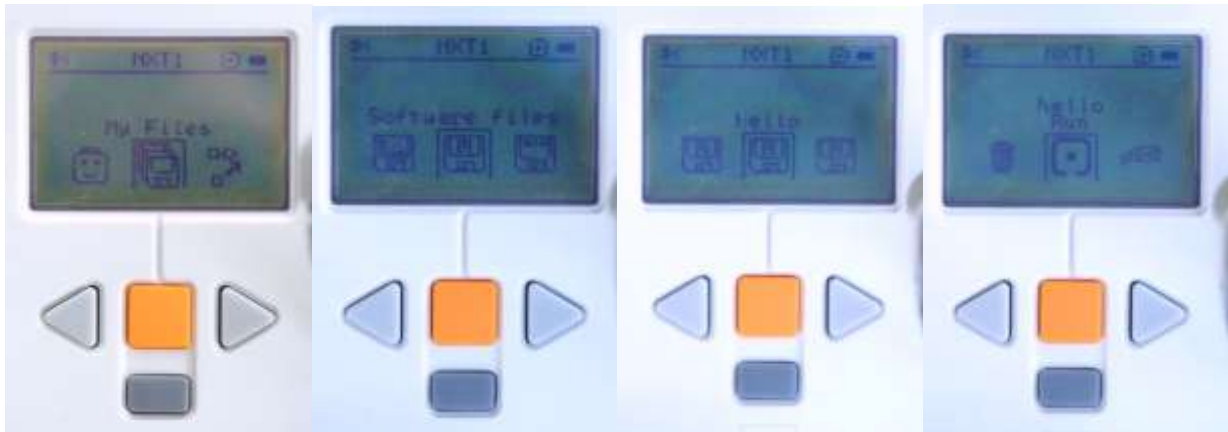
Fordítsd le és töltsd fel a robotra.



☞ Ellenőrizd a számítógép össze van-e kötve a robottal. Az Inicializing..., Compiling..., Downloading... és Complete! Az előkészítés (inicializáció) során a fordító szoftver ellenőrzi a kapcsolatot a számítógép és a robot között, fordítás (compile) – a gép által értelmezhető nyelvre fordítja a forrásprogramodat, letöltés (download) – igazából áttöltés a robot memóriájába és végül a kész (complete) folyamatok zajlanak a gombra kattintás után.



Válaszd le a számítógépről a robotot és indítsd el a programot.



☞ A robot menüjében a háromszög alakú gombok a lapozást szolgálják. A narancssárga gomb a kiválasztás, az alsó szürke a visszalépés.

Az első csalódottság után vond le tanulságot. Valami nem stimmel. Többszöri programindítás után vedd észre, hogy valami történik csak olyan gyorsan, hogy nem értelmezhető. Sejtsd meg, hogy a program jól működik, csak olyan gyorsan lefut, hogy emberi léptékkal nem érzékelhető.

Tedd fel a kérdést, akkor most mi történjen, hogy lásd a kijelzőn a feliratot?

A válasz a kiírás után rá kell venni a robotot, hogy még ne hagyja abba a program futását.

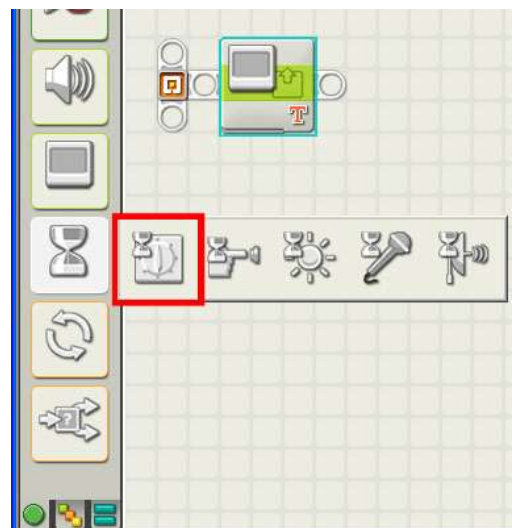
Milyen lehetőségek állnak rendelkezésedre?

Találkoztál már valamilyen megoldással más programokban, Interneten, játékoknál?

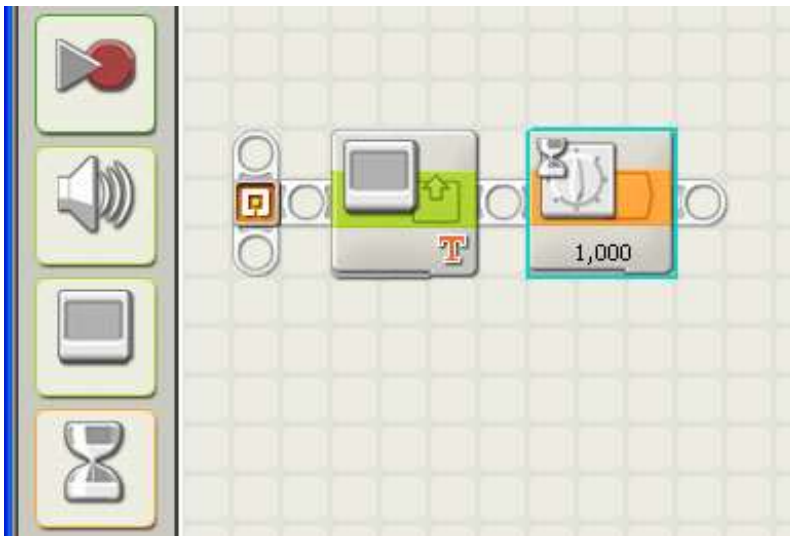
Bizonyára több megoldás eszedbe jutott, de most csak egy egyszerű, a kiírás utáni várakozással old meg a problémát.

Várakozás az eredményre

Válaszd ki a Common palette Várakozás blokkcsoportja közül a Time blokkot.



Helyezd el a Display blokk után.



A várakozási idő tulajdonságát pedig állítsd be 3 másodpercre.



Mentsd el a változásokat.

Töltsd fel a robotra.

Futtasd a programot.

A tapasztalat az, hogy most 3 másodpercig látszik a felirat és utána leáll a program futása.



Display blokk beállításai

Action (működési lehetőségek):

Image: Kép megjelenítése a kijelzőn. A kép .ric (raster image file) kiterjesztésű állomány.

☞ Ha nem elegendő a felkínált kép mennyiség és saját képet szeretnél, akkor találsz az interneten szerkesztő programot, ☺



Display: Clear kijelölt állapotban a művelet végrehajtása előtt töröli a kijelzöt.

File: C:\Program Files\LEGO Software\LEGO MINDSTORMS Edu NXT\engine\Pictures könyvtárban található RIC kiterjesztésű képállományokból készült lista.

Position: A kép helyzete a kijelzőn. Kiinduló pont $X = 0$ és $Y = 0$ a kijelző bal alsó sarka a képernyő mérete, pedig 100×64 képpont.

Text: Egyszerű szövegek, számok kijelzése.

☞ Az első lecke rámutatott, hogy csak az angol abc betűit használhatod. Viszont nagyon hasznos a mérési eredmények visszajelzése miatt.



Display: Clear kijelölt állapotban a művelet végrehajtása előtt töröli a kijelzőt.

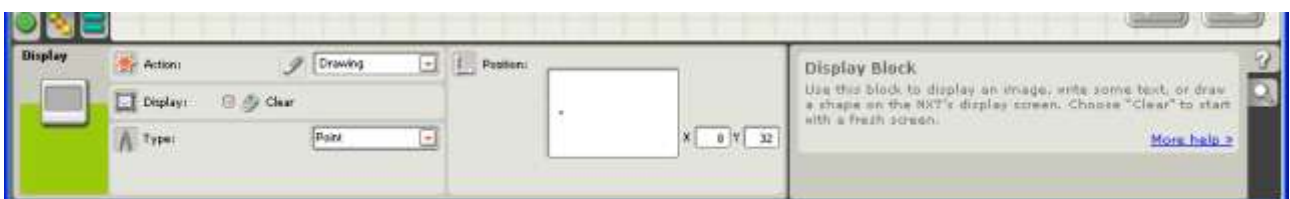
Text: a képernyőre kiírandó szöveg.

Position: A kép helyzete a kijelzőn. Kiinduló pont $X = 0$ és $Y = 0$ a kijelző bal alsó sarka a képernyő mérete, pedig 100×64 képpont.

Line: A sor tulajdonsággal 1-8. sorig lehet pozícionálni a a kiírt szöveg helyzetét.

Drawing: Egyszerű rajzi elemekből pont (point), vonal (line) és kör (circle)

☞ Egyszerű ábra, vagy garfikon rajzolás, például bejárt útvonal egyszerű térképe, vagy mérési eredmények diagramon való ábrázolása a lehetséges alkalmazása.



Display: Clear kijelölt állapotban a művelet végrehajtása előtt töröli a kijelzőt.

Type:

Point: Egyetlen képpont rajzolása

Position: X és Y koordinátákkal a pont helyének meghatározása

Line: Vonallal rajzolás

Position: X és Y kezdő koordinátákkal.

End point: X és Y vég koordinátával.

Circle: Kör rajzolása

Position: X és Y a kör közepének koordinátája.

Radius: a kör sugara képpontban megadva.

Összefoglaló táblázat a Display blokk beállításairól:

Action	Image	Text	Drawing
File	✓	x	x
Position	X: 0, Y:0 bal alsó sarok		
Line	x	✓ (1-8 sor)	x
Type	x	x	Point (X,Y)
Type	x	x	Line (kezdő: X, Y - vég: X, Y)
Type	x	x	Circle (középpont: X, Y sugár: hossz)
Display Clear	✓	✓	✓

Gyakorló feladatok:

A feladatok megoldásakor ügyeljenek a csapatmunkára. Minden feladat végén cseréljenek szerepet. Az első feladatokban ez nem lesz túl szórakoztató, hiszen a feladatok egyszerűek, így a csapatmunka előnyeinek nyilvánvalóan elő. Ám hosszútávon a feladatok nehezedésével egyre nagyobb lesz az egymásra utaltságotok. Legyetek türelmesek egymással, és igyekezzetek a saját gondolataitokat világosan megfogalmazni, mert nem csak a társaitokon segítesz, hanem a te tudásod is alaposabb és biztosabb lesz. **Docendo discimus.** (Tanítva tanulunk)

1.2. ☞ **Jeleníts meg 5 másodpercig egy bombát a kijelző közepén. (bomb.ric)**

1.3. ☞ **Rajzolj egy kört a kijelző közepére.**

1.4. ☞ **Rajzolj egy olyan kört a kijelző közepére, amely érinti legalább két oldalát a kijelzőnek.**

1.5. ☞ **Jelenítsd meg a kijelző közepén egymás után 1 másodpercenként a következő helyzetű vonalakat:**

1.6. ☞ **Rajzolj egy 30 x 50 képpont oldalhosszúságú téglalapot!**

1.7. ☞ **Rajzolj egy kb. 50 képpont oldalhosszúságú egyenlőszárú háromszöget!**

1.8. ☞ **Írj szöveget a Talk 01 szövegbuborékba!**

2. Lecke

MOZGASD MEG A ROBOTOT!

A legérdekesebb tűnő feladat, ha a robotot mozgásra készítjük. Menj oda, fordulj meg mozgasd a karod, fogd meg, integess nekem, vagyis mozdulj meg.

MOZGÁS BLOKK PROGRAMOZÁSA

A központ egységhez (tégla) 3 porton (A, B és C) keresztül lehet vezérelni a szervó motorokat. A motorok három legfontosabb paramétere a sebesség (power) és a működtetés ideje (duration) és a forgás iránya. Az alapbeállítás szerint a B és C porton keresztül 75-ös sebességgel 1 fordulatot tesz a két motor előre.

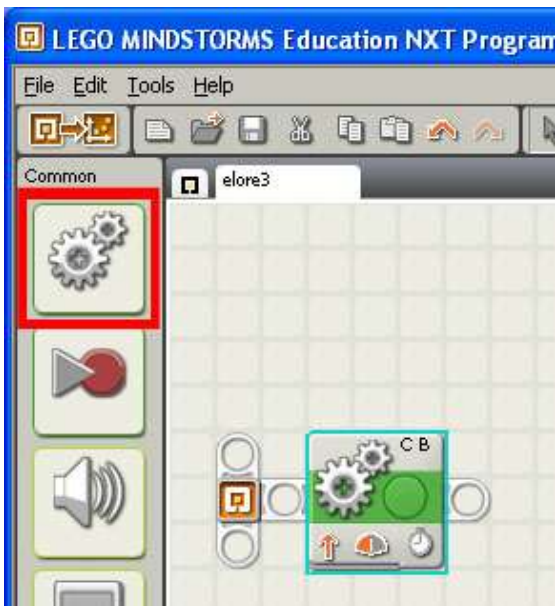
2.1. 🗨 Haladjon előre 3 másodpercig 50-es sebességgel a robot!

Hozz létre egy új forrásállományt.

File → New menüpont (CTRL + N)

Mentsd el a megbeszélt helyre a elore3.rbt néven.

Mozgasd a motorblokkot a forrásprogramban a megfelelő helyre



Állítsd be a megfelelő tulajdonságokat, amelyek a motorok mozgását meghatározzák.



Ellenőrizd le, hogy a Port (adatkapu) a B és C van-e kijelölve és a roboton a is ezekben aljzatokban van-e a vezeték. A Power-t állítsd a csúszka segítségével.

vel vagy beírással 50-re. Majd a Duration listából válaszd a Seconds működési időtartamot és állítsd be 3-ra.

Port: a B és C portra csatlakoztatott motor vezérlése az alapértelmezett.

☞ Ellenőrizd a motor portjait!

☞ Ha 1 vagy 3 portot jelölsz ki egyszerre, akkor nem minden beállítási lehetőség elérhető.

Direction: a forgásirány beállítása, vagy a motor leállítása.

Steering: kormányzás.

☞ A beállítással a motorok sebességének elosztásával a kormányzást lehet megoldani. A csúszka mozgatásával a két motor között megosztva az erőt, fordulásra késztesd a robotot. A teljesen oldalra húzott csúszka esetén a motorok ellentétes irányba kezdenek forogni, így helyben a közeptengelye körül fordul el a robot.

Power: a motor forgási sebessége. 0 – 100-ig állítható.

☞ Ha feladat másképpen nem rendelkezik, állítsd mindig 50-re a sebességet.

Duration: a mozgás időtartamát állíthatod be.

Unlimited: korlátlan

☞ Ennek akkor lesz jelentősége, amikor már érzékelőkkel (sensor) együtt programozzuk a motort.

Degrees: az elfordulás szöge.

☞ 0° nincs fordulás, 360° egy körbefordulás és 720° pedig a két körbefordulás.

Rotations: körbefordulások száma:

Seconds: időtartam

☞ Először mindig a működés módját és utána a mértékét állítsd be!

Next Action: a Duration (időtartam) lejártá után hogyan vieszkedik.

Brake: a motort befékezi, sebességét 0-ra csökkenti.

Coast: a robot lendületének elfogyása után áll csak le, szabadon fut.

Gyakorló feladatok:

2.2. ☞ **Mozgasd úgy a robotot, hogy a mozgása az ABC valamely betűjét utánozza.**

Egyszerű betűk: C, O, N, Z, M, D
Bonyolultabb betűk: T, A, B, E, S

3. Lecke

CSINÁLJ ZAJT!

SOUND BLOKK PROGRAMOZÁSA

A téglá képes hang kibocsátásra és kisebb zenei (a memória mérete a korlát) állományok lejátszására. Ezáltal a display blokkhoz hasonlóan nem csak vizuális, hanem akusztikus figyelmeztetésekre, jelzésekre. R2-D2.

Két alapelethezésre számíthatsz, az egyik hang lejátszása, ahol a hangmaga, lejátszás ideje és hangereje a beállítható paraméter. A másik a hang állomány lejátszása, ahol a listából kiválasztott hang állomány és annak hangereje a beállítási lehetőség.

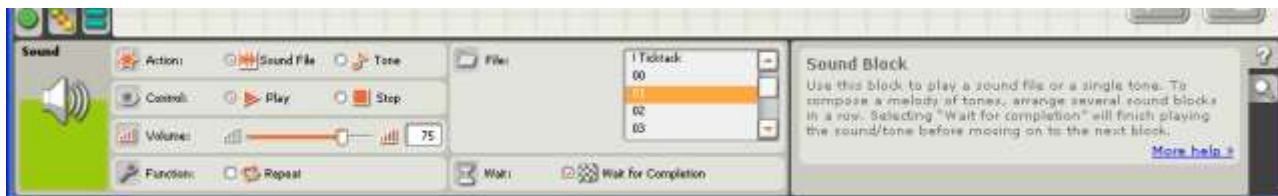


3.1. ☞ Háromig számoljon el a robot angolul hangosan és utána induljon egyenesen előre 3 kerékfordulatot, majd amikor megállt mondja, hogy stop.

Hozz létre egy új üres forrásprogramot.

Mentsd el hang.rbt néven megfelelő könyvtárba!

Mozgasd a Sound blokkot a kezdő pozícióba, és állítsd be a paramétereit úgy, hogy az „one” szót játssza le.



Az **Action** legyen az egyébként is alapértelmezett **Sound File** válaszd, a **Control** lehetőségei közül az alapértelmezett a **Play**. A hangerő (**Volume**) maradjon 75. A **File** listából válaszd a 01 nevűt, a **Wait for Completion** jelölőnégyzet legyen bejelölve.

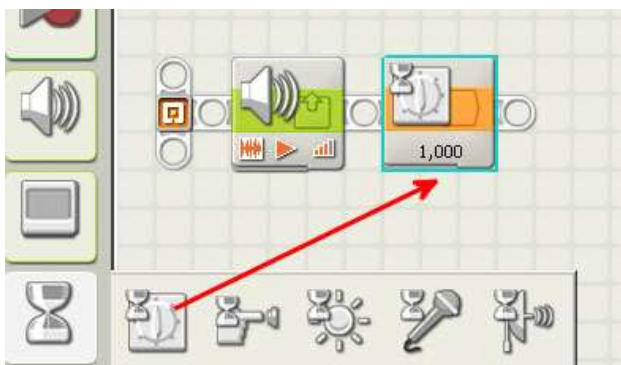
Helyezz el egy várakozás blokkot a Sound Blokk után.

Állítsd a 1 másodpercre a várakozási időt.

Mozgass még egy Sound blokkot a szerkesztő területre. Az előző blokk után helyezd el.

Állítsd be a paramétereit ugyan úgy, mint az előbb, csak a File a 02 legyen, amit a listából kiválasztasz.

Helyezz el egy várakozás blokkot a Sound Blokk után.



Állítsd a 1 másodpercre a várakozási időt.

Mozgass még egy Sound blokkot a szerkesztő területre. Az előző blokk után helyezd el.

Állítsd be a paramétereit ugyan úgy, mint az előbb, csak a File a 03 legyen, amit a listából kiválasztasz.



Helyezd el a mozgatás blokkot az utolsó hang blokk után.

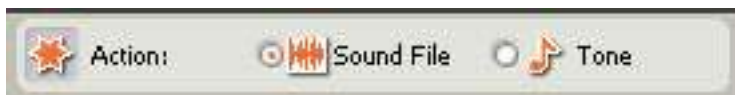
Helyezz el még egy Sound blokkot, amelynél a File a Stop nevű.

Ments (CTRL + S)

Töltsd át programot a robotra és indítsd el.

SOUND BLOKK PARAMÉTEREI

Action: File vagy egyetlen hang lejátszása.



Sound File: a \Program Files\LEGO Software\LEGO MINDSTORMS Edu NXT\engine\Sounds könyvtárban található .rso állományok közül játszhat le egy modullal egyet.

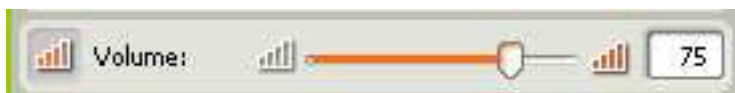
Tone: Egyetlen hang lejátszása, a beállításban majd a hang magasságát és a megszólalás hosszát határozhatod meg.

Control: A hang lejátszása vagy elhallgattatása.

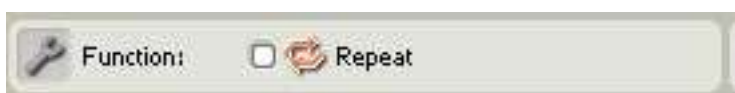


☞ Alapértelmezetten a Play-t gondolnánk mindig megfelelőnek, hiszen a hang állomány is rövid és egyetlen hangot sem játszunk le általában a végtelenségig. Mégis szükség lehet, hogy egy hangot, amely éppen szól elhallgattatni. Egyszerre egy hang szólaltatható meg.

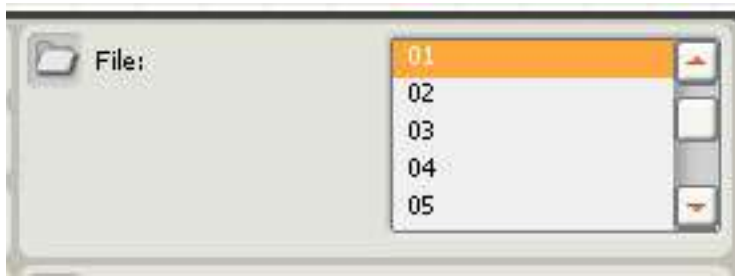
Volume: a hangerő, amely 0 – 100 között állítható.



Repeat: a hang vagy hangállomány ismétlése a megállításig, tehát addig míg a Control: Stop Sound blokk utasítás kiadásra nem kerül.



File: Ha Sound File action választottad, akkor egy állomány listát látsz, amely neve utal a kimondott (angolul) vagy megszólaló hangállomány tartalmára.



Tone: Egy zongorabillentyűn kiválaszthatod a megszólaltatni kívánt hangot, rögtön beállíthatod, hány másodpercig szóljon a hang. Az alapérték az A hang 0,5 másodpercig.



Wait for Completion: ha bejelölt, akkor következő utasítás csak akkor hajtódik végre, ha a hang vagy hangállomány már elhallgatott.

Gyakorló feladatok:

- 3.2. ☞ **A mozgás blokk programját írd át úgy, hogy jelezze az indulás és megállást hanggal, valamint minden fordulás előtt mondja meg a robot merre fog fordulni.**
- 3.3. ☞ **Játsszon le a robot valamilyen egyszerű dallamot (pl: boci, boci ...)**
- 3.4. ☞ **Mozogjon úgy és adjon olyan hangokat, mint R2-D2.**

4. Lecke

ÉRZÉKELD A KÖRNYZETED!


A robothoz számos sensor (érzékelő) tartozik, amelyek a környezetben való tájékozódást segítik. Az érzékelőkkel mért adatok alapján dönthet a robot arról, hogyan folytatja egy feladat megoldását, merre kell haladni a küldetésben, és meg vannak-e egy feladat végrehajtásához szükséges feltételek.

Az alapkészlet tartalmaz távolságérzékelőt, fény- és hangérzékelőt, és két darab érintésérzékelőt is.

Először ismerkedjünk meg az érintésérzékelő (touch sensor) működésével.

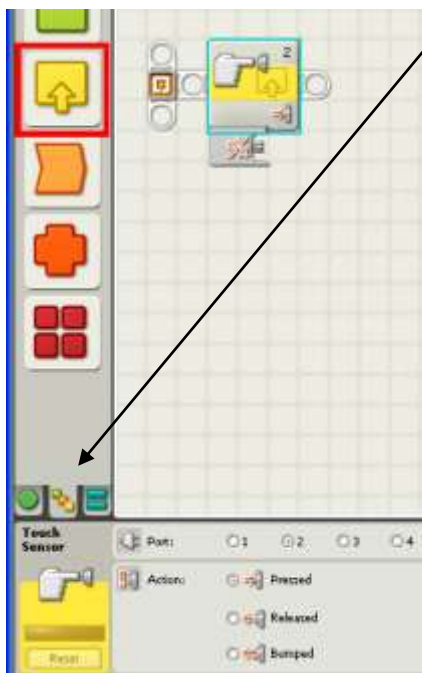
TOUCH SENSOR HASZNÁLATA

4.1. **Az érintésérzékelő állapotától függően induljon a robot előre vagy hátra 3 másodpercig, majd egy rövid 1 mp-es felső C hang kiadásával álljon meg!**

 A fal mellett álló robot érintésérzékelője benyomott (pressed) állapotban van ezért „logikus”, hogy ne hátra, hanem előre induljon el.


A feladat látszólag igen egyszerű, de arra jó, hogy ne csak az érzékelő állapotát figyelhesd meg, hanem azt is, hogyan adhatsz át adatot az érzékelőtől a motornak.

Legyen neve a forrásprogramnak (nyomas.rbt) és mentsd is el a megfelelő helyre.



Váltsd át az utasításkészletet „Complete palette”-re, majd az érzékelők közül válaszd a Touch sensor-t.

A tulajdonságait állítsd be.

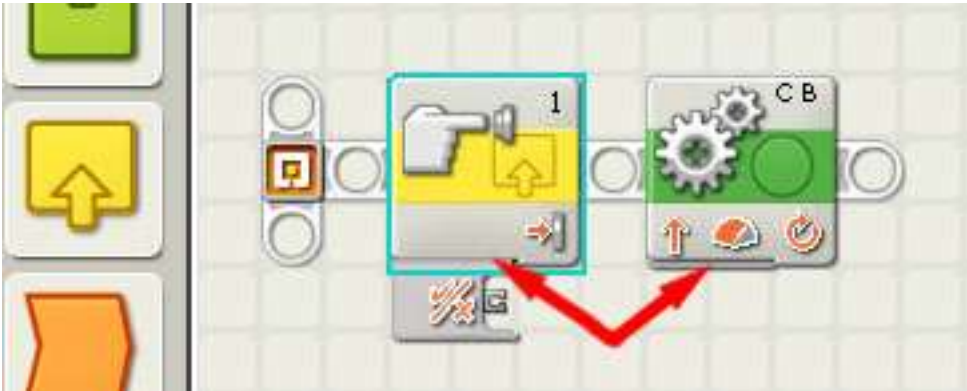
 A Portot ellenőrizd le, hogy a roboton hová van dugva az érintésérzékelő kábele. Az Action hagyd az alapértelmezett Pressed állapoton.

A Common készletből helyezd el a Move blokkot a Touch sensor után.

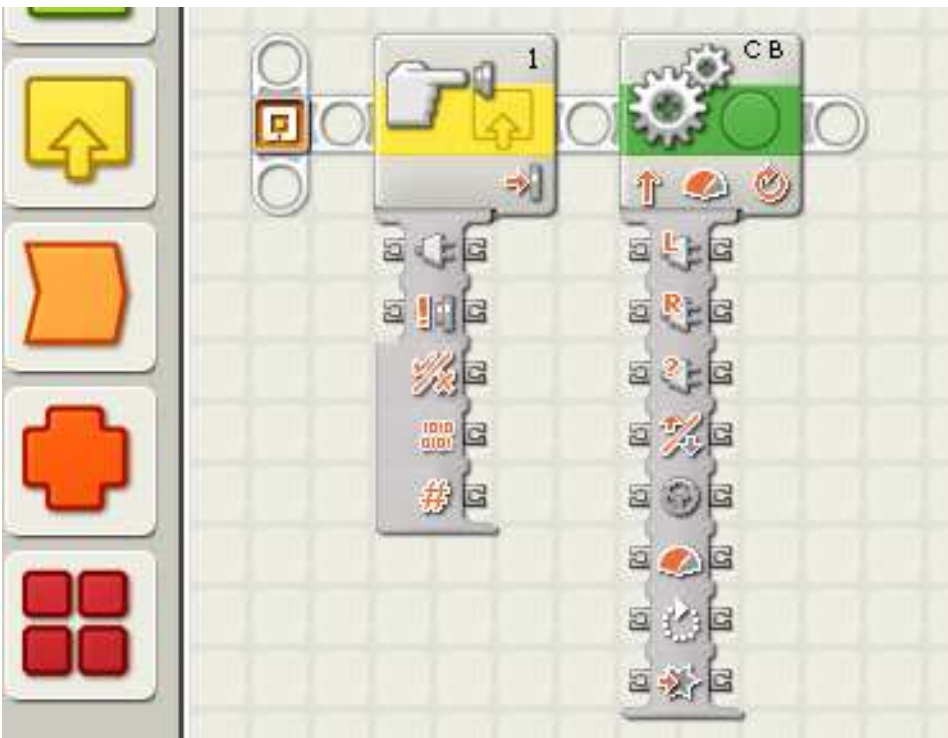


Állítsd be a tulajdonságait feladatnak megfelelően.

Nyisd le a csatlakozó készletét a Touch Sensor-nak és a Move blokknak is.



☞ Kattints blokk alsó részén látható vízszintes vonalkára



☞ Amit itt látsz azok a két blokk paraméterei (Data Hubs - adatkapcsolat), amelyek alapállapotát eddig a tulajdonság panelen te is meghatározhatad. Ám amikor a program lefordításra kerül és elindítod a végrehajtását a roboton, akkor már nem tudsz rajta változtatni.

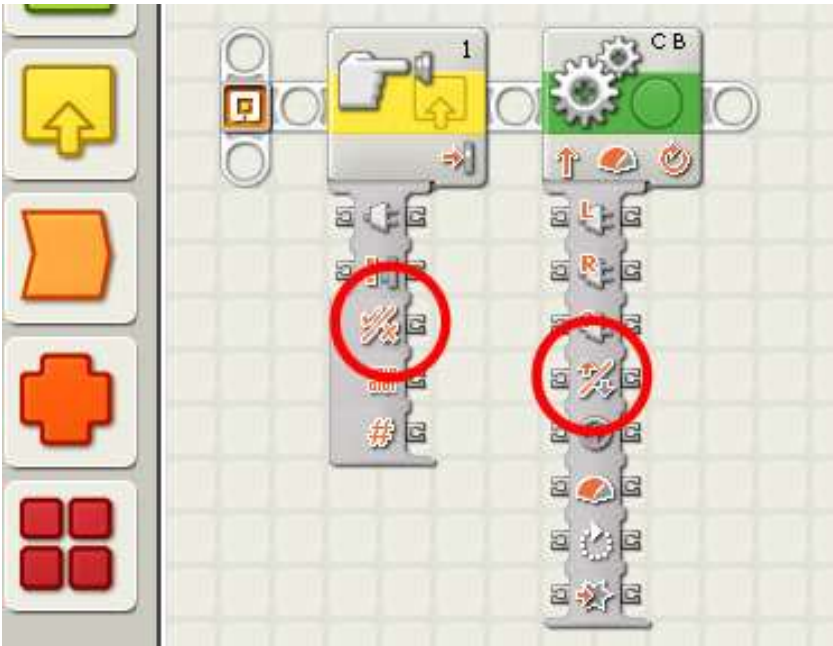
A feladatban az a kihívás, hogy a kerekek forgási irányát a program indítása után kell meghatároznod, attól függően, hogy az érintés érzékelő éppen a megnyomott vagy kiengedett állapotban van-e.

Igen vagy nem – LOGIKAI ADAT TÍPUS

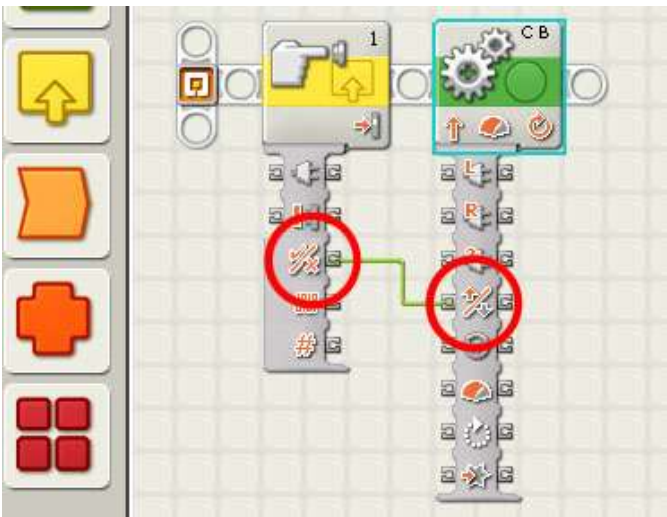
Ezek a kapcsolók tulajdonképpen adatok, amelyek legfontosabb tulajdonságuk a típusuk. Esetünkben egy logikai típus, amely az adattípusok közül a leg-

kiseb és legegyszerűbb adat. Két állapota van: igen vagy nem (yes or no). Mint a nyomógomb.

Tehát a nyomógomb állapotát kell átadni a move blokk irányparaméterének (Direction).



Kösd össze a Touch blokk yes/no kapcsolóját a Move blokk direction kapcsolójával.



☞ A bal egérgombbal kattints a yes/no plug-ra (konnektor, dugó), majd a direction konnektorra. A zöld szín esetünkben azt jelzi, hogy az adatkapcsolat megfelelő, tehát olyan konnektorokat kötöttünk össze, amelyek adattípusa megfelel egymásnak.

Mentsd el és töltsd át a programot a robotra.

TESZTELÉS

Eleddig a programok vagy működtek vagy nem, vagy azt csinálták amit kértünk a robottól indítás után vagy nem. Most azonban az indítás után két különböző viselkedést várunk a robottól.

- 1. eset:** A robot a falnál áll és az érintés érzékelő benyomott állapotban.
- 2. eset:** A robot ácsorog valahol a padlón, az érzékelő nincs benyomva.

Neked mind két esetet meg kell vizsgálnod és ki kell próbálnod!

MIT ÉRZÉKELHETEK MÉG?

Minden érzékelőnek van egy Yes/No konnektora (plug) az adatkapcsolati panelen (data hubs).

Azt is megfigyelheted, hogy a konnektorok két csoportba oszthatók. Az egyik csoport az, amelyik input (bemenet) és output (kimenet) egyszerre. Adatok esetében szerencsésebb, ha írható és olvasható adatról beszélünk, még a másik csoport a csak olvasható adat.



Ez egy írható és olvasható adat (tulajdonság).

Írható: a bemeneten keresztül a motor forgásirányát határozhatod meg.

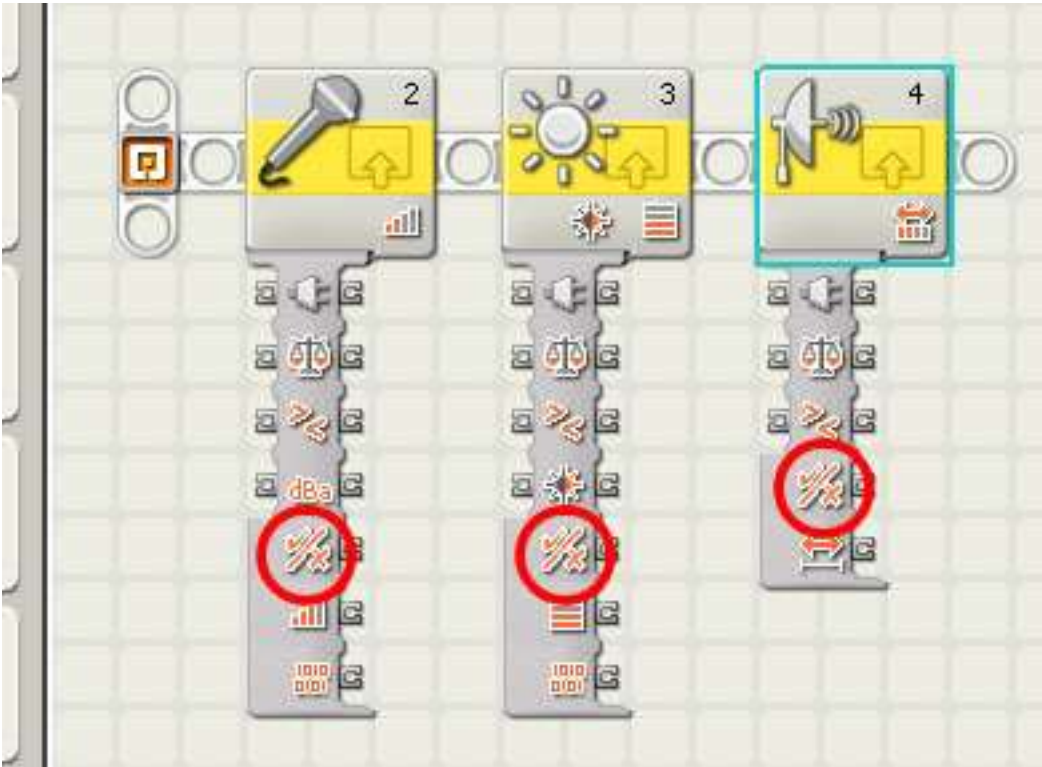
Olvasható: meg tudod kérdezni a motortól, hogy éppen melyik irányba fog forogni.

Tehát a külső vagy belső körülmények (érezelők állapota, program végrehajtás helyzete), alapján tudod befolyásolni, hogy melyik legyen a forgásirány. Illetve a motor forgási irányától függően tudod meghatározni a program további haladását.



Ez egy csak olvasható adat (tulajdonság).

Az olvasható ki belőle, hogy a beállított akció bekövetkezett-e, vagy más érzékelőknél, hogy elérted a megadott távolságot stb.

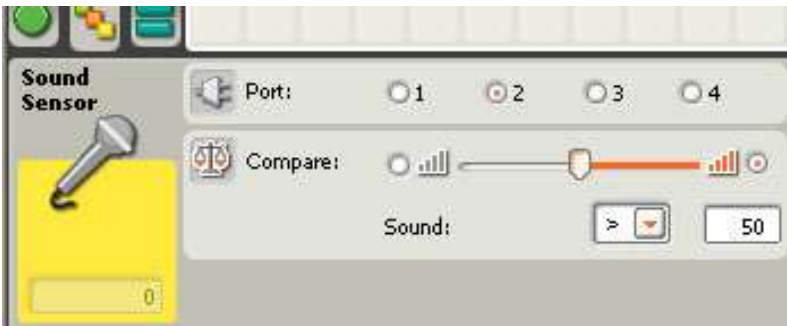


Három érzékelőt látsz a képen. Balról jobbra haladva hangérezkelő (Sound Sensor), fényérezkelő (Light Sensor) és Távolságérezkelő (Ultrasonic Sensor).

A teljes igény nélkül haladunk, hiszen ha Sensor csoportot kinyitod, választhatsz még egyéb érzékelők közül, amelyek később kerülnek ismertetésre.

SOUND SENSOR

A neve is mutatja a hangerő érzékelésére használható, amelyet egy 0-100-ig terjedő skálán értékelhetünk ki.



Port: melyik portra (kapu) van csatlakoztatva az eszköz.

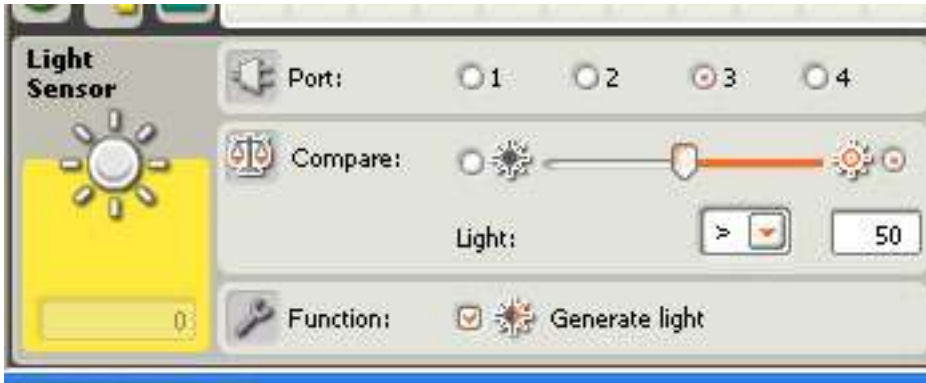
☞ Mindig ellenőrizd a roboton, még akkor is, ha biztos vagy benne, mert annál kínosabb nincs, mint hogy programhibát keresel perceként keresztül és csak egyszerűen nem abban a csatlakozó aljzatban van a vezeték, amely portot beállítottál a programban.

Compare: az érzékelő által mért értéket összehasonlíthatod egy általad beállított értékkel.

☞ Ez egy reláció, amely IGAZ vagy HAMIS (TRUE or FALSE) lehet. Az előző feladat megoldásakor ezt az értéket használtuk, az előre vagy hátrame-
nethez.

LIGHT SENSOR

Ezzel az érzékelővel a világosság mértékét (tónus) érzékeljük, egy 0-100-ig terjedő skálára vetítve, ahol a nincs fény 0 (fekete) a nagyon rávilágítok pedig a 100 (fehér). Bizonyos mértékig színek megkülönböztetésére is használhatjuk, hiszen a kék általában sötétebb, mint a piros.



Port: melyik portra (kapu) van csatlakoztatva az eszköz.

☞ Mindig ellenőrizd a roboton, még akkor is, ha biztos vagy benne, mert annál kínosabb nincs, mint hogy programhibát keresel perceként keresztül és csak egyszerűen nem abban a csatlakozó aljzatban van a vezeték, amely portot beállítottál a programban.

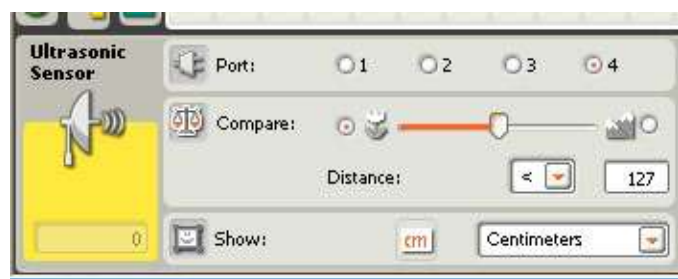
Compare: az érzékelő által mért értéket összehasonlíthatod egy általad beállított értékkel.

Function: Az érzékelő saját vörös fényű lámpával van felszerelve. Ezzel a kapcsolóval be és ki kapcsolhatod az érzékelő saját fényét.

☞ Azért van rá szükség, mert a robot különböző fényviszonyok között dolgozik, és ha környezeti hatásokat a legkisebbre akarjuk mérsékelni, akkor a saját fény közel azonos mérési eredményeket produkál délelőtt, délután és este.

ULTRASONIC SENSOR

A távolságérzékelés a feladata. Meg tudja mérni kb. 0 cm-től 2,5 méterig a tárgyak távolságát. Persze a mérés pontossága nem túl megbízható, de gyakorlásra és a programozási elvek elsajátítására tökéletes.



Port: melyik portra (kapu) van csatlakoztatva az eszköz.

☞ Mindig ellenőrizd a roboton, még akkor is, ha biztos vagy benne, mert annál kínosabb nincs, mint hogy programhibát keresel percekig keresztül és csak egyszerően nem abban a csatlakozó aljzatban van a vezeték, amely portot beállítottál a programban.

Compare: összehasonlítja egy általad beállított értékkel a mérés eredményét és a yes/no konnektorra a reláció eredményét küldi (IGAZ vagy HAMIS).

Show: beállíthatod, hogy centiméterben vagy inch-ben akarod a mérés eredményét.

☞ Vigyázz, ha nem a mértékegységet nem állítod be, akkor előfordulhat, hogy az általad beállított 10 centiméternek gondolt távolság 10 inch, ami 25,4 cm.

A következő gyakorló feladatokban a minta feladathoz hasonló programokat kell írnod. Egyszerően egy előre beállított értéket összehasonlítasz az érzékelő mérési eredményével és az összehasonlítás eredményként kapott IGAZ vagy HAMIS érték alapján a robot cselekszik. Mindig a programindítás előtt kell beállítanod a körülményeket.

Gyakorló feladatok

- 4.2. ☞ **A robot haladjon előre 2 másodpercig, ha egy fehér papíron áll és hátra, ha nem.**
- 4.3. ☞ **Ha a robot 50 centiméternél közelebb a falhoz, akkor tolasson hátra 1 kerékfordulatnyit.**
- 4.4. ☞ **Készíts két feliratot, az egyikben zaj van, a másikon a csend van szöveggel. A robot annak a papírnak az irányába induljon, amelyik a környezetben tapasztalt zajnak jobban megfelel.**

5. Lecke

Ebben a leckében a feladat már elég összetett ahhoz, hogy szót ejtsünk a programozás (probléma megoldás) egyik fontos és elengedhetetlen lépéséről az algoritmizálásról.

Algoritmus

Utastássorozat, mely megadja egy feladat megoldásmenetének pontos leírását

- véges sok utasítást tartalmaz
- nem feltétlenül véges végrehajtási idejű
- megfelelő sorrendű (szemantikailag helyes)
- utasításonként megfelelően paraméterezett (szintaktikailag helyes)

Most már bekövetkezett!

WAIT BLOKK

5.1. ☞ **A robot küldetése az legyen, hogy addig haladjon, amíg akadályba nem ütközik.**

El kell dönteni, hogy ezt az általánosan megfogalmazott feladatot, hogyan lehet konkretizálni, magyarul pontosan és formálisan meghatározni.

Nincs meghatározva a feladatban:

- Melyik irányba induljon?
- Mekkora legyen a robot sebessége?
- Melyik érzékelőt használja az akadály érzékeléséhez?
- Mi számít akadálynak?
- Mi történjen, amikor elérte a robot az akadályt?

A programozásnál általában mindig nagyon általánosan határozzák meg a feladatot. Aki számítógépet (robotot) akar használni egy feladat megoldásához és programozóhoz fordul, hogy készítsen egy programot, amely az adott feladatot elvégzi, nem tudja, hogy milyen lépésekre (utasításokra) kell bontani a feladatot a megoldás érdekében.

Ha csapatban dolgoztok, osszátok négy részre a feladatot:

Értelmező, Tervező, Operátor, Főnök

Értelmezd a feladatot! (Értelmező)

A kérdésekre adott válaszok, sokszor egymástól is függenek.

Tehát az indulási irány attól függ, hogy melyik érzékelőt választod. Az érintés és a távolságérzékelő jöhet legegyszerűbben szóba. Válaszd elsőnek az érintés érzékelőt.

Akadálynak az számít, amit nem tud eltolni, amikor nekiütközik és átmenni se tud rajta a robot.

A sok lehetséges viselkedés az akadállyal való találkozáskor - például megáll, másik irányba indul, megáll és hangot ad - közül egyelőre csak álljon meg.

A feladat pontos meghatározása:

A robot 50-es sebességgel haladjon az érintésérzékelője irányába egyenesen addig, amíg akadályba nem ütközik, ekkor álljon meg.

Tervezzük meg a programot! (Programtervező)

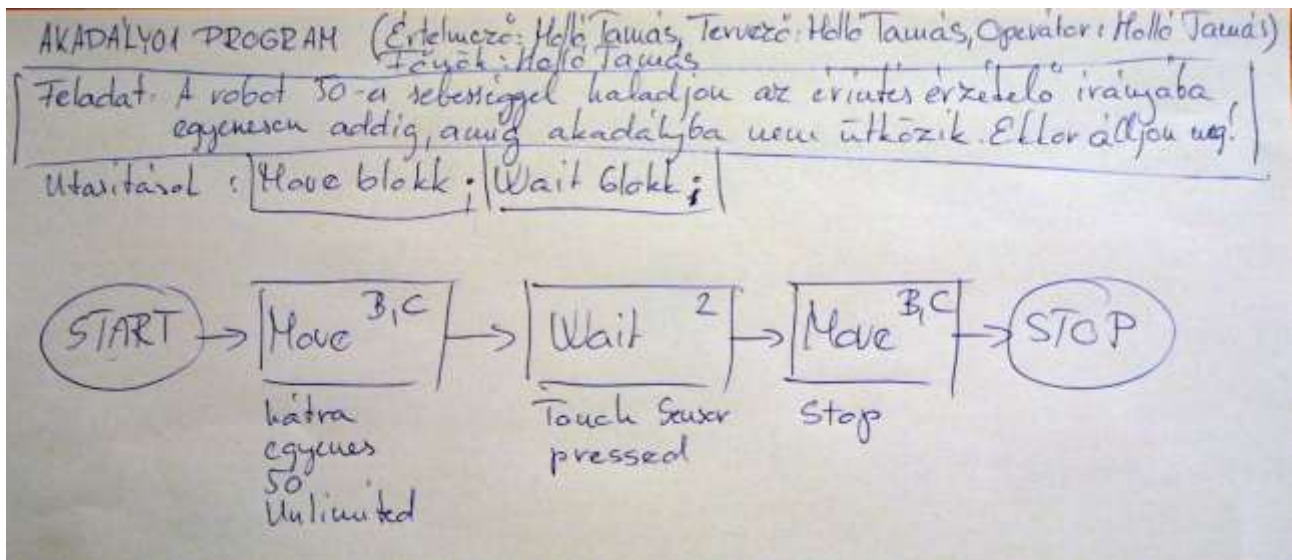
Egyszerű mondatszerű leírással, paraméterekkel:

Haladjon előre: Motor blokk → Port: B,C; Direction: hátra, Steering: egyenes; Power: 50; Duration: Unlimited;

Várjon a nyomásérzékelőre: Wait blokk → Control: Sensor; Sensor: Touch Sensor; Port: 2; Action: Pressed;

Álljon meg: Motor blokk → Port: B, C; Direction: Stop;

Folyamatábra, paraméterekkel:



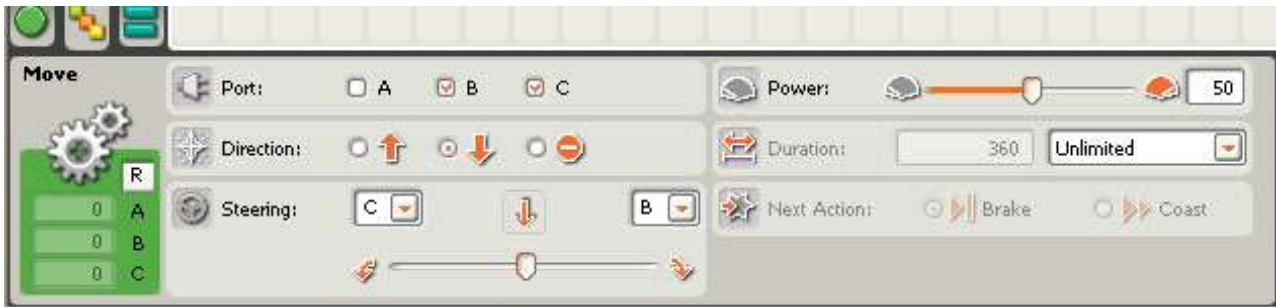
Írd meg a programot! (Operátor)

Indítsd el a Lego Mindstoms Programing programot.

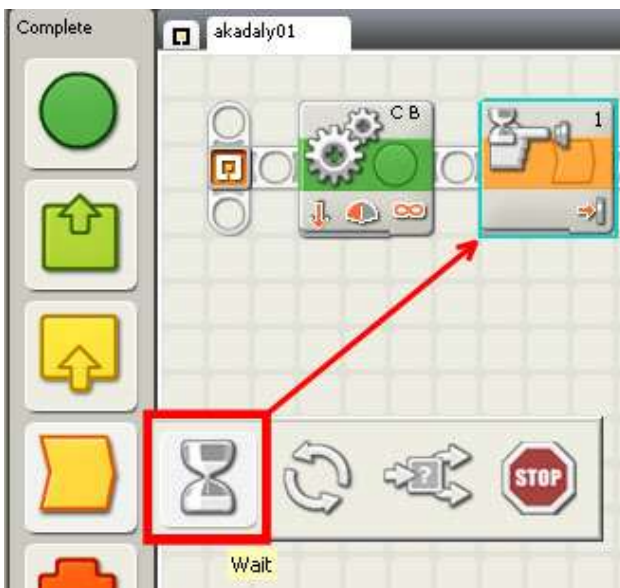
Add akadaly01 nevet a programnak.

Helyezd el a Move Blokkot a program szerkesztő területén

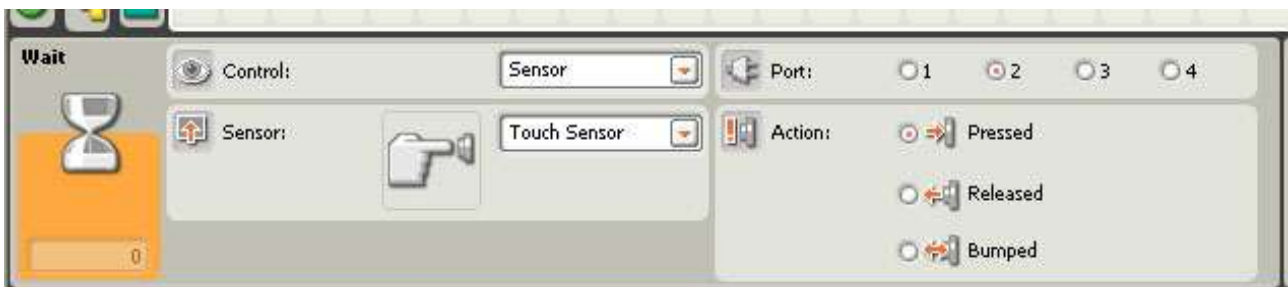
Állítsd be a paramétereit.



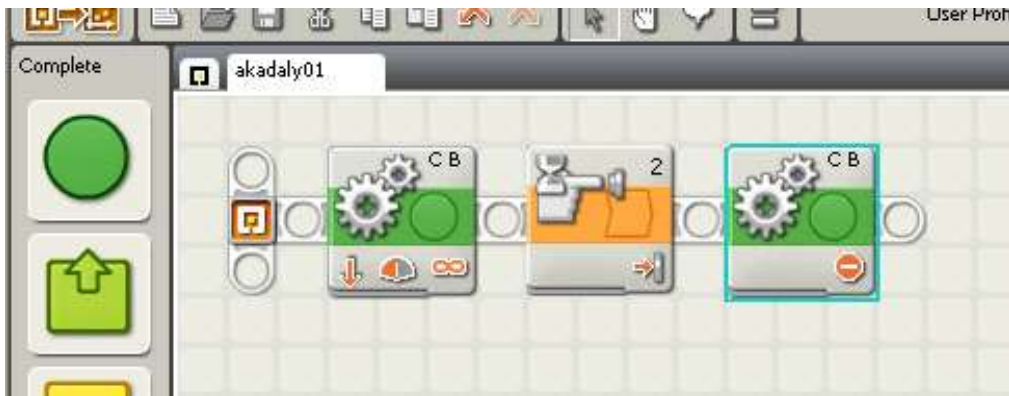
Helyezd el a Wait Blokkot a program szerkesztő területén



Állítsd be a paramétereit.



Helyezd el a Move Blokkot a program szerkesztő területén



Állítsd be a paramétereit.



Mentsd el a megadott helyre a programot.

Próbáld ki, hogy működik-e! (Főnök)

Kösd össze a robotot a számítógéppel.

Ellenőrizd a kapcsolatot.

Kattints a nyíl által mutatott gombra.

Kattints a Scan gombra.

Szólj, ha az elem állapota 7,1 alá csökkent!

Töltsd fel a programot a robotra.

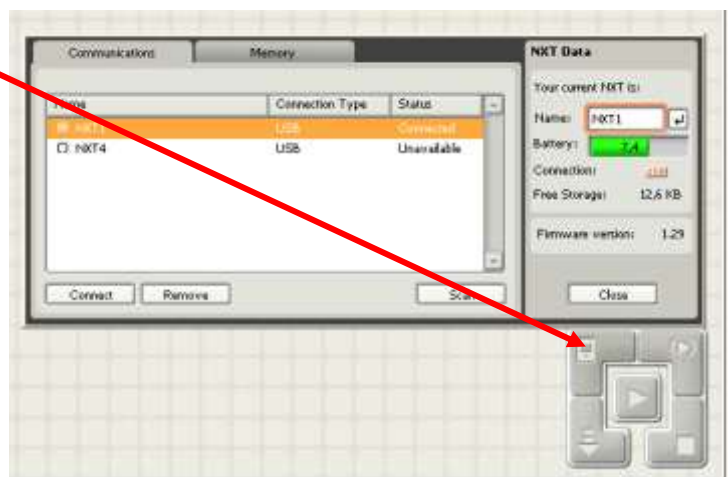
Húzd ki az USB kábelt a robotból.

Vidd a robotot a teszt helyére.

Indítsd el a program végrehajtását.

Próbáld ki több helyzetben, távolságból.

Ha a feladatban leírtak szerint működik, akkor készen vagy.



☞ Ha nem a feladatban leírtak szerint viselkedik a robot, akkor kezdődik a hibakeresés.

HIBAKERESÉS

Nézd át a programot, a blokkok beállításait egyenként ellenőrizd!

Ha megtaláltad a hibát rendben van!

Ha nem, akkor nézd át az algoritmust, gondold végig, hogy milyen utasításokat adtál ki egymás után. Most ez nem igen történhetett meg, de a későbbiekben ez a tanács is jól jöhet.

A hibakeresésről bővebben később.

Gyakorló feladatok

- 5.2. ☞ Írjon programot, amelyet végrehajtva a robot egyenesen halad mindaddig, amíg az ütközésérzékelőjével neki nem megy egy akadálnak! Ekkor tolasson 1 mp-ig, majd helyben forduljon kb. 180°-ot és utána haladjon egyenesen 3 mp-ig!
- 5.3. ☞ Írjon programot, amelyet végrehajtva a robot áll mindaddig, amíg a hangérzékelője kb. 60 as értéknél kisebb értéket mér! Ekkor induljon el egyenesen előre 3 mp-ig!
- 5.4. ☞ Írjon programot, amelyet végrehajtva a robot egyenesen halad mindaddig, amíg a távolságérzékelője 15 cm-nél kisebb távolságot nem mér! Ekkor tolasson 2 mp-ig, forduljon kb. 180°-ot, majd haladjon egyenesen előre 2 mp-ig!
- 5.5. ☞ Írjon programot, amelyet végrehajtva a robot egyenesen halad előre mindaddig, amíg a fényérzékelője az alapszintől eltérő szintet nem észlel, ekkor álljon meg.
- 5.6. ☞ Írjon programot, amelyet végrehajtva a robot megáll az asztal szélén! (Csak óvatosan!)
- 5.7. ☞ Írjon programot, amelyet végrehajtva a robot egy alapszintől jól elkülöníthető csíksor fölött halad és megáll a harmadik (vagy megadott számú) csík után. (A csíkok párhuzamosak, de távolságuk és szélességük nem féltetlenül azonos.)



- 5.8. ☞ Írjon programot, amelyet végrehajtva a robot tolja mindaddig, amíg az ütközésérzékelőjével neki nem ütközik egy akadálnak. Ekkor elindul egyenesen előre egy alapszintől jól megkülönböztethető csíkig. Ezt elérve megáll.

6. Lecke

ISMÉTELD AZ UTASÍTÁST!

Ebben a leckében az ismétlésé lesz a főszerep. Azért alkalmazzuk számítógépeket, mert ugyan azt a feladatot a végtelenségig ismételve pihenés nélkül tökéletes pontossággal elvégzik.

Ha eddig nem tűnt volna fel, akkor most mondom, egy robotot programozol. Bizonyos döntéseket hozott már önállóan a programjának megfelelően, de a feladatokat, utasításokat még nem ismételte.

CIKLUSOK

6.1. 🗨️ Küldetés: Haladjon a robot 2 másodpercig egyenesen 2 másodpercig, majd forduljon kb. 90°-t és ismételje ez a végtelenségig.

Értelmezés:

Itt sok értelmezni való nincs, mert a feladat elég pontos.

Nincsenek bemenő adatok, nincs környezeti feltétel.

Mindegy melyik irányban indul.

A fordulás szöge is meg van határozva 90°.

Íránya mindegy, tehát legyen jobbra.

A sebesség, ami ha nincs meghatározva, akkor az általános tanács az 50-es sebesség beállítása.

A programnak nincs vége, addig megy a robot, amíg ki nem merül az akkumulátora vagy a Főnök meg nem szakítja a program futását.

Tervezés:

A program neve „végtelen”.

Mondatszerű leírás

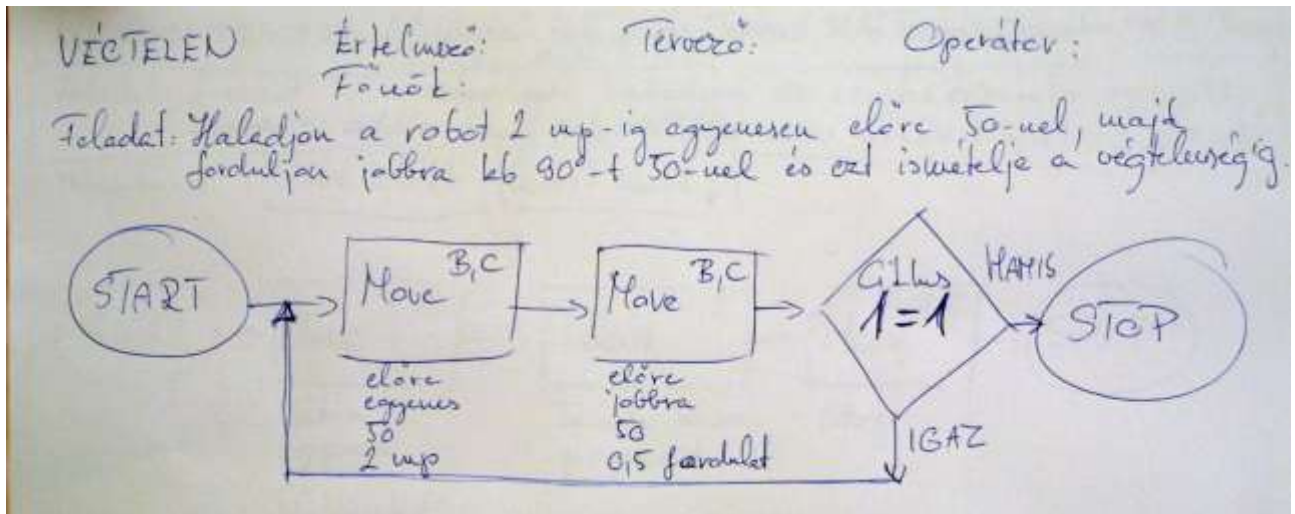
Ciklus

Move Blokk: B, C; előre; egyenes; 50; 2 mp;

Move Blokk: B, C; előre; teljesen jobbra; 50; 0,5 fordulat;

amíg végtelen (1=1)

Folyamatábra



- ☞ A ciklus folyamatábrán való jelölése megegyezik az elágazás jelölésével. Ebben az esetben a végtelenséget egy azonosság jelöli ($1=1$), ami sohasem lehet hamis. Ezt a megoldást a programozás kezdetén alkalmazzuk, majd később a valós élet megoldásaihoz közelebb álló megoldás fogunk alkalmazni.

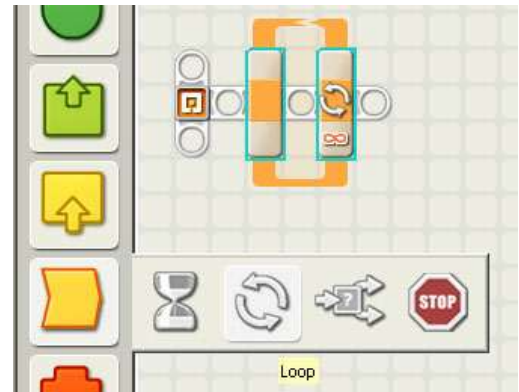
Operátor

Indítsd el a Lego Mindstorms Programming programot.

Add végtelen nevet a programnak.

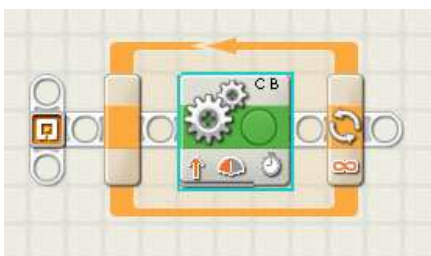
Helyezz el ciklust (Loop) a programszerkesztőben.

Állítsd be a paramétereit.



- ☞ Természetesen ez most csak fiktív beállítás, hiszen az alapbeállítása a ciklusnak a Control: Forever;

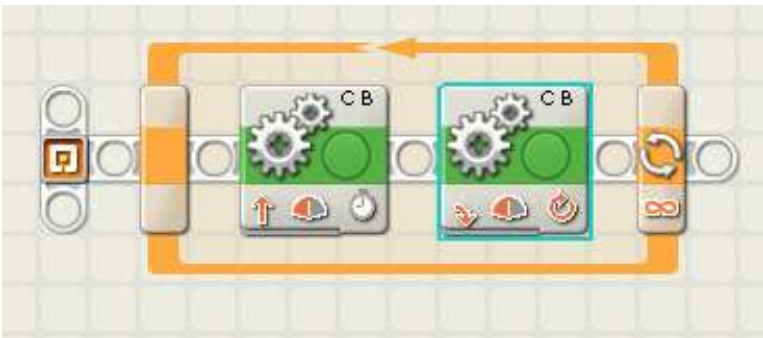
Helyezd el a Move Blokkot a Ciklus belsejében.



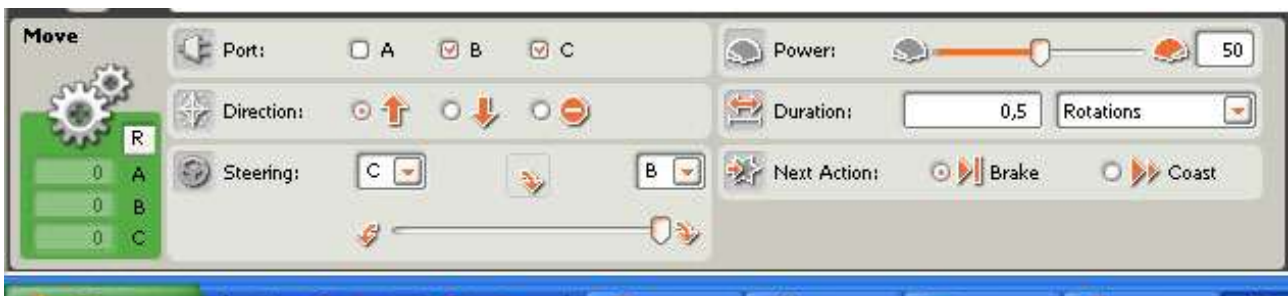
Állítsd be a paramétereit.



Helyezd el még egy Move Blokkot a Ciklus belsejében.



Állítsd be a paramétereit.



Mentsd el a programot!

Főnök

Kösd össze a robotot a számítógéppel.

Ellenőrizd a kapcsolatot.

Töltsd fel a programot a robotra.

Húzd ki az USB kábelt a robotból.

Vidd a robotot a teszt helyére.

Indítsd el a program végrehajtását.

AZ NXT buttons alsó szürke gombjával tudod megszakítani a program futását.

Próbáld ki több helyzetben, távolságból.



Általában a ciklusokról

A programozás során a ciklusok képezik a strukturált algoritmusok egyik alap-pillérét.

Strukturált algoritmus elemei:

a., szekvencia (soros algoritmus, blokk)

☞ Eddig látszólag csak ezt használtuk, jól paraméterezett display, move és sound blokkot.

b., szelekció (elágazás)

- feltételes
- többirányú

☞ Burkoltan már ezt is használtuk, hiszen eldöntöttük, hogy be van-e nyomva az érintésérzékelő vagy sem!

c., iteráció (ciklus, ismétlés)

A programozás elmélet három ciklus félélet különböztet meg.

Számláló ciklus: Meghatározott ismétlés szám jellemzi.

Ciklus n-től m-ig lépésköz=d

Ciklus mag

Ciklus vége

Előli tesztelő ciklus: Egy feltétel teljesüléséig fut a ciklus mag, az ismétlések száma így nem előre meghatározott, valamint fontos tulajdonsága, hogy előfordulhat a ciklus mag végrehajtásra sem kerül.

Ciklus amíg a feltétel igaz

Ciklus mag

Ciklus vége

Hátul tesztelő ciklus: A ciklus mag legalább egyszer lefut és a ciklus csak addig ismétlődik amíg a feltétel nem teljesül.

Ciklus

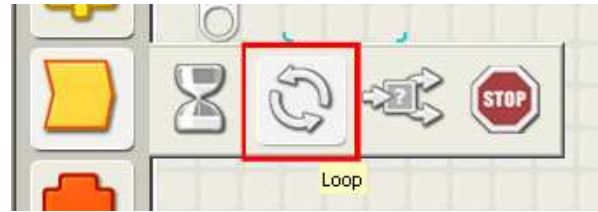
Ciklus mag

Ciklus amíg a feltétel hamis

A LEGO ezen programozási nyelvén csak számláló ciklus és hátul tesztelő ciklust használhatsz.

LOOP BLOKK

Minden ciklus (ismétlés) ugyan abból a Complete palette, Flow csoportjának Loop blokkjából indul ki. A következő oldalakon a legfontosabb ciklusszervezési lehetőségeket fogod áttekinteni.



SZÁMLÁLÓ CIKLUSOK

Count ciklus

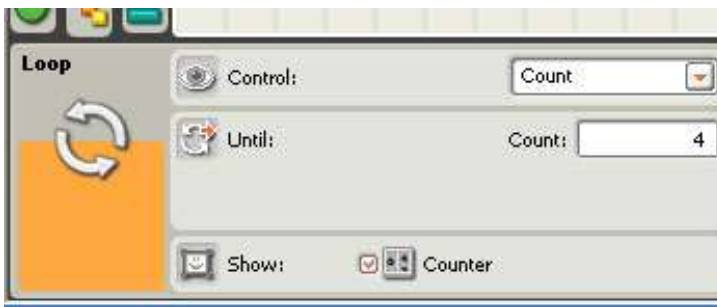


A számláló ciklus pontosan annyiszor fut le, amennyi a beállított count érték.

Control: Count (számláló)

Until: amíg a számláló nem 4, vagyis tetszőleges pozitív egész szám. Ennyiszor fog a ciklus lefutni.

Show: A ciklus számláló konnektorának bekapcsolása, amellyel a ciklus magban tájékozódhatsz, hányadik ismétlésnél tart a ciklus.



A másik számláló típusú ciklus a Time ciklus, amelyik meghatározott ideig fut.

Time ciklus

Control: Time (Idő)

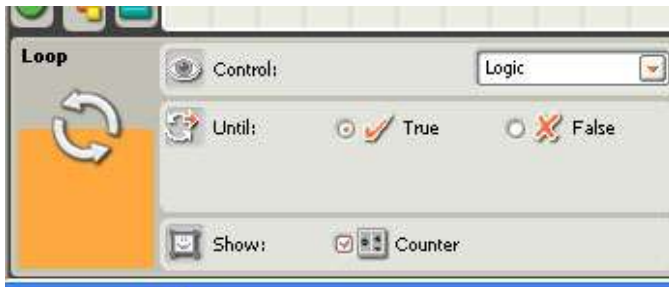
Until: a beállított másodpercig fut a ciklus.

Show: A ciklusszámláló konnektorának bekapcsolása, az ismétlések száma. (Azt adja meg, hányszor futott le a ciklus.)



HÁTUL TESZTELŐ CIKLUSOK

Logic ciklus



Control: Logic (Logikai – Igaz/Hamis)

Until: a konnektoron keresztül kapott érték azonossá nem válik a beállított True (Igaz) vagy False (Hamis) értékkel.

Show: A ciklusszámláló konnektorának bekapcsolása, az ismétlések száma. (Azt adja meg, hányszor futott le a ciklus.)

Bármilyen reláció eredménye (logikai adat), vagy érzékelő yes/no konnektorának értéke beköthető és a ciklus Until (amíg) konnektorába.

ÉRZÉKELŐHÖZ KÖTÖTT CIKLUSOK

Touch Sensor



A ciklus addig fut, amíg az érzékelő az előre beállított állapotba nem kerül.

Control: Sensor

Sensor: Touch Sensor

Show: A ciklusszámláló konnektorának bekapcsolása, az ismétlések száma. (Azt adja meg, hányszor futott le a ciklus.)

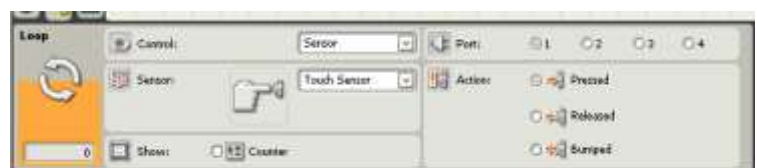
Port: az érzékelő melyik portra van csatlakoztatva.

Action:

Pressed: a gomb benyomva

Released: a gomb kiengedve

Bumped: a gomb benyomva és kiengedve



Light Sensor



A ciklus addig fut, amíg a fényérzékelő paraméteriben beállított reláció kiértékelése igazgá (true) nem válik.

Control: Sensor

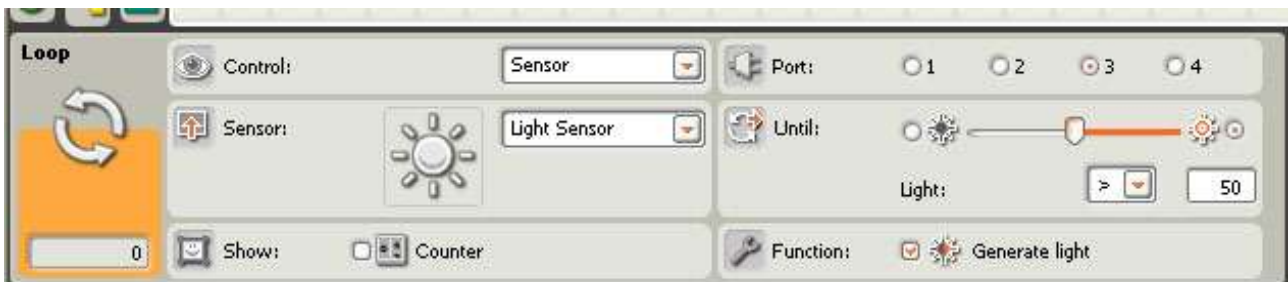
Sensor: Light Sensor

Show: A ciklusszámláló konnektorának bekapcsolása, az ismétlések száma. (Azt adja meg, hányszor futott le a ciklus.)

Port: az érzékelő melyik portra van csatlakoztatva.

Until: amíg a fényérés eredménye és a megadott érték közti reláció igazra nem válik, addig fut a ciklus.

Function: a saját fény bekapcsolása



Esetünkben a képen látható Until beállítása mellett a ciklus addig fut, amíg a fényérzékelő nagyobb értéket nem mér mint 50. Tehát a ciklusmag végrehajtása addig folytatódik, amíg a mérés eredménye kevesebb vagy egyenlő, mint 50.

Sound Sensor



A ciklus addig fut, amíg a hangérzékelő paraméteriben beállított reláció kiértékelése igazra (true) nem válik.

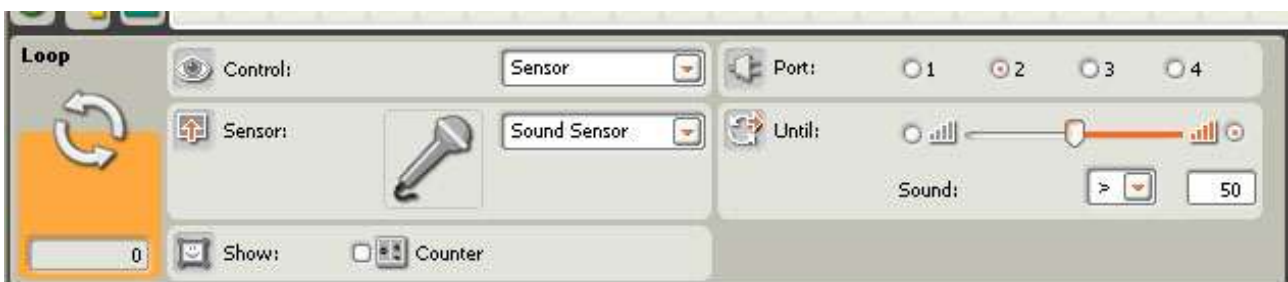
Control: Sensor

Sensor: Sound Sensor

Show: A ciklusszámláló konnektorának bekapcsolása, az ismétlések száma. (Azt adja meg, hányszor futott le a ciklus.)

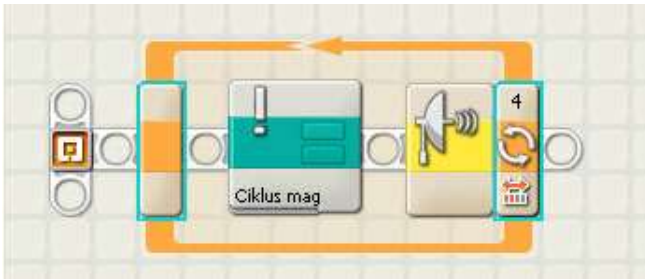
Port: az érzékelő melyik portra van csatlakoztatva.

Until: amíg a hangerő mérés eredménye és a megadott érték közti reláció igazra nem válik, addig fut a ciklus.



Esetünkben a képen látható Until beállítása mellett a ciklus addig fut, amíg a hangérzékelő nagyobb értéket nem mér mint 50. Tehát a ciklusmag végrehajtása addig folytatódik, amíg a mérés eredménye kevesebb vagy egyenlő, mint 50.

Ultrasonic Sensor



A ciklus addig fut, amíg a távolságérzékelő paraméteriben beállított reláció kiértékelése igazgá (true) nem válik.

Control: Sensor

Sensor: Ultrasonic Sensor

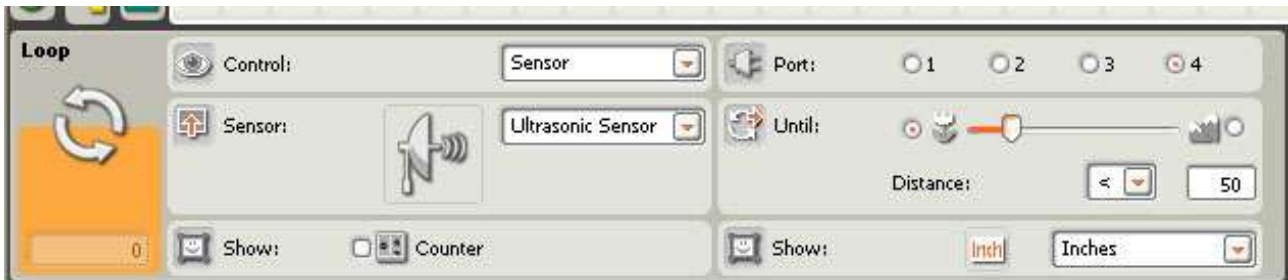
Show: A ciklusszámláló konnektorá-

nak bekapcsolása, az ismétlések száma. (Azt adja meg, hányszor futott le a ciklus.)

Port: az érzékelő melyik portra van csatlakoztatva.

Until: amíg a távolság mérés eredménye (distance) és a megadott érték közti reláció igazgá nem válik, addig fut a ciklus.

Show: a távolság mértékegysége. Inch (alapértelmezett) vagy centiméter.



Esetünkben a képen látható Until beállítása mellett a ciklus addig fut, amíg a távolságérzékelő kisebb értéket nem mér mint 50 inch. Tehát a ciklusmag végrehajtása addig folytatódik, amíg a mérés eredménye nagyobb vagy egyenlő, mint 50 inch.

Gyakorló feladatok

- 6.2. Írj programot, amelyet végrehajtva a robot ütközésig halad előre, majd tolat 1 mp-ig, fordul kb. 120°-ot, majd mindezt ismétli kikapcsolásig!
- 6.3. Írj programot, amelyet végrehajtva a robot az alapfelülettől eltérő színű csíkig halad előre (fényérzékelővel mérve), majd tolat 1 mp-ig, fordul kb. 120°-ot, mindezt ismétli kikapcsolásig!
- 6.4. Írj programot, amelyet végrehajtva a robot egyenesen halad előre mindaddig, amíg a fényérzékelőjére rá nem világítunk, ekkor forduljon kb. 180°-ot. Mindezt ismétlje kikapcsolásig!
- 6.5. Írj programot, amelyet végrehajtva a robot ultrahang szenzorával mért 15 cm-nél kisebb távolságig halad előre, majd ütközésérzékelőig tolat hátra, ezt ismétli kikapcsolásig!
- 6.6. Írjon programot, amelyet végrehajtva a robot egyenesen halad előre 2 mp-ig, fordul jobbra kb. 60°-ot, majd ezt ismétli 6-szor!

- 6.7.** ☞ Írj programot, amelyet végrehajtva a robot egy ultrahang-szenzorral képes egy adott távolságon belül lévő akadály észlelésére és követésére. A robot egyenletes sebességgel forog és ha egy adott távolságon belül észlel valamit (az ultrahangszenzora jelzi), akkor elindul felé mindaddig, amíg az adott távolságon belül van az észlelt akadály. Ha elveszítette (kikerült az akadály a távolságon kívülre), akkor forog újra. Mindezt kikapcsolásig ismétli.
- 6.8.** ☞ Írj programot, amelyet végrehajtva a robot egy fényérzékelővel képes egy lámpa fényének észlelésére és követésére. A robot egyenletes sebességgel forog és ha egy lámpa fényét észleli, akkor elindul felé. Ha elveszítette, akkor forog újra. Mindezt kikapcsolásig ismétli.
- 6.9.** ☞ Írj programot, amelyet végrehajtva a robot egyenesen halad előre mindaddig, amíg ultrahang szenzorával 20 cm-es távolságon belül nem érzékel akadályt. Ekkor fordul jobbra kb. 90°-ot. Mindezt az ütközésérzékelő benyomásáig ismétli.
- 6.10.** ☞ Írjon programot, amelyet végrehajtva a robot a hangérzékelője által mért értéket használja fel a motorok sebességének vezérlésére! Annál gyorsabban forogjon helyben a robot, minél hangosabb a környezete!

7. Lecke

Merre induljak? Melyiket válasszam?

A robotok, számítógépek sokszor megtévesztenek bennünket, mert a környezeti hatásokra, bevitt adatokra, olyan válaszokat adnak, amelyek tudatos cselekedeteknek, gondolkodásra utaló válaszoknak hatnak.

Aki egy kicsit is járatos a programozásban, az viszont tudja, hogy az amit a robottól látott reakció, a számítógéptől kapott válasz az csak előre a programozó által végiggondolt lehetőségek (elágazások) és arra adott igen vagy nem válaszok.

Elágazások

7.1. ☞ Legyen a robot küldetése, hogy hangjelzéssel jelezze, ha áthalad egy csíkon!

☞ Már volt egy olyan feladat, amelyikben csíkokon kellett áthaladni és megtalálni a harmadikat.

Gondolkozzatok el, miben különbözik a két feladat egymástól?

Értelmező és kérdései?

<p>Merre kell haladnia a robotnak és milyen sebességgel? Meddig kell haladnia? Mit jelent a csík? Milyen széles, milyen színű? Milyen hangjelzést adjon?</p>
--

A robot előre egyenesen halad, 50-es sebességgel.

Addig megy, amíg a program futását meg nem szakítjuk.

A csík a padló színétől jelentősen eltérő (sötétebb) szigetelő szalag csík.

A szélessége változó.

A hangjelzés rövid sípolás, A hang 0,5 másodperc.

Tervező

A probléma az, hogy nem tudjuk, hogy a megtett út alatt, hány darab csíkkal és milyen széles csíkkal találkozik a robot.

Tehát egy végtelen ciklusba ágyazott mozgás, amelyik viszi előre a robot és egy elágazás, amelynél ha fénymérő kisebb értéket mér a beállítottnál, akkor sípolunk, különben kikapcsolod a hangot.

Mondatszerű leírás:

Program Csiksip.rbt

Ciklus

Motor B,C; előre; egyenes; 50; Unlimited

Ha fénymérő < 50

akkor

hangjelzés A; 0,5 másodperc

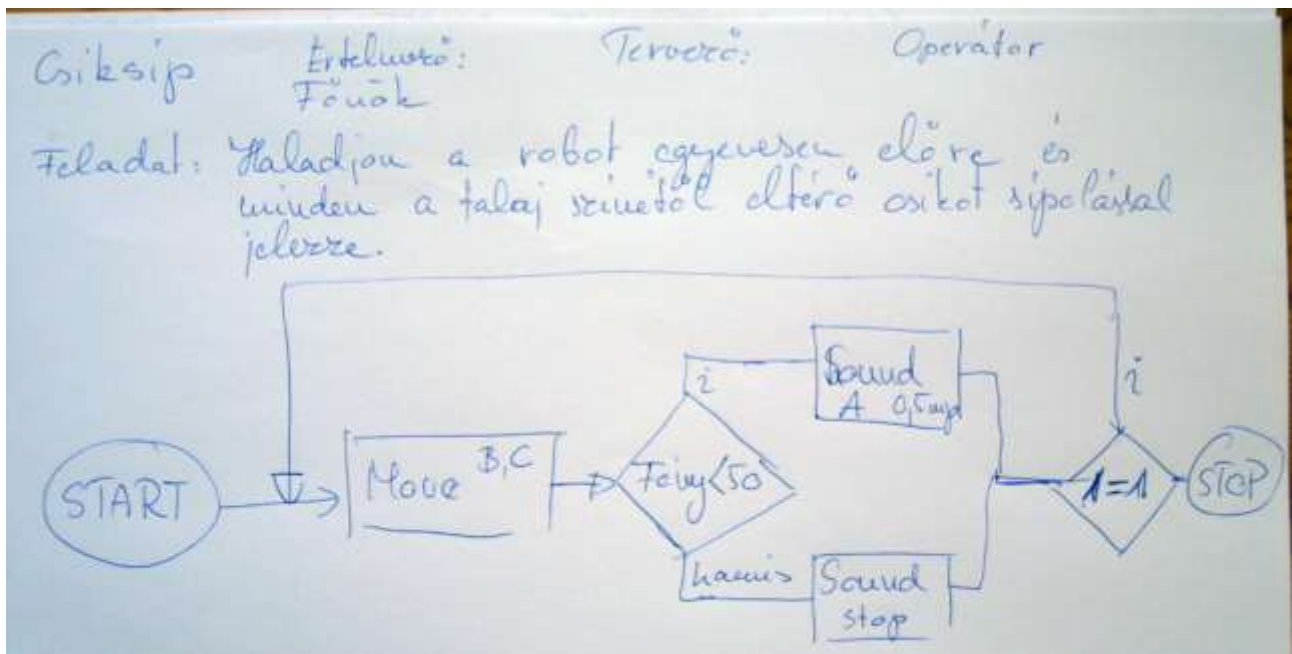
különben

hangjelzés Stop

amíg 1=1 (végtelen)

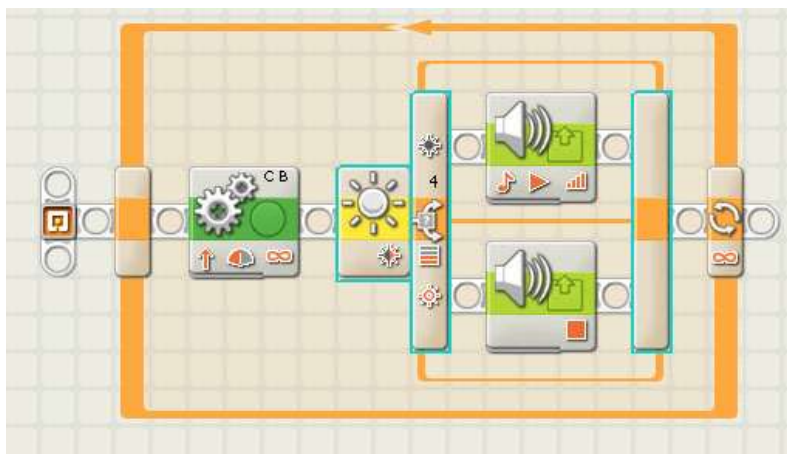
Program vége

Folyamatábra



Operátor

A programozási felület elindítása után, hozd létre a következő programot.



Illeszd be a végtelen ciklust.

Szúrj be egy elágazást.

- ☞ Ebben a feladatban az új ismeret (program blokk) az elágazás (switch), amelyet a flow csoportban találsz!

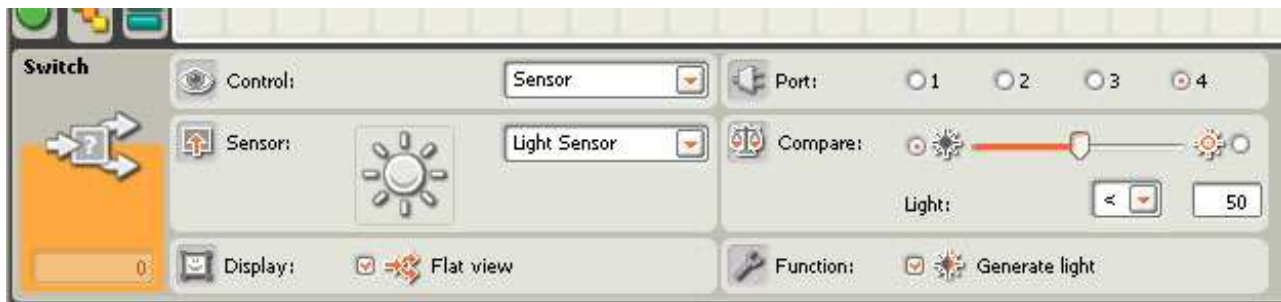


- ☞ Az elágazás vezérlője (Control) a sensor. Ez azt jelenti, hogy az érzékelő által mért adatok alapján döntjük el milyen további utasításokat fog elvégezni a robot.

Állítsd be az elágazás típusát.

- ☞ Az érzékelő a feladatban a fényérzékelő (Light Sensor), tehát a beillesztés után erre kell átállítani az érzékelő (Sensor) típusát.

A fényérzékelő beállításai pedig az Értékelő és Tervező által kitaláltak alapján állítsd be.



- ☞ Ügyelj a Portra és a reláció megfelelő beállítására.

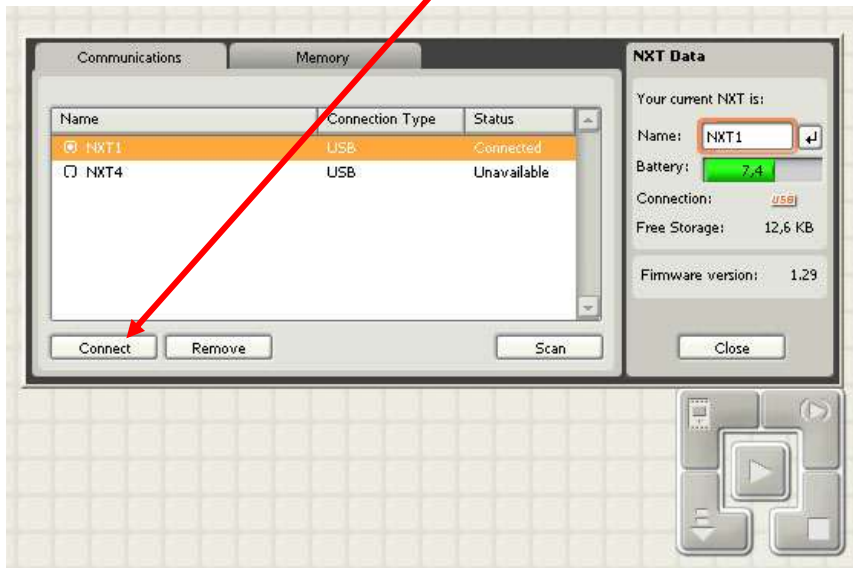
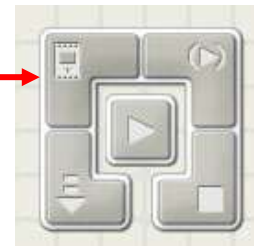
Compare: Ha a mért érték kevesebb, mint 50, akkor az igaz ágon (felső útvonal) halad a program végrehajtása, különben ha nagyobb vagy egyenlő, mint 50, akkor a hamis úton (alsó útvonal) halad a programvégrehajtás.

Display: Bejelölve az elágazás mindkét ága látszik.

- ☞ Az elágazások megjelenítése, sokszor túl áttekinthetatlenné teszi a programunkat. Ezzel a beállítással egyszerűsíthetsz a program megjelenésén, úgy, hogy egyszerre csak egy ágát jeleníted meg az elágazásnak.

Érzékelő kalibrálása

Az érzékelők különböző környezetben más és más értéket mérhetnek. Ezért az NXT ablak (NXT window) bekapcsolása után a kapcsolat ellenőrzésével létrehozott összekötetés segítségével megnézheted, hogy milyen értékeket mér a fényérzékelő.



Ha megnyitottad az NXT ablakot, kattints a kapcsolt (Connected) NXT téglára, majd a Connect gombra.

Ezután zárd be (Close) az NXT ablakot.



Most jelöld ki a szerkesztő felületen az elágazás blokkot, és néhány másodperc adatcsere után látni fogod az aktuálisan mért értéket a beállítás panelen.

Tegyél a fénymérő alá más és más színű papírt és látni fogod, hogy az adott fényviszonyok között mennyi a mért érték.

Ezután, a kis kitérő után folytasd a program a megírását.

A tegyél egy-egy hang blokkot az elágazás két irányába.

Az egyiket állítsd be úgy, hogy adjon ki egy A hangot 0,5 másodpercre.

A másikat úgy állítsd be, hogy hallgasson el.

Ne felejtse el elmenteni a forrásprogramot a megfelelő helyre.

Főnök

Töltsd fel a programot!

Próbáld ki.

Úgy viselkedik, ahogy a feladatban meghatároztuk?

Hányszor sípol? Hány csík van?

Ha többet sípol, vajon miért?

Általában az elágazásokról

A számítógép a döntéseit elágazások kiértékelésével hozza. A legbonyolultabb feladat is lebontható elemi logikai (igaz-hamis) döntések sorozatára.

Egyszerű elágazás

Ha *logikai feltétel*
akkor
 Utasítás blokk
különben
 Utasítás blokk
Elágazás vége

A **logikai feltétel** a formális logika által használt operátorok használatával létrehozott összetett kifejezés, amely kiértékelésének eredménye IGAZ vagy HAMIS lehet.

Az **utasítás blokk** azon utasítások sorozata, akár újabb elágazások és ciklusok együttese, amelyeket a logikai feltétel kiértékelése után végre kell hajtani. Az „akkor” és „különben” ág közül csak egyik hajthat végre egy kiértékelés alkalmával.

Egyéb elágazás

A programozási nyelvek többsége az egyszerű elágazás mellett több különböző lehetőséget is megenged az elágazások létrehozására, azért, hogy egyszerűsítse a forrásprogramot, ezáltal hatékonyabb legyen a program.

A jelenleg bemutatásra kerülő programozási környezetben nincs másik elágazás, így most eltekintek azok bemutatásától.

Gyakorló feladatok

- 7.2. ☞ Írj programot, amelyet végrehajtva a robot egyenesen halad előre egy alapszintől jól megkülönböztethető színű csíkokat tartalmazó felületen! A csíkokon áthaladva adjon hangjelzést!
- 7.3. ☞ Írjon programot, amelyet a robot végrehajtva egy alap színétől jól megkülönböztethető csíkot (nem feltétlenül egyenes) követ egyetlen fényérzékelőjével.
- 7.4. ☞ Írj programot, amelyik követi a kezéd oly módon, hogy ha közelíted a távolságérzékelőhöz, és ha távolítod a kezéd, akkor követi. Ha nincs semmi a közvetlen látóterében, akkor egyhelyben áll.
- 7.5. ☞ Írj programot, amelyet végrehajtva a robot egyenesen halad előre addig, amíg hanghatás nem éri, ekkor a hangerő mértékétől függően forduljon balra vagy jobbra 90°-t.

8. Lecke

Tárolj adatokat!

Minden érzékelő adatokkal dolgozik. A számítógép nevében is hordozza a szám fogalmát. A program végrehajtása során a mért vagy kiszámolt adatok azonnali feldolgozására nem lehetséges vagy nem szükséges. Ezeket az adatokat ideiglenesen vagy hosszútávon.

Ebben a leckében az ideiglenes adattárolással fogunk foglalkozni. Az ideigleenség itt az jelenti, hogy az adatot legfeljebb a program futásának idejéig tároljuk.

Változó blokk

Az adatok tárolása minden programozási nyelv és feladat alap problémája. Akár ideiglenesen, akár hosszútávon szeretnéd tárolni a mérési adatokat szükséged lesz egy tároló blokkra, amelyet a programozásban változónak (Variables) neveznek.

A változók két legfontosabb tulajdonsága:

Neve: Ezzel különböztetjük meg a különböző változókat. Csak az angol ABC betűit használhatod a változó elnevezésére.

Típusa: szám, szöveg vagy logikai adat tárolható a változóknak, de hogy ezek közül melyiket, azt előre a változó létrehozásakor meg kell határozni. Ezzel egyúttal általában a változó méretét is meghatározod.

8.1. 🗡️ A robot küldetése, hogy megállapítsa két tanuló közül, melyik tapsolt hangosabban.

A feladat elsőre elég jól meghatározottnak tűnik, de nem egy programozó számára. Kezdjük konkretizálni a feladatot.

Értelmező

A feladatban leírásában nincs meghatározva,
hogyan jelezze, a hangosabb taps irányát?
hogyan jelezze, a hangosabb taps irányát?

Egyáltalán honnan tudja, hogy melyik irányból jön a hang? Az ember is csak abból tud tippelni, a hang irányára, hogy két füle van. Akkor tegyünk két mikrofont a robotra?

Az értelmező feladata, hogy ezekre a kérdésekre, a többiek véleményének figyelembe vételével választ adjon és meghatározza a feladat pontos specifikációját.

Most a döntés a következő:

A robot két egymás után tapsoló (hangoskodó, kiabáló vagy doboló) diák közül kell, hogy megállapítsa, hogy melyik a hangosabb. Egy lámpa kigyulladás jelzi

a zajkeltés kezdetét és a kialvása a végét. A robot a képernyőre írja ki, hogy az első vagy a második zaj volt az erősebb.

Tervező

A feladat leírása elég pontos, ezért kezdek bele rögtön a mondat szerű leírásba. Van eleje és vége, hiszen csak meg kell jegyezni az első zaj maximális erejét (Upsz, micsoda? Ez azért keménynek tűnik, lehet, hogy erről lemondok.). Helyette mérek egyet valamikor véletlenszerűen a lámpa felkapcsolása és lekapcsolása között. (☺). Majd a két mérés után összehasonlítom a két mérést és kiírom melyik volt a nagyobb.

Program hangoska.rbt

```
//Inicializálás – magyarul a kezdő értékek és paraméterek beállítása
```

```
Elsohang = 0
Masodikhang = 0
Kinyert = ""
```

```
//Adatbevitel
```

```
Lámpa be //felkapcsolja a lámpát
ido = Véletlen(2-5) //sorsol egy véletlen számot 2 és 5 között
Meresido = 0 //Ciklus számláló nullázása
```

```
Ciklus
```

```
Ha ido = Meresido akkor Elsohang = Hangerő mérés
```

```
Vár(1mp) //1 mp vár, így 6 mp-ig fut a ciklus
```

```
Meresido = Meresido + 1 //ciklus számláló növelése
```

```
Amíg Meresido < 6
```

```
//addig fut amíg a ciklus számláló el nem éri a 6-t
```

```
Lámpa ki
```

```
//felkapcsolja a lámpát
```

```
Lámpa be
```

```
//felkapcsolja a lámpát
```

```
ido = Véletlen(2-5)
```

```
//sorsol egy véletlen számot 2 és 5 között
```

```
Meresido = 0
```

```
//Ciklus számláló nullázása
```

```
Ciklus
```

```
//6 szor lefutó ciklus
```

```
Ha ido = Meresido akkor Masodikhang = Hangerő mérés
```

```
Vár(1mp) //1 mp vár, így 6 mp-ig fut a ciklus
```

```
Meresido = Meresido + 1 //ciklus számláló növelése
```

```
Amíg Meresido < 6
```

```
//addig fut amíg a ciklus számláló el nem éri a 6-t
```

```
Lámpa ki
```

```
//felkapcsolja a lámpát
```

```
//Feldolgozás
```

```
Ha Elsohang > Masodikhang
```

```
akkor
```

```
Kinyert = "1. zajongó volt hangosabb"
```

```
különben
```

```
Kinyert = "2. zajongó volt hangosabb"
```

```
//Adatkiírás
```

```
Kiírás Kinyert
```

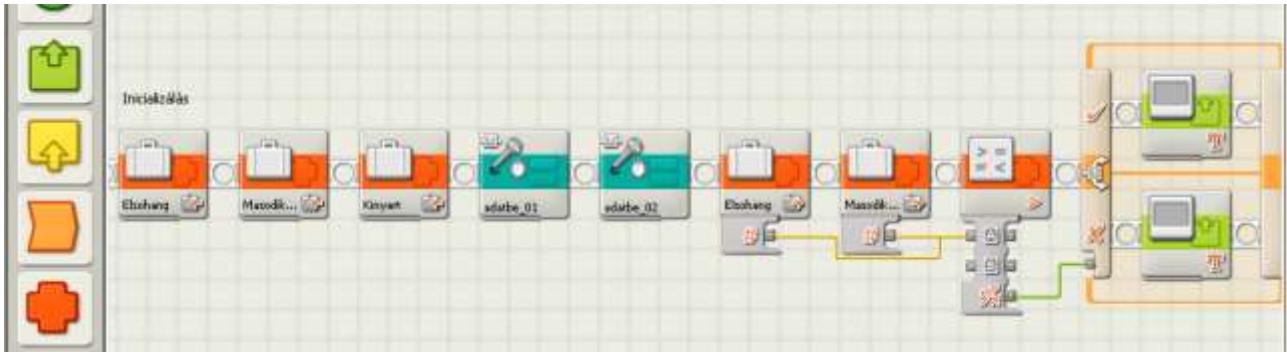
```
Program vége
```


Sajnos közben kiderült számomra, hogy ez így igazságtalan, de erre már nektek kell rájönnötök, hogy miért és megoldani a problémát. Van egyéb gond is, például hatékonysági probléma, de nem lehet mindent egyszerre.

Főnök, piszkosul ráérsz! Szerelj fel egy lámpát a robotra!

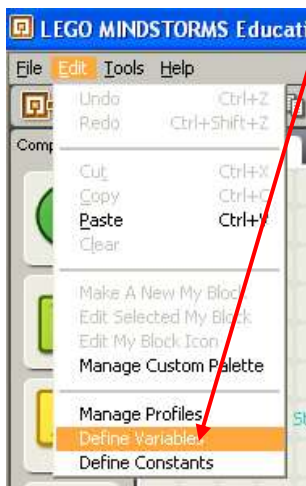
Operátor

Indítsd el a LEGO MINDSTORMS NXT Programing-t, majd add a neki a „hangoska.rbt” nevet. Ne felejtse el a mentési útvonalat beállítani.



☞ Ez a forrás program teljes listája, azt vedd észre, hogy az adatbevitel helyett két blokk van elhelyezve. A feladat megoldása során a blokk létrehozásának lépéseit is bemutatom.

Definiáld egy Elsohang nevű szám (Number) típusú változót! Ebben fogod tárolni az első hangoskodó mérési adatát.

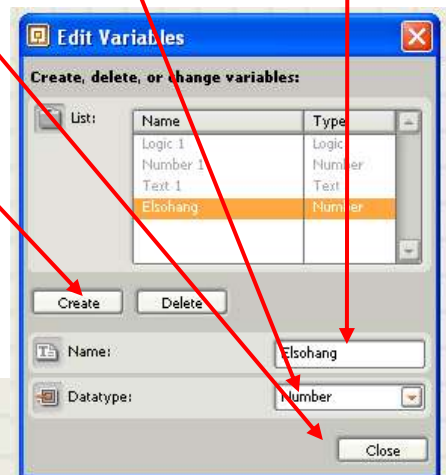


Edit → Define Variables menüpontot válaszd.

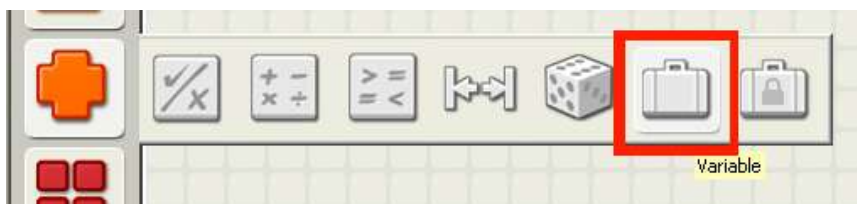
Kattints a Create gombra. Állítsd be a változó nevét (Elsohang) és a típusát (Number).

Zárd be az ablakot a Close gombbal!

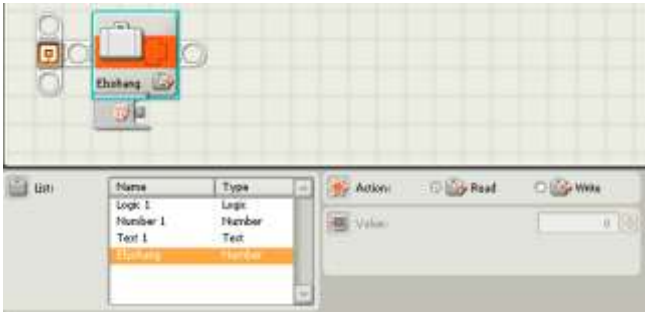
☞ A változó nevében nem használhatsz ékezeses vagy különleges karaktereket. Ez egyébként is így van minden programozási környezetben. Válassz mindig rövid, de beszédes nevet a változóidnak.



Helyezz el egy változót a forrás programban.



Állítsd be a paramétereit.



Válaszd az Elsohang nevet a listából az **Action**-t (tevékenység) pedig állítsd Write-ra.

A **Value** (érték) legyen 0.

-
- ☞ Az action (tevékenység) lehet Read, ekkor kiolvasni szeretnénk a változó értékét, vagy Write, ekkor el helyezhetünk egy a változó típusának megfelelő értéket a változóban.
-

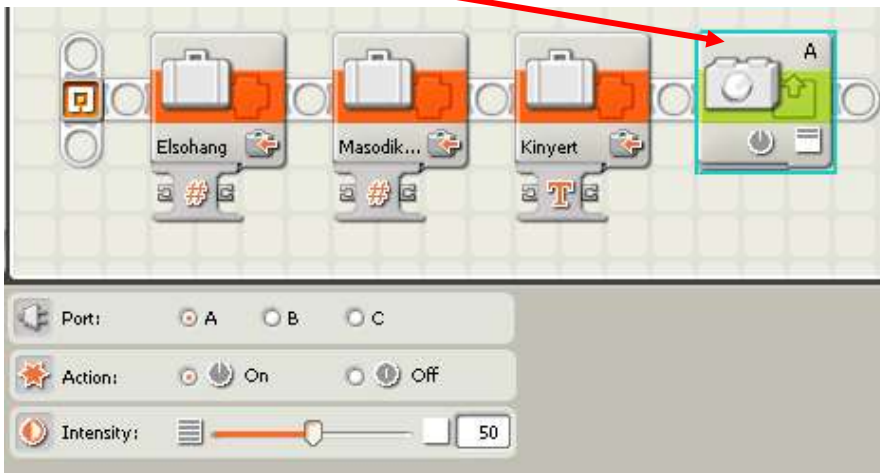
Még két változót kell definiálnod: Masodikhang (Number) és Kinyert (Text).

Ezeket a változókat helyezd el a forrás programban és állítsd be a változók nevét, ezzel az inicializáló rész végére értél.

Most következnek az adatbevitel

Az adatbevitel első lépése a lámpa felkapcsolása

Lámpa blokk



Port: Azokat a kapukat kell használni, amelyeket a motorok használnak.

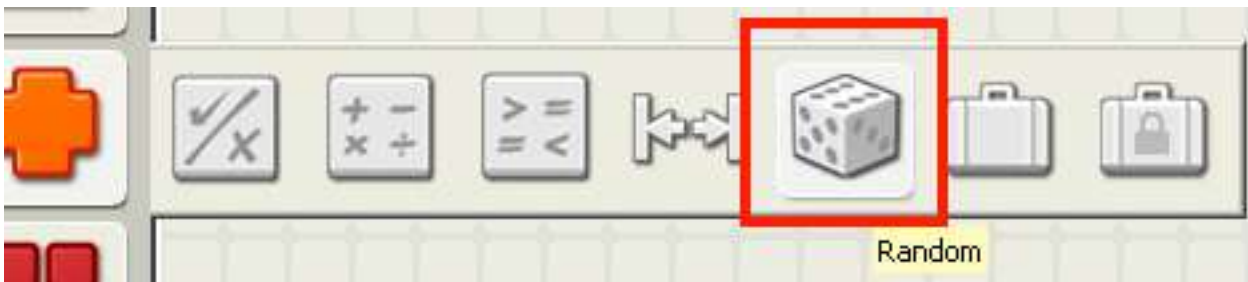
Action: Bekapcsolod (On) vagy kikapcsolod (Off)

Intensity: A lámpa fényereje 0 – 100 állítható. (0 esetén, be van kapcsolva még se világít).

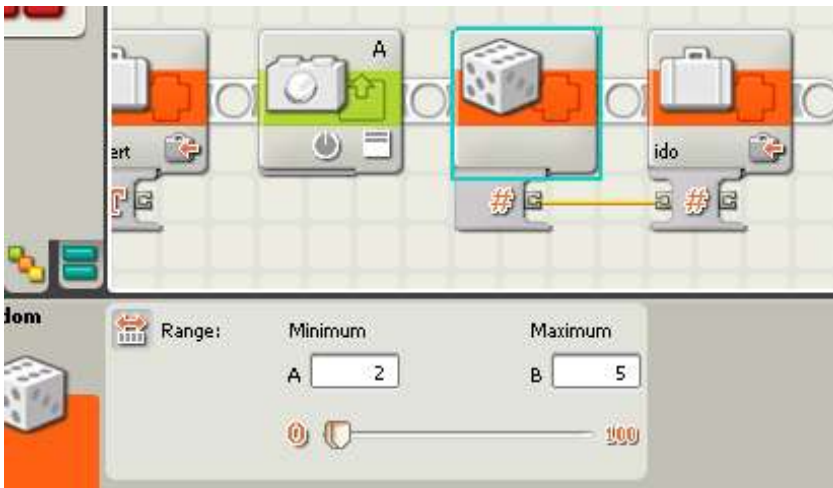
Sorsolj egy számot, amely azt fogja megadni, hogy a 6 mp zajmérés időn belül, mikor méri meg a robot a hangerőt.

A véletlen számot az ido nevű, szám típusú változóban fogod tárolni. Tehát definiálni kell egy ido változót.

Véletlen szám előállítása



Szúrd be a véletlen szám (Random) utasítást.



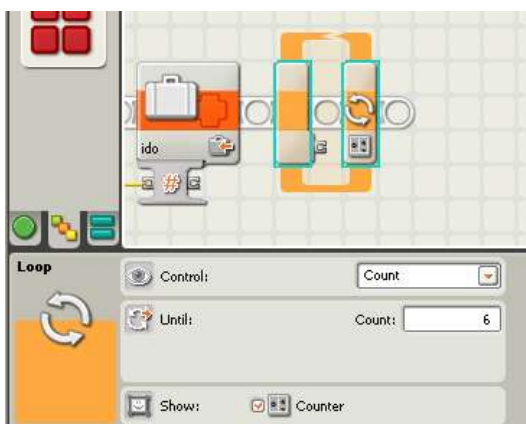
Range: A beállított minimum és maximum érték között előállít egy számot véletlenszerűen.

Az intervallum jobbról és balról zárt, tehát a határai is a kisorsolt számok között lehetnek.

☞ Szúrd be az idő változót és állítsd az action-t Write-ra. Hiszen ebbe írjuk bele a generált véletlen számot.

Majd kösd össze a konnektoraikat.

Szúrj be egy számláló ciklust és állítsd be a paramétereit!



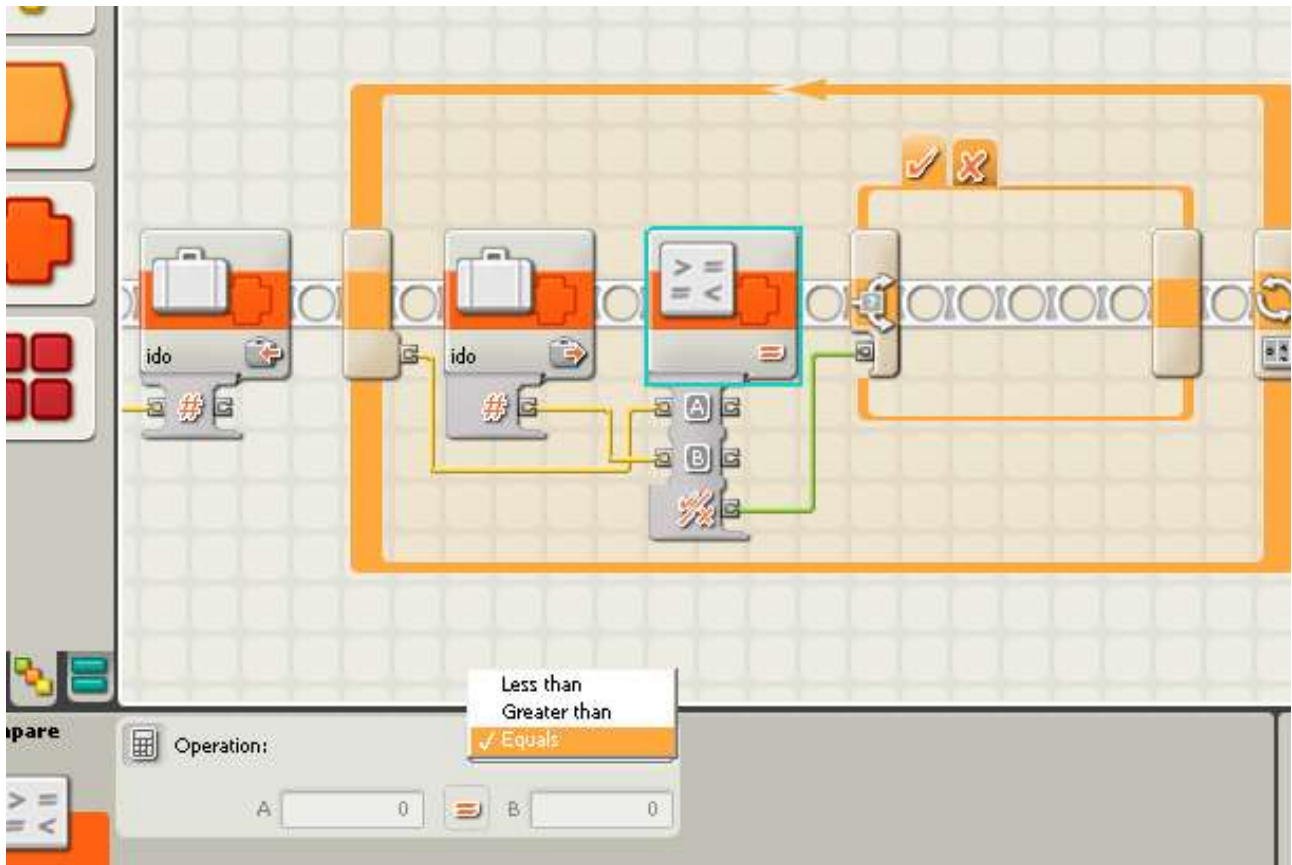
Control: Count (számláló)

Until Count: hányszor fusson le a ciklus.

Show: Counter kiválasztva. Láthatóvá válik az a konnektor, amelyiken keresztül kiolvashatjuk, hányadszor fut le éppen a ciklus.

☞ A számláló (Counter) miatt nincs szükségünk a Meresido változóra és cikluson belül a növelésére sem, mert ez a ciklus automatikusan növeli a számláló értékét, amely a konnektoron keresztül kiolvasható.

Hasonlítsd össze a számláló állását a ido változó értékével. Majd a kapott logikai érték alapján a robot dönt, hogy mérjen hangerőt vagy sem.



Tehát szűrj be a ciklusba az ido változót, állítsd Read-re az action.

Szűrj be egy összehasonlító utasítást és kösd a konnektoraiba a számláló és az ido változó konnektorát.

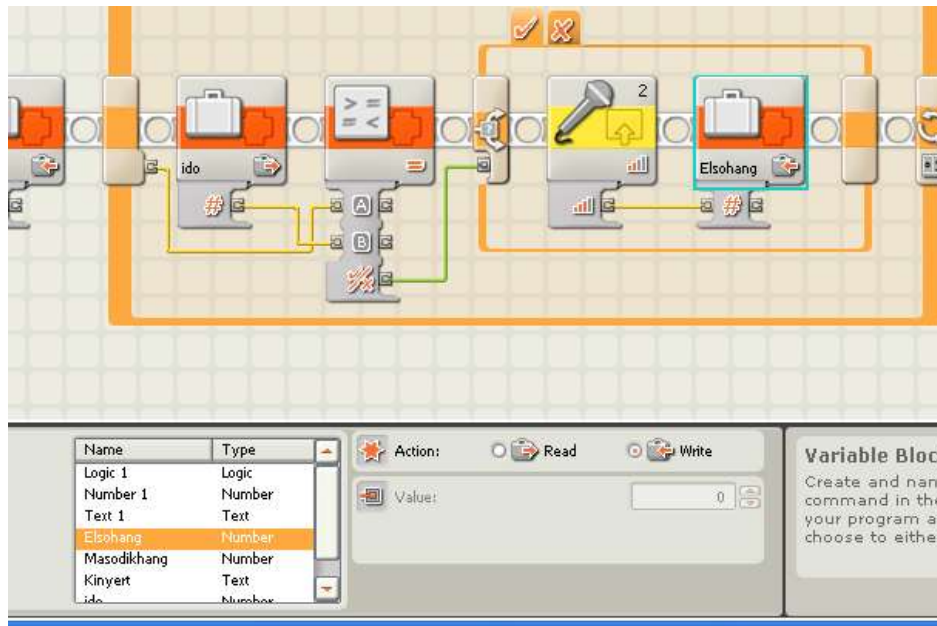
-
- ☞ Mindegy, hogy melyiket kötöd A és B konnektorba, mert a relációt Equals-ra (egyenlő) kell állítani, mert amikor a számláló értéke egyenlő a sorsolt idővel, akkor a reláció értéke igaz (TRUE) lesz és végrehajtódik az elágazás igaz ága, ahol a robot a hangerő mérésére kap majd utasítást.
-

Szűrj a ciklusmagba egy érték (value) elágazást, amelyik típusa logikai (Logic) és A Display paraméterét állítsd jelöletlenre, hiszen csak az igaz ágát fogod használni.

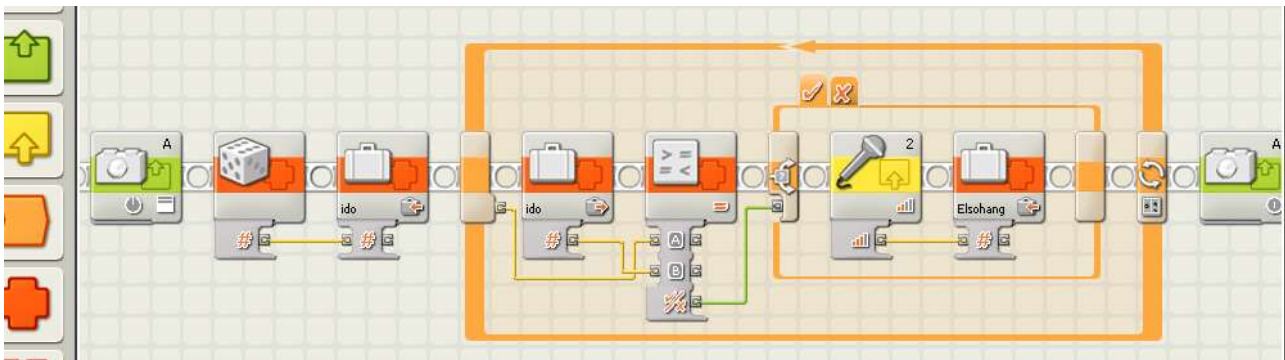
Az elágazásba szúrd be a hangérzékelő utasítást,

Ugyan ide szúrd be a Elsohang nevű változót és állítsd Write-ra.

Kösd össze a hangérzékelő hangerő konnektorát a változó bemenetével.



Már csak a lámpát kell kikapcsolnod.



- ☞ Tehát létrehoztad az első hangmérés adatbevitelét a lámpa felkapcsolásától a lekapcsolásáig. Vedd észre, hogy ez a második hangmérés teljesen ugyan ez, mindössze a mérés eredményét a Masodikhang változóba kell beletenni.
- ☞ Másolhatnánk is az egészet, de vedd észre, hogy a forrás program a már kezd nagyon hosszú lenni. Ezért az áttekinthetőség kedvéért most megtanuljuk a saját blokk készítését.

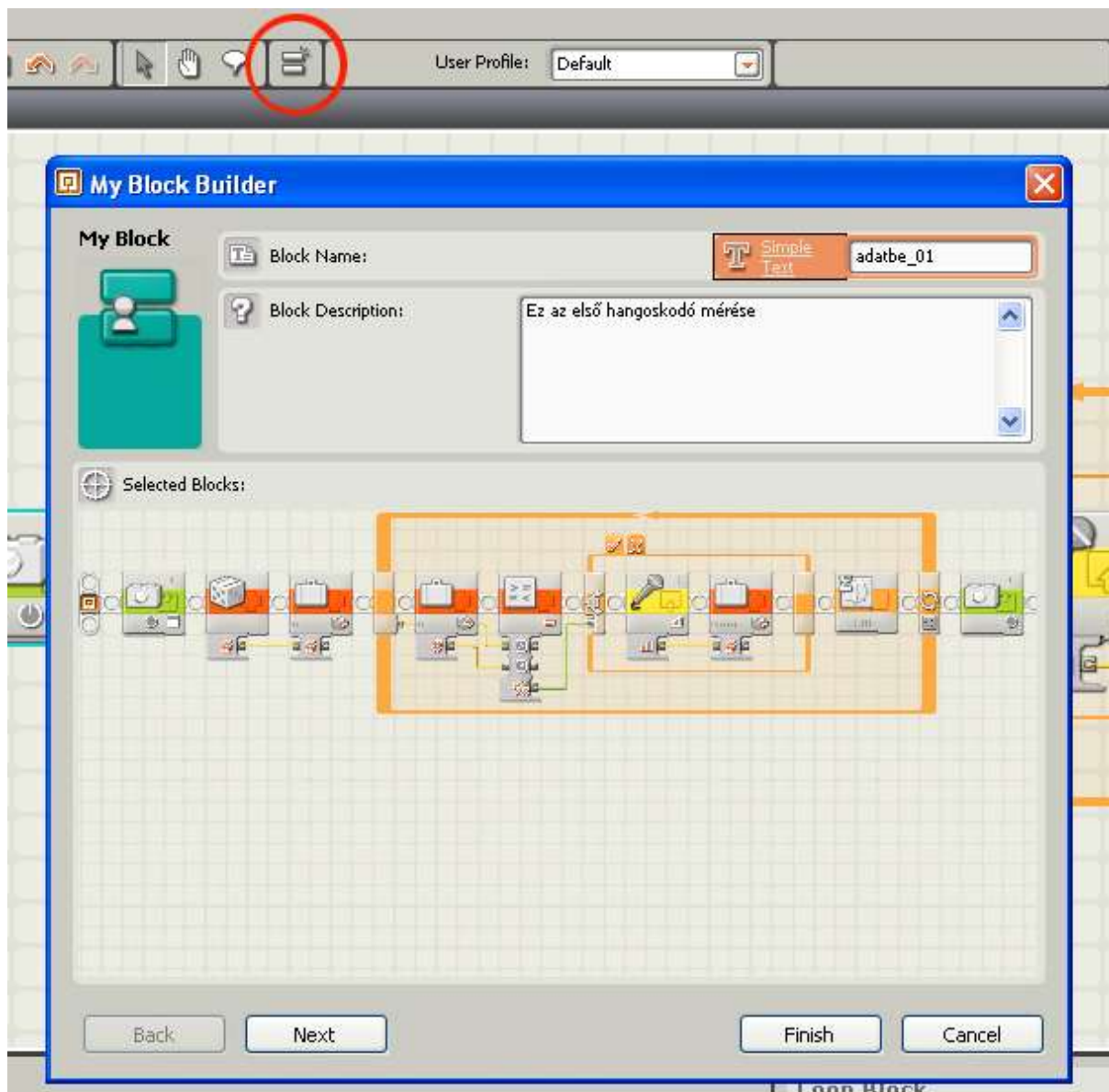
Saját blokk készítése

A forrásprogramok áttekinthetőségének érdekében, valamint a gyakran esetleg több forrásban is felhasználható utasítássorozatokat saját blokkba menthetjük, és így több kódban is felhasználhatjuk.

Most az egyszerűsítés a cél.

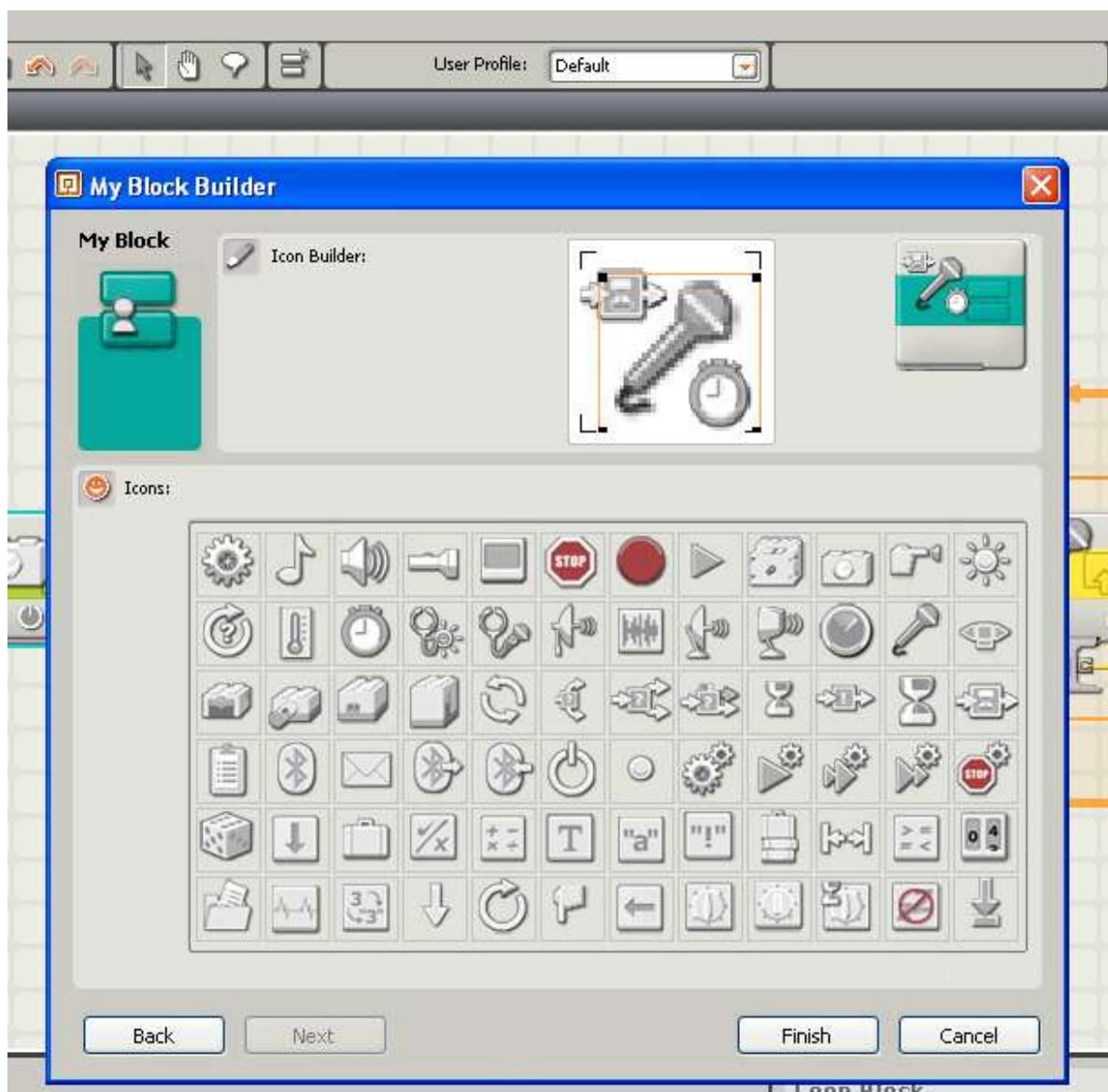
Jelöld ki a az adatbeviteli részt a lámpa be és kikapcsolásával együtt.

Kattints a piros körrel megjelölt ikonra, így kinyílik a blokk készítő ablak!



Add a blokk nevének: `adatbe_01` esetleg valami leírást (Description) is adhatsz arról, hogy mit csinál a blokk.

Kattints a Next gombra.



Az ikonok ábráiból építs valami ikon-t a saját blokkodnak is.

Kattints a Finish gombra.

☞ Ne ijedj meg, csak keresd meg a szerkesztő területen a forrás program elejét. Valahol balra lehet. Ezt fogod látni!



A blokk el lett mentve a következő könyvtár struktúra mélyére:

Dokumentumok\LEGO Creations\MINDSTORMS Projects\Profiles\Default\Blocks\My Blocks
Ha szerkeszteni kell, vagy csak meg akarod nézni, akkor meg kell nyitni.

Állj át a Custom palette-ra.

☞ A My blocks csoportban megjelent az Adatbe_01 nevű blokk.



Ezt ne most ne válaszd ki, vagy ha rákattintottál nyomj egy ESC-t.

Kattints inkább duplán a forráskódban az adatbe_01 ikonjára, ekkor megnyílik egy új szerkesztőablakban a forráskód.



Mentsd el másként (File → Save As), legyen a neve Adatbe_02.rbt. A mentési útvonal egyelőre maradjon.

☞ Ezt fogod átalakítani a második hangoskodó hangerejének megméréseire.



Most már a szerkesztőben az adatbe_02.rbt szerkesztését folytathatod. Itt kell kicserélned az elágazásban a mérés eredményét tároló változót. Az Elsohang változót a Masodikhang változóra.

Töröld az Elsohang változót.

Illessz be a helyére egy változó blokkot.

☞ Most meglepetten tapasztaltad, hogy a listából nem tudod kiválasztani a Masodikhang változót.

Hozd létre újra a Masodikhang változót.

☞ Ügyelve nehogy betűt tévessz, mert ha főprogram és a blokk változó neve nem egyezik, akkor az két különböző változó, tehát ha beleteszel az egyikbe egy értéket, akkor az nem lesz benne a másikban.

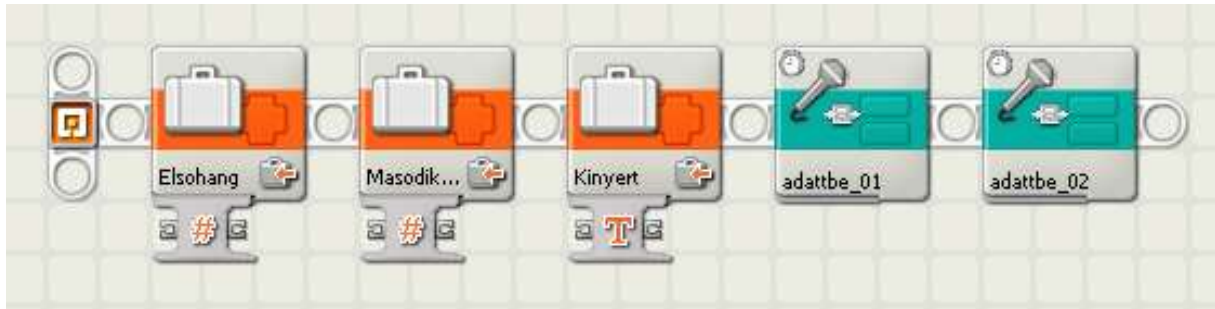
Most már ki tudod választani a listából a változó blokk paramétereként.

Állítsd át az Action-t Write-ra és kösd össze a konnektorokat.

Lépj át a hangoska szerkesztőbe!

Szúrd be az adatbe_01 blokkot az adatbe_02 elé!

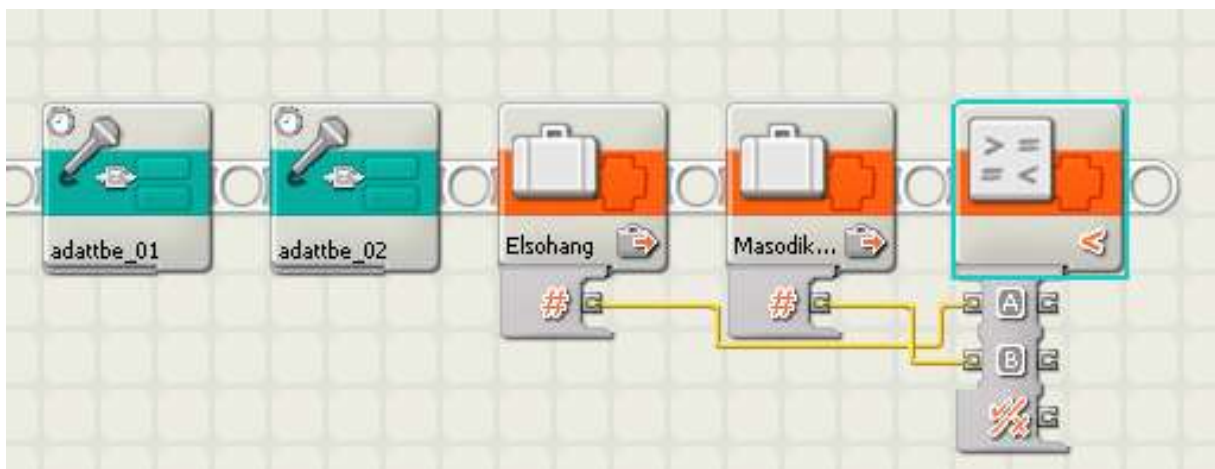
☞ Figyelem! Nem írtam el a blokk nevét ellenőrizd. Azzal, hogy másként mentetted az adatbe_01-t, lecserélted adatbe_02-re a főprogramban is a blokkot.



Következik a feldolgozó rész.

Össze kell hasonlítanod az Elsohang változó tartalmát a Masodikhang változó tartalmával és az összehasonlítás eredményétől függően kell a Kinyert változót feltölteni.

Illeszd be a hangoska forrásprogramba az Elsohang és Masodikhang változót és egy összehasonlító utasítást.

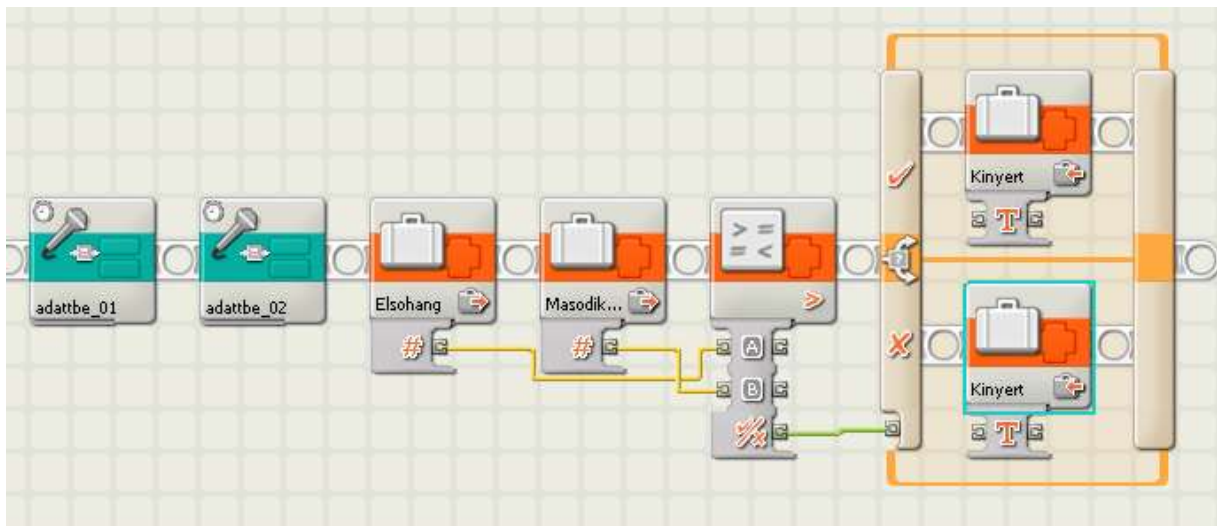


Kösd össze a konnektorokat. Elsohang „A” konnektorba, a Masodikhang „B” konnektorba.

Állítsd át az összehasonlító utasítás relációját Greater than (nagyobb, mint) állásba.

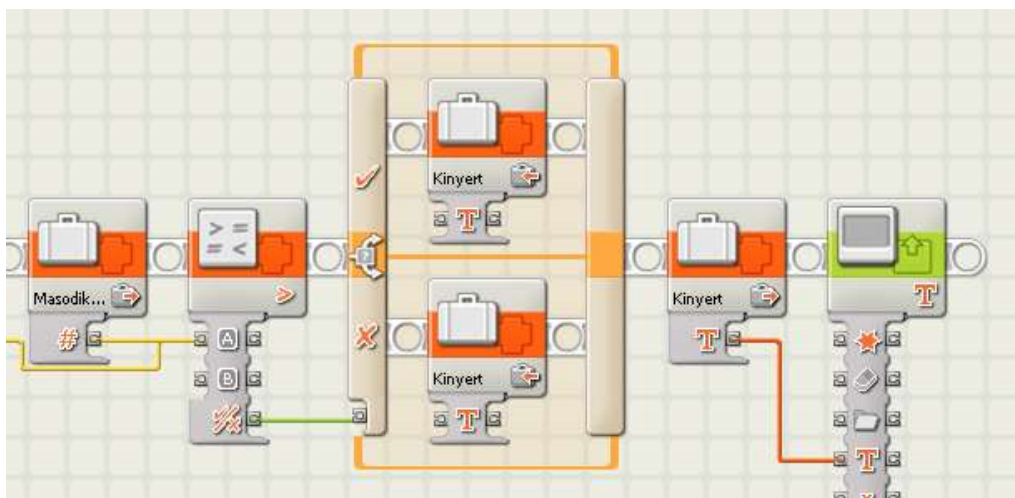
☞ Ez a következő relációnak felel meg $A > B$.

Illessz be egy érték vezérelte (Control: Value) elágazást logikai (Logic) típusút és kösd bele az összehasonlítás eredmény konnektorát az elágazás konnektorába.



Az igazágra és a hamis ágra is a Kinyert változót illeszse be, Action Write és a Text mezőbe az igaz ágon (felső) „1. hangosabb” szöveget írja, míg a hamis ágon a Text mezőbe a „2. hangosabb” szöveget írja.

Már csak az eredmény kiírása maradt hátra!



Szűrj be egy változót, állítsd át Kinyert nevű változóra és az Action-t Read-re.

Szűrj be egy display utasítást. Állítsd át a tevékenységet (action) Text-re. Csatlakozó készletét lenyitva kösd össze a változó kimenetével a Text bemenetet.

Ments és töltsd fel a robotra.

Főnök

Rendezd be a játékosokat. Figyelj. Mert a mikrofon elég érzékeny, ezért könnyedén mindkét csapat 100-as értéket érhet el.

Magyarázd el a szabályokat. Lámpa felgyulladás után első hangoskodó hangoskodik addig amíg a lámpa el nem alszik. Második csapat ugyan így.

Állítsd a hangoskodókat egyenlő távolságra a robottól és indítsd el a programot.

Ha lefutott a program akkor, már látod a problémákat. Foglald össze:

- ☞ a lámpa felkapcsolása az indítás után túl gyors.
- ☞ az első hangoskodás vége nem látszott. Nem kapcsolt le a lámpa látszólag.
- ☞ a végén meg nem látszott az eredmény.

Tegyél javaslatokat a megoldásra:

- ☞ tegyél az első beolvasás elé egy várakozást kb. 3 mp-t.
- ☞ tegyél a két beolvasást közé is egy várakozást, kb 5 mp-t.
- ☞ a végére pedig egyelőre szintén egy várakozást, hogy legyen idő meg nézni az eredményt.

☞ Persze az idővárakozások helyett később valamely érzékelőre várást is be lehet tenni (pl: nyomásérzékelő).

Javítsd ki a hibát az algoritmusban a tervezővel és a programban az operátorral.

Most már lehet tesztelni.

Amennyiben egyéb hibás működéssel találkoztl, akkor kénytelen vagy az operátorral együtt lépésről lépésre végignézni a kódot. Ez a „fehér doboz” teszt módszer egyik lépése, a kód ellenőrzés.

Ha megtaláltátok a hibát, akkor jöhet a „fekete doboz” módszer. Elindítod a programot és futtatod. A módszer lényege, hogy a program által várt beviteli adatokat változtatod.

Először csak az első hangoskodó hangoskodik. A második, akkor se, ha rákerülne a sor. Ha program az elsőt hozza ki eredményül, akkor rendben. Ha nem akkor újra a „fehér doboz” módszer, vagyis kódvizsgálat.

Másodsor az első hangoskodó nem hangoskodik a második hangoskodó viszont igen. Ha program a másodikat hozza ki eredményül, akkor rendben. Ha nem akkor újra a „fehér doboz” módszer, vagyis kódvizsgálat.

Harmadszor mindketten csöndben maradnak. Vajon miért a második nyer?

A harmadik teszt azt mutatja, hogy a program már szintaktikailag hibátlan, de szemantikailag hibás, hiszen az egyenlő hangerejű hangoskodás esetén a másodikat hozza ki győztesnek. Ez egy kicsit egyenlőtlenek a győzelmi feltételek.

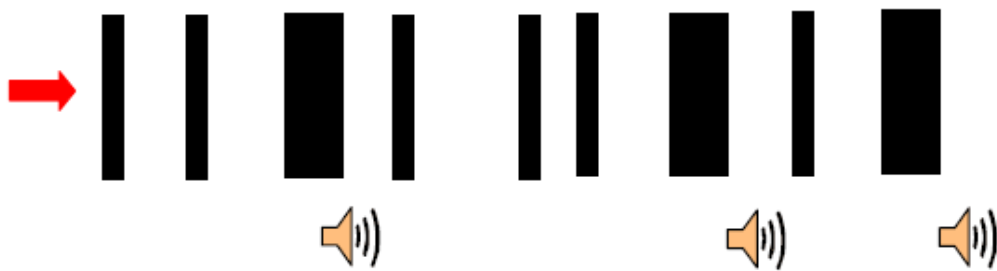
Főnök a feladatod:

A csapattal együtt ezt a problémát megbeszélve a program továbbfejlesztésének lehetőségeit feltárjátok és módosítátok úgy, hogy egyenlő esélyekkel induljon a két hangoskodó.

Gyakorló feladatok

- 8.2. ☞ A robot véletlenszerűen sorsol az 1-3 közötti számot. A kisorsolt számot kiírja a kijelzőre és ki is mondja angolul.
- 8.3. ☞ Írjon programot, amelyet a robot végrehajtva egy 1 és 10 közötti véletlen számot sorsol és ír a képernyőre. Azt is írja a képernyőre, hogy a szám páros, vagy páratlan-e.
- 8.4. ☞ Írjon programot, amelyet végrehajtva a robot a képernyőn egyesével növekedő számokat jelenít meg (a régi számot mindig törli a képernyőről)! A számok növekedését az ütközésérzékelő benyomása szabályozza.
- 8.5. ☞ Írjon programot, amelyet végrehajtva a robot a képernyőn egyesével növekedő számokat jelenít meg (a régi számot mindig törli a képernyőről) tízig, majd utána egytől újra kezdi a számlálást! A számok növekedését az ütközésérzékelő benyomása szabályozza.
- 8.6. ☞ Írjon programot, amelyet végrehajtva a robot nyomásérzékelőjét figyeli! Minden benyomás után írja a képernyőre, a „páros” vagy „páratlan” szavakat minden benyomás után váltakozva. Mindezt addig ismételje, amíg a fényérzékelőre erős fénnel rá nem világítunk.
- 8.7. ☞ Írjon programot, amelyet végrehajtva a robot előre halad és 100 ms-onként mintát vesz a fényérzékelőjével és azt kiírja a képernyőre egymás alatti sorokba! Mindezt 6-szor ismételje!
- 8.8. ☞ Írjon programot, amelyet a robot végrehajtva addig forog, amíg ultrahang szenzorával meg nem lát 20 cm-es távolságon belül valamit, vagy 3 másodpercig. Ezután álljon meg!
- 8.9. ☞ A robot küldetése legyen az, hogy tárgyak mellett elhaladva a képernyőjén a tárgyaktól mért legkisebb távolságot jelenítse meg.

Pl.:

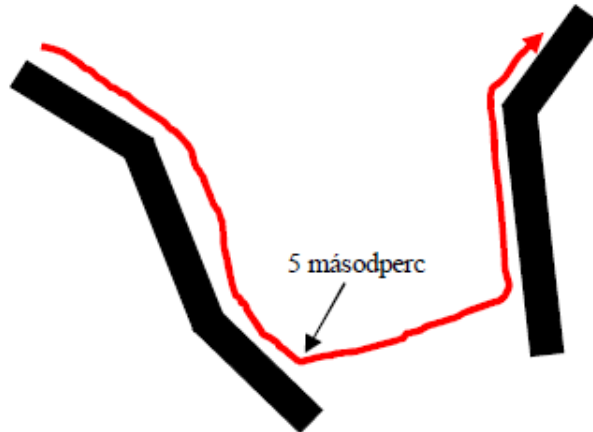


9.10. Írjon programot, amelyet a robot végrehajtva a robot váltakozva fekete-fehér színű vonalat követ egyetlen fény szenzorával! Az alapszíne jól megkülönböztethetően különbözik a feketétől és fehértől is.

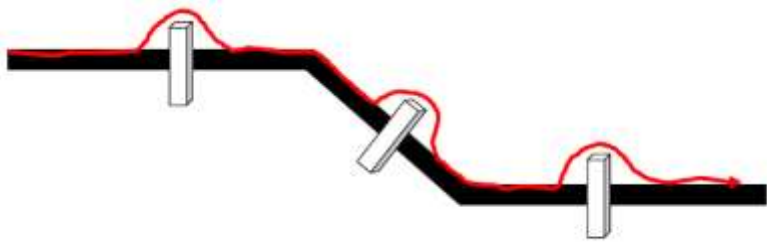
Pl.:



9.11. Írjon programot, amelyet végrehajtva a robot egyetlen fény szenzorával 5 másodpercig követ egy alaptól jól megkülönböztethető színű csíkot, majd adott pozícióba mozog, ahol ismét követni kezdi a csíkot! Lásd az ábrát!



9.12. Írjon programot, amelyet a robot végrehajtva egyetlen fény szenzorával követi az alaptól jól megkülönböztethető színű csíkot (nem feltétlenül egyenes), amelyen az ultrahang szenzorral észlelhető akadályok vannak. Egy ilyen akadályhoz érve a robotnak valamelyik oldaláról megkerülve az akadályt a csíkra visszatérve kell folytatnia az útvonalkövetést. Lásd ábra!



Bibliográfia

A feladatokat és az ötletek a Bányai Júlia Gimnázium weboldalán található dokumentumok felhasználásával készítettem: www.banyai-kkt.sulinet.hu

Kovács László: Programozás-elmélet

Daniele Benedettelli: Programming LEGO NXT Robots using NXC

Weboldalak:

<http://robotics.benedettelli.com/>

<http://bricxcc.sourceforge.net/nbc/nxcdoc/index.html>

<http://mindstorms.lego.com/en-us/Default.aspx>

<http://www.nextwork.hu/tutorial>

<http://ldd.lego.com/>

<http://www.jataka.hu/rics/lego/index.html>

http://www.ortop.org/NXT_Tutorial/index.html

Versenyek:

www.banyai-kkt.sulinet.hu

<http://sagv.gyakg.u-szeged.hu/fil/>

<http://sagv.gyakg.u-szeged.hu/szumo/>