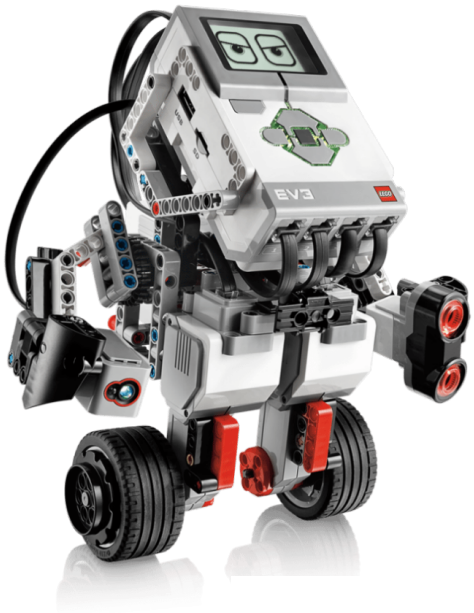
Belépő a tudás közösségébe

Lego Mindstorms EV3 robotok programozása II.

Szakköri segédanyag tanárok számára

Solymos Dóra, Solymosné Barbalics Dóra Krisztina



SZÉCHENYI 2020
MAGYARORSZÁG KORMÁNYA
Európai Unió
Európai Szociális Alap
BEFEKTETÉS A JÖVŐBE

Lego Mindstorms EV3 robotok programozása II.

Szerzők

Solymos Dóra, Solymosné Barbalics Dóra Krisztina

Felelős kiadó

ELTE Informatikai Kar

1117 Budapest, Pázmány Péter sétány 1/C.

ISBN szám

ISBN 978-963-489-280-9

A kiadvány „A felsőoktatásba bekerülést elősegítő készségfejlesztő és kommunikációs programok megvalósítása, valamint az MTMI szakok népszerűsítése a felsőoktatásban” (EFOP-3.4.4-16-2017-006) című pályázat keretében készült 2020-ban.

Tartalomjegyzék

1. Bevezető

Az oktatásban való innováció, megújulás fontos eszköze a hatékony tanulási-tanítási folyamatnak. Az informatikában ez különösen lényeges, hiszen ez az egyik legdinamikusabban változó tudományág. Az oktatásnak lépést kell tartania ahhoz, hogy a diákoknak naprakész tudást tudjon biztosítani. Nem csak tartalmában, de módszereiben is nyitottnak kell lennie a változásra. Ennek egy jó eszköze a robotok világa.

A programozás tanítása számos készséget, képességet fejleszt. Nemcsak azoknak jó lehetőség ez, akik informatikai pályán képzelik el életüket, hiszen logikus gondolkodásra, algoritmizálási képességre, tudatos tervezésre mindenkinek szüksége lesz a felnőtt életben. A robotokkal közelebb hozhatjuk a diákokhoz a programozás talán elsőre idegen témakörét. Az órák játékossá vállnak, miközben a diákok szinte észrevétlenül sajátítanak el jövőjüket meghatározó ismereteket.

A LEGO© robotok nem csak a programozás világával való ismerkedést teszik lehetővé, biztosítják a gyerekek számára az oly szórakoztató építés élményt is. A több száz LEGO© elemből szabadon építkezhetnek, az okostégla, motorok, érzékelők felhasználásával, miközben kézügyességük és kreativitásuk is fejlődik. Ez a tevékenység alkalmas csoportmunkára is, mely által fejleszthetjük diákjaink kooperációs készségét [1].

Tananyagunk második részével azon pedagógus kollégáink munkáját szeretnénk segíteni, akik haladó szintű csoport oktatását tűzték ki célul. Ez a tananyag épít a Lego Mindstorms EV3 robotok programozása című könyvünkben szereplő ismeretanyagra [2], így ennek ismeretét javasoljuk a második rész felhasználása előtt. A következő 9 szakköri alkalmat a 12-18 éves korosztálynak ajánljuk.

A szerzők

2. A szakkör felépítése

A szakköri anyagot a LEGO© Mindstorms EV3 robotot már alapszinten ismerő tanároknak és diákjaiknak (főként 12-18 éveseknek) készítettük. A szakkör 9 db 90 perces órát foglal magába, melyeken a diákok kettesével használnak egy-egy előre felépített robotot.

Eszközszükséglet 10 diákra és egy tanárra:

- 11 db asztali számítógép vagy laptop egérrel
- 5 db LEGO© Mindstorms EV3 összerakva
- 5 vagy 10 db USB kábel a számítógép és a robot összekötéséhez
- 1 db projektor

2.1. A könyv felépítése, jelölések

Minden szakköri alkalom során egy új szenzort vagy funkciót ismerünk meg, így a foglalkozások egy új elem bemutatásával kezdődnek, majd feladatmegoldással folytatódnak. A foglalkozásokon rendelkezésre álló időt igyekeztünk jobban beosztani, mint az előző tananyagban és ezért inkább több kisebb tananyagmennyiséget tartalmazó szakköri anyaggal töltöttük fel ezt a segédanyagot.

Ebben a tananyagban csupán az első foglalkozás elejére készítettünk Kahoot![1] -ot, hogy a haladó ismeretek elsajátítása előtt rendszerezzük az eddig megtanultakat. A teszt kérdéseit ebben a tananyagban is narancssárga szegélyű dobozba helyeztük (olyanba, mint például ez itt).

A tananyag online jellege és terjedelme miatt kattinthatóvá tettünk mindent, amit lehetett, hogy megkönnyítsük a navigálást a dokumentumban. A szövegben elhelyezett ábra- és táblahivatkozások sorszámára kattintva az adott részhez ugrik a dokumentum. A feladatok szövegére kattintva megnézhetitek a megoldásokat, és ha a megoldásoknál lévő dőlt betűs feladatokra kattintotok, akkor visszajuttok az eredeti feladatleírásokhoz. A tananyag lábjegyzeteiben szereplő külső hivatkozásokra kattintva, megnézhetitek az adott weboldal(aka)t, amelyek sok érdekes információt, példát tartalmazhatnak még az általunk felhasználtakon kívül. Továbbá, ilyen aláhúzással jelöltük a folyó szövegben elhelyezett hivatkozásokat, amik szintén kattinthatóak (kivéve ezt, mert ez csak egy minta :D).

A feladatokat külön oldalra tettük, ha esetleg kinyomtatva szeretnétek felhasználni őket, akkor ne kelljen az átszerkesztéssel bajlódni.

A fentihez hasonló lila/rózsaszín szegélyű szövegdobozokban szeretnénk megosztani Veletek a saját tapasztalatainkat, véleményünket ill. egyéb fontos információkat, mint amit itt feljebb olvashattok.

[1] `https://kahoot.com/`

Szerzők és munkáik:

Solymos Dóra: A szakkör felépítése, Szakköri alkalmak (1., 4., 5., 9.)

Solymosné Barbalics Dóra Krisztina: Bevezető, Szakköri alkalmak (2., 3., 6., 7., 8.), Az EV3 belső memóriájának kezelése

2.2. Az általunk használt tesztrobot

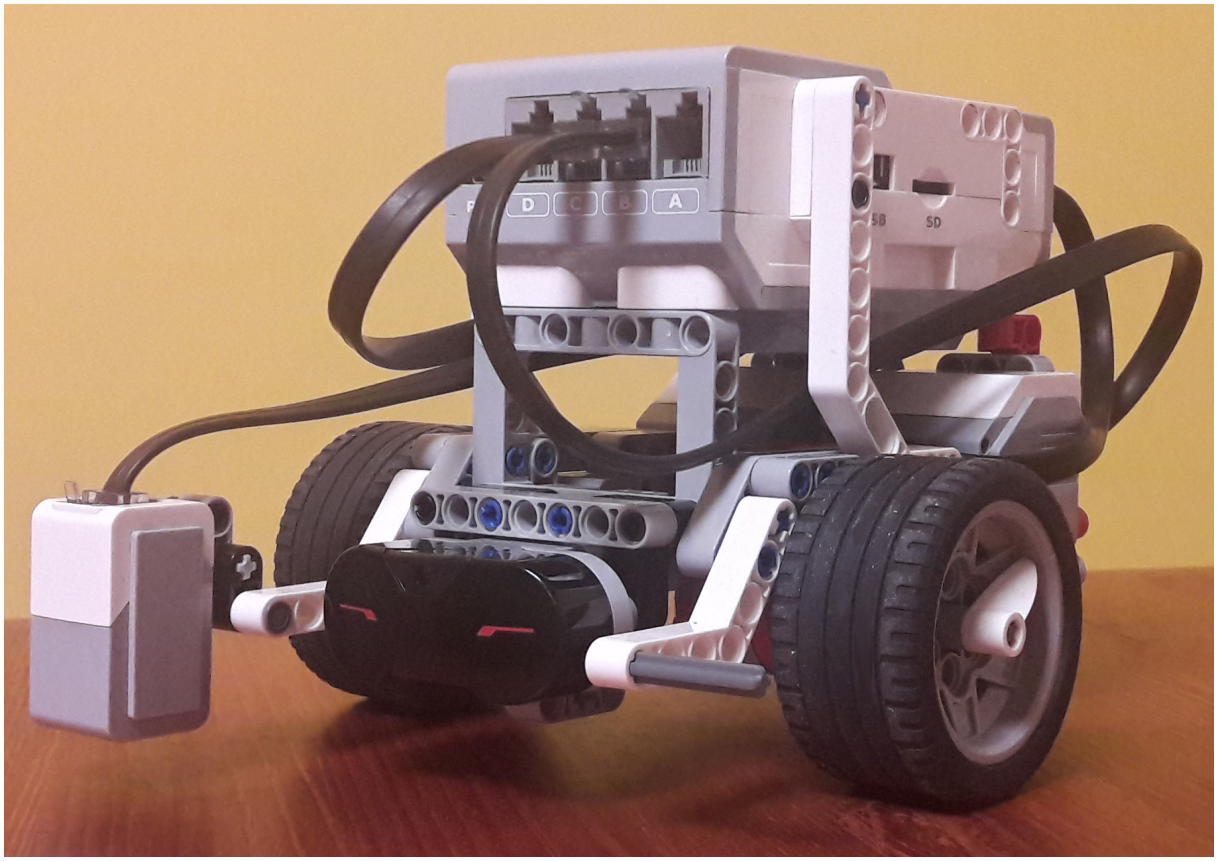
A szakköri alkalmak során a már megszokott felépítésű (1. ábra) robotunkat használtuk, és mindig az aktuálisan használt szenzorral egészítettük ki. Ez az új szenzor vagy valamelyik első szenzor helyére került vagy közvetlenül a tégla mellé vagy a hőmérsékletérzékelő esetében nem került rögzítésre.
A robot építési útmutatói az alábbi honlapon érhetőek el (a „Building Instructions for Robot Educator" rész alatt) : `https://education.lego.com/en-us/support/mindstorms-ev3/building-instructions#robot` vagy a programozó szoftver Education verziójának Lobby részéből. Mindkét felületen elég sok hasonló útmutató van fent, mi a következőket használtuk:

„Color Sensor Down",

„Driving Base",

„Touch Sensor Driving Base",

„Ultrasonic Sensor Driving Base".



1. ábra. A robot (Forrás: [2])

3. Szakköri alkalmak

3.1. 1.alkalom

Mielőtt belekezdünk a haladóbb anyag elsajátításába érdemes lehet picit átismételni, összefoglalni az eddig tanultakat. Ehhez készítettem egy rövidnek nem nevezhető kvízt, amit akár az előző tananyagban megismert Kahoot!-tal, akár más feladatsor-készítő programokkal (Google Űrlap, Redmenta, Socrative) kitöltethetünk a diákokkal. A kvízt, az előző tananyag [2] Kahoot! kvízeiből állítottam össze, nehogy véletlenül olyat kérdezzek, amit nem tanultunk. A kérdéssort lentebb tekinthetitek meg a szokásos narancs dobozban.

- *A betűvel jelölt portokhoz mit kel csatalkoztatni?*

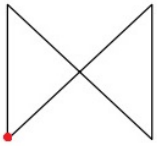
  - *motorokat*
  - *érzékelőket*

- *Ahhoz, hogy robotunk tolasson, ...*
  - *negatív tengelyfordulat értéket kell megadnunk.*
  - *meg kell fordítanunk rajta a motorokat.*
  - *negatív sebességet kell megadnunk.*
  - *a két motor sebességét ellentétes előjellel kell megadnunk.*

- *Igaz-e, hogy ha a tengelyfordulatot 90-re állítjuk, akkor a robotunk 90°-ot fog fordulni?*
  - *Igaz*
  - *Hamis*

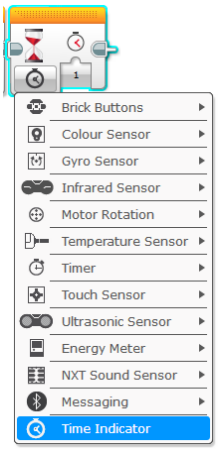
- *Mire jó a ciklus*
  - *Csak így lehet a robottal megismételtetni a mozgásokat.*
  - *Az ismétlődéseket meg tudjuk adni vele rövidebb formában.*
  - *Mindig mindent végtelenszer meg tud ismételni vele.*
  - *Csak így lehet a motort működtetni.*

- *A képen látható alakzatot (csokornyakkendőt) ki lehet rajzolni a piros pontból indulva és felfelé haladva ciklus használatával is.*



  - *Igaz*
  - *Hamis*

- *A homokórás blokk (általában) arra jó, hogy...*



  - *várni tudjon a robot pár másodpercig.*
  - *várni tudjon a robot az érintésérzékelő megnyomásáig.*
  - *várni tudjon a robot, amíg nem lát valamit.*
  - *várni tudjon a robot egy beállított esemény bekövetkezéséig.*

- *Mi a különbség az érintésérzékelő „Pressed” (1) és „Bumped” (2) állapota között?*

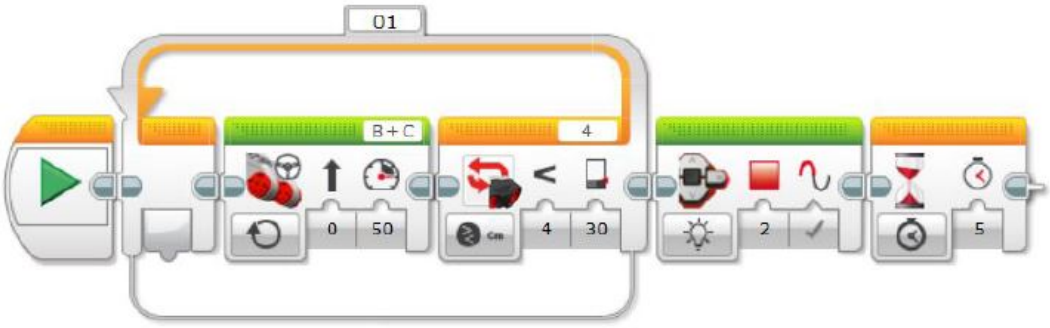
  - *Semmi, ezek lényegében ugyanazt jelentik.*
  - *A „Pressed” a benyomott állapotot jelenti, míg a „Bumped” egy benyomás és kiengedés együtteséből áll.*
  - *A „Pressed” a benyomott állapotot, míg a „Bumped” a kiengedettet jelenti.*
  - *A „Pressed” a kiengedettet állapotot, míg a „Bumped” a benyomottat jelenti.*

- *Párhuzamos programot...*
  - *egy erre való, speciális blokk segítségével tudunk létrehozni.*
  - *nem lehet ebben a szoftverben megvalósítani.*
  - *úgy tudunk létrehozni, hogy két külön indítás blokkot használunk.*
  - *„vezetékek” segítségével tudunk létrehozni.*

- *Egy párhuzamos program úgy működik, hogy...*
  - *robotunk a különböző szálakon lévő utasításokat egyszerre hajtja végre.*
  - *robotunk először azt az ágát hajtja végre a programnak, amely a szoftverben följebb van.*
  - *robotunk önkényesen eldönti milyen sorrendben hajtja végre a különböző szálakon lévő utasításokat.*
  - *az a szál fut le előbb, melynek feltétele előbb teljesül, csak aztán a többi.*

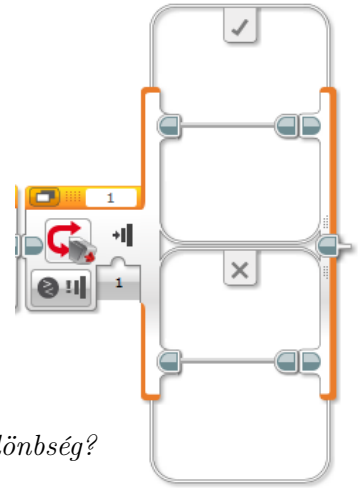
- *A színérzékelő arra jó, hogy...*
  - *visszavert fény erősségét mérje.*
  - *színeket ismerjen fel.*
  - *visszavert fény és környezetében lévő fény erősségét mérje és színeket ismerjen fel.*
  - *színesen tudjon világítani a robotunk.*

- *Ha a képen látható programot elindítjuk, a roboton mikor fog pirosan villogni a led?*



  - *Ha be van nyomva az érintésérzékelő.*
  - *Ha 30 cm-en belül észlel valamit az ultrahangos távolságérzékelő.*
  - *Ha 30 cm-en kívül észlel valamit az ultrahangos távolságérzékelő.*
  - *Ha ki van engedve az érintésérzékelő.*

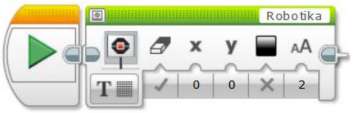
- *Elágazás szerkezet használatakor a pipával jelölt rész...*



  - *mindenképp lefut.*
  - *csak akkor fut le, ha a megadott feltétel hamis.*
  - *csak akkor fut le, ha a megadott feltétel igaz.*
  - *az X-el jelölttel párhuzamosan fut le.*

- *Mi az elágazás szerkezet és a többszálú program között a különbség?*
  - *Semmi.*
  - *Elágazásnál egyszerre csak az egyik ágba megy bele a program.*
  - *Többszálú programnál csak az egyik ágba megy bele a program.*
  - *Az elágazásnak csak két ága lehet.*

- *Mi kerül az elágazás alapértelmezett ágába, ha színérzékelőt használunk?*
  - *Az, amit akkor csinál a robot, ha feketét észlel.*
  - *Az, amit akkor csinál a robot, ha zöldet észlel.*
  - *Az, amit akkor csinál a robot, amikor egyik feltétel sem teljesül.*
  - *Az, amit akkor csinál a robot, ha fehéret észlel.*
- *Mi történik, ha az alábbi programot szeretnénk futtatni a roboton?*



  - *A program azonnal leáll, látszólag semmi nem történik.*
  - *Kiírja a képernyőre, hogy „Robotika”.*
  - *Villogni kezd a képernyőn a „Robotika” felirat.*
  - *Hibás a program, ezért el sem indul.*
- *Ki tudjuk írni az érzékelők értékét a képernyőre?*
  - *Igen*
  - *Nem*
- *A Stop program blokk mit csinál?*
  - *Az adott szálat leállítja a programban.*
  - *Kikapcsolja a robotot.*
  - *Kiírja a képernyőre, hogy STOP.*
  - *Megállítja a teljes program futását.*
- *Milyen új blokkot használtunk arra, hogy megszámoljuk az érintésérzékelő megnyomásait?*
  - *Konstans blokkot*
  - *Változó blokkot*
  - *Ütköző blokkot*
  - *Motor blokkot*
- *A változó blokk arra jó, hogy...*
  - *robotunk folyamatosan tudja változtatni sebességét.*
  - *robotunk folyamatosan tudja változtatni a téglán lévő led színét.*
  - *eltároljunk segítségével egy értéket, melyre később még szükségünk lesz.*
  - *eltároljunk segítségével blokkokat, hogy máskor könnyebben elő tudjuk állítani őket.*

3.1.1. Gyro szenzor használata

Mi az a giroszkóp és mire használják?

Érdemes megkérdezni a gyerekeket, hogy tulajdonképpen mi az a giroszkóp, tudják-e hogy néz ki, láttak-e esetleg ilyen eszközt, és tudják-e mire/hogyan használjuk. (Valószínűleg nem fogják tudni, hogy milyen eszközről van szó, így érdemes tisztázni, esetleg egy-két rövid animációt[2] megnézni a működéséről.)

2. ábra. Búgócsiga

A giroszkóp a szögelfordulás és a szögsebesség mérésére szolgáló eszköz. A giroszkóp „elődjét" már az ókori rómaiak és görögök is ismerték, viszont a most is használatos giroszkópokat csak a 19. században találták fel. A giroszkópot használhatjuk a stabilitás növelésére is, melynek legismertebb példája a bicikli kereke. Ugyanis, minél gyorsabban forog a bicikli kereke, annál stabilabb lesz. Azonban, ezen kívül használhatjuk iránytűként vagy annak kiegészítéseként, mint ahogy például az okostelefonokban, tabletekben tesszük ezt. Továbbá, számos olyan játék (2. ábra) létezik, melyben az eszköz döntésével tudjuk irányítani a játékot, vagy akár ezzel a módszerrel kereshetünk a telefonkönyvünkben. Azonban offline játékoknak is lehet a működtető elve, például a jojónak, frizbinek vagy a búgócsigának.

Mi most a szögelfordulás és szögsebesség mérésére fogjuk használni. Nézzük is meg hogyan!

A gyro szenzor működése

3. ábra. A gyro szenzor teteje.

A szenzor (elvileg) két értéket tud meghatározni. Az egyik az elfordulás szöge, amit fokban ad meg a legutóbbi nullázáshoz (reset-hez vagy kalibráláshoz) viszonyítva, a másik pedig a szögsebesség, amit fok/másodperc mértékegységben mér. A mérést egy irányban, egy tengely szerint végzi a szenzor, amely tengelyt magán az eszközön is jelöltek a készítők a tetején lévő két piros nyíllal. (3. ábra)

De ez mit jelent? - merülhet fel benned a kérdés. Gyakorlatilag azt jelenti, hogy az eszköz jobbra vagy balra fordításának szögét és szögsebességét tudja meghatározni a szenzor. Nézzünk egy példát rá! Rakd a tégla egyik oldalára két valamilyen rögzítéssel a szenzort, majd rakd az asztalra a „robotot". Ha az eszköz felemelése nélkül, egy helyben forgatod a robotot, akkor ezt meg tudja mérni a szenzor, de ha már felemeled az asztalról az eszközt forgatás közben, akkor azt nem tudja mérni. (Valójában kiírhat valamilyen értéket, mert felemelés közben picit oldalirányban is elfordulhat az eszköz, azonban nem ezt hivatott mérni.) Ezt ellenőrizheted, ha csatlakoztatod a szoftverhez a robotot és a program jobb alsó sarkában lévő menüben a *Port View* (bástya jelű) opciót választod. Itt jelennek meg a csatlakoztatott motorok és szenzorok, valamint az általuk mért értékek.

A mérés eredménye a forgatás irányától függően pozitív vagy negatív. Ha jobbra forgatod a szenzort, akkor pozitív eredményt fogsz kapni, ha balra, akkor pedig negatívat (mint matematika órán az alakzatok forgatásánál). A *fok* mértékegységben mért értékekhez nincs legfelső határ, míg a szögsebességhez

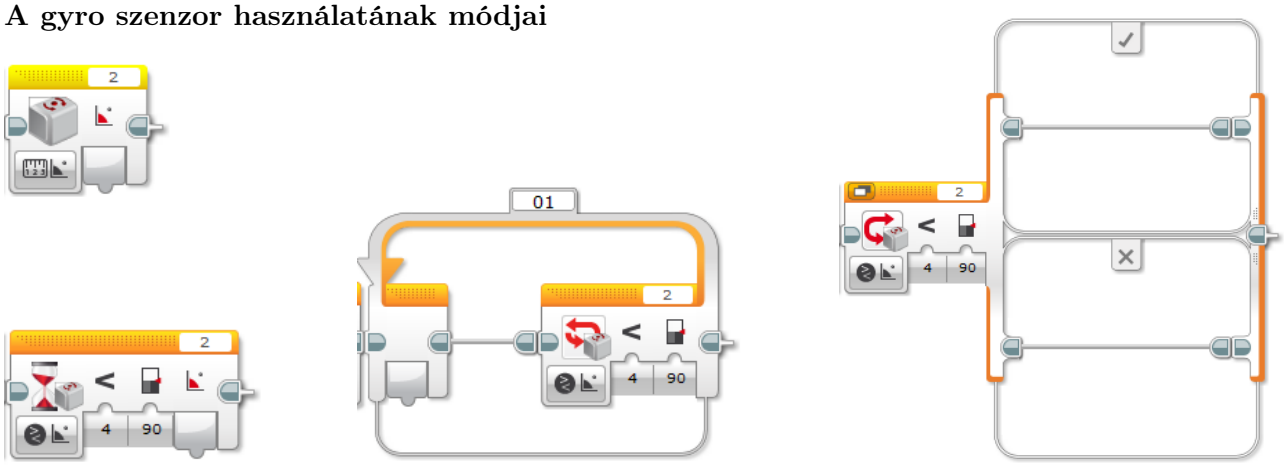
[2] `https://commons.wikimedia.org/wiki/File:Gyroscope_operation.gif`

igen, mégpedig 440 fok/ms-t[3], azonban a próbálgatások során ennél nagyobb értéket is mutatott már a szenzor.

Ez így szép és jó, de miért nem használtuk eddig? Mint ahogy más érzékelőknek, ennek is van néhány gyengesége. Először is, az EV3 gyro szenzorát nem szögmérésre találták ki, vagyis nem ez a fő funkciója, hanem a szögelfordulás mérése. Emiatt a szenzor a következőképpen méri a szögelfordulást (leegyszerűsítve): megnézi milyen sebességgel forgatod az eszközt és mennyi ideig, majd ebből meghatározza, hogy mekkora szögben fordítottad el. Pl. 9 fok/ms-es sebességgel forgattad 10 másodpercig, akkor 90 fokot fordultál. Viszont ez a sebesség nem biztos, hogy pont 9 fok/ms, amivel elérkeztünk a másik hátrányhoz, a pontatlansághoz. Tehát, lehet, hogy 8,9 vagy 9,1 vagy 9,2 fok/ms a szögsebesség, ami 89-92 fokos elfordulást fog jelenteni. Egy-két fordulás esetében ez még tűrhető hiba, azonban egy négyzet rajzolásakor emiatt már nem tudunk csak a giroszkóp által mért értékekre hagyatkozni. (Később visszatérünk arra, hogy hogyan lehet ezt kiküszöbölni.)

A gyro szenzor használata során különösen figyeljetek az irányokra! Ugyanis nagyon nem mindegy, hogy milyen irányba néz a robotodon a motor (normál helyzetben építetted be vagy fejjel lefelé) és milyen irányba akarod forgatni a robotodat. Ezen kívül a gyro szenzor helyzete is fontos és az általa mért értékek is. Ha nem működik elsőre egy program, akkor gondold át, hogy az általad elvárt forgás pozitív vagy negatív irányú és ennek megfelelően jó irányba fordul-e a robot.

A gyro szenzor használatának módjai

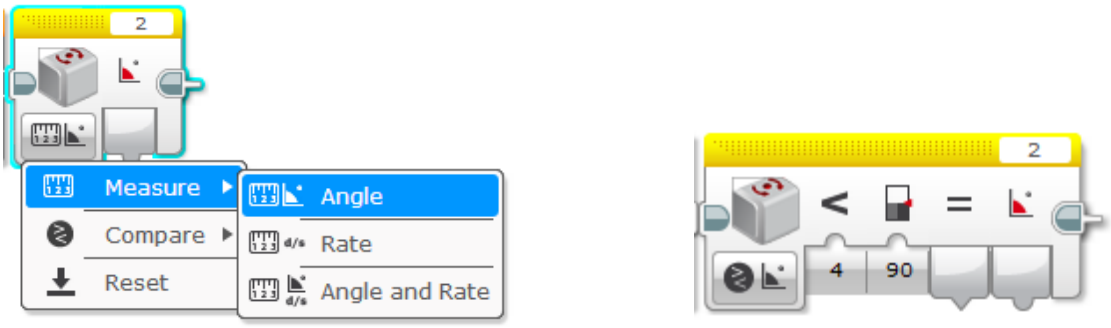


4. ábra. A gyro szenzor használata (balról jobbra) az érzékelő (fent), a várakozás (lent), a ciklus és az elágazás blokkokkal.

Azt már korábban megtanultuk, hogy a szenzorokat többféleképpen használhatjuk. A narancssárga (Flow Control) menüben többek között a várakozás, a ciklus és az elágazás programozói blokkok használata során választhatjuk a giroszkóp módot, azonban a citromsárga opciónál is találunk ehhez az érzékelőhöz programozható blokkot. Ezek láthatóak a 4. ábrán.

[3] https://education.lego.com/en-us/products/lego-mindstorms-education-ev3-gyro-sensor-/45505

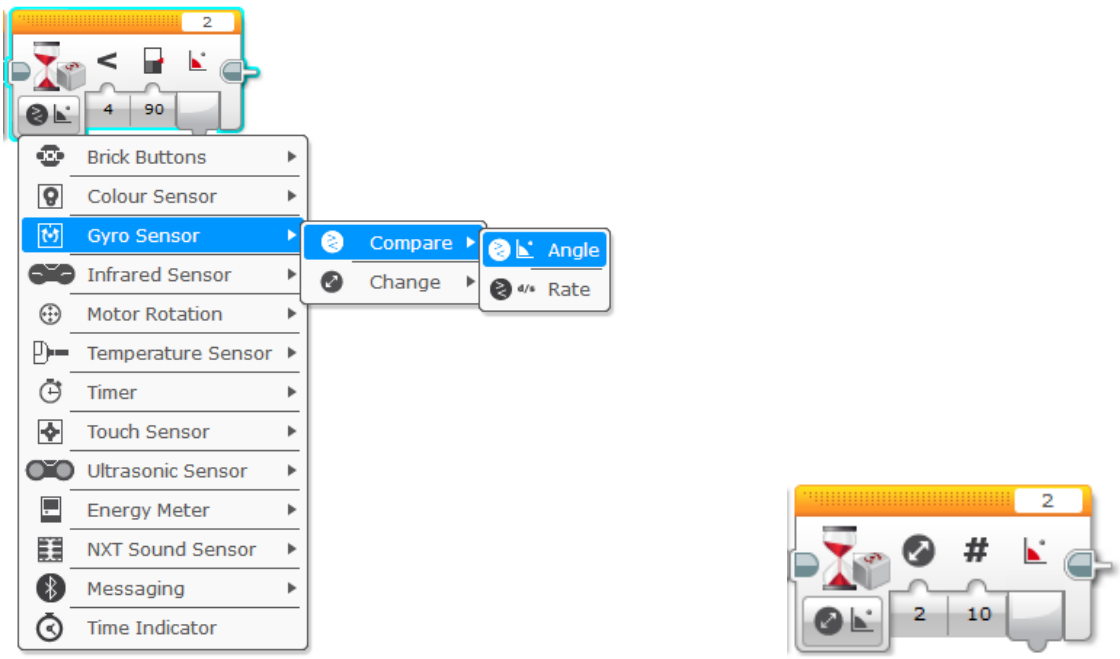
A giroszkóp érzékelő blokkjában (citromsárga) három funkció közül választhatunk, melyek az 5. ábra bal oldalán láthatóak és a következő nevekkel rendelkeznek: Measure, Compare és Reset. A Measure módban az érzékelt értékeket tudjuk továbbadni a csatlakozók segítségével. A mérés funkción (Measure) belül háromféle opciót különböztetünk meg, melyből az első az elfordulás szögét (Angle), a második a szögsebességet (Rate), míg a harmadik, egyszerre mindkét értéket továbbíthatja.



5. ábra. A gyro szenzor érzékelő blokkjának módjai és ugyanannak a blokknak a Compare módja szög mérésére állítva.

Ugyanezen blokk Comapre módjában viszont már csak kétféle opciónk van, a szög (Angle) és a szögsebesség (Rate). Itt egy általunk megadott értékhez viszonyítja a robot a mért értéket, amihez a következő adatok megadása szükséges: összehasonlítás típusa (kacsacsőr), küszöbérték. Ezen kívül találunk két kimeneti értéket is a blokkban, amiket vezetékek segítségével átadhatunk egy másik blokknak. Az egyenlőségjellel jelölt átadható érték az összehasonlítás eredménye (Compare Result) nevet kapta, amely egy igaz/hamis (logikai) értéket ad tovább attól függően, hogy az összehasonlítás igaz vagy hamis. A másik átadható érték pedig a mért szög vagy szögsebesség, a kiválasztott módnak megfelelően. Az érzékelő blokk utolsó módja a Reset, ami nem azonos az újrakalibrálással. Ez a resetelés csak annyit tesz, hogy a megjelenített értékeket átírja úgy, hogy a jelenlegi pozíció legyen a 0, a jobbra lévők a pozitívak, a balra lévők pedig a negatívak. A kalibráláshoz szükségünk van a várakozás blokkra is. Nézzük is meg, az hogyan működik!

A várakozás blokkban kétféleképpen használhatjuk a gyro szenzort, melyek a Compare és Change módok. (Ezek láthatóak a 6. ábrán.) A Compare módban, ahogy már korábban tanultuk, egy általunk megadott értékhez viszonyít a program, és addig vár, amíg az általunk megadott hatást (kisebb, nagyobb, kisebb vagy egyenlő, stb.) el nem éri. Ezzel szemben, a Change (változás) módban a program elindításakor mért értékhez viszonyított változást mérhetjük meg. (6. ábra jobb oldala) Ezen blokk első paraméterénél a változás irányát kell kiválasztanunk, ami 3-féle lehet: nő, csökken vagy bármilyen irányú (kétirányú nyíl). A második paraméternél, amit egy hashtag (régi nevén kettős kereszt) jelöl, a várt változás mértékét kell megadni, tehát egy számot. Itt is van egy mért érték átadó „paraméterünk”, ami a korábban kiválasztott módnak megfelelően szög vagy szögsebesség.



6. ábra. A giroszkóp várj blokkal történő használatának módjai.

A ciklus és elágazás blokkokban ugyanazokkal a beállításokkal tudjuk használni a gyro szenzort, mint a várakozás blokkban, azonban itt csak a Compare mód választható a kettő közül.

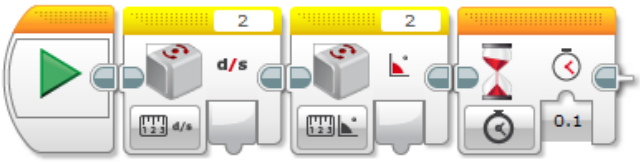
A gyro szenzor kalibrálása

Első és legfontosabb kérdés: Mikor lehet szükség kalibrálásra? Nos, ha a szoftver jobb alsó sarkában lévő Port View (bástya jelű) ablakban észreveszitek, hogy a gyro szenzor értéke álló helyzetben is változik, akkor újra kell kalibrálni a szenzort. Egy másik ok a kalibrálása az, amikor a gyro szenzor használatával próbáltok egyenes mozgást elérni a robottal, de az észrevehetően az egyik irányba „húz".

Mielőtt elmélyednénk a kalibrálás rejtelmeiben, tisztázzuk mi a különbség a kalibrálás és a resetelés között. Ahogy korábban is említettem, a resetelésnél egyszerűen átírja a szenzor a jelenlegi „pozíciót" (a szögelfordulást) 0-ra. Ezt hajtja végre a sárga érzékelő blokkban lévő Reset mód.
A kalibrálás kifejezés, habár ismerősen cseng, a gyerekek számára nem biztos, hogy érthető. Helyette, az angol „calibrate" kifejezés egy másik fordítását, a hitelesítést is használhatjuk. Ez a kifejezés magát a folyamatot is jobban jellemzi, mint a kalibrálás, ugyanis a gyro szenzor akkor végzi el a kalibrálást, ha mindent mozdulatlannak érzékel. Ugyanis ekkor tudja csak kijelenteni (hitelesíteni), hogy a szenzor nem mozog, tehát a szögelfordulás és a szögsebesség is 0.

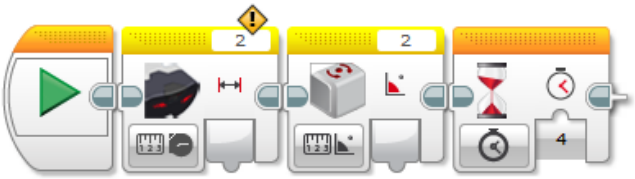
A szenzort kétféleképpen lehet kalibrálni, hardveresen és szoftveresen. A hardveres megoldás során a bekapcsolt téglából ki kell húzni a szenzor kábelét, majd pár másodperc után újra csatlakoztatni kell. Ez kevésbé megbízható megoldás (néhány versenyző nem is javasolja ezt), ugyanis az újraindítás során megmozdíthatjuk a szenzorunkat és ezáltal nem érjük el a várt eredményt. A szoftveres kalibrálás megbízhatóbb és többféle megvalósítása is létezik, melyekből csupán néhány rövidebbet mutatok be. (Ezek olyan kódok, amelyeket minden olyan programban lefuttathatunk, amelyek használnak gyro

szenzort. Azonban arra figyelni kell, hogy a hitelesítés (konfigurálás) során ne mozogjon a robot.)

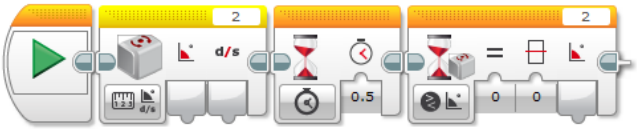


7. ábra. A gyro szenzor kalibrálásának első verziója.

8. ábra. A gyro szenzor kalibrálásának második verziója.



9. ábra. A gyro szenzor kalibrálásának harmadik verziója.



10. ábra. A gyro szenzor kalibrálásának negyedik verziója.

A 7. és a 8. ábrákon látható megoldásokban is mozdulatlanul kell hagyni az eszközt, azonban nem emiatt fog kalibrálódni a szenzor, hanem a módok változtatása miatt. Ilyen módváltoztatás a 7. ábrán a szögsebesség mérése után a szögelforudlás mérése, vagy a 8. ábrán a második blokkban a két különböző érték lekérése. Itt merül fel a kérdés: szinte ugyanaz a két kód, akkor miért van szükség mindkettőre? Azért, mert nem minden szenzorra működik ugyanaz a kalibrálási mód/kód. Ha a képen lévők nem működnek, akkor ki lehet próbálni azzal a módosítással őket, hogy az elejére beszúrunk néhány másodperc várakozást. Ezáltal biztosan nyugalmi állapotban lesz a robot, ill. egy másik opció a végén lévő idő növelése, amellyel a kalibrálási időt tolhatjuk ki. (Vannak olyan szenzorok, amik picit lassabbak, így szükségük lehet több időre.)

a 9. ábrán lévő kódban azért történik meg a kalibrálás, mert először a gyro szenzor portján az infravörös érzékelőt olvastatjuk be és csak utána váltunk szögelforudlás mérésre. Ezt követően pedig csak várni kell, hogy kalibrálódjon a szenzor és nem szabad elfelejteni, hogy itt sem mozgatjuk a robotot.

A 10. ábra kicsit érdekesebb az utolsó várakozás blokk miatt. A kód készítője (Builderdude35[4]) azt tapasztalta, hogy nem minden esetben számértékre kalibrálódik a szenzor. Ennek a kiküszöbölésére találta ki azt, hogy addig várakoztatja a szenzort (kalibráltatja), amíg nulla értéket nem ad eredményként.

A bemutatásra nem kerülő, hosszabb kódokban az eredmények összehasonlításával figyelik a szenzor értékeit és addig kalibrálják, amíg egy számértéket nem ad eredményül. A kód részletesebb leírása az [3] diasorban található.

[4]YouTube felhasználó, aki az EV3 szett felhasználásáról készít angol nyelvű tutoriál videókat. (Sokan használják a videóit az FLL és WRO versenyekre való felkészüléshez.)

3.1.2. Feladatok

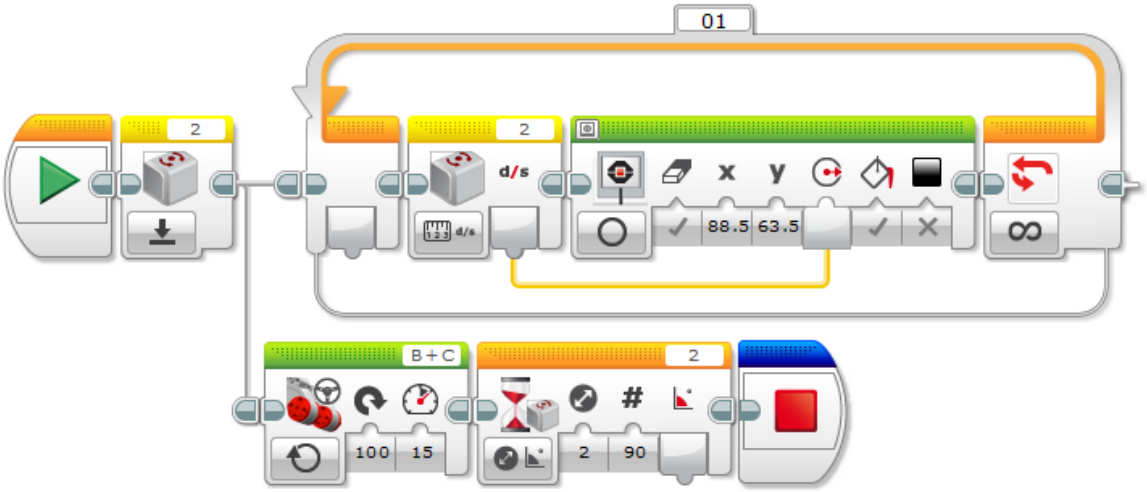
1. Írd ki a kijelzőre az aktuális szögelfordulást! Ügyelj arra, hogy mindig az aktuális értéket jelenítsd meg!

2. Jeleníts meg a kijelző közepén a robot sebességével azonos szélességű téglalapot! (A téglalapot színezd ki, hogy jobban látszódjon az eredmény! A teszteléshez kézzel mozgasd a robotot vagy állíts be valamilyen mozgást a képernyőre rajzolás mellé!)

3. Készíts programot, melyben a robot egy tengelyfordulás után fordul 90°-ot, majd ismét megy előre egy tengelyfordulást! A fordulást a várakozás blokk (gyro szenzor módjának) segítségével hajtsd végre!

4. Hogyan lehetne a gyro szenzor más blokkját használva 90°-ot fordulni?

5. A következő programot futtatva a robot 90°-ot fordul jobbra, miközben a kijelzőre kirajzol a szögsebességének megfelelő sugarú kört. Próbáld ki! Mi lehet az oka annak, hogy nem a korábban leírtakat hajtja végre a robot?



6. Készíts programot, amelyben a robot pontosabb egyenes vonalú mozgást végez! Ügyelj arra, hogy a program során legfeljebb 1 értékkel térjen el a 0-tól a szögelfordulás mértéke!

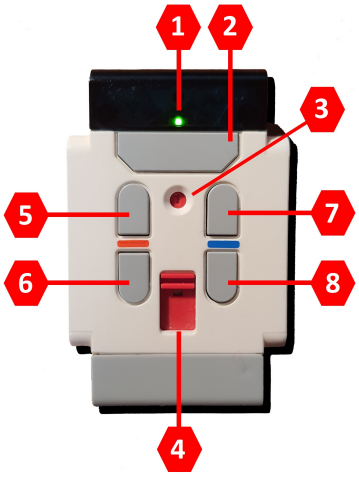
3.1.3. Az infravörös szenzorhoz tartozó jeladó használata

A jeladó

A jeladó a 11. ábrán látható kis eszköz, amely kettő darab AAA típusú elemmel működik. (Ha túl könnyűnek érződik, akkor valószínűleg nincs benne elem.) Az angol neve Remote Infrared Beacon, amit úgy is fordíthatunk, hogy külső infravörös jeladó. (Én csak jeladóként fogok rá hivatkozni a továbbiakban.)

Ahogy a neve is mutatja, az eszköz fő funkciója a jeladás, amire elég sok blokk is épül a szoftverben, de ezen kívül távirányítóként is használhatjuk az eszközön lévő négy gomb segítségével. Első ránézésre több, mint négy gomb van az eszközön, ami igaz, azonban nem mind programozható. Nézzük meg, hogy melyik gombnak/kapcsolónak milyen funkciója van! Ehhez a 11. ábrán lévő jelöléseket fogom használni.

1. bekapcsolt állapotot jelző fény
2. ki-, bekapcsoló gomb (Beacon Mode)
3. aktuálisan használt csatorna
4. csatornaválasztó kapcsoló
5. gomb 1
6. gomb 2
7. gomb 3
8. gomb 4

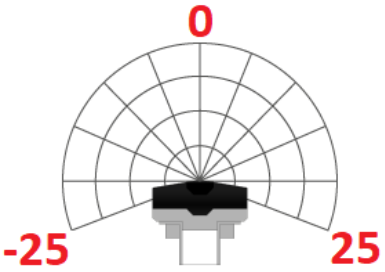


11. ábra. A külső infravörös jeladó.

Ezek közül az 5-8. számokkal jelölt gombok programozhatóak, a többinek van gyári funkciója. Ezek közül a csatornaválasztó az, ami fontos. Ez arra szolgál, hogy megkülönböztessük az azonos helyen lévő infravörös érzékelő és jeladó párosokat a működésük közben. A csatornaválasztó kapcsolót függőlegesen mozgatva tudjuk kiválasztani az aktuálisan használandó csatornát, amit a programozó felületen szintén be kell állítani használat előtt. Amint az imént írtam a távirányító az érzékelővel együtt, párban használható csak, önmagában nem lehet használni.

A jeladó használata (Beacon mód)

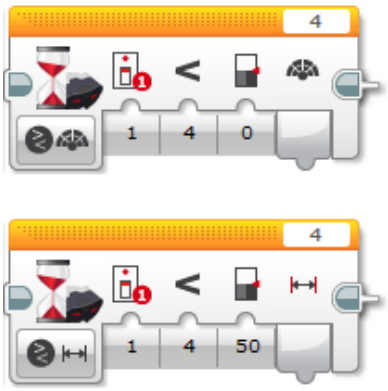
Az infravörös érzékelő a 12. ábrán látható tartományban észleli a jeladóként használt eszközt. Az érzékelő háromféle értéket tud megadni a jeladóról: a relatív távolságát (Proximity), az irányát (Heading), valamint azt, hogy érzékeli-e a jeladót. Ezeket az értékeket a várakozás, a ciklus, az elágazás és az érzékelő blokkok használata közben tudjuk felhasználni.



12. ábra. Az infravörös érzékelő érzékelési tartománya. (Forrás: [4])

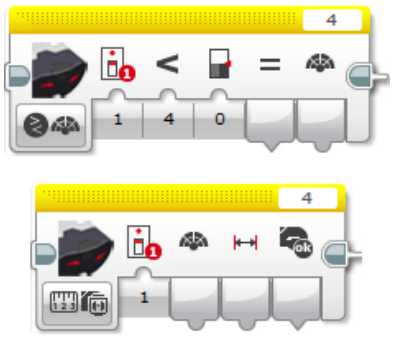
A távolság lekérdezésekor egy 0 és 100 közötti értéket fog megadni az infravörös érzékelő, ahol a 0 azt jelenti, hogy nagyon közel van a jeladó, míg a 100 azt, hogy nagyon messze. Viszont, a 100 értéket fogja visszaadni abban az esetben is, ha

nem érzékeli a jeladót. Az irány lekérdezésekor a 12. ábrán is látható, -25 és 25 közötti értéket fogunk kapni. Ez az érték nem fok, csupán egy jelölő, ahol a 0 azt jelenti, hogy a jeladó közvetlenül előttünk van, míg negatív értéknél a jeladó tőlünk balra, pozitív értéknél pedig tőlünk jobbra helyezkedik el.



13. ábra. A jeladó használata várakozás - összehasonlítás blokkokkal.

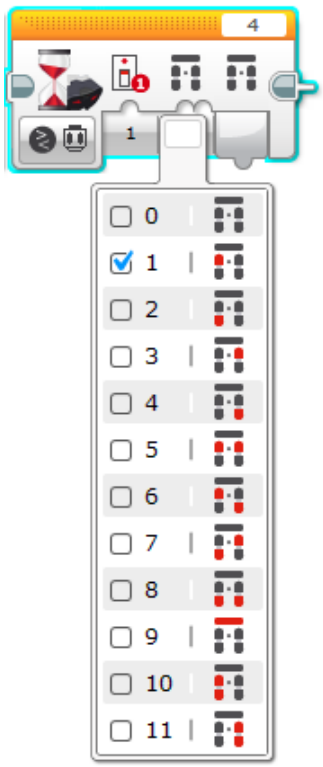
A programozás során, a jeladó és a távirányító használatához is, az infravörös érzékelőt kell választanunk és majd a blokkon belüli módoknál kell kiválasztanunk, hogy melyik funkcióra van szükségünk. A várakozás blokkban nézzük meg az összehasonlítás (Compare) szerinti működését a jeladónak! A 13. ábrán a jeladó iránya és távolsága szerinti beállításban látható a várakozás blokk összehasonlítás (Compare) módban. A felső blokk a Beacon Heading, vagyis a jeladó iránya szerinti értékeket vizsgálja, míg az alsó a Beacon Proximity-t, a jeladó távolságát vizsgálja. Mindkétt blokkban az első paraméternél kell megadni a jeladón használt csatorna számát. Ez a szám 1-4 közötti érték lehet. A második és harmadik paraméterek a szokásos tulajdonságokkal rendelkeznek. A negyedik paraméterrel pedig le tudjuk kérni a mért értéket. Ennek a paraméternek az ikonja változik a két módban, a 13. ábra felső blokkján az irányt jelöli, míg az alsón a távolságot. A várakozás blokk változást figyelő módjának (Change) a középső két paramétere tér el az összehasonlítástól. Ezek a paraméterek, a változás irányát és mértékét meghatározó értékek.
A ciklus és az elágazás blokkok Compare módjának ugyenezek a paraméterei, használatuk a korábbiakhoz hasonló.



14. ábra. A jeladó használata az érzékelő blokkokkal.

Az érzékelő (citromsárga) blokk Compare módjánál a paraméterek listája kiegészül még eggyel. Ez látható a 14. ábra felső blokkján. Ez a plussz paraméter egy logikai értéket ad át, amely az összehasonlítás eredménye. (Pl.: A mért érték kisebb, mint az általunk megadott küszöbérték. Ha ez az állítás igaz, akkor a true eredményt adhatjuk át a vezetékkel, ha hamis, akkor pedig a false-t.)
Az érzékelő blokk Measure módját választva három további opció közül választhatunk. Az első, a korábban megismert távolságmérés (Proximity), a második a Beacon és az utolsó a Remote. Ezek közül a Beacon szolgál a jeladóval kapcsolatos értékek átadására. Ezt a blokkot láthatjuk a 14. ábra alján. A blokk egy általunk beírt paraméterrel rendelkezik és három átadható értékűvel. A módosítható paraméter a jeladó által használt csatorna száma, ami a blokk használatához elengedhetetlen. Az átadható értékek balról jobbra a következők: jeladó iránya, jeladó távolsága, jeladó érzékelése. Az utóbbi egy logikai érték, ami azt adja meg, hogy az infravörös érzékelő a megadott csatornán érzékel-e jeladót, az első kettő, pedig a korábban megismert értékeket jelöli.

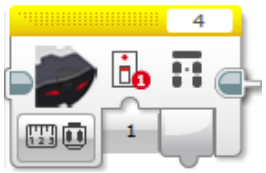
A távirányító használata (Remote mód)



15. ábra. A távirányító használata a várj blokkal és a gombok sorszámai.

Az eszköz távirányító módban történő használatakor a gombok helyzetét (be van-e nyomva) tudjuk lekérdezni és felhasználni. Habár csak négy programozható (és egy már beállított) gomb van a távirányítón, a blokkban 12-féle opció közül választhatunk, ami így is elég sok lehetőséget biztosít. Ebben a 12-ben ugyan benne van a bekapcsolást jelző gomb lenyomása is, azonban csak annyit tudunk szabályozni vele, hogy a távirányító továbbítsa-e az infravörös jeleket vagy sem. Azért, hogy a gombnak ezt a speciális tulajdonságát kiemeljék, külön nevet is adtak neki, mégpedig a következőt, Beacon Mode. A gombok programozásakor egy sorszám alapján tudjuk beazonosítani, hogy melyik gombra/gombokra vonatkozik az adott feltétel. Ezek a számok láthatóak a 15. ábrán.

A várakozás blokk Compare módjában a Remote opciót választva (15. ábra) meg kell adnunk a csatornát, ahol a kommunikáció zajlik az érzékelő és a távirányító között, valamint, hogy mely gomb vagy gombok lenyomására várakozunk. Viszont, ha megtörtént a lenyomás, akkor a blokkból le tudjuk kérdezni, hogy melyik gomb lett megnyomva, így erre nem kell egy másik blokkot használnunk.

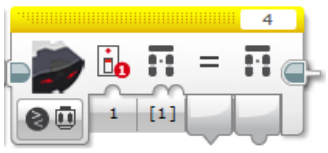


16. ábra. A távirányító használata az érzékelés blokk Measure módjában.

A várakozás blokk Change módjánál szintén a Remote opciót választva, hasonló blokkot fogunk kapni, mint a Compare módban. Azonban, itt a változást figyeljük, így nem kell megadnunk, hogy mely gomb/gombok megnyomására várakozunk. Emiatt, az előzőhöz képest csak a két szélső paramétert tartalmazza ez a mód.

A Wait - Change - Remote blokk megjelenésében (és paramétereiben) hasonló az érzékelő blokk Measure módjához, ami a 16. ábrán látható. Azonban, míg a várakozás blokknál várakozunk gomb megnyomására és utána kérdezhetjük le, hogy melyik(ek) lett(ek) megnyomva, addig az érzékelő blokknál az aktuálisan megnymott gomb értéket tudjuk paraméterként átadni és esetleg megjeleníteni.

A 17. ábrán látható a másik érzékelő blokk, amellyel használhatjuk a távirányítót. Ezzel a blokkal megvizsgálhatjuk, hogy az általunk megadott gomb(ok) le van(nak)-e nyomva az aktuális pillanatban. Ennél a blokknál is átadhatjuk az összehasonlítás eredményét paraméterként, amelyet az egyenlőségjellel jelölt paraméterrel tehetünk meg.



17. ábra. A távirányító használata az érzékelés blokk Compare módjában.

Az eddig említésre nem kerülő ciklus és elágazás blokkok paraméterei a fent tárgyalt blokkok paramétereivel megegyezik, így ezek használatára külön nem térek ki.

3.1.4. Feladatok

1. Készíts programot, amelyben az eszköz mozgása során, a jeladót követi! (Tipp 1: A megoldásban használd fel a jeladó irányát. Tipp 2: Használj valamilyen matematikai műveletet az irány felhasználásakor.)

2. Készíts programot, amelyben a távirányítót használva mozgatod a robotot! Állítsd be, hogy ha nincs lenyomva gomb, akkor a Tick tack hangot adja ki, valamint ha az egyes és hármas gombokat egyszerre lenyomjuk, akkor megálljon a motor. (Pl.: gomb 1 előre, gomb 2 hátra, stb.)

3. Készíts egy olyan programot, amelyben a jeladó távolságát az érzékelőtől hanggal jelzed! Minél messzebb van a jeladó, annál magasabb legyen a hang.

4. Készíts egy olyan programot, amiben a távirányítóval irányítod a robotot, hogy melyik irányba forduljon el ill. mikor álljon meg! Egy ezektől különböző gomb segítségével írasd ki a kijelzőre a gyro szenzor elfordulási szögét! A program, a távirányító bármely gombjának megnyomásakor induljon el!

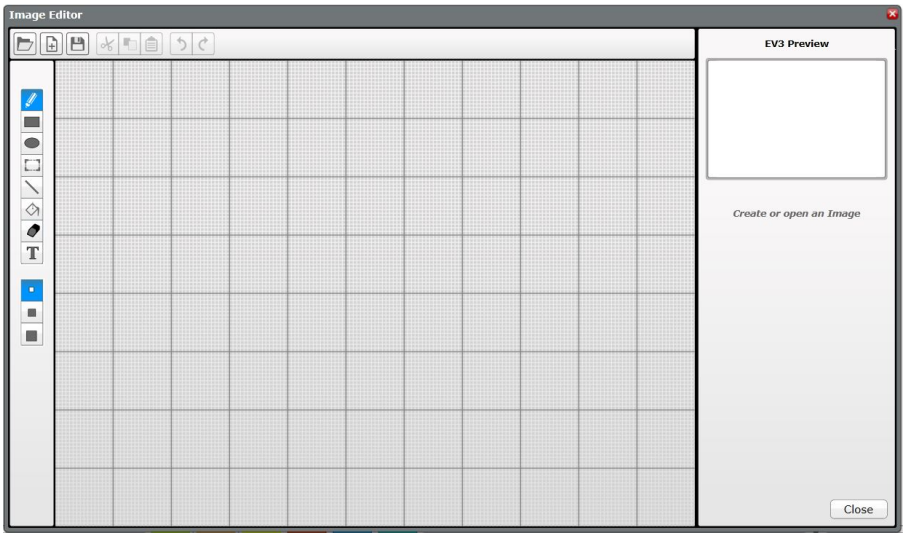
3.2. 2.alkalom

3.2.1. Saját kép rajzolása és megjelenítése

Az első tananyagban megismerkedtünk a Display blokkal, melynek segítségével robotunk kijelzőjén szövegeket, képeket, alakzatokat tudtunk megjeleníteni. Pontról pontra megnéztük milyen lehetőségeket rejt ez a funkció, hogyan tudunk navigálni robotunk képernyőjén pixels és grid módban egyaránt és hogy mégis mi az a Reset Screen, ha nem a megjelenített kép törlésére szolgál. Most ezen számos ismeret birtokában bővítjük tudásunkat, és megismerkedünk néhány izgalmas, haladó funkcióval a képernyőre rajzolással kapcsolatban.

Image Editor használata [1]

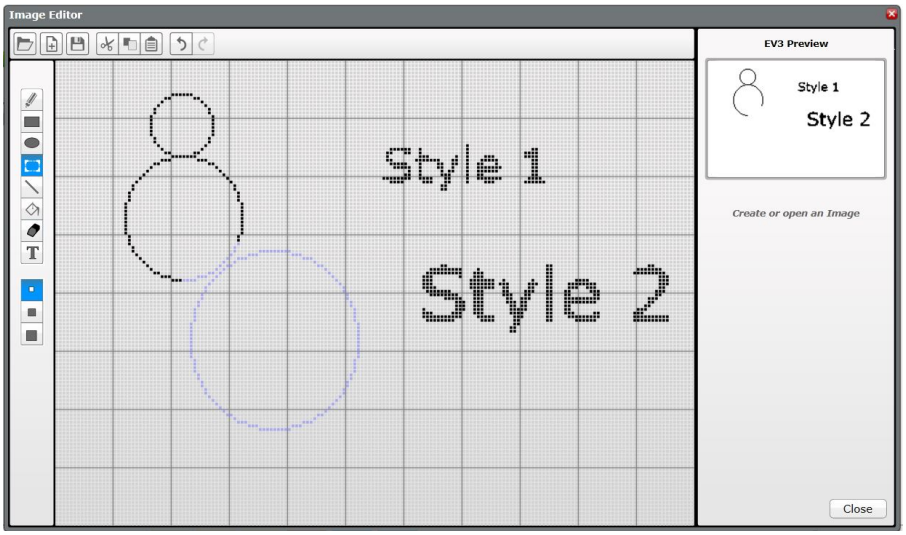
A szoftverben lehetőségünk nyílik a korábban megismerteken kívül arra is, hogy saját rajzot készítsünk. Ez különösen izgalmas lehet a gyerekek számára, hiszen ennek segítségével a robot képernyőjén egy általuk létrehozott, egyedi alkotás jelenhet meg. A beépített „rajzprogramot" csak akkor tudjuk megnyitni, ha van egy éppen megnyitott projektünk (tehát a szoftver kezdőképernyőjéről nem érhető el). A megnyitásához válasszuk a képernyő felső részén lévő Tools - Image Editor lehetőséget. Erre kattintva a 18. ábrán látható rajzolóprogram jelenik meg.



18. ábra. Image Editor

Bal oldalon található az eszköztár. A legfelső ceruza elem segítségével szabadkézi rajzokat tudunk létrehozni. A következő lehetőséggel téglalapot, az ezt követővel pedig ellipszist tudunk rajzolni. A kitöltetlen téglalap jelölésű gomb kijelölésre szolgál. Egy már megrajzolt objektumot vagy annak egy részét tudjuk kijelölni (egy téglalapba foglaljuk a kijelölendő részt) és tetszőlegesen mozgatni a rajzfelületen. Itt érdemes megjegyezni, hogy ez a kis beépített rajzolóprogram nem rendelkezik rétegkezeléssel. Ennek következtében tehát a kijelölésnél ha egymásra lóg két objektum nem tudjuk csak az egyiket kijelölni. Például ha van két egymást metsző kör, nem tudjuk áthelyezni egyiket sem úgy, hogy a másikból ne vágnánk le egy darabot. Valójában még metszeni sem kell a két körnek egymást ahhoz, hogy belefussunk az előbb említett problémába. Jól látszik a 19. ábrán, hogy a legnagyobb kört nem tudjuk áthelyezni a középső kör megsértése nélkül, mivel a nagy kör bennfoglaló téglalapja belelóg a másik körbe.

A kijelölő gomb alatti eszközzel egyenest tudunk rajzolni. A vödröcske segítségével kitölthetünk alakzatokat, azok hátterét tudjuk "beszínezni". Mivel robotunk képernyője csak fekete és "fehér" színt tud megjeleníteni, ezen eszköz segítségével alakzatainkat feketére színezhetjük. Vigyázat, ha véletlenül nem folytonos vonal határolja rajzunkat, könnyen a háttér is fekete lesz! Mivel a kitöltő eszköz csak feketére festésre szolgál, fehérre nem tudunk festeni vele. Ha mégsem szeretnénk kitölteni egy alakzatot, akkor azt a fölső eszközsáv utolsó előtti visszavonás elemével szüntethetjük meg. Persze lehet olyan tervünk is, hogy az egész hátteret feketére festjük és benne fehér alakzatokat jelenítünk meg. Ebben az esetben első lépésként az alakzatokat kell megrajzolni - ügyelve a folytonos vonalakra - majd ha ezzel készen vagyunk, jöhet a háttér feketére festése. Ahhoz, hogy apró javításokat tudjunk végezni, szükség lesz egy radírra, erre szolgál a következő eszköz.
A "T" jelölésű gombra kattintva szöveget írhatunk készülő rajzunkra. Első lépésként döntenünk kell a betűméretről. Style 1 választásával kisebb, míg Style 2 választásával nagyobb betűmérettel írhatunk. A 19. ábrán látható, hogy melyik beállítás mekkora méretet eredményez. Szöveg beviteléhez csak a rajfelület azon részére kell kattintanunk, ahol kezdeni szeretnénk a feliratot és már gépelhetünk is.
Az alsó 3 gomb a méret kiválasztására szolgál majdnem az összes eszköz esetében. Kivételt képez a kijelölő eszköz és a kitöltő eszköz, mivel ezeknél nincs értelme a méretválasztásnak, illetve a szövegbeviteli eszköz, mivel ennél a méret beállítását magánál a gombnál végezhetjük el.

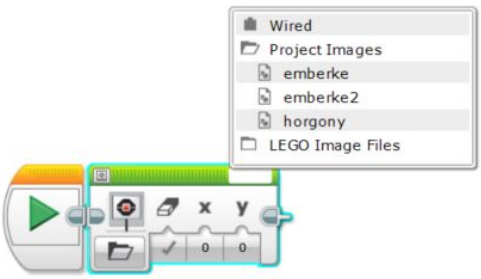


19. ábra. Kijelölés és betűméret

A felső menüsorban tudunk új rajzlapot nyitni a második elemre kattintva. Vigyázat! Ha már van egy készülő rajzunk és erre a gombra kattintunk, a program kérdés nélkül új rajzlapot nyit és elveszik korábbi munkánk! Erre érdemes felhívni a gyerekek figyelmét még mielőtt értékes rajzok vesznek el mentés nélkül! Munkánk mentéséhez a floppy ikonra kell kattintanunk. Mentéskor egy ékezeteket és speciális karaktereket nem tartalmazó nevet is kell adnunk elkészült alkotásunknak. Ahogy projektjeinket is érdemes beszédesen elnevezni, úgy rajzainknak is jó, ha olyan nevet adunk, melyről könnyen be tudjuk azonosítani a nélkül, hogy látnánk azt. Ennek akkor van jelentősége, ha több rajzunk van és azokat programunk egyes részein meg szeretnénk jeleníteni robotunk képernyőjén.

A következő olló gomb egy kijelölt rész törlésére, vagy áthelyezésére szolgál. A gomb hatására a kijelölt terület tartalmát töröljük, ami a vágólapra kerül, így újra beilleszthető. E mellett a másolás gombot találjuk, mely - ahogy a kivágás gomb is - csak akkor aktív, ha van egy kijelölt rész a rajzlapunkon. Abban az esetben ha kijelölés után erre kattintunk, vágólapra helyezzük a kijelölt részletet rajzunkból. Ha ez után a másolás melletti, beillesztés opciót választjuk megjelenik egy másolat az előbbi kijelölt részből. Ezen eszközöktől jobbra a már korábban említett visszavonás gombot találjuk, illetve a "visszavonás visszavonására" szolgáló gombot. Tehát az előre mutató nyilat arra tudjuk használni, hogy ha valamit visszavontunk, de meggondoltuk magunkat, akkor újrarajzolás helyett ezzel megjeleníthetjük. Jobb oldalon az EV3 Preview felirat alatt a fehér téglalapban azt látjuk, hogy a készülő rajz hogy mutatna a robot képernyőjén.

Az elkészített rajzokból akár történeteket is alkothatunk beillesztve azokat programunk kódjába. Ehhez a már jól ismert Display blokk lesz segítségünkre. A módok közül most az Image opcióra lesz szükségünk. A blokk jobb felső sarkába kattintva válasszuk ki a Project Images mappába mentett rajzunkat, melyet a mentéskor megadott név alapján ismerünk fel. Az én példámban emberke és emberke2 néven szerepel a két rajz a 20. ábrán látható módon. Ahhoz, hogy robotunk képernyőjén megjelenjen a rajz még legalább egy blokkra szükségünk lesz. A várakozás itt sem maradhat el!

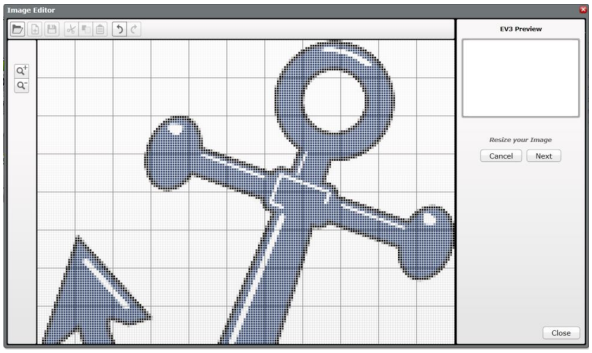


20. ábra. Saját rajz megjelenítése

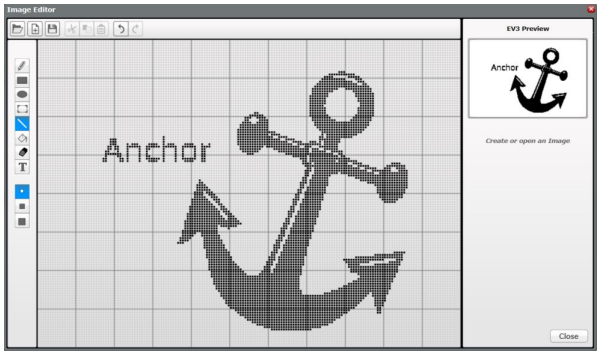
Több blokk felhasználásával számos rajzunkat megjeleníthetjük a képernyőn, történeteket mesélhetünk el ezen funkció segítségével.

3.2.2. Kép megjelenítése a számítógépről

Eddig a szoftverbe épített képeket, illetve saját rajzainkat jelenítettük meg robotunk képernyőjén. Azonban a lehetőségek sora itt nem ér véget. A számítógépünkről bármilyen képet kirajzoltathatunk robotunkkal. Akár saját rajzainkkal is kombinálhatunk internetről letöltött képeket, ezzel még nagyobb teret engedve a kreativitásnak. De hogyan kell ezeket a képeket a szoftverbe varázsolni? Ehhez is az Image Editor lesz segítségünkre. Fölül az első opciót (Open) választva tallózhatjuk be a képet fájljaink közül. Miután ez megtörtént még lehetőségünk nyílik a kép módosítására. Átméretezhetjük, mozgathatjuk, beállíthatjuk telítettségét és akár szabad kézzel is rajzolhatunk rá további elemeket. Az alábbi két ábrán jól látszik, mennyire átalakítható egy kép.

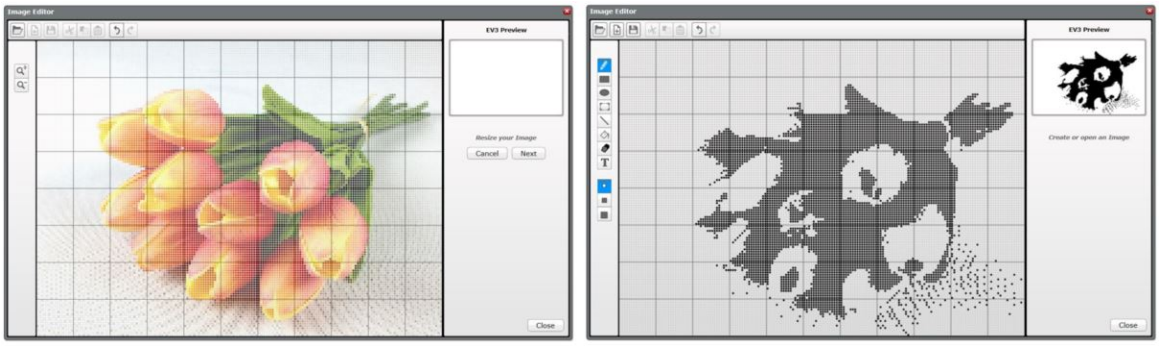


21. ábra. Tallózás utáni állapot



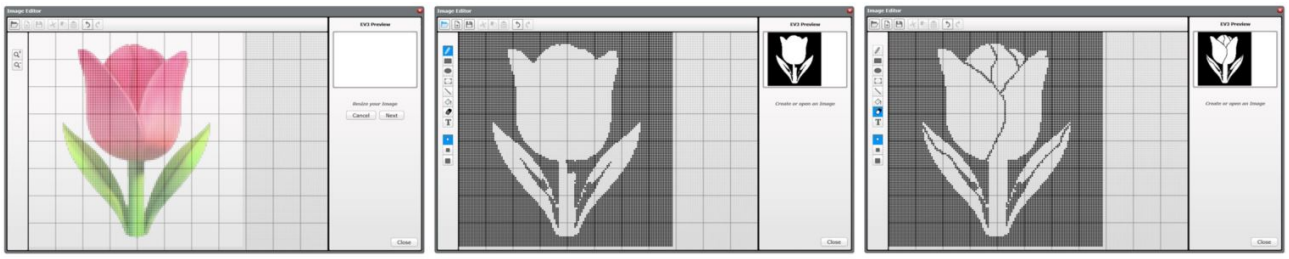
22. ábra. Szerkesztés utáni állapot

A szép megjelenítéshez azonban nem mindegy milyen kiinduló képet választunk! Érdemes háttér nélküli, letisztult, éles körvonallal rendelkező képekkel próbálkoznunk, mivel robotunk csak ezeket tudja felismerhetően megjeleníteni.Ha időnk engedi szabadon hagyhatjuk próbálkozni diákjainkat, majd egy kis egyéni felfedezés után együtt összegyűjthetjük a tanulságokat, hogy milyen képeket tudott jó minőségben megjeleníteni a robot, és melyekkel nem birkózott meg. A 23. ábrán látható tulipán csokor sajnos nem túl mutatós robotunk képernyőjén. A szálak összemosódnak és a kontraszt hosszas állítgatása mellett is csak ennyit sikerült kihozni a képből.



23. ábra. Összetett kép megjelenítése

Ha azonban beérjük a csokor helyett egy szál tulipánnal is, máris felismerhetőbb ábrát kapunk. A 24. ábrán balról jobbra a kiindulási kép, a robot által értelmezett kép és a kézi korrekció utáni állapot látható.



24. ábra. Letisztult kép megjelenítése

Az így készült képek szintén a Project Images mappában érhetőek el. Ugyanúgy ki tudjuk választani a Display blokk Image funkcióját használva, mint a már előbb említett saját rajzokat. Ezekkel az ismeretekkel még színesebbé tehetjük a robot képernyőjéhez kapcsolódó feladatokat. A diákoknak hagyjunk elég időt az alkotásra, és figyeljük a készülő munkákat.

3.2.3. Feladatok

1. Rajzolj az Image Editor segítségével egy vigyázzállásban álló pálcika emberkét. Munkádat mentsd emberke néven. Rajzolj egy ugyanekkora méretű pálcika emberkét, aki terpeszállásban áll, kezeit felemelve. Ezt a rajzot mentsd emberke2 néven. Alkoss programot, melyben a robot képernyőjén csillagugrást végez pálcika emberkéd. Használd elkészített rajzaid.

2. Engedd szabadjára fantáziád! Rajzolj az Image Editorban több képet: háttér, szereplők és a programozás segítségével játssz el velük egy történetet, melyet a robot képernyőjén jeleníts meg!

3. Tölts le egy képet az internetről. Szerkeszd az Image Editorban tetszés szerint és jelenítsd meg ezt robotod képernyőjén. Állíts be egy hozzá illő hangot is.

4. Alkoss egy olyan robotot, ami színekről asszociál valamire (pl.: gyümölcs, állat), ami hasonló színű. Jeleníts meg egy képet arról a dologról, amire "robotod gondol", amikor meglátja az adott színt. Például citromsárga szín láttán banánra, míg piros színt észlelve almára gondol. A képet internetről is letöltheted, de akár saját kezűleg is megrajzolhatod a szoftverben. (Szükséges: pálya színes elemekkel, melyeken a robot végighalad.)

3.3. 3.alkalom

3.3.1. Saját hang készítése és lejátszása

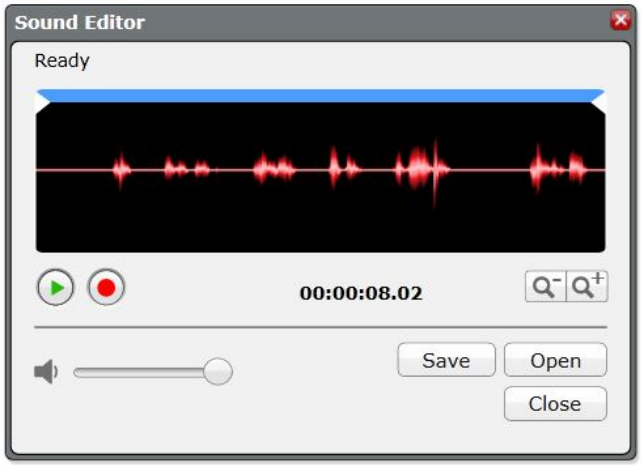
Tananyagunk első részében (5.alkalom) [2] már megismerkedtünk az Action (zöld) blokkok között található Sound blokkal, mely a hangok roboton való lejátszásáért felelős. Több módban is játszottunk le hangokat, akár zenét is komponáltunk a zongorabillentyűk segítségével. Érdemes ezeket az ismereteket néhány egyszerű feladattal feleleveníteni, hogy amikor elkészülnek diákjaink saját hanganyagukkal, azt már zökkenőmentesen tudják kipróbálni a roboton.

A hang - egy kis fizika

Mivel a következő részben egyszerű hangszerkesztéssel is fogunk foglalkozni, érdemes diákjainkkal beszélgetni a hangról, mint fizikai jelenségről. Ha csoportunk tagjai már elvégezték a 7. osztályt valószínűleg sok ismerettel rendelkeznek a témában. Ha viszont 7.-eseknél fiatalabb korosztályt tanítunk, érdemes erre a bevezetőre több időt szánni, hogy a felfedezés élménye teljes legyen. Érdemes a korosztálynak megfelelő szinten kitérni a hang mechanikai hullám voltára, illetve a hozzá kötődő legfontosabb jellemzőkre, mértékegységekre. Ezek szemléltetéséhez felhasználható a hangszerkesztő eszközben megjelenő hanghullám ábrázolás is.

Sound Editor használata

Az eszköz, mely ezen szakköri alkalmunk főszereplője szintén a Tools menüpontban érhető el, akárcsak a képszerkesztésre alkalmas Image Editor. Most a menü első elemére lesz szükségünk, ami nem más, mint a Sound Editor.

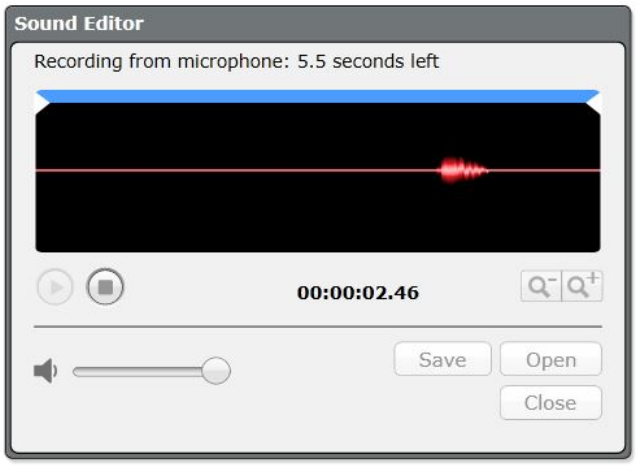


25. ábra. Sound Editor

Ez az eszköz két funkcióval is rendelkezik. Egyik a hangfelvétel, amivel most részletesen megismerkedünk, a másik pedig egy kész hanganyag szerkesztése, lejátszása, melyet számítógépünkről importálunk a programba (erről kicsit később lesz szó). Tehát nézzük, mit kínál nekünk ez a hangfelvételre alkalmas Sound Editor.

Az eszköz kezelőfelülete letisztult, így a diákoknak rövid idő alatt megtanítható használata és már kezdődhet is az alkotás! A piros (Record) gombbal tudunk új hangfelvételt indítani. Ugyanezzel a gombbal tudjuk leállítani a folyamatban lévő felvételt. Fontos felhívni a gyerekek figyelmét arra, hogy a hangfelvétel nem folytatható, ha egyszer leállítottuk. Akkor már csak a jelenlegi szerkesztésére, mentésére van lehetőségünk vagy egy új hangfelvétel indítására. Abban az esetben, ha nem állítjuk le a hangfelvételt, körülbelül 8 másodperc után magától leáll. A 25. ábrán egy ilyen kézi megszakítás nélküli felvétel látható. Ennél hosszabb felvételt nem tudunk egy fájlba készíteni.

Hangfelvétel közben az ablak felső részén látjuk hány másodpercünk van még az automatikus leállításig. Eközben, ahogy a 26. ábrán is látható, a hanghullámokat rajzoló ablak alatt az eddigi felvételünk hossza jelenik meg.



26. ábra. Másodperc kijelzés

Új hangfelvételt szintén a piros (Record) gombbal indíthatunk. Erre akkor is lehetőségünk van, hogyha nem mentettük el korábban felvett munkánkat. Ebben az esetben biztonsági kérdés figyelmeztet, így még meggondolhatjuk, hogy biztosan mentés nélkül akarunk-e új felvételt indítani (hiszen ekkor korábbi felvételünk elveszik). A felvett hangot a zöld (Play) gombbal tudjuk meghallgatni. A visszajátszás hangerejének állítására szolgál a csúszka. Ez nincs hatással arra, hogy a roboton milyen hangosan kerül lejátszásra a felvett hang (azt a Sound blokk első paraméterével szabályozhatjuk).

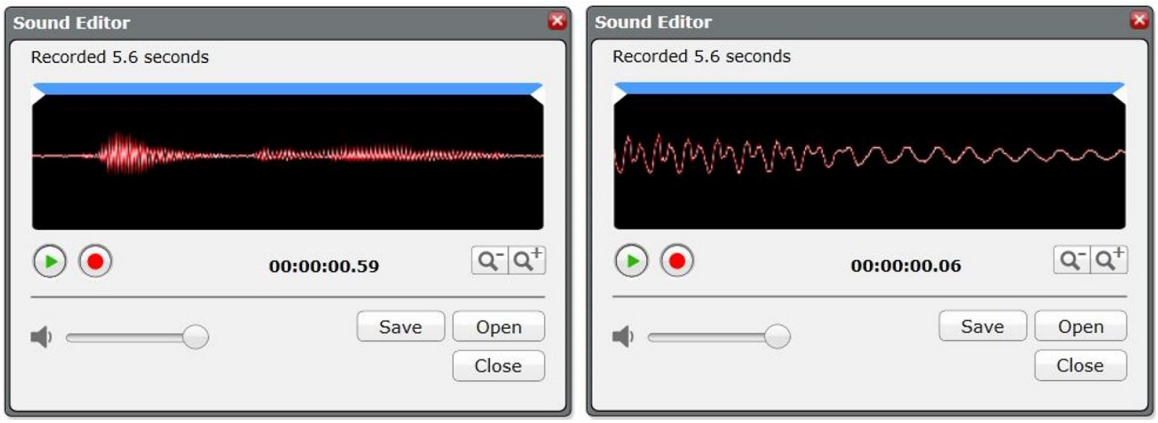
A Save gombbal tudjuk elmenteni munkánkat egy ékezeteket és speciális karaktereket nem tartalmazó név megadása után. Ahogy más munkák mentésénél, itt is nagyon fontos a beszédes elnevezés, hogy könnyen használhatóak legyenek elmentett anyagaink.

Felvételünket nem csak az eredeti formában menthetjük, lehetőségünk van minimális szerkesztésre is.

27. ábra. Kijelölő eszköz

Kiválaszthatjuk felvételünk egy kisebb részletét, melyet önmagában el is menthetünk. A hanghullámokat megjelenítő ablak felső részén látható kék sáv mutatja az aktuális kijelölt tartományt. Ezt az intervallumot a két fehér jelölő mozgatásával változtathatjuk. Felvételünkből kiválaszthatunk akár egy nagyon rövid részt is, ahogy a 27. ábrán látható.

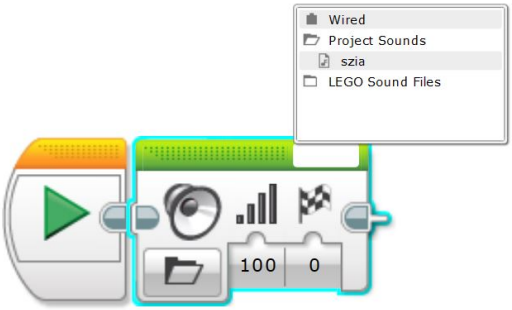
Abban az esetben, ha felvételünk egy részét jelöltük csak ki, ráközelíthetünk az adott intervallumra a + jeles nagyító segítségével. Ez praktikus lehet akkor, ha egy ennél még kisebb részlet kivágását tűztük ki célul, mert ilyenkor jobban látjuk a kirajzolódó hanghullámokat, és így az elkülönülő szavakat. Érdemes a diákoknak egy, a 28. ábra jobb oldali ábrájához hasonló képen megmutatni a hanghullámokat. Beszéljük meg mit látunk, mit jelenthetnek a sűrű, ritka, magas és ellaposodó hullámok!



28. ábra. Nagyítás - vágás

Bátran ráközelíthetünk felvételünk kisebb részeire, ettől az eredeti állományunk még nem veszik el. A - jeles nagyítóra kattintva újra eggyel bővebb részletét látjuk a felvételnek.
Ahhoz, hogy hangfelvételünk csak egy darabját mentsük elég kijelölnünk az adott részt, nem szükséges a ráközelítés.

Miután minden lehetőséget megnéztünk, nincs más hátra, próbáljuk ki a roboton alkotásunkat. Ehhez, ahogy a beépített hangok esetén is a Sound blokk Play File módja lesz segítségünkre. A jobb felső sarokra kattintva a Project Sound mappából válasszuk ki azt a fájlt, amit robotunkon szeretnénk lejátszani. Az én példámban, ahogy a 29. ábrán is látható, ez a "szia" nevezető fájl lesz. Bíztassuk diákjainkat a blokk egyéb paramétereinek változtatására is, hiszen ez egy jó lehetőség az ismétlésre.



29. ábra. Saját hang lejátszása

3.3.2. Hang lejátszása a számítógépről

A Sound Editorban minden gombot megismertünk az előző részben, egyet kivéve. Az Open feliratú gomb segítségével kész hanganyagot importálhatunk számítógépünkről a programba. Ezzel ugyanolyan műveleteket végezhetünk, mint saját hangfelvételünkkel - visszahallgathatjuk, kivághatjuk egy részletét, menthetjük projektünkbe, majd beilleszthetjük a Sound blokk segítségvel programunkba ezzel megszólaltatva robotunkon.
Van azonban egy-két feltétel, melyet figyelembe kell vennünk, amikor hangfájlt választunk! A Sound Editor a .wav, .mp3 és .rsf kiterjesztésű fájlokat képes lejátszani, azonban ezek közül sem mindegy milyen minőségűt választunk! Saját tapasztalataim szerint 320 kbps MP3-at már nem képes importálni az eszköz. Csak az ennél rosszabb minőségűeket tölti be sikeresen. Nagyjából bármilyen hosszú lehet az importálni kívánt hang, ha figyelünk a megfelelő kiterjesztésre és minőségre, valószínűleg sikeres lesz a feldolgozás. Ha nem, akkor azt az eszköz hibaüzenettel jelzi a felső sorban.

Egy zeneszám hanghullámai egészen máshogy néznek ki, mint az általunk felvett beszédé. Érdemes mutatni a diákoknak egy ilyen példát, hogy lássák a különbségeket. Például a U2 együttes Beautiful Day című számát a 30. ábrán látható módon jeleníti meg.

30. ábra. Zeneszám importálása és mentése

Bár ezt a számot sikeresen importáltuk attól függetlenül, hogy több mint 4 perc hosszú, elmenteni mégsem tudjuk ebben az állapotban. Ahogy a 30. ábra bal oldali ablakában látjuk, a Save gomb inaktív. Ezt a mentési korlát okozza, ami ilyen számítógépről betöltött fájlok esetén körülbelül 10 másodperc. Maximum ilyen hosszú darabot enged elmenteni egy fájlba a program. Ahogy a 30. ábra jobb oldali ablakában látjuk, ez egy zeneszámból igen kicsi részlet. A fehér jelölőket állítva tudjuk megkeresni azt a legnagyobb intervallumot, amit már el tudunk menteni - ekkor a Save gomb automatikusan aktívvá válik.

Mivel a szakkör alkalmával nagy valószínűséggel nem áll rendelkezésre a gyerekek gépén letöltött hanganyag, érdemes olyan weblapötletekkel készülni, ami ingyenes letöltésre ad lehetőséget. Ilyen weblap például a ZapSplat[5]. Az oldal előnye, hogy több mint 75.000 hang letöltését biztosítja ingyenesen a legkülönbözőbb témákban. Több mint 25 kategória közül választhatunk. Hátránya, hogy a letöltéshez regisztráció szükséges, melyre egy jó megoldás lehet, ha mi tanárként regisztálunk egy fiókot, a diákok pedig ezzel a felhasználónév - jelszó párral lépnek be. Így elkerülhetjük az elfelejtett e-mail címek, felhasználónevek, jelszók problémás világát.

[5] `https://www.zapsplat.com`

3.3.3. Feladatok

1. Készíts olyan programot, melyben robotod köszön és röviden bemutatkozik. Elmondja hogy hívják és miket szeret csinálni a szabadidejében. Engedd szabadjára fantáziád, találj ki minél érdekesebb, viccesebb "robot tulajdonságokat". A szöveget több különböző fájlba is mentheted, hiszen egyszerre csak 8 mp rögzítésére van lehetőség.

2. Az előző szakköri alkalmak egyikén saját rajzaidból alkotott történethez készíts mesélő szöveget. Az egyes felvételeket a történet megfelelő pontjaihoz illeszd! A hangfelvételek alatt folyamatosan legyen látható az adott részhez tartozó kép, jelenet is. (A motormozgatást érdemes kivenni a programból, mivel olyan hangos, hogy mellette nem lehet jól érteni a felvett szöveget.)

3. Készíts osztályhíradót vagy mókás időjárás jelentést! Használj internetről letöltött hangokat és saját felvett szövegeket egyaránt! A teljes anyag hossza, melyet a robot megszólaltat legalább fél perces legyen! Munkádat akár a robot képernyőjén megjelenített képekkel is színesítheted.

3.3.4. NXT hangérzékelő használata

Mielőtt belevágnánk az új érzékelő felfedezésébe, jöjjön egy kis fizika. Erre azért van szükség, mert a hangérzékelővel kapcsolatban két új, talán a diákok számára még ismeretlen mértékegység jelenik meg, mely a programozás során kérdéseket vethet fel.
Az emberek a tízszeres intenzitású hangot csupán kétszer olyan hangosnak érzékelik. Emiatt a hangerőt logaritmikus skálán mérjük, melynek mértékegysége a decibel (dB). 10 dB növekedés így tízszeres hangintenzitást és kétszer olyan hangosnak érzékelt hangot jelent. Emellett a magasabb frekvenciájú hangokat halkabbnak érzékeljük ugyanolyan dB érték mellett. Ennek korrigálására született meg a dBa skála. Az ebben kifejezett hangerő még közelebb van a valós hangerőérzetünkhöz, mivel figyelembe veszi a vizsgált hang frekvenciáját is. A gyakorlatban mindkettő jól használható a robotikában, de érdemes ismerni a különbséget.

Technikai feltételek

Mielőtt belevágnánk a következő témába fontos meggyőződnünk arról, hogy a technikai feltételek adottak a szakkör megtartásához.

31. ábra. Hangérzékelő

Először is szükségünk lesz minden robothoz egy hangérzékelőre. Ez nem része az EV3 csomagnak, ehhez a robothoz nem gyártottak ilyen szenzort. A hangérzékelő az EV3 elődjének, az NXT csomagnak volt része. Ugyan az EV3-hoz nem készült ilyen elem, az NXT hangérzékelője szerencsére kompatibilis robotunkkal. Véleményem szerint érdemes beszerezni külön ezt a szenzort, mert nagyon sok lehetőség van benne, érdekes és szórakoztató feladatok oldhatóak meg segítségével. Amikor ebben a témában tartottam szakkört csoportom nagyon élvezte, hogy a robotot hangoskodással vagy éppen a csend megtartásával tudja irányítani.

32. ábra. Letöltés

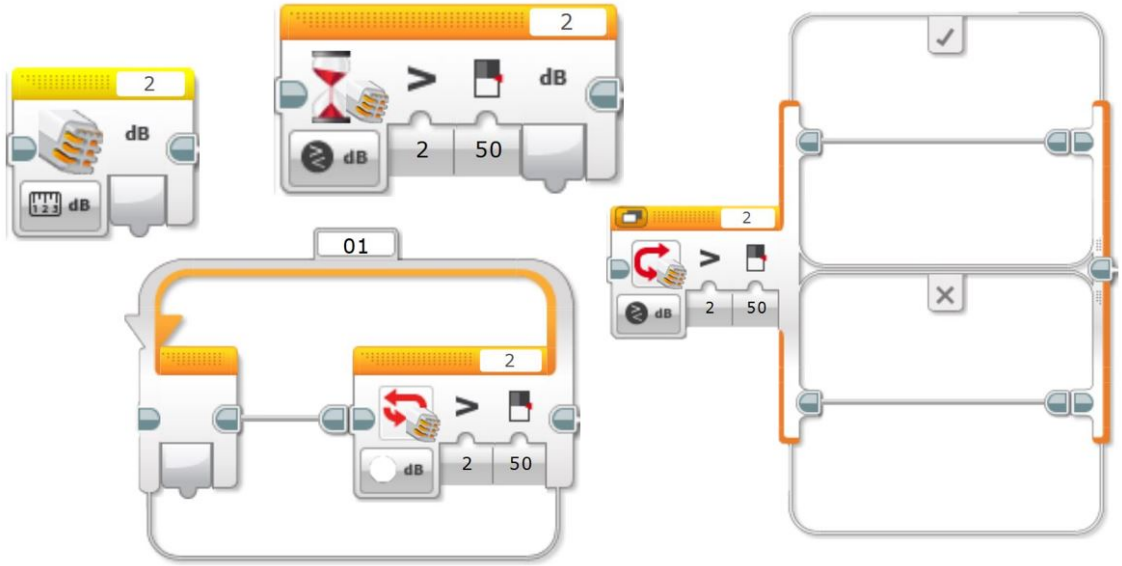
Ha megvan a hangérzékelő, már csak egy dolgunk maradt az előkészületekből. A szoftver nem tartalmazza a hangérzékelő kezeléséhez szükséges blokkokat, így ezt importálnunk kell programunkba. Első lépésként le kell tölteni a Lego honlapjáról az EV3-as szoftver blokkok között található hangérzékelő blokkot (Sound.ev3b)[6].

Ezt követően a szoftver felső menüsorából a Tools - Block Import lehetőséget választva eljutunk a megfelelő ablakhoz. Itt az Import fülön a Browse gombra kattintva válasszuk ki fájlkezelőnkből az előbb letöltött Sound.ev3b fájlt és importáljuk a szoftverbe az Import gombra kattintva. Ha jól dolgoztunk a szoftver újraindítása után megjelennek a hangérzékelőhöz tartozó blokkok. Ezt a legegyszerűbben a Sensor (citromsárga) blokkok között tudjuk ellenőrizni - a sorban az utolsó az NXT Sound Sensor.

[6] `https://www.lego.com/hu-hu/themes/mindstorms/downloads`

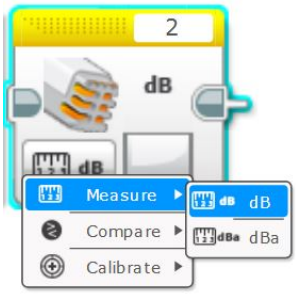
Hangérzékelő használata

Hasonlóan a már korábban tanult érzékelőkhöz, a sound sensor is több blokkban megjelenik. Megtaláljuk a Sensor (citromsárga) menüben, illetve feltételként állíthatjuk vezérlési szerkezetek (narancssárga blokkok) használatakor. Ezen szakköri anyag 1. alkalmának egyik témája a gyro szenzor volt. Itt részletesen megismerkedtünk a blokkok beállítási lehetőségeivel, melyek ezen érzékelő használatakor rendelkezésünkre állnak. Most, a hangérzékelővel való ismerkedésnél felhasználhatjuk ezen korábbi tudásunkat, hiszen a blokkok és a lehetőségek megegyeznek, csak jelentésben és apró beállításokban mutatkozik jelentős különbség. Így a következőkben a hangérzékelőre vonatkozó specifikus beállításokon lesz a hangsúly.



33. ábra. Hangérzékelőhöz köthető blokkok a programban

Most pedig vegyük sorra a blokkokat. Kezdjük az érzékelő (citromsárga) blokk megismerésével. Ezen blokkcsoport lényegében arra hivatott, hogy az egyes érzékelők értékét lekérdezzük és ezt az értéket valamely más blokk paraméterértékeként felhasználjuk. Erről részletesen a Lego Mindstorms EV3 robotok programozása I. tananyag 6. szakköri anyagában olvashatsz, mely ide kattintva elérhető.



34. ábra. Érzékelő blokk 3 módja

Az érzékelő blokk módválasztójára kattintva 3 lehetőség közül választhatunk. Az első kettő, a Measure és Compare lehetőségek már ismerősek lehetnek. Ha bizonytalan vagy használatukban, olvasd el a gyro szenzornál található leírást a 12. oldalon. A két mód működésének alapja azonos a két érzékelő esetében. A különbség csak a két szenzor eltérő funkciójából adódik. Míg a gyro szenzor az elfordulás szögét és a szögsebességet tudja vezetéken továbbadni, a hangérzékelő a mért hang erősségének közvetítésére képes. Measure és Compare módban is két opció közül választhatunk: dB és dBa. A módválasztó harmadik eleme eltér a két szenzor esetében. A hangérzékelőnél ez a Calibrate, vagyis a kalibrálás.

A hangérzékelő összesen 1024 különböző hangerőszint megkülönböztetésére képes. Alapértelmezetten ezt az 1024 értéket képezi le a 0-100 tartományra, ami ilyenkor megközelítőleg a már korábban említett dB skálával egyezik meg. Lehetőségünk van arra, hogy megadjunk egy kalibrációs minimum értéket. Ilyenkor a megadott érték szerint levágásra kerül a tartomány alsó része. Ez hasznos lehet abban az esetben, ha minimumnak egy alap zajszintet adunk meg, a 35. ábra bal oldalán látható blokkpár segítségével. Ekkor a robot figyelmen kívül fogja hagyni az ezzel egyenlő vagy halkabb hangokat, így csak olyanokat vesz figyelembe, melyekkel dolgozni szeretnénk. Ugyanez a felső tartománnyal is megtehető, amikor maximum értéket adunk meg. Ezt akkor érdemes megtenni, ha elegendő számunkra a halkabb hangok érzékelése, azokat viszont nagyobb pontossággal szeretnénk mérni. Alsó és felső korlátot megadhatunk beolvasott érték alapján vezeték segítségével vagy egy konkrét szám megadásával is. A 35. ábra jobb szélső blokkja a reset lehetőséget mutatja. Ezzel visszaállíthatjuk az alapértelmezett kalibrációt, vagyis újra a 0-1023 intervallum lesz leképezve a 0-100 tartományra.

35. ábra. Kalibrálás lehetőségei

A várakozás blokk feltételeként két módon használhatjuk a hangérzékelőt (Compare, Change). Erről részéletesen szintén a 12. oldalon olvashatsz. Ciklus feltételeként is használhatjuk szenzorunk értékét.

Vigyázat! Ne felejtsük el, hogy a ciklus kilépési feltételét adhatjuk meg, tehát a program addig ismételgeti a ciklus belsejében lévő blokkokat, amíg nem teljesül a megadott feltétel. Amint teljesül, kilép és folytatja futását.

Az elágazás szerkezet használatakor ugyanaz a helyzet, mint a korábban tanult érzékelők esetén, a pipával jelölt ágba írjuk, amit a robot a feltétel teljesülése esetén hajtson végre, míg az x ágba, amit akkor csináljon, amikor a feltétel nem teljesül. Ha azt szeretnénk, hogy robotunk többször is megvizsgálja a feltételt, ciklusba kell tennünk az elágazás szerkezetet!

3.3.5. Feladatok

1. Írasd ki robotod képernyőjére a mért hangerősséget. Az érték kövesse a zajszint változását 30 másodpercen keresztül. Gondoskodj róla, hogy például egy taps esetén az érték leolvasható legyen, ne tűnjön el túl gyorsan. Állapítsd meg mennyi a mért érték suttogás, normál beszéd, illetve taps esetén.

2. Készíts programot, melynek hatására robotod tapsra elindul. Egyenesen halad, míg újabb tapsot nem hall.

3. Robotod most egy engedelmes kutyus szerepét öltse magára! Egy helyben vár, amíg nem mondod neki, hogy "Gyere ide!". Ekkor egyenesen elindul feléd, majd körülbelül 20 centiméterrel előtted megáll és ugat. Így örül a gazdájának. :) (Vigyázz! Kutyusod csak akkor engedelmeskedik pontosan parancsodnak, ha csend van körülötted!)

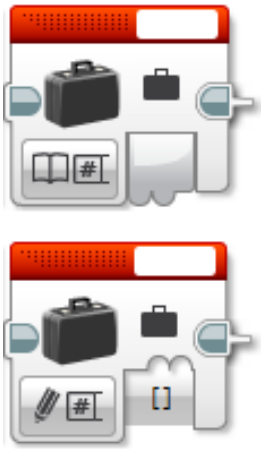
4. Alkoss programot, melynek hatására robotod pirosan villogva, dühös arccal gyorsan hátrálni kezd, ha túl nagy a zaj. Ha csendesebb a helyzet akkor zölden villogva, mosolyogva elindul lassan előre.

5. Robotod forogjon körbe saját tengelye körül 30 másodpercig. Sebessége a mért hangerősségtől függjön. Nagy zaj esetén gyorsan, míg egyre halkuló hangok mellett egyre lassabban forogjon.

3.4. 4.alkalom

3.4.1. Tömbök használata

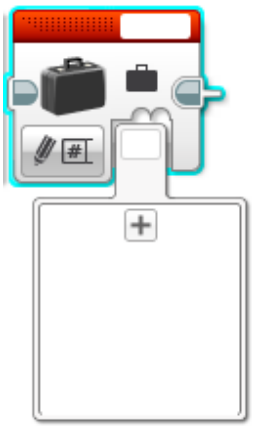
Tömböket a Data Operations (piros) szakaszban lévő blokkokkal tudunk létrehozni, azon belül pedig az első blokkal, amivel a változókat is készítjük. Kétféle tömböt hozhatunk létre, számokat vagy logikai értékeket tartalmazókat. Ugyanúgy, mint a változóknál, külön mód van az értékek kiolvasására (36. ábra felső blokkja) és írására (36. ábra alsó blokkja), valamint a névadás is ugyanúgy történik, a jobb felső sarokban lévő üres téglalapra kattintva. Fontos, hogy olyan neveket használjunk, amik beszédesek, tehát jól leírják, hogy mire használtuk az adott blokk elemeit.



36. ábra. Tömb értékeinek kiolvasása és írása.

> *Ezen kívül segíthet, ha megjegyzéseket írunk az adott blokk fölé/alá. Ilyen kommentet a menüsor jobb oldalán lévő mentés ikon bal szomszédja segítségével hozhatunk létre. (A komment gomb ikonja hasonlít a szövegszerkesztőkből ismert balra igazít ikonra.)*

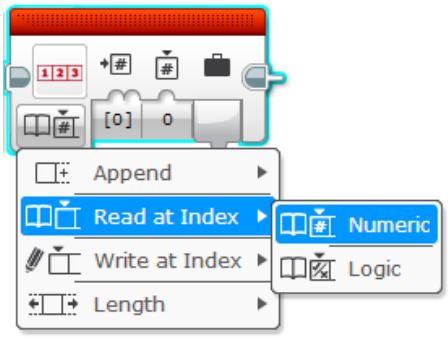
A változó és a tömb között a paraméterátadásnál annyi különbség van, hogy a változónak egy pici félkör van a vezetéke végén, míg a tömbnek kettő. Látható a 36. ábrán, hogy a tömb értékeinek olvasása és írása blokkoknál is két „hupszlis” végű vezetéket kell használnunk. A tömböt, létrehozása után, fel kell töltenünk elemekkel, amelyet az aktatáska ikon alatt lévő szögletes zárójelre kattintva tehetünk meg. Ekkor a 37. ábrán látható kis ablak ugrik elő, ahol a plusz jelre kattintva egy új „sor” jelenik meg az ablak tetején, amelynek az alapértelmezett értéke 0. Ezt az értéket a sorba kattintva módosíthatjuk. Logikai értékeket tartalmazó tömb esetén nem kézzel kell beírnunk az értékeket, hanem egy legördülő listából kell kiválasztanunk. A listában pipa vagy x közül választhatunk, azonban a négyzetes zárójelek között 0 vagy 1 értékek fognak szerepelni (0 - x/hamis, 1 - pipa/igaz).



37. ábra. Szám értékek felvétele tömbbe.

Műveleteket egy másik blokk segítségével tudunk végrehajtani a tömbökön, amely a piros szakasz harmadik eleme (38. ábra) és az Array Operations (tömb műveletek) nevet kapta. Négy különböző műveletet tudunk megvalósítani vele, amelyek a következők:

- elem hozzáadása (Append),
- adott sorszámú elem kiolvasása (Read at Index),
- adott sorszámú helyre érték írása (Write at Index),
- tömb méretének lekérdezése (Length).



38. ábra. Műveletek tömbökkel.

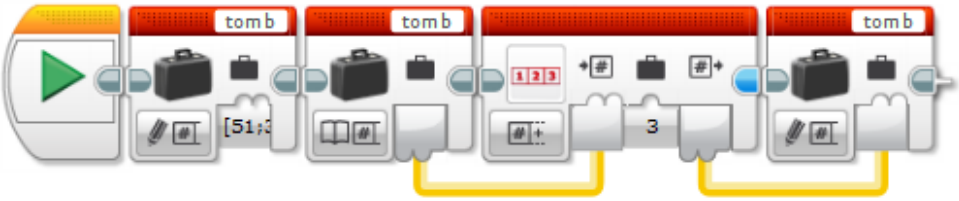
Mindegyik esetben két további opció közül lehet választani, amely lehetőségek a tömb elemeinek típusát határozzák meg. Tehát azt, hogy számot tartalmazó

vagy logikai értéket tartalmazó tömbön szeretnénk elvégezni az adott műveletet.
A következőkben a tömbökön végezhető műveleteket fogom részletesebben bemutatni. (Mind a négy esetben a számokat tartalmazó tömböt fogjuk megnézni, de mivel a paraméterek és a funkciók megegyeznek mindkét típusú tömbben, így ezek a beállítások a logikai értékeket tartalmazó tömböknél is használhatóak.)

Elem hozzáadása tömbhöz (Append)

39. ábra. Elem hozzáadása tömbhöz blokk.

A blokknak három paramétere van, ahogy a 39. ábrán is látható. Az első paraméterbe (Array In) azt a tömböt kell beírnunk vagy paraméterként átadnunk, aminek a végéhez elemet szeretnénk fűzni. Viszont, ha üresen hagyjuk ezt, akkor egy új tömböt hozhatunk létre, aminek az első eleme (0. indexű) a blokk második paraméterében megadott érték lesz. Tehát, a blokk aktatáska ikonja alá kell beírnunk azt az értéket, amit a tömb végére szeretnénk fűzni. Az utolsó paraméter egy kimeneti csatlakozó, amivel át kell adnunk egy másik blokknak az új tömbünket. Azaz, egy meglévő tömb végére egy új elem fűzése a 40. ábrán látható blokkokból áll. Az első blokkban létrehoztam az [52;36;15] elemeket tartalmazó *tomb* nevű blokkot. A középső kettővel megnyitom és hozzáadom a 3-as számot, majd az utolsó blokknak átadom az új tömböt, hogy az előzőt felülírja. Ekkor a *tomb* a következő elemeket tartalmazza: [52;36;15;3].



40. ábra. Egy meglévő blokkhoz új elem fűzése.

Adott sorszámú elem kiolvasása (Read at Index)

41. ábra. Adott sorszámú elem kiolvasása tömbből blokk.

Egy elemet, egy már létező blokkból a 41. ábrán látható blokk segítségével tudunk megnézni és felhasználni. A blokkban az első tényező, itt is az a tömb, aminek fel szeretnénk használni az elemét. A blokk második paraméterével határozhatjuk meg a felhasználandó érték indexét, míg az utolsó paraméterrel, ami ismét egy csatlakozó, átadhatjuk a kiolvasott értéket. Nézzünk egy példát! Ahhoz, hogy a korábbi példánkból ([52;36;15]) a 15-t fel tudjuk használni máshol, a blokk középső paraméterébe a 2-es értéket kell írnunk, majd át kell adnunk a kiolvasott értéket egy másik blokknak.

> *Ne felejtsük el, hogy eggyel kisebb az elemek indexe, mint sorszáma! Tehát a tömb első eleme a nulladik indexű helyen szerepel, a tömb második eleme az első indexű elem és így tovább. Az index és a sorszám nem ugyanaz!*

Adott sorszámú helyre érték írása (Write at Index)

42. ábra. Adott sorszámú helyre érték írása blokk.

Ha a tömb egy elemének értékét szeretnénk módosítani, akkor a 42. ábrán lévő blokkot kell használnunk. A blokk, paramétereit tekintve, az első műveleti blokkra, az Append-re hasonlít a leginkább, ugyanis az 1., 3. és 4. paraméterei teljes mértékben megegyeznek. A kimaradt második paraméterrel tudjuk megadni, hogy mely indexű érték helyére szeretnénk beírni az aktatáska alatt lévő értéket.

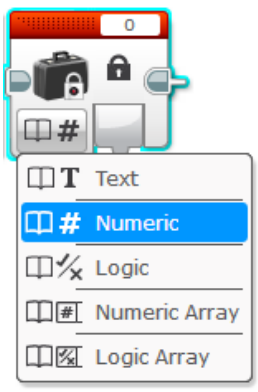
Működése a következő: a kapott blokk adott sorszámú értékét felülírja a meghatározott értékkel, majd a csatlakozó segítségével átadja egy másik blokknak. Ha a megadott index nagyobb, mint a tömb hosszúsága, akkor nem ad hibát, hanem a tömbhöz fűzi az értéket. Ha esetleg keletkezne „üres” index, akkor az ott lévő űrt, 0 értékekkel tölti fel. Pl.: Az előző tömbünk három elemű volt ([52;36;15]), de erről elfelejtkeztünk és az 5. indexű helyre szúrtuk be a 3-t. Ekkor a tömb elemei a következők: [52;36;15;0;0;3].

Tömb méretének lekérdezése (Length)

43. ábra. Tömb méretének lekérdezése blokk.

A tömbök méretét (ha esetleg elfelejtjük, hogy mekkora) le tudjuk kérdezni a 43. ábrán szereplő blokkal. Egyetlenegy paramétert kell megadni, mégpedig a tömböt, aminek le szeretnénk kérdezni a méretét/hosszát. A kimeneti csatlakozóval a tömb méretét tudjuk átadni egy másik blokknak felhasználásra.

3.4.2. Konstansok használata



44. ábra. Konstans blokk.

A konstansok olyan értékek, amelyek nem változnak a program futtatása során, emiatt akkor érdemes őket alkalmazni a programunkban, amikor egy értéket többször is felhasználunk a kódban. Ilyen érték lehet például a sebesség, az idő, de akár egy érzékelőnél használt küszöbérték is.

Az EV3 szoftverében ötféle érték lehet konstans, amiket a 44. ábrán láthatsz a blokk módválasztó listáján. Tehát konstans lehet szöveg, szám, logikai érték és tömb is. Amint látható a blokkon, nincs benne bemeneti csatlakozó, ami azt jelenti, hogy a konstansként használandó érték(ek)et kézzel kell beírnunk. Ezt a blokk jobb felső sarkában lévő fehér kis téglalapba kell megtennünk.

3.4.3. Feladatok

1. Hozz létre egy tömböt, aminek az elemei a következők: 51, 36, 15! Fűzd a tömb végére a 3-as értéket! Ezt követően jelenítsd meg a kijelzőn az értékeket külön sorokban! Az értékek megjelenítése akkor érjen véget, ha megnyomjuk a tégla valamelyik gombját.

2. Az előző feladatban létrehozott tömb 5. indexű helyén lévő értéket módosítsd 23-ra! Ezt követően jelenítsd meg az értékeket az előző feladatban leírt módon!

3. Hány értéke van/volt az előző feladatban szereplő tömbnek a feladat végrehajtása előtt és után? A tömb értékein kívül, jelenítsd meg „mindkét” tömb méretét a kijelző jobb oldalán!

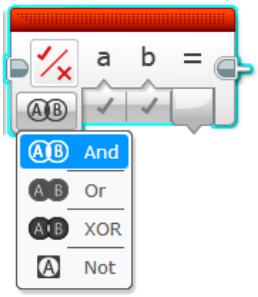
4. Készíts egy olyan programot, amiben egy általad megadott értékeket tartalmazó tömbnek, az összes elemét összeadod! A tömbben legalább 5 érték legyen és a végeredményt jelenítsd meg a kijelzőn.

5. Készíts egy programot, amelyben a robotod sebessége folyamatosan nő! Ehhez használd fel a 2-t, mint konstans értéket!

3.4.4. További adatmanipulációs elemek használata

Logikai műveleteket végző blokk (Logic Operations)

Ezzel a blokkal a logikai és (And), vagy (Or), kizáró vagy (XOR), nem (negálás; Not) műveleteit végezhetjük el, amelyeket a 45. ábrán is láthattok.



45. ábra. Logikai műveleteket végző blokk.

Mivel logikai műveletekről van szó, így logikai értékeket kell használnunk, tehát igaz vagy hamis értékeket. A negálás kivételével, mindegyik módban két értékre van szükség a művelet elvégzéséhez, amiket az első két paraméterben kell megadni. (A negálásnál csupán egy értéket kell megadnunk.) A művelet eredményét a harmadik paraméterben lévő csatlakozóval használhatjuk fel a későbbi blokkokban, amely eredmény a következő táblázatok szerint fog alakulni:

a	b	a And b
i	i	i
i	h	h
h	i	h
h	h	h

a	b	a Or b
i	i	i
i	h	i
h	i	i
h	h	h

a	b	a XOR b
i	i	h
i	h	i
h	i	i
h	h	h

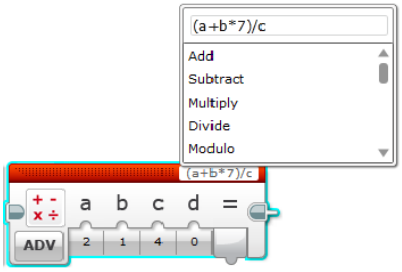
a	Not a
i	h
h	i

Matematikai műveleteket végző blokk (Math)

A matematikai blokkot már használtuk korábban is, azonban az Advanced (haladó) módjával (46. ábra) még nem ismerkedtünk meg. Ez a mód annyiban különbözik a többitől, hogy itt több műveletet el lehet végezni egyszerre és ezt a műveletsort mi határozzuk meg. Ez az a blokk, ahol bonyolultabb képleteket is meg lehet adni, mint például $(a + b * 7)/c$, $\sqrt{a * b}$, $\cos^2(a) - \sin^2(a)$.

A műveletekehez négy értéket adhatunk meg vezetékek segítségével (a, b, c, d), de ha kevesebbre van szükségünk, az is megoldható, egyszerűen ki kell hagyni a képletből, mint ahogy én is tettem a példákban. Ha több értéket kellene felhasználnunk, akkor vagy a konstans értékeket számként adjuk meg a képletben (az első példában a 7), vagy több részre osztva, külön blokkokban végezzük el a műveleteket. Az eredményt pedig a blokk egyenlőségjellel jelölt csatlakozójával adhatjuk át egy másik blokknak. Ebben a módban, a blokk jobb felső sarkában lévő fehér téglalapra kattintva adhatjuk meg a műveletsort, ahol nemcsak az összeadás, kivonás, stb. közül válogathatunk, hanem a legördülő lista elemeiből is. A listában a következő műveletek találhatóak meg:

- Add - összeadás
- Subtract - kivonás
- Multiply - szorzás
- Divide - osztás
- Modulo - maradék meghatározása
- Exponent - hatványozás

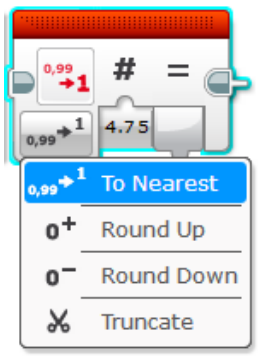


46. ábra. Matematikai műveleteket végző blokk haladó módja.

- Negate - negálás/tagadás
- Floor - lefelé kerekítés
- Ceil - felfelé kerekítés
- Round - legközelebbi egészre kerekítés
- Absolute - abszolútérték
- Log - logaritmus
- Ln - természetes alapú logaritmus
- Sin - szinusz
- Cos - koszinusz
- Tan - tangens
- Asin - arkusz szinusz
- Acos - arkusz koszinusz
- Atan - arkusz tangens
- Square Root - négyzetgyök

Kerekítést végző blokk (Round)

A kerekítést végző blokk (47. ábra) a megadott valós számot fogja le (Round Down), fel (Round Up) vagy a legközelebbi egészre (To Nearest) kerekíteni. Megadjuk az értéket a blokk első paraméterébe kézi beírással vagy paraméterként átadva egy másik blokkból. Ezután a második paramétert átadva egy másik blokknak, fel is használhatjuk a kerekített értéket.



47. ábra. Kerekítést végző blokk.

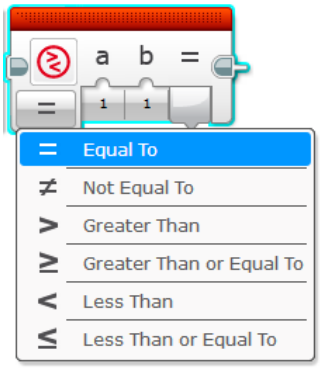
A negyedik módja a blokknak a Truncate (48. ábra) nevet kapta, ami azt jelenti, hogy levágni, tehát az olló nagyon jól szimbolizálja a mód tevékenységét. Ennek a műveletnek két számszerű paraméterre van szüksége, hogy a harmadik helyen lévő csatlakozóba meghatározza számunkra az új értéket. Ez szép és jó, de hogyan lesz új értékünk? Nos, a blokk a megadott tizedestört végéről levág annyi számjegyet, hogy az általunk megadott mennyiségű legyen a tizedespont után. Ehhez az első paraméterbe egy nem egész számot kell megadnunk. Ez az a szám, aminek a végéről le szeretnénk vágni néhány vagy a tizedespont utáni összes számjegyét. A második paraméterbe pedig, a megtartandó számjegyek számát kell megadni. A 48. ábrán látható példában a 4.75 értéket adtuk meg és azt szeretnénk, hogy egy számjegy maradjon a tizedespont után. Tehát a 4.7 értéket

48. ábra. Számjegyek levágása mód.

tudjuk továbbadni a blokk kimeneti csatlakozójával. Fontos megjegyezni, hogy ez nem kerekítés, ítt csupán elhagyunk számjegyeket.

Összehasonlítást végző blokk (Compare)

A Compare kifejezés már remélem, hogy nagyon ismerős mindenkinek. Az érzékelőknél tapasztaltakhoz hasonlóan, ez a blokk (49. ábra) is összehasonlítást végez, azonban itt mindkét értéket mi adjuk meg, nemcsak az egyiket. (Az érzékelőknél az egyik értéket mindig a szenzor adta, a másikat pedig mi. Ott az általunk megadott értéket, küszöbértéknek is neveztük.)

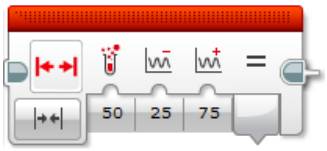


49. ábra. Összehasonlítást végző blokk.

A blokknak hat különböző módja van, melyek rendre a különböző relációs jeleknek megfelelő összehasonlításokat végzik el. Mindegyik típusnál két bemeneti értéket kell megadni, amit kézzel vagy paraméterként tehetünk meg. A harmadik érték az összehasonlítás eredménye, ami egy kimeneti logikai típusú adat és paraméterként átadva használható fel.

Intervallumot vizsgáló blokk (Range)

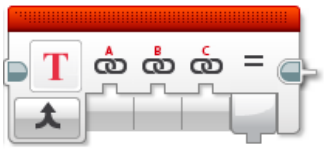
Az intervallumot vizsgáló blokk segítségével megnézhetjük, hogy egy általunk megadott intervallumon belül vagy kívül van-e egy adott érték. A blokknak két módja van, az 50. ábrán látható az intervallumon belüli vizsgálatot végzi el, míg a másik módja az intervullomon kívülit. A blokk első paraméterébe kell megadni a vizsgálandó értéket, míg a másik kettőbe az intervallum határait. A második paraméterbe az intervallum bal oldali határát, míg a harmadikba a jobb oldali határát. A negyedik „paraméter” a művelet eredménye, egy logikai érték.



50. ábra. Range blokk.

Szövegösszefűzés blokk (Text)

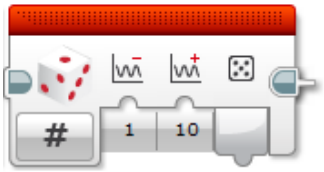
A Text blokknak egyetlen módja van, a Merge, amivel legfeljebb három szövegrészt tudunk összefűzni. A blokkban (51. ábra) a bemeneti kapcsolók teteje négyzet alakú, ami szöveg típusú bemeneti értéket jelent, azonban ugyanúgy megadható szám érték és logikai érték is, ugyanis a blokk ezeket átalakítja szöveggé az összefűzés során. Tehát, a kimeneti csatlakozóval biztosan szöveg típusú értéket fogunk átadni. Továbbá, itt is érvényes az, ami a matematikai blokk haladó módjában volt, hogy amelyik paraméterbe nem adunk meg értéket, azt nem fogja figyelembe venni az összefűzésnél.



51. ábra. Szövegösszefűzés blokk.

Véletlen értéket generáló blokk (Random)

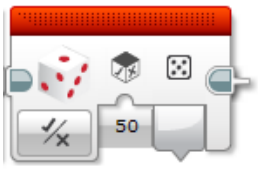
A Random blokknak két módja van, az egyikkel véletlen számot tudunk generálni egy adott intervallumon belül, míg a másik módjával egy logikai értéket. Nézzük meg először a véletlen számot generáló módot! Ez a blokk



52. ábra. Véletlenszám generáló blokk.

látható az 52. ábrán. A blokknak két paramétere van, amikkel a lérejövő számunk minimum és maximum értékeit tudjuk meghatározni. A harmadik értékkel a keletkezett számot tudjuk átadni egy másik blokknak.

A logikai értéket generáló blokkot már annyira nem nevezném véletlen értéknek, mert mi határozzuk meg, hogy mekkora valószínüséggel legyen igaz a létrejövő érték. Az 53. ábrán látható példában a megadott 50-es érték azt jelenti, hogy 50% az esélye annak, hogy igaz érték fog létrejönni. Viszont, ha átírjuk 80-ra, akkor már 80% eséllyel fogunk igaz értéket kapni.



53. ábra. Véletlen logikai értéket generáló blokk.

3.4.5. Feladatok

1. Készíts egy programot, amiben a nyomásérzékelő megnyomására 250 és 7000 Hz közötti hangot játszik le a robot! Ezt addig folytassa, amíg a középső gombot meg nem nyomjuk!

2. Hozz létre egy ötelemű tömböt, amiben 2 és 5 közötti értékek szerepelnek! Az értékeket közvetlenül egymás után írva jelenítsd meg a kijelzőn!

3. Bővítsük ki az előző feladatot! A tömbben lévő értékeket értelmezzük a színérzékelő által érzékelt színeknek és ez alapján „mondja ki" a robot az adott értékhez tartozó szín nevét angolul. (Ha rendelkezésre áll több idő, akkor javaslom, hogy vegyétek fel magyarul a színek neveit a Sound Editor segítségével és ezt használjátok fel a színek kimondásakor.

A témával kapcsolatos további feladatok megtalálhatóak a hátralévő szakköri foglalkozások feladatai között. (Lásd: 1., 2., 3., 4., 5., 6., 7.)

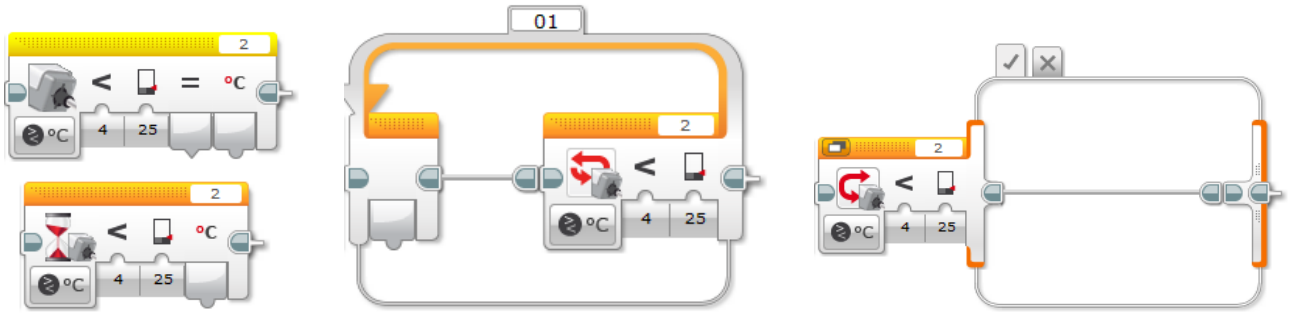
3.5. 5.alkalom

3.5.1. Hőmérsékletmérő használata

54. ábra. Hőmérsékletérzékelő.

A hőmérsékletérzékelő eredetileg az NXT eszközhöz készült és 2009-ben jelent meg, azonban az EV3-mal is jól használható. A LEGO© honlapján nem, de egy másik weboldalon megtaláltam a hivatalos mérési határokat, melyek a következők: -20°C-tól +120°C-ig vagy -4°F-től +248°F-ig. Az eszköz elég masszív, a folyadékok tesztelését is túléli, így bátran kísérletezhetünk vele.

Azok, akik nem az Education verzióját használják a szoftvernek, nem fogják látni a blokkokban a hőmérsékletérzékelőt, mert abba a programba külön be kell importálni a blokkokat. Ehhez a következő honalpról lehet letölteni a kiegészítő szoftvert: `https://www.lego.com/hu-hu/themes/mindstorms/downloads`.



55. ábra. A hőmérsékletérzőkelő használata (balról jobbra) az érzékelő (fent), a várakozás (lent), a ciklus és az elágazás blokkokkal.

Programozás szempontjából az 55. ábrán látható blokkok nagyon hasonlóak, használatuk megegyzik a korábban tanult szenzorokéval. Mindegyik blokkban kiválaszthatjuk, hogy az értékeket Celsius fokban vagy Fahrenheitben szeretnénk megvizsgálni.

Ezeken kívül, lekérhetjük a mért hőmérséklet értékét is, a szenzor (sárga) blokk Measure módjával, vagy megvizsgálhatjuk a hőmérséklet változását a várakozás blokk Change módjával.

3.5.2. Feladatok

1. Készíts programot, melyben megjeleníted az indításkor érzékelt hőmérsékletet, majd gombnyomás után valamilyen folyadék hőmérsékletét! A program befejezése előtt legyen látható a program elején és végén mért hőmérséklet is!

2. Készíts programot, melyben ha a jelenlegi átlagos hőmérséklettől magasabbat mérsz, akkor pozitív hangjelzést ad a robot, míg ha alacsonyabbat, akkor negatívat hangot játszik le!

A szenzort leginkább projektek megvalósítása során lehet kihasználni, így a következőkben néhány ötletet osztok meg veletek.

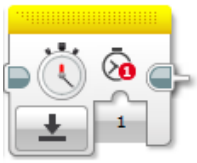
- óriás hőmérő készítése
- Annak a vizsgálata, hogy az ablakon bejutó napsugarak milyen mértékben melegítenek fel testeket, folyadékokat.
- Meddig melegítsük/hűtsük mini okosotthonunkat?
- Forró/meleg folyadékot addig keverget a robot, amíg szobahőmérsékletűre ki nem hűl.
- Addig tartja a robotkar a folyadékban a hőmérsékletérzékelőt, amíg fel nem forr. Ekkor a kar kiemelkedik a vízből.

3.5.3. Stopperek használata

A stopperrel a program kezdete (vagy a legutóbbi stopper resetelés) óta eltelt időt mérhetjük meg (ezred)másodpercben. Egy programon belül legfeljebb nyolc stoppert helyezhetünk el, melyet minden blokk használata előtt be kell állítanunk. (Ez a beállítás hasonlít az infravörös távirányító csatornaválasztó paraméteréhez.)
Stoppert négy különböző blokk segítségével használhatunk, melyek a következők: érzékelő blokk, várakozás blokk, ciklus és elágazás blokkok. A várakozás és a ciklus blokkoknál a mód választásakor figyeljünk arra, hogy ne keverjük össze az egyszerű idővel (Time). (A szoftver education 1.4.2-es változatában már *Time Indicator* néven szerepel a korábbi *Time* mód, ennek ellenére fennáll az összetévesztés lehetősége.)

Stopper resetelése

A gyro szenzorral ellentétben, itt jól működik a reset funkció és ténylegesen újraindítja az adott sorszámú stoppert a blokk. (56. ábra) Resetre legfőképp akkor van szükség, ha már használtuk az adott stoppert és ugyanazon a programon belül újra használnunk kell vagy nem a program elejétől szeretnénk mérni az időt. Ennek ellenére, sokan a program elején is resetelnek, ami nem tiltott dolog, de nem feltétlenül szükséges, ha az egész program idejét szeretnénk megmérni.



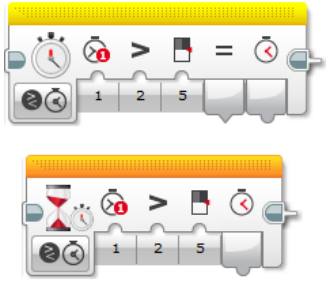
56. ábra. Stopper reseteléséhez szükséges blokk.

Stopper használata az érzékelő Measure módjában

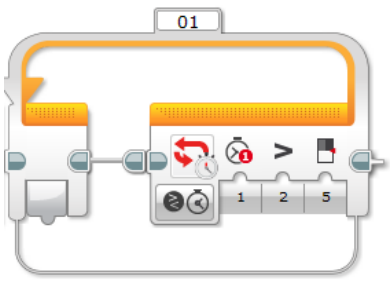
Az stopper az adott program futtatásakor automatikusan lenullázódik és elkezdi mérni az eltelt időt attól függetlenül, hogy le fogjuk kérdezni vagy sem. A legegyszerűbben az érzékelő blokk Measure módjának használatával adhatjuk meg a program indítása óta eltelt időt. Ez a blokk látható az 57. ábrán. A blokk egyetlen bemeneti paraméterében a stopper sorszámát kell kiválasztani. (Ezzel főként akkor kell foglalkozni és figyelni rá, ha egy programon belül több stoppert is használunk, addig különösebb teendőnk nincs vele.) A kimeneti csatlakozóval pedig átadhatjuk más blokkoknak a mért időt.

57. ábra. Timer blokk Measure módban.

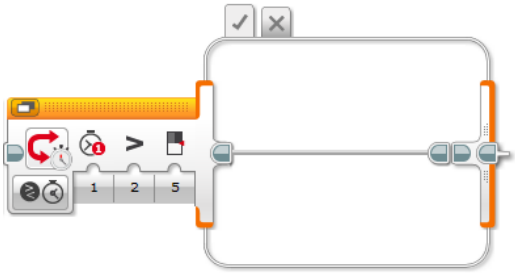
Stopper használata Compare módban



58. ábra. A stopper használata Compare módban a Timer és a Wait blokkban.



59. ábra. Stopper használata ciklussal.



60. ábra. Stopper használata elágazással.

Az 58., az 59. és a 60. ábrán látható blokkokban nagyrészt megegyeznek a használt paraméterek. Mindegyik blokkban az első paraméternél meg kell határozni a stopper sorszámát, tehát, hogy melyik azonosítójút használjuk. A második paraméternél a jól ismert relációs jelet kell kiválasztani, míg a harmadiknál a küszöbértéket kell megadni. Az 58. ábrán látható blokkokban vannak további paraméterek, amik kiementi csatlakozók. A várakozás blokkban lévővel lekérdezhetjük a mért értéket, ami szintén megtalálható a Timer blokkban is, azonban ott még szerepel egy egyenlőségjellel jelölt kimeneti csatlakozó is. Ezzel az összehasonlítás eredményét, ami egy logikai érték, tudjuk átadni egy másik blokknak.

Stopper használata a várakozás blokk Change módjával

61. ábra. Stopper használata a várakozás blokk Change módjával.

A várakozás blokk Change módját (61. ábra) választva addig várakozik a program, amíg el nem telik bizonyos „mennyiségű” idő a program indítása vagy a stopper legutóbbi resetje óta. Ehhez a blokkban ki kell választanunk, hogy melyik stoppert használjuk a nyolc közül, ill. meg kell adnunk, hogy mennyi időnek kell eltelnie. Ezeken kívül lekérhetjük a ténylegesen eltelt idő számszerű értékét is az utolsó paraméterben. Tehát, működését tekintve ez a blokk hasonló az egyszerű várakozás blokkhoz, ahol csupán néhány másodpercet várakoztatjuk a programot (Time Indicator), azonban ezt kiegészítették néhány újabb funkcióval. Viszont, ez nem azt jelenti, hogy mostmár ezt kell mindig használnunk, sőt inkább csak akkor használjuk, amikor idő mérése a cél.

3.5.4. Feladatok

1. Készíts programot, amelyben megméred, hogy egy fordulatot mennyi idő alatt tesz meg a robot! Az időt jelenítsd meg a kijelzőn is.

2. Teszteld a gyorsaságod! Készíts egy reakcióidőt mérő játékot, melyben egy hang lejátszása után, meg kell nyomnod a középső gombját a robotnak. A hang lejátszása és a gomb megnyomása közötti időt mérd meg és jelenítsd meg a kijelzőn! Azért, hogy ne legyen olyan könnyű a játék, oldd meg, hogy véletlen mennyiségű idő teljen el az indítás és a hang lejátszása között! Teszteljétek a játékot! Ki a leggyorsabb?

3. Készíts egy programot, amelyben folyamatosan megjeleníted a kijelzőn a futtatás elindítása óta eltelt időt két tizedesjegy pontossággal a következő formátumban: „Ido: _mp”! (Az _ vonal jelöli az eltelt időt.)

4. Készíts egy visszaszámlálót, ami 30 másodperctől indul és hangjelzéssel figyelmeztet, ha már csak 5 másodperc van hátra! Ezen kívül akkor is jelezzen, ha lejárt az idő! A rendelkezésünkre álló időt mindneképp jelenítsd meg a kijelzőn! (Tipp: 30-ból vond ki az eltelt időt.)

3.6. 6.alkalom

3.6.1. Fájlok elérése és módosítása

A korábbi alkalmak során sok szenzorral, vezérlési szerkezettel ismerkedtünk meg. Előkerült néhány olyan téma is, melynek során külső adatokkal dolgoztunk. Ilyen volt az Image és Sound Editor használata, ahol számítógépünkön található képet és hangot használtunk fel programunkban. Most középpontba kerülnek a fájlok, azon belül is a szöveges és számokat tartalmazó fájlok olvasásával, írásával fogunk foglalkozni.

Mielőtt speciálisan a robothoz kötődő fájlkezeléssel megismerkednénk érdemes csoportunkkal beszélgetni több fájlokhoz kapcsolódó kérdésről. Ilyen témák lehetnek például: Mi az a fájl, mappaszerkezet, fájlkezelő? Fájlok elnevezése? Fájlok kiterjesztése? Nézhetünk konkrét példákat is: milyen elérési útvonal tartozik a legutóbbi szakkörön elmentett projekthez? Mi a kiterjesztése egy, a számítógépen lévő képnek, zenének, videónak, szöveges dokumentumnak, robotos projektnek?

Milyen fájlokat szeret robotunk és hol tárolja azokat?

Ahogy az előző részben említettem, most a szöveges és számokat tartalmazó fájloké lesz a főszerep. Fájlokat számítógépünkről a robotra tölteni, illetve a robotról számítógépünkre menteni a Tools menüpontban található Memory Browser segítségével tudunk. Az Upload gombbal a kijelölt elemet tudjuk számítógépünkre menteni, a tőle jobbra található Download gombbal pedig számítógépünkről tudunk fájlt másolni robotunkra. A Memory Browser kezeléséről részletesen a 71. oldalon olvashatsz.

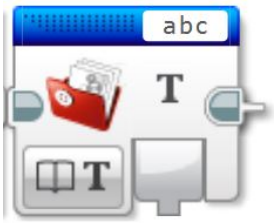
A robotra számítógépről feltöltött fájlok, illetve a robotunk által programok futtatása során létrehozott fájlok is robotunk saját memóriájába kerülnek. Ezen fájlok hatóköre projektszintű, robotunk az adott projekt mappájába mentett fájlokat éri el, ezeken tud módosításokat végrehajtani. Ez azt jelenti, hogy egy projekten belül létrehozott programok mind elérik a projekt fájljait, azok csak projekten kívül nem láthatóak.

Most hogy már tudjuk hol laknak az EV3 által használt fájlok, nézzük meg mit kell tudni a kiterjesztésről. Robotunk a szoftverben képes fájlokat létrehozni. Ezek a szöveges fájlok *rtf* formátumúak. Ha robotunkra akarunk menteni egy általunk a számítógépen létrehozott fájlt, ugyanezt a kiterjesztést kell választanunk. Az *rtf* a Rich Text Format rövidítése, mely egy formázásokat tartalmazó szöveges dokumentumtípus. Robotunk az általa létrehozott fájlt nem formázza és kezelni sem tud formázott dokumentumokat. Windows operációs rendszer alatt alapértelmezetten az *rtf* kiterjesztésű fájlokat Word vagy Word Pad nyitja meg. Ezek a programok azonban formázási információkat is írnak a fájlba, még akkor is, ha csak néhány soros szöveget írtunk enterekkel tagolva. Mivel robotunk ezt nem tudja értelmezni, használjunk inkább a notepad vagy notepad++ programokat a fájlok szerkesztésére, létrehozására.

A robot által létrehozott fájlok tartalmát számítógépünkre mentve ellenőrizhetjük. Az általunk létehozott és robotra töltött fájlok tartalmát pedig számos módon használhatjuk fel programunkban. Ehhez ismerkedjünk meg a fájlkezelésért felelős blokkal a szoftverben!

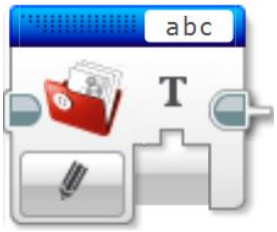
File Access blokk

A szoftverben az Advanced vagyis haladó (kék) blokkok között találjuk a fájlok eléréséért, létrehozásáért, módosításáért felelős File Access blokkot (ez a legelső a sorban). A blokk módválasztójára kattintva 4 lehetőség közül választhatunk. Nézzük meg ezeket föntről lefelé haladva részletesen.



62. ábra. Fájl olvasása

Az első mód a fájlok olvasásáért felelős (**Read)**. Ezzel tudjuk a robotunk memóriájában már meglévő fájlokat elővenni és tartalmukat felhasználni más blokkok bemenő paramétereként. A Read módon belül a Text és a Numeric opció közül kell választanunk. Ez azt jelenti, hogy el kell döntenünk, az adott fájl tartalmát szövegként vagy számként szeretnénk-e kiolvasni. Szöveget szövegként, számot számként nyilván sikerülni fog értelmeznie a programnak. De mi a helyzet a többi variációval? Számot szövegként ki tudunk olvasni, de fordítva, tehát szöveget számként nem tud értelmezni a program.
A blokk jobb felső sarkában a téglalapba írt szöveg a fájl neve, melyet megnyitunk olvasásra. Egy, a 62. ábrán szereplő blokk felhasználásával egy sort tudunk kiolvasni az *abc* nevű fájlból. Ezt a sort a blokk output mezője segítségével értékül tudjuk adni egy másik blokk input paraméterének. Például ki tudjuk írni a robot képernyőjére a szöveget, vagy akár el is tudjuk küldeni egy másik robotnak üzenetben - ehhez olvasd el az 52. oldalon kezdődő EV3 robotok közötti kommunikációról szóló fejezetet! Ahhoz, hogy több sort is kiolvassunk a fájlból, több File Access blokk szükséges - soronként egy. Az első File Access blokk programunkban az adott fájl első sorát fogja kiolvasni, a második a második sort és így tovább (persze csak akkor, ha azonos fájlnevet adtunk meg). Ha a fájlból kiolvasta az összes adatot tartalmazó sort nem ad hibát, hanem ettől kezdve 0-val tér vissza. Ez a 0 jelzi, hogy a fájl végére értünk.
Jogosan merül fel benned a kérdés, kedves Olvasó, hogy de akkor egy programon belül minden sort pontosan egyszer tudunk csak kiolvasni?! Szerencsére erre is van megoldás, de erre még egy kicsit várni kell, később lesz róla szó!



63. ábra. Fájl írása

A következő mód az írás (**Write)**. Ezt a módot akkor válasszuk, ha fájlt hozunk létre vagy ha már meglévő fájl tartalmát szeretnénk bővíteni. Az első esetben a jobb felső sarokban nevet kell adnunk a létrehozandó fájlnak, utóbbi esetben pedig a már meglévő fájl nevét kell begépelnünk. Ahogy az előző, olvasás módnál láttuk, nagy jelentősége van a sorok elválasztásának. Egy, a 63. ábrán szereplő blokkal egy sort írunk az *abc* nevű fájlba. A fájl következő sorát egy újabb blokk felhasználásával tudjuk létrehozni és így tovább. Ha a megadott nevű fájl még nem létezik, automatikusan létre fogja hozni és látható lesz a Memory Browserben. Ha már létező fájlba írunk, a hozzáfűzött adat a fájl végére, új sorba fog kerülni. A fájlból már meglévő adat nem fog törlődni.
Az írás blokk egyetlen bemenő paraméterrel rendelkezik. A fájlba írandó első sor megadható közvetle-

nül, begépelve a szöveget vagy számot, illetve vezetéken keresztül, felhasználva egy másik blokk kimenő paraméterét. Ilyen lehet például egy érzékelő blokk, melynek mért értékét fájlba írjuk vagy egy kapott üzenet, melynek tartalmát elmentjük - ehhez olvasd el az 52. oldalon kezdődő EV3 robotok közötti kommunikációról szóló fejezetet!

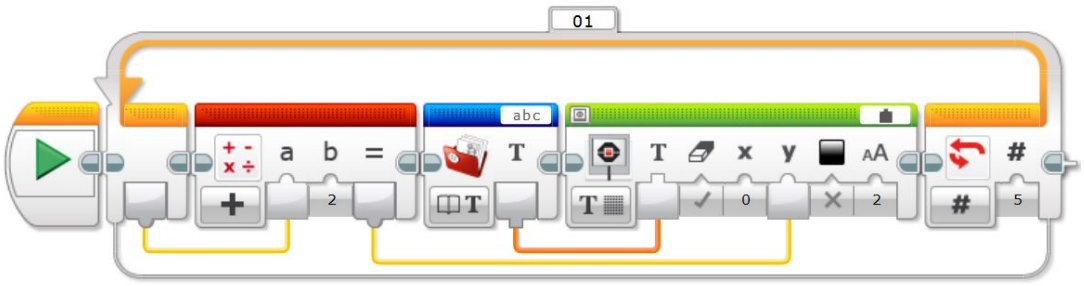
A harmadik mód fájlok törlésére szolgál (**Delete**). A jobb felső sarokban megadott nevű fájl kerül törlésre a robot memóriájából. Ezen blokk futtatása után az már nem lesz elérhető a szoftver fájlkezelőjében. A törlés művelet azért nagyon fontos, mert robotunk memóriájában kevés hely van. Ha sok nem használt fájlt tárolunk ott, hamar betelik és esetleg nem lesz hely új projekt létrehozására. Erről részletesen a 71. oldalon kezdődő, a robot belső memóriájáról szóló részben olvashatsz.

64. ábra. Fájl törlése

Az utolsó választható lehetőség a fájl bezárása opció (**Close**). Ezzel az adott programon belül korábban már olvasásra megnyitott fájlt tudunk bezárni. Elképzelhetjük úgy is, mintha egy könyvet visszatennék a polcra, de nem raknánk bele könyvjelzőt. A program nem jegyzi meg hol tartottunk az olvasással. Ha bezárás után újra megnyitjuk Read módben a fájlt, ismét az elejéről fogunk tudni olvasni. Ahogy a Delete mód, ez sem rendelkezik egyetlen paraméterrel sem.

65. ábra. Fájl bezárása

Tipp: Az elkészült fájl tartalmát akár azonos programon belül ki is írhatjuk robotunk képernyőjére. Ehhez először be kell zárni a fájlt! Ha a kiolvasott sorokat a képernyőn egymás alatt szeretnénk megjeleníteni figyelnünk kell y értékének változtatására. Ha egymás mellett szeretnénk megjeleníteni az értékeket érdemes a fájl írásakor elválasztó karaktereket tenni (például vesszőt) annak érdekében, hogy az adatok jól elkülönüljenek egymástól. E mellett persze figyelnünk kell x értékének változtatására is, nehogy fedjék egymást a kiírandó sorok. A koordináták szabályos változtatására sok jó megoldás létezik. Ciklus használata nélkül kézzel állíthatjuk be minden egyes Display blokk esetén a megfelelő értéket. Ha ciklust használunk, eszünkbe juthat változó használata. Ennek egy praktikus módja magának a ciklusváltozónak a felhasználása. A 66. ábra ennek használatát mutatia be.



66. ábra. Ciklusváltozó használata koordináták módosítására

3.6.2. Feladatok

1. **Dobókocka robot:** Robotod hozzon létre egy *dobokocka* nevű fájlt. Ennek tartalma az általa véletlenszerűen generált 100 db dobókocka dobás eredménye legyen (vagyis 1 és 6 közötti egész számok). Ellenőrizd a fájl tartalmát letöltve azt számítógépedre!

2. Hozz létre egy sebesseg.rtf nevű fájlt, mely egyetlen szám értéket tartalmaz! Robotod haladjon egyenesen, sebességét a fájlból kiolvasott adat határozza meg!

3. **Hirdetőtábla robot:** Találj ki egy rövid hirdetés szöveget és töltsd fel azt robotodra hirdetes.rtf néven! Készíts programot, amely kiolvassa a *hirdetes* nevű fájl tartalmát és az első négy sort kiírja a robot képernyőjére félkövér betűtípussal! Mivel egy hirdetésnél a lényeg a nagy nézettség, robotod villogtassa ledjét, hogy felhívja magára a figyelmet!

4. **Felderítő robot:** Robotod nagy kalad előtt áll! Küldetése, hogy felderítést végezzen egy távoli bolygón. Ehhez a következő feladatokat kell teljesítenie: tetszőleges mozgást végezve pásztázzon át egy területet, eközben másodpercenként mérje meg a hőmérsékletet, a környezeti fényerősséget és állapítsa meg milyen színű a talaj! Ezeket az információkat jegyezze fel három különböző fájlba! (A fájloknak beszédes nevet adj, mert még szükség lesz rájuk!)

5. **Felderítő robot 2:** Amikor robotod sikeresen végzett a bolygó feltérképezésével, hazatérve beszámol a mérési eredményekről. Másodpercenként kiolvas 1 elmentett hőmérséklet értéket, 1 fényerősséget és egy talajszínt a mentett fájlokból. Ezeket az adatokat egymás alá kiírja képernyőjére. (Továbbfejlesztési lehetőség: a képernyőn az értékeket úgy jeleníti meg, hogy minden sor erején feltünteti a mentett adat típusát. Pl.: hőmérséklet: *mért érték*)

3.7. 7.alkalom

3.7.1. EV3 robotok közötti kommunikáció

Ezek az EV3 robotok egyedül is számos feladat megoldására képesek, hát még ha összefognak és egymást segítve dolgoznak! A következő részben olyan lehetőségekkel ismerkedhetsz meg, melyek alapja a robotok közötti kommunikáció üzenetküldés segítségével. Ennek technikai feltétele, hogy legalább 2 robot rendelkezésre álljon. Minél több robottal dolgozunk, annál több lehetőséget rejt ez a téma, hiszen nem csak két robot tud egymással "beszélgetni", hanem akár többen is összedolgozhatnak. Vágjunk is bele, fedezzük fel hogy tudnak kommunikálni egymással robotjaink!

Robotok közötti kapcsolat létrehozása

Az emberek közötti kommunikáció egyik fontos eleme a csatorna, melynek feladata a jelek továbbítása adó és vevő között. Így van ez a robotok világában is! Az EV3-ak esetén Bluetooth kapcsolatot használunk, így a robotok rádiófrekvencia segítségével tudnak beszélgetni egymással.

Ebben a kis bevezető részben beszélgethetünk csoportunkkal a kommunikációról, annak fajtáiról. Besorolhatjuk a robotok közötti kommunikációt iránya-, résztvevői-, kódja szerint. Ezzel nagyban támogathatjuk, erősíthetjük a tantárgyközi kapcsolatokat.
A Bluetooth is nagyon érdekes téma, melyről érdemes beszélgetni! Akár kiadhatjuk kutatómunkának és meghallgathatjuk lelkes, szorgalmas diákjaink beszámolóját erről a rádiófrekvenciás kommunikáció protokollról.

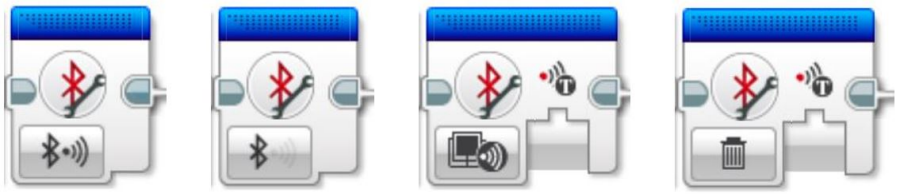
Most nézzük meg milyen lépéseket kell tennünk ahhoz, hogy létrejöjjön két robot között az előbb említett Bluetooth kapcsolat! Két mód közül is választhatunk. Az egyik a tégla menüjében történő beállítás közvetlenül a roboton, a másik a szoftveres, blokkok segítségével.

Csatlakoztatás a tégla menüjében lépésről lépésre:

1. Egymáshoz csatlakoztatni kívánt robotokat bekapcsolása
2. Beállítások - 4. menüpont
3. Bluetooth és láthatóság bekapcsolása mindkét roboton
4. Bluetooth menüben Connections kiválasztása kapcsolatot kezdeményező roboton
5. Ha volt már korábban egymáshoz csatlakoztatva a két robot, megjelenik a másik robot neve a kedvencek között - ezt bepipálva csatlakoztatás
6. Ha nem voltak még soha egymáshoz csatlakoztatva vagy töröltük a kedvencek közül, akkor a keresés (Search) opció választása
7. Keresés eredményeként megjelenik a másik robot neve - ezt bepipálva csatlakoztatás
8. A két robot első csatlakozásánál ellenörző kódot kérhet az egyik robot a másiktól. Ekkor a kapott kódot írjuk a másik roboton megjelenő téglalapba, majd fogadjuk el a csatlakozási kérést.

Bluetooth Connection blokk

A szoftverben az Advanced vagyis haladó (kék) blokkok között találjuk a Bluetooth kapcsolat létesítéséhez szükséges Bluetooth Connection blokkot. A módválasztóra kattintva 4 lehetőség közül választhatunk: On, Off, Initiate, Clear. Nézzük meg most ezeket részletesen.

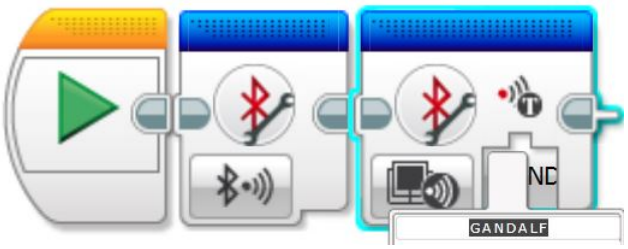


67. ábra. Bluetooth Connection blokk On, Off, Initiate és Clear módja

Az **On**, paraméter nélküli blokk futtatásakor a robot bekapcsolja magán a Bluetooth-t. **Off** módban futtatva a blokkot a Bluetooth kikapcsol. Az **Initiate** módot futtatva kapcsolatot kezdeményez azzal a robottal, amelyiknek nevét paraméterül átadtuk. Végül a **Clear** mód arra szolgál, hogy a meglévő Bluetooth kapcsolatot megszüntessük azzal a robottal, amelyiknek nevét paraméterül átadtuk. Az utolsó két mód esetén a paraméter értékének megadása történhet vezetéken való értékátadással és szöveg közvetlen begépelésével egyaránt.

Csatlakoztatás a szoftverben blokkok segítségével lépésről lépésre:

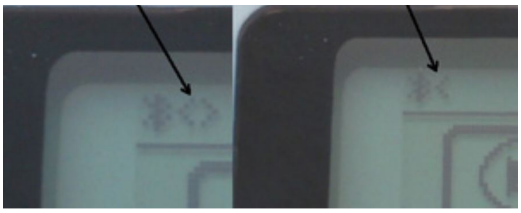
1. Egymáshoz csatlakoztatni kívánt robotokat bekapcsolása
2. Kapcsolatot kezdeményező robot csatlakoztatása a számítógéphez (rajta kell futtatni a programot)
3. Advanced (haladó - kék) menü kiválasztása
4. Bluetooth Connection (balról a harmadik) blokk használata kétszer egymás után
5. Először Bluetooth Connection blokk ON módban
6. Utána Bluetooth Connection blokk Initiate módban (Paraméter értékének megadása: azon robot neve, akivel a kapcsolatot kezdeményezi)
7. Program futtatása a roboton



68. ábra. Kapcsolat kezdeményezése Gandalffal

(Megjegyzés: abban az esetben, ha a másik roboton is a szoftver segítségével szeretnénk bekapcsolni a Bluetooth-t, őt is csatlakoztatni kell a számítógéphez és lefuttatni rajta a 68. ábrán látható programnak csak a Play utáni első blokkját - Bluetooth Connection ON.)

Bármelyik módot is választjuk a robotok közötti kapcsolat létrehozásához, ennek sikerességéről visszajelzést kapunk a robot képernyőjén. A bal felső sarokban a Bluetooth jel mellett jobbra látható ikon megváltozik. Bluetooth kapcsolat esetén a 69. ábra[7] bal oldalán látható ikon jelenik meg. Ha robotunk nincs semelyik másik robottal párosítva, azt a jobb oldali képen látható módon jelzi.



69. ábra. Bluetooth kapcsolat jelzése

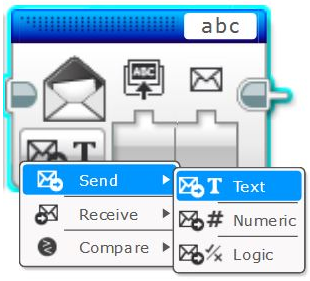
A robotok közötti kommunikációnak fontos, működést és felhasználhatóságot befolyásoló tulajdonsága a kapcsolat fajtája. Az EV3 robotok közötti bluetooth kapcsolat **Master-Slave elvű**. Ez azt jelenti, hogy van egy kitüntetett Master (mester), ő kezdeményezi a kapcsolatokat. Ez a "vezér" robot legfeljebb 7 másik robothoz tud csatlakozni egyidőben - ők a "Slave-ek", vagyis szolgák. A mester bármelyik hozzá csatlakoztatott robotnak tud üzenetet küldeni. A szolgák azonban csak a mesternek küldhetnek üzenetet, célzottan másik alárendelt robotnak nem.

Üzenetek küldése és fogadása

A sok-sok előkészület után végre elérkeztünk a legizgalmasabb részhez. Már vannak párosított robotjaink, jöhet a üzenetküldés!

> *Mielőtt nagyon belemélyednénk, érdemes diákjainkkal beszélgetni arról, milyen előnyökkel járhat a robotok kommunikációja. Mi az amit egyedül egy robot nem tudna teljesíteni, de ketten már elboldogulnának vele. Bemelegítő feladatként érdemes robotok közötti interakciót megvalósítani üzenetküldés nélkül. Ehhez ad ötletet az 59. oldalon található 1. feladat.*

Az üzenetekért felelős blokkot - a Bluetooth Connection blokkhoz hasonlóan - a haladó (kék) menüben találjuk (balról a második).



70. ábra. Messaging blokk módválasztó

Ezen blokk segítségével tudunk üzeneteket küldeni, fogadni, illetve összehasonlítani a 70. ábrán látható megfelelő mód kiválasztásával. Mindhárom lehetőség esetében szükséges az üzenet típusának megjelölése is. Ez lehet szöveg (text), szám (numeric), logikai (logic) típus.

Minden üzenetet három összetevő jellemez: a téglák amelyek között az üzenet továbbításra kerül; az üzenet címe, mely azonosításra szolgál és az üzenet típusa, ami az előbb említett három lehetőség közül kerül ki.

Nézzük meg részletesen melyik módban milyen lehetőségeink vannak.

[7] `https://www.ucalgary.ca/IOSTEM/ev3-programming/ev3-communicating-between-robots`

Send - üzenet küldése

Amikor üzenetet szeretnénk küldeni egyik robottal a másiknak, először az üzenet típusát kell meghatároznunk. A Messaging blokk paraméterei ennek függvényében változnak. Ha ezzel megvagyunk, értéket kell adnunk a blokk 2 paraméterének. Az első annak a robotnak a neve, akinek címezzük az üzenetet. Ezt kiválaszthatjuk a listából (ha korábban már csatlakoztunk az adott robothoz) vagy begépelhetjük. A második paraméter az üzenet szövege. Text esetén ez bármilyen szöveg lehet, néhány szabály betartása mellett. Az üzenet nem tartalmazhat ékezetes és speciális karaktereket. Numeric esetén bármilyen valós számot megadhatunk. Ha pedig a logikai típust választjuk, igaz vagy hamis értéket küldhetünk a másik robotnak. Végül egy nagyon fontos lépés: címet, azonosítót kell adnunk üzenetünknek. Ezt a blokk jobb felső sarkában tehetjük meg. Itt is korlátozva vannak a felhasználható karakterek, ahogy az összes többi szöveg esetén a szoftverben. A 71. ábrán a három különböző típusú üzenet címzettje egyaránt Gandalf. Az üzeneteket érdemes beszédes névvel azonosítani, ahogy a példában a "hello" tartalmú üzenet címe "koszon", vagy a logikai típusúé "igaze". Így amikor sok üzenettel dogozunk, nem fogjuk összekeverni azokat.

71. ábra. Üzenetküldés

Receive - üzenet fogadása

A 72. ábrán az előző részhez kapcsolódó üzenetek fogadása látható. Ezeknek a blokkoknak nincs egy bemeneti paramétere sem, csupán egy kimeneti paraméterrel dolgozhatunk. Ennek segítségével vezetéken felhasználhatjuk az üzenet tartalmát. Figyeljük meg, hogy mindhárom blokk esetén más a kimeneti paraméter "alakja". Az első, négyzet végződésű szöveges üzenetet tartalmaz, a második, kör végződésű számot, míg a harmadik, háromszög jelölésű logikai értéket. Ennek megfelelően használhatjuk fel bemeneti paraméter értékének megadásához.

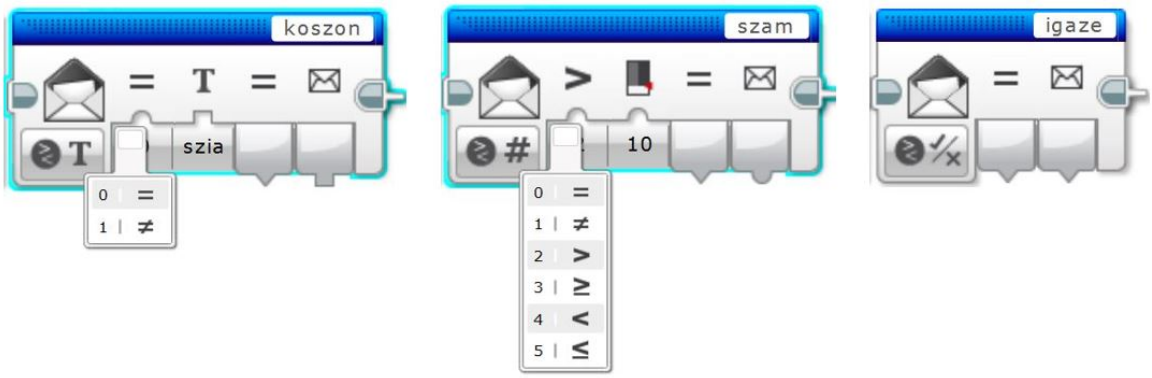
Már rengeteg blokkal megismerkedtünk. Ez egy jó alkalom lehet az ismétlésre. Diákjainkkal gyűjtsük össze brainstorming jelleggel, hogy melyik blokk melyik paraméterének adhatnánk a "koszon", "szam", illetve "igaze" üzeneteket bemeneti értékként. Ennek eldöntésében nagy segítség a blokkok input és output mezőjének formája , hiszen azonos alakzatot csak azonoshoz van értelme bekötni.

72. ábra. Üzenet fogadása

Az előző részben megismert, Send módban küldött üzenet azonnal megérkezik a címzettként megjelölt robothoz. Erről nem ad visszajelzést a robot, akinek küldtük. Valójában ő csak eltárolja memóriájában az információt, az üzenet minden tulajdonságát. Amikor programja elér egy Receive (üzenet fogadása) blokkhoz, megnézi, kapott-e már ilyen azonosítójú üzenetet és ha igen, akkor tud dolgozni az abban tárolt értékkel.

Compare - üzenet összehasonlítása

Amikor robotunk kap egy üzenetet, lehetősége van annak tartalmát összehasonlítani egy adott értékkel. Azon értéket, amellyel összeveti az üzenetet megadhatjuk vezetéken keresztül egy másik blokk adatának kiolvasásával vagy közvetlenül begépelve. Utóbbira jó példa a 73. ábra, mely a három különböző adattípus esetét mutatja be. Egy adott üzenetet csak vele megegyező típusú adattal hasonlíthatunk össze. Így például a "koszon" azonosítójú üzenetet szöveggel.



73. ábra. Kapott üzenet összehasonlításának 3 típusa

Nézzük meg milyen kimeneti és bemeneti paraméterek állnak rendelkezésünkre a különböző üzenettípusok esetén.

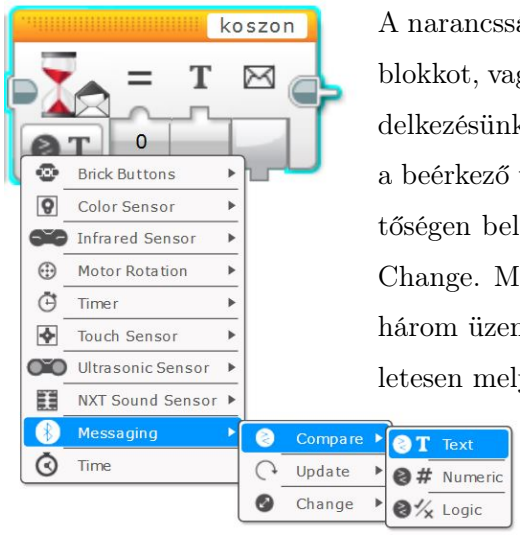
A szöveges (text) üzenet egyenlőségét vizsgálhatjuk egy másik szöveggel. Az első bemeneti paraméter értéke 0 vagy 1 aszerint, hogy az egyenlőséget vagy a különbözőséget akarjuk-e vizsgálni. A második input az a szöveg, mellyel összevetjük a kapott üzenet tartalmát. Az első kimeneti érték logikai típusú. Igaz, ha teljesül a vizsgált egyenlőség (vagy különbözőség) és hamis, ha nem. Két szöveg különböző, ha csak egy karakterében is eltér! Ezt a logikai kimenetet felhasználhatjuk egy másik blokk bementeként. Az utolsó output érték magát az üzenetet tartalmazza. Ez egyenlőség esetén nem meglepő, ugyanaz, mint amivel összehasonlítottuk. Ha viszont eltérést találtunk, érdekes lehet a tartalma!

Ha szám típusú üzenetet kaptunk nagyon hasonló felépítésű blokkal találjuk szemben magunkat. A különbség csak annyi, hogy többféle relációt is vizsgálhatunk és az érték, mellyel összevetjük az üzenet tartalmát szám kell legyen.

Végül a 73. ábra jobb szélén található blokk a logikai üzenettel való összehasonlítást mutatja. Ez különbözik az előző két esettől, hiszen itt csupán két kimeneti paraméter találunk. Ebben az esetben alapértelmezetten igaz értékkel hasonlít a program. Tehát ha igaz volt az üzenetben, igazat fog adni, míg hamis esetén az érték hamis lesz.

Várj blokk használata üzenetek kezelésekor

A várj blokkot számos esetben hívtuk már segítségül feladataink megoldásakor. Az összes eddig tanult érzékelőnél használtuk, várakoztunk a szenzorok egy bizonyos értékére vagy éppen állapotuk megváltozására. Most, az üzenetek fogadásakor is fontos szerepe lesz. De hogyan és mire tudjuk majd használni? Olvass tovább és kiderül!



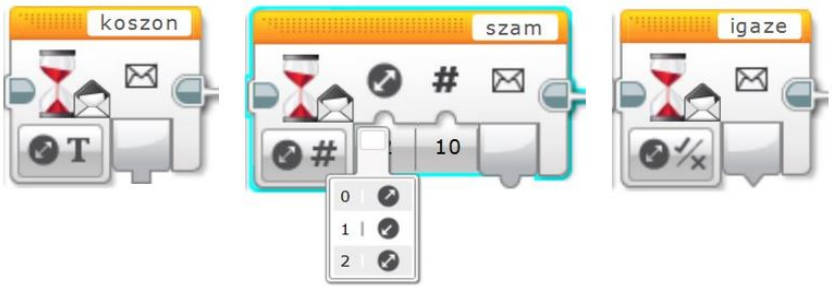
74. ábra. Várakozás egy üzenetre

A narancssárga, vezérlési szerkezeteket tartalmazó menüből a homokórás blokkot, vagyis a várakozást fogjuk használni. Ebben számos mód áll rendelkezésünkre. Válasszuk ki a Messaging feliratút. Ezzel tudjuk kezelni a beérkező üzeneteket. Ahogy a 74. ábrán látható, az üzenetküldés lehetőségen belül is három opció közül választhatunk: Compare, Update és Change. Majd ezen belül is elénk tárul három, már ismerős lehetőség, a három üzenettípus, melyekkel már találkoztunk. Most nézzük meg részletesen melyik lehetőséget hogyan és mire érdemes használni!

Az első, Compare opció már ismerős lehet, hiszen az előző oldalon végig erről volt szó csak nem a várakozás blokkban hanem az üzenetküldésnél. A hasonlóság nem véletlen és éppen ezért most csak az eltéréseket fogom részletesen leírni. Az első szembetűnő különbség és megjelenésében az egyetlen, hogy eggyel kevesebb kimenő paraméterünk van. Itt nem tudjuk felhasználni az egyenlőségvizsgálat logikai típusú eredményét. A többi ki- és bemeneti paraméter jelentése megegyezik. A sok hasonlóság mellett azonban van egy nagy használatbeli különbség. Míg a Messaging blokk Compare opciójának használatakor nem voltunk kíváncsiak arra, mikor érkezett az üzenet, a várakozás blokk esetén ez lesz a lényeg. A homokórás blokkot akkor érdemes haszálni, ha várakozni szeretnénk adott feltétel teljesüléséig. A Messaging - Compare opciót akkor válasszuk, ha az érkező üzenet tartalmát össze szeretnénk hasonlítani egy általunk adott értékkel. Programunk akkor lép tovább a várakozás utáni blokkokra, ha ez a vizsgálat igaz eredményt adott.

Az Update módot akkor érdemes választani, ha várakozni szeretnénk adott nevű és típusú üzenet érkezésére. A program az üzenet tartalmától függetlenül vár, amíg az meg nem érkezik a másik robottól, majd folytatja futását. Azt sem veszi figyelembe, hogy korábban érkezett-e már ugyanilyen nevű, tartalmú üzenet. Update mód esetén mindhárom típusnál egyetlen kimeneti paraméterrel tudunk dolgozni, az üzenet tartalmát tudjuk felhasználni vezetéken keresztül egy másik blokk bemeneti paraméterértékeként.

75. ábra. Messaging - Update



76. ábra. Messaging - Change

Végül, de nem utolsó sorban nézzük a Change módot. Ennek használatakor a robot arra vár, hogy érkezzen adott nevű és típusú üzenet a korábbitól eltérő tartalommal. Tehát ezt akkor érdemes hasz-

nálni, ha egy üzenet tartalma a program futása során megváltozhat és ilyen esetben robotunktól más működést várunk. Szöveg és logikai típusú üzenet esetén a blokknak egyetlen kimeneti paramétere van, akárcsak Update módban. A szám típusnál viszont több paraméter megadásáról is gondoskodnunk kell, ahogy azt a 76. ábra középső blokkján láthatjuk. Az első paraméter arra szolgál, hogy megadjuk a várt változási irányt. Ez lehet növekvő, csökkenő vagy bármely irány elfogadása. A második bemeneti paraméter egy számot vár, ez a változás várt mértékét fejezi ki. A harmadik, kimeneti paraméter a többi típusnál is látható, az üzenet értékének átadásáért felelős rész.

Összefoglalva, a várakozás blokk üzenetfogadás esetén azt a célt szolgálja, hogy arra azonnal reagálni tudjon robotunk. Ezt a reakciót feltételhez is köthetjük, de várakozhatunk csak az üzenet megérkezésére, jelezve ezzel, hogy a kézbesítés sikeres volt.

3.7.2. Feladatok

A feladatok leírásában a könnyebb érthetőség érdekében robotneveket is használok, így megkülönböztetve a szerepeket egy-egy feladat esetén.

1. **Bemelegítő feladat** - Robotok interakciója üzenetküldés nélkül: A következő feladatban két robotra lesz szükséged! Az egyik robot feladata, hogy "énekelve" egyenesen haladjon előre közepes sebességgel. A másik robot nem szereti, ha barátja hangosan énekel a fülébe, így ha közel ér hozzá, a nagy hangerő hatására kicsit távolabb megy. Mindig elmenekül, hogyha hangoskodnak mellette. (A feladat végrehajtásához az éneklő robot a másik robot mögött helyezkedjen el, tőle legalább 1 méteres távolságot hagyva kezdetben!)

2. Készíts programot, melynek hatására Rozi (az egyik robot) a nyomásérzékelő változásának hatására szöveges üzenetet küld barátjának! Gandalf (az üzenetet fogadó robot) az üzenet érkezésekor azonnal kiírja az üzenet szövegét a képernyőjére és hangosan örül barátja jelentkezésének.

3. Alkoss távirányító robotot! Most az egyik robot távirányítóként fog szolgálni, és ezzel fogjuk irányítani a másik robotot. Használd a tégla programozható gombjait! Amíg nyomva tartjuk az irányító robot felső gombját, az irányított robot menjen folyamatosan előre, az alsó gomb hatására hátra. A bal gombot megnyomva forduljon balra 90°-ot, a jobb gomb hatására jobbra ugyanennyit. Ha a középső gombot nyomjuk meg, villogtassa pirosan ledjét és rajzoljon STOP táblát képernyőjére. Ez a rajz ne legyen a képernyőn, amikor a robot mozgásban van! (Tipp: az elágazás szerkezet Text módjában felhasználhatjuk a kapott szöveges üzenet tartalmát.)

4. A következő feladatban Gandalf üzenetben felkéri Rozit táncolni. Rozi a véletlenre bízza a dolgot, 50% valószínűséggel elfogadja a felkérést. Amikor döntött, kiírja képernyőjére, hogy "dontottem" és válaszol Gandalfnak. Ha Rozi igennel válaszolt, akkor mindketten mosolyognak és körbe forognak 3 másodpercig. Ha Rozi elutasította a felkérést Gandalf szomorú lesz és távolabb megy.

5. Táncoló robotok - csak szinkronban! A következő feladathoz akár 3 robotot is használhatsz! Legyen egy kitüntetett robot, aki a legjobb táncos. Ő random generált értékek szerint állítgatja motorjait. Erről információt küld folyamatosan a két "háttértáncosnak" akik próbálják őt ennek segítségével utánozni. Ha jól dolgoztál, a robotok szinkronban táncolnak! :)

6. **Projektfeladat:** Mesedélután - A következő feladathoz több robotra lesz szükséged! Akár 3-4 robottal is összedolgozhattok! Találjatok ki egy egyszerű kis mesét, melynek a robotok lesznek a szereplői. Legyen egy főszereplő, hozzá kell csatlakoztatni a többi robotot. Készítsetek hangfelvételeket! A történet mozgásokból és mesélésből fog állni. A robotok üzenetekkel jelezzék egymásnak, ha a másik következik a mesében.

3.8. 8.alkalom

3.8.1. Eszközök láncba kötése

Az előző részben megismerkedtünk egy lehetőséggel, mellyel robotjaink egymást segítve, összefogva tudtak feladatokat megoldani. Kommunikációjuk alapja a Bluetooth kapcsolat volt, így tudtak üzeneteket küldeni egymásnak. A következő részben robotjaink megint egymást fogják segíteni, de egészen más módon, mint korábban.
A Lego robotprogramozásban elmélyülve idővel eljuthatunk olyan komplex feladatokig, melyek esetén az egy robothoz csatlakoztatható motorok/érzékelők száma kevésnek bizonyul. Ilyenkor segíthet a láncba kapcsolásos technika, vagyis a Daisy Chaining. Nézzük meg mit is jelent ez!

Összekapcsolás és átépítés

A Daisy Chaining név arra utal, hogy robotjainkat úgy kapcsolhatjuk láncba, mint ahogy a százszorszépeket koszorúba fonhatjuk. Most az EV3 téglákat kapcsoljuk össze kábellel. Az összekapcsolt robotok közül csak az elsőt lehet programozni, de rajta keresztül elérhető a többi téglához csatlakoztatott érzékelő, motor is. Így nem csak 4 bemeneti és 4 kimeneti portja van robotunknak, hanem annyiszor 4 ahány robotot sorbakötöttünk. Maximum 4 robotot kapcsolhatunk össze a következő módon:

77. ábra. Téglák láncba kötése

Válasszuk ki azt a téglát, aki a "főnök", az irányító lesz! Azért fontos ez a lépés, mert az összekapcsolásnál számít a robotok sorrendje! Miután láncba kötöttünk több téglát nem választhatjuk meg szabadon melyiken futtatjuk a megírt programot! Az első, kitüntetett téglán az USB feliratú csatlakozóba kell dugnunk az USB kábelt. A 77. ábrán[8] látható, 2-es jelzésű tégla PC feliratú mini USB csatlakozójába pedig a kábel másik végét. Ha folytatni szeretnénk a sort egy harmadik tégla csatlakoztatásával, akkor a 2-es USB csatlakozóját és a 3-as tégla PC, mini USB csatlakozóját kell használnunk. A robotok csatlakoztatása mellett nagyon fontos az is, milyen sorrendben kapcsoljuk be a téglákat. Ahogy az előbb említettem, van egy kitüntetett, 1-es számú robot (rajta fogjuk futtatni a szoftverben megírt programokat). Az 1-eshez közvetlenül kapcsolt tégla legyen a 2-es számú, az ő szomszédja a 3-as, majd a 4-es zárja a sort.

[8] `https://ev3-help-online.api.education.lego.com/Retail/en-us/page.html?Path=editor%2FDaisyChaining.html`

A téglák bekapcsolásánál éppen fordítva kell haladnunk. A legutolsó láncszemet (most a 4-es) kell először bekapcsolni, utána a 3-as téglát, majd utánuk jöhet a 2-es és végül az 1-es, kitüntetett robot!

Mivel diákjainknak egy új, az előzőektől nagyon eltérő témát mutatunk valószínűleg izgatottak lesznek és minél előbb szeretnék kipróbálni ezen robotlánc működését. Érdemes még az összekapcsolás előtt felhívni figyelmüket ezekre a fontos szabályokra, mivel be nem tartásuk rossz működéshez és így csalódáshoz vezethet!

Tehát a Daisy Chaning módszer ereje abban rejlik, hogy egy téglával sokkal több motort és érzékelőt lehet használni, mint amennyit alapból lehetne. Ez főleg nagy projektek megvalósításakor nyer értelmet. Az eddig használt, "autónak" megépített robotot mindenképp érdemes szétszedni és csapatokat alkotva több robot alkatrészeiből egy nagy robotot építeni több tégla, motor, érzékelő felhasználásával.

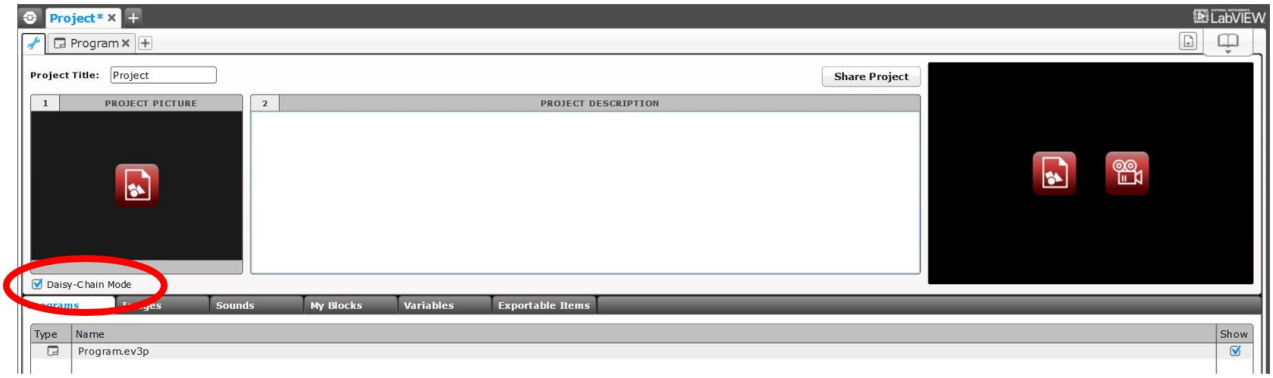
Ajánlott terv a szakkör megtartásához:

- *Csapatok alkotása (legalább 2-3 robot álljon rendelkezésre 1 csapatban)*
- *Nagy projektfeladat kihírdetése, melyben szabad kezet lehet adni diákjainknak az építésben és program megvalósításában (motivációtól és életkortól függően akár a teljes projektfeladat kitalálását is a kis csoportokra bízhatjuk)*
- *Nagy robotok megépítése*
- *Programozás*
- *Csapatok bemutatják egymának elkészült munkájukat*

Projektötletekért nézd meg a 64. oldalon található Feladatok részt!

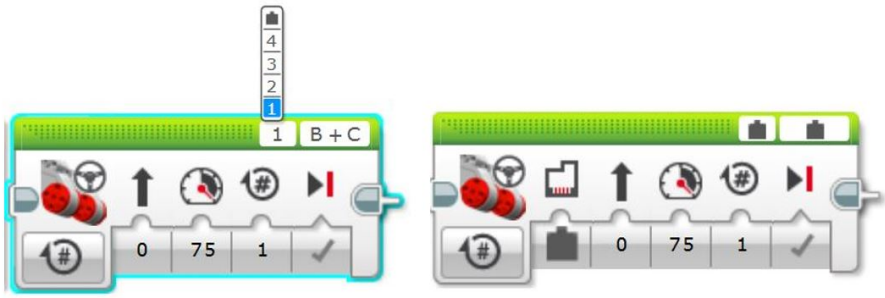
Összekapcsolt robotok programozása

Ahhoz, hogy ezt a sok összekapcsolt téglát egy kitüntetett tégla segítségével irányíthassuk a projekt beállítások fülön be kell pipálnunk a Daisy-Chain Mode jelölőnégyzetet. Ennek megtalálásában nyújt segítséget a 78. ábra.



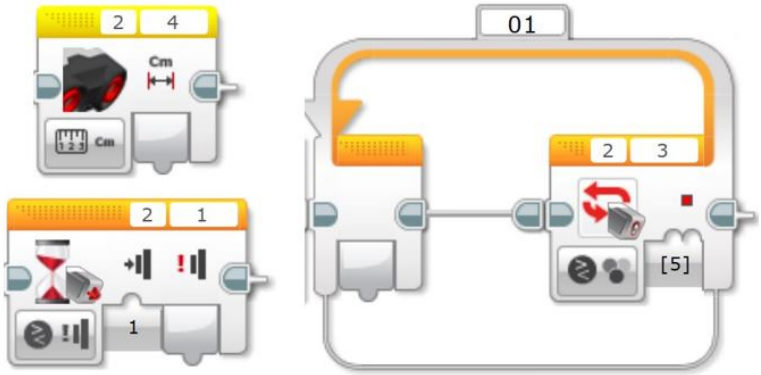
78. ábra. Daisy-Chain Mode bekapcsolása

Ha bepipáltuk az előző ábrán pirossal karikázott módot, a programozási felület támogatni fogja a láncolt működést és blokkjaink is picit megváltoznak. A motorblokkok és szenzorokhoz kapcsolódó blokkok esetén a jobb felső sarokban lévő portbeállítás előtt megjelenik egy új téglalap egy számmal. Itt tudjuk beállítani, hogy az adott parancs melyik téglához csatlakoztatott motort/érzékelőt irányítsa.



79. ábra. Tégla választása motorblokk esetén

A 79. ábrán látható 5 lehetőség közül választhatunk. Az alsó 4 a sorba kapcsolt téglák sorszáma. A legfelső lehetőséget a Display blokkból már jól ismerjük. Ha ezt választjuk blokkunk a 79. ábra jobb blokkjához hasonlóan eggyel több bemeneti paramétert kap. Az első paraméter szolgál a tégla sorszámának kiválasztására, amit ekkor egy másik blokk kimeneti értéke segítségével tudunk vezetéken átadni. (Fontos, hogy ez a szám csak 1-4 közötti lehet!)



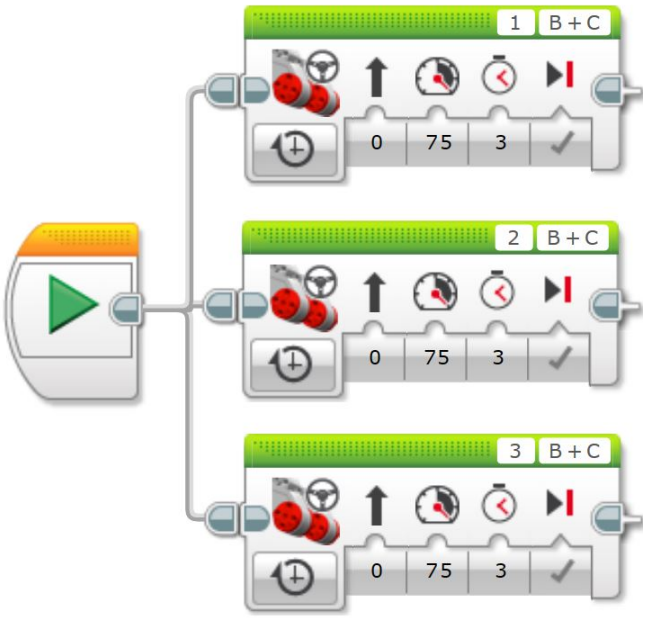
80. ábra. Blokkok szenzorok használatához

Ezen plusz téglalap, a téglaválasztó sorszám azon blokkok esetén jelenik meg, melyeknél eredetileg is ki lehet választani a portokat. Ilyenek a motorblokkokon kívül a szenzorokhoz tartozó érzékelő blokkok, illetve vezérlési szerkezetek is, melyből néhány példa látható a 80. ábrán.

A képernyő, LED és hang kezeléséért felelős blokkok kiesnek ebből a körből, így ezek csak az első téglán vezérelhetőek, melyen közvetlenül futtatjuk a programot. A többi tégla képernyőjét, ledjét és hangszóróját nem tudjuk irányítani.

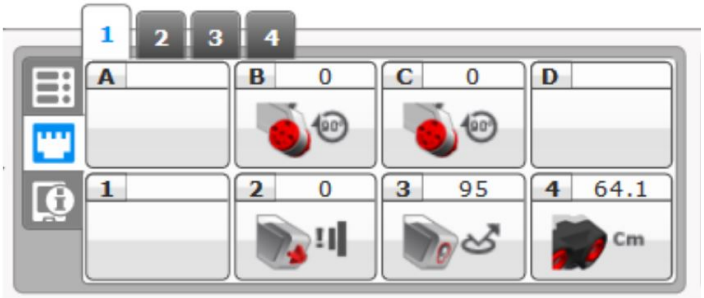
Bár egy blokk segítségével egyszerre csak 1 érzékelőnek vagy 2 motornak tudunk utasítást adni, párhuzamos programozáshoz kötődő ismereteinket alkalmazva több motort és szenzort is kezelhetünk egyszerre külön ágakon. Ennek lehetőségét mutatja be a 81. ábrán látható program. Itt egyszerre 6 motort mozgatunk, hiszen az 1-es, 2-es és 3-as tégla B és C portjában lévő motornak is adtunk utasítást a három különböző szálon.

Ezt nem csak a motorokkal, az érzékelőkkel is megtehetjük, tehát egyszerre várakozhatunk több különböző érintésszerzor értékének megváltozására vagy akár a robot más-más pontjaira felszerelt ultrahangérzékelők segítségével vizsgálhatjuk robotunk különböző részei milyen messze vannak tárgyaktól. A Daisy-Chain módban tehát nem tekintjük téglánként külön robotnak nagy építményünk. A legfeljebb 4 téglából és sok-sok érzékelőből, motorból álló alkotás egy nagy robot. Ennek része egy kitüntetett tégla (1-es számú), melyen futtatjuk a programokat. A többi tégla csak közvetítő szerepet tölt be. Valójában csak kimeneti és bemeneti portjaik miatt használjuk fel őket. Így szerepük, hogy a motorokhoz információt továbbítsanak és a szenzorok értékét közvetítsék az 1-es, "főtégla" felé.



81. ábra. Több motor irányítása külön szálon egyidőben

Ahogy egyetlen tégla használatakor, láncba kapcsolt téglák esetén is követhetjük a csatlakoztatott motorok, érzékelők adatait. A Daisy-Chain mód bekapcsolásával ez a menü is módosul. A 82. ábra felső részén látható a 4 új választófül megjelenése. Ezekre kattintva váltogathatjuk melyik tégla perifériáit szeretnénk látni a szoftver jobb alsó sarkában. A választófülek száma a láncba kapcsolt robotok számától független, mindig 4 darab.



82. ábra. Csatlakoztatott eszközöket tartalmazó panel

3.8.2. Feladatok

A következőkben néhány projektötletet ismerhetsz meg. Ezek inkább inspirációként szolgálnak, hiszen minden csoport más és más. A kiadott projektfeladatot fontos saját csoportunkra szabni. Vegyünk figyelembe több tényezőt: motiváció, érdeklődési kör, Lego építési gyakorlat, mennyi idő áll rendelkezésre, mire szeretnénk helyezni a hangsúlyt.

Több EV3 készletből egy nagy robot megépítésének csak a képzelet szab határt! Létrehozható egy félelmetes soklábú robot, egy robotkar, melyet még Vasember is megirigyelne, rubik kocka kirakó gépezet vagy egy mini gyártósor is. Ha diákjaink profi Lego építők, érdemes szabad kezet adni az építkezésnél. Így nem csak a programozás terén, de az alkotási fázisban is megmutathatják tudásukat, melyből egyedi, kreatív munkák születhetnek!

Íme néhány konkrét munkáról készült videó, melyben jól látszik a Daisy-Chain mód lényege:

1. Sorba kapcsolt daruk adogatnak egy elemet[9]
2. Soklábú robotbogár lépked[10]
3. Vasember jobb keze[11]

[9] `https://www.youtube.com/watch?v=0z01g9VKdbA`
[10] `https://www.youtube.com/watch?v=FKGCMm7T76c`
[11] `https://www.youtube.com/watch?v=7JsDiZxz_fQ`

3.9. 9.alkalom

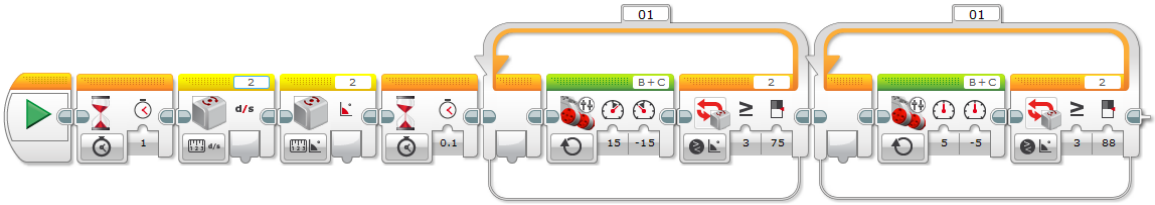
3.9.1. Saját blokk létrehozása

Gondolom, hogy már észrevettétek a blokkcsoportok között az utolsót, amiben nincs semmi. Azt már az előző tananyagban ([2]) is említettem, hogy itt a saját blokkjaink lesznek, viszont azt akkor nem mondtam/írtam le, hogy ehhez el kell készítenünk őket. Mielőtt bemutatom, hogyan kell egy MyBlock-ot elkészíteni, nézzük meg, hogy miért érdemes ezeket használni és mit tudnak.

Saját blokkot gyakorlatilag bármikor készíthetünk, azonban úgy gondolom, hogy akkor érdemes, ha bizonyos kódrészeket többször is felhasználunk és/vagy nagyon hosszú a kódunk és szeretnénk áttekinthetőbbé tenni. Mint ahogy a beépített blokkokban szerepelnek bemeneti értékek, mi is adhatunk meg ilyeneket a blokkunkban, sőt, akár kimeneti értékeket is. Az eddigiek alapján tekinthetünk úgy a saját blokkra, mint egy függvényre, azonban a függvényeket csak az adott programkódban használhatjuk, a saját blokkjainkat viszont más projektekben is alkalmazhatjuk.

Saját blokk létrehozásának bemutatása példán keresztül

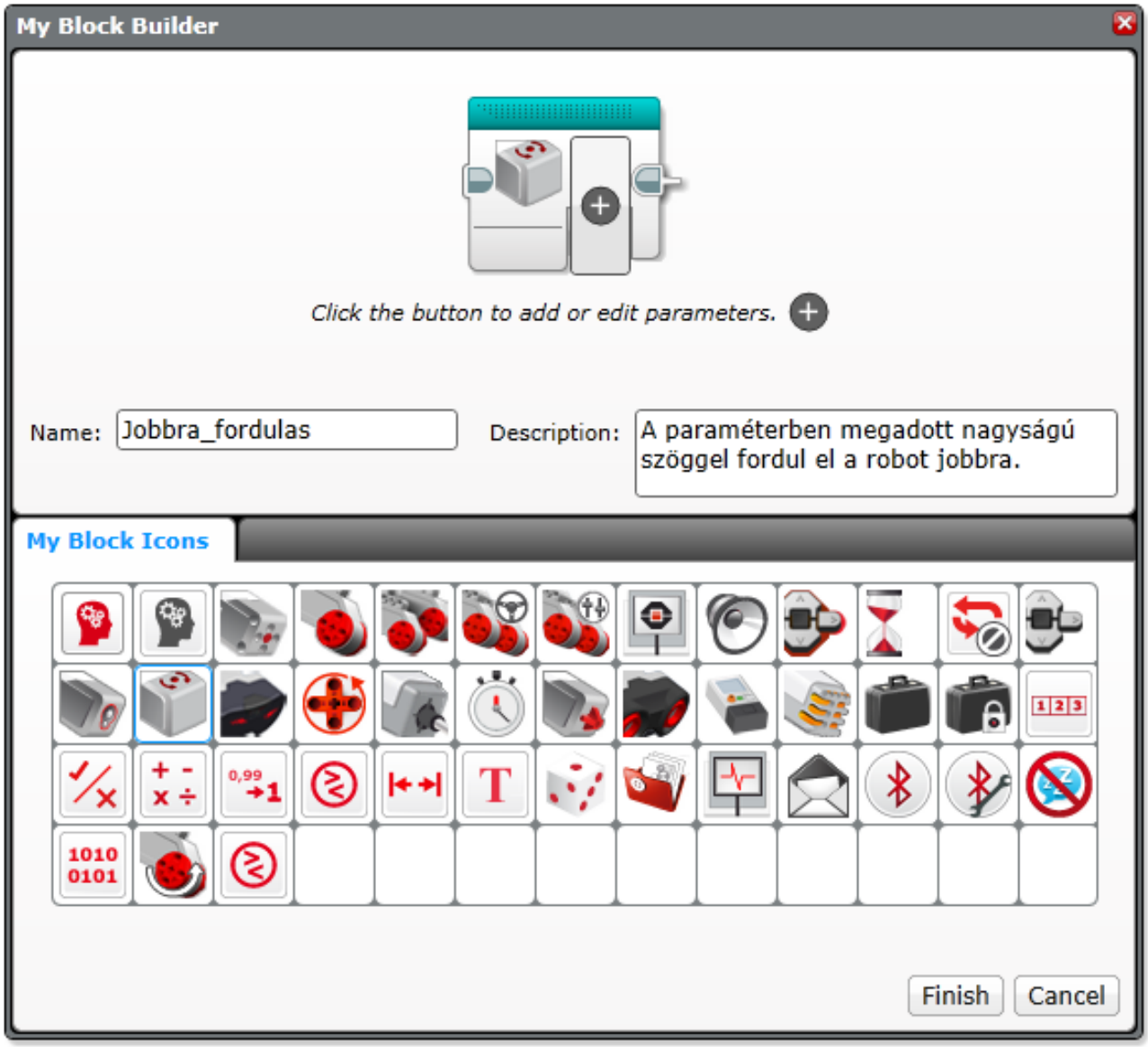
A 83. ábrán látható, a robotot jobbra 90°-ot fordító programot fogjuk felhasználni kiindulási alapnak a blokkunkhoz. Ezt a kódot úgy fogjuk átalakítani, hogy bármekkora szögben el tudjon fordulni a robot. Ehhez a saját blokk készítése során létre fogunk hozni egy változót, amiben eltároljuk a kapott szöget (amennyit szeretnénk fordulni) ill. két műveletet fogunk elvégezni, hogy a megfelelő mértékben forduljon el a robot.



83. ábra. A saját blokk általános beállításai és paramétereinek meghatározása látható a képen.

Első lépésként ki kell jelölni a saját blokkban felhasználandó kódot, de úgy, hogy ne legyen benne az indító blokk. Ha véletlenül így jelöljük ki, akkor hibát fog jelezni a szerkesztő és nem enged addig továbbmenni, míg enélkül ki nem jelöljük a kódot.
Tehát, megvagyunk a kijelöléssel, ezután a bal felső sarokban lévő menüsorban kiválasztjuk a *Tools* menüt, azon belül pedig a *My Block Builder* opciót. Ezután a 84. ábrán látható ablak fog megjelenni, ahol elnevezhetjük a blokkunkat és egy kis leírást is adhatunk róla. Továbbá, itt kell megadnunk a paramétereket is, az ablak felső részében látható minta blokkon a paraméterben szereplő + jelre kattintva. A blokk elkészítése után még szerkeszthető lesz a paraméter lista is, azonban érdemes az elején átgondolni, hogy milyen be- és kimeneti paraméterekre lesz szükségünk és a megfelelő mennyiségűt felvenni a blokkba. Erre azért is van szükség, mert a blokk működését meghatározó értékeket be kell kötnünk a megfelelő helyre a kódba, és ha valami kimarad, akkor az lehet, hogy a helyes futtatás gátja lesz.

A szerkesztő ablak alsó részében azok az ikonok láthatóak, amikből választhatunk egyet a saját blokkunkba. Én most egy gyro szenzort választottam a blokk jelölésére, hogy el ne felejtsem, hogy ennek a blokknak a működéséhez szükség van egy ilyen szenzorra.



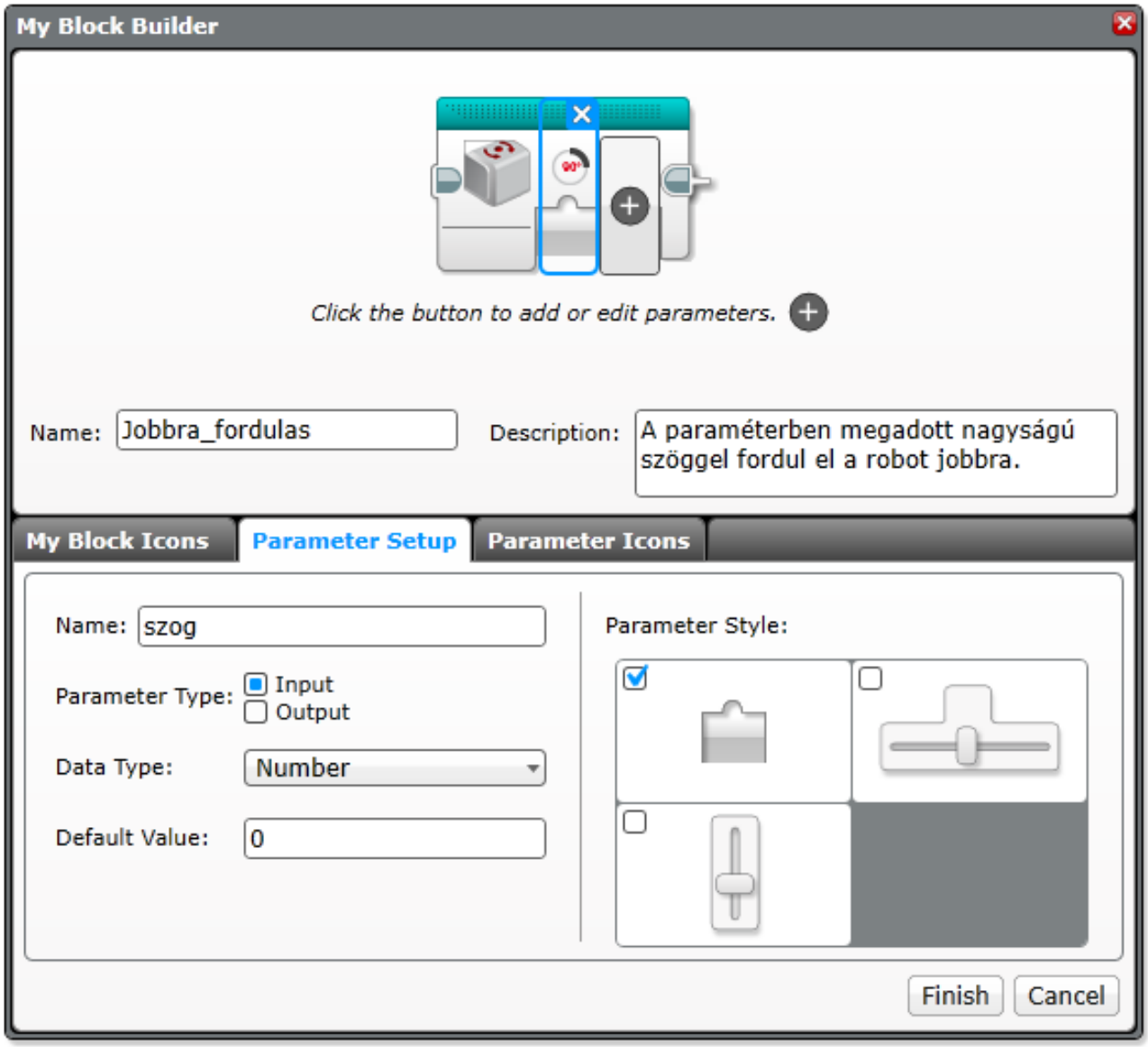
84. ábra. A saját blokk általános beállításai.

Ha az lenne a célunk, hogy a blokk egy 90°-os forgást végezzen el, akkor itt készen is lennénk a blokk külcsínjének kialakításával. (Tehát nem szükséges minden blokknak paramétert tartalmaznia.) Azonban, mi nem feltétlenül derékszögben szeretnénk elfordulni, hanem mondjuk csak 60°-ban, ezért azt valahol meg kell adnunk, hogy mekkora legyen az elfordulási szög. Azaz szükségünk van egy paraméterre.

Ha felvettük a blokkba a paramétert, akkor a 85. ábrán látható két további opció jelenik meg az ablak alsó részében. A középső, Paraméter Setup, menü bal oldalán az aktuálisan kiválasztott paraméter nevét (Name), típusát (Paraméter Type), adattípusát (Data Type) és alapértékét (Default Value) tudjuk beállítani. A paraméter típusánál azt kell meghatároznunk, hogy bemeneti (Input) vagy kimeneti (Output) értéket szeretnénk. Az adat típusánál a szokásos ötféle érték közül választhatunk, melyek a következők: szám, logikai érték, szöveg, számokat tartalmazó tömb, logikai értékeket tartalmazó tömb.

A menü jobb oldali részén a paraméter megadásának formáját állíthatjuk be. Ha valamelyik csúszkás opciót választjuk, akkor a Default Value érték alatt megjelenik két további értéket kérő beviteli mező, ahol a csúszka két határát kell megadnunk.

A másik újonnan megjelenő menüben választhatjuk ki a paraméter fölött megjelenő ikont. Itt is érdemes valami olyasmit keresni, ami utal az adott paraméterre. (Ez nem mindig könnyű, így érdemes beszédes névvel ellátni a paramétert.)

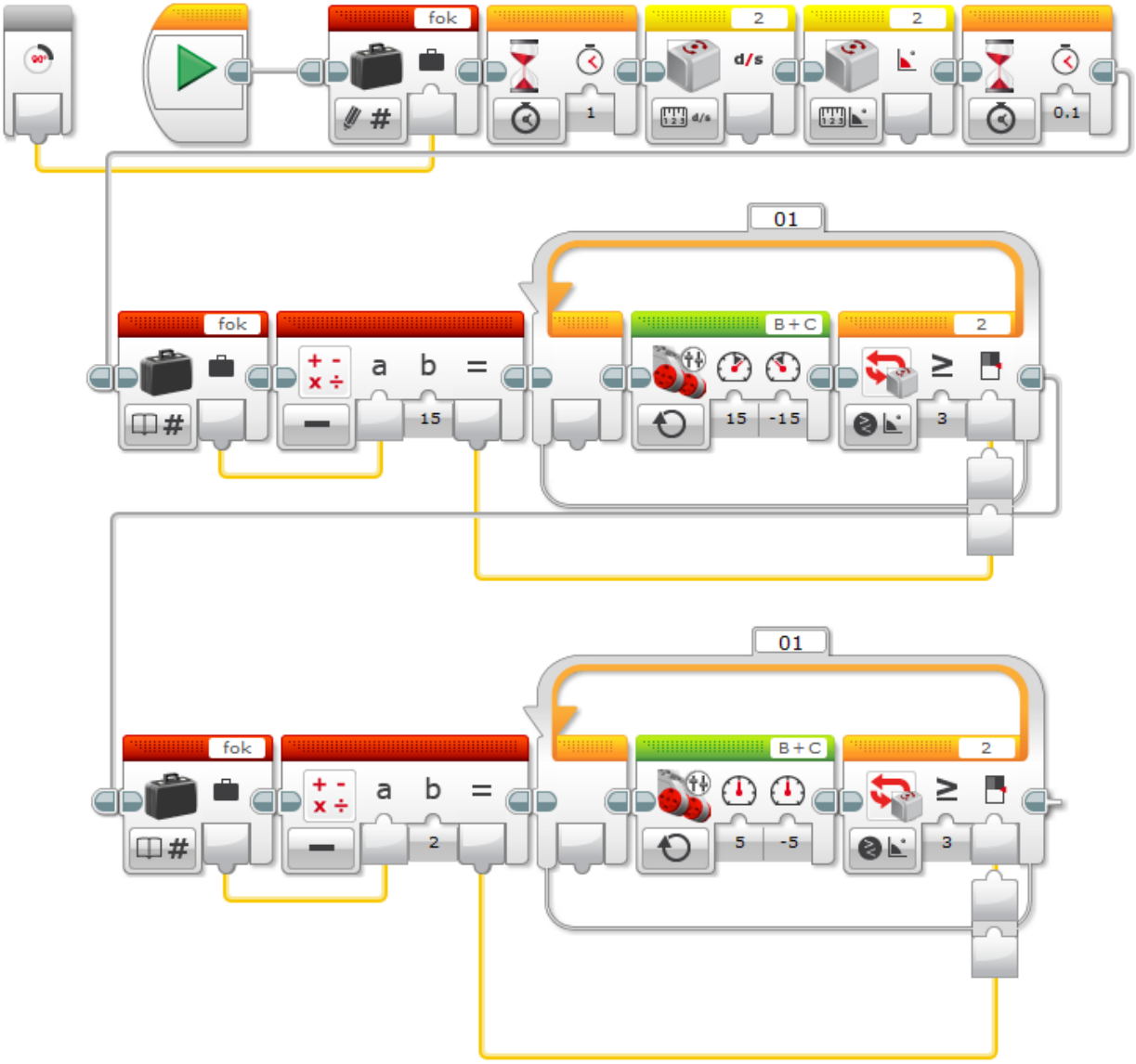


85. ábra. A saját blokk paraméterének szerkesztése.

Ha beállítottunk mindent, akkor az ablak jobb alsó sarkában lévő Finish gombra nyomjunk rá, majd a megjelenő szerkesztő felületen elkezdhetjük módosítani a korábban kijelölt kódunkat. (Erre nem minden esetben van szükség, például a korábban említett derékszögű fordulás elkészítésénél, nem kellene szerkesztenünk a kódot.)

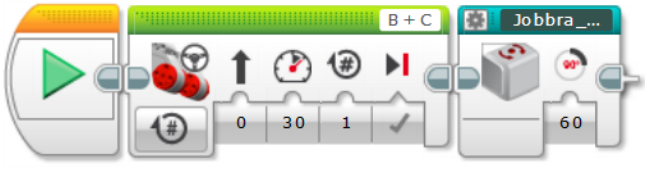
A kód bal oldali részén megjelent egy szürke blokk, amivel eddig még nem találkoztunk. Ebben a szürke blokkban kapjuk meg azt az értéket, amit a blokkban bemeneti paraméterként adtunk meg (itt a szog nevű paraméterről van szó). Tehát, mivel mi egy általánosabban használható blokkot szeretnénk készíteni, ahol mi adjuk meg a szöget, így minden olyan helyen, ahol fix értékként szerepeltettük a

fordulás mértékét, át kell alakítanunk változtathatóvá. Itt van szükség a korábban említett változó létrehozására és a matematikai műveletek alkalmazására. Ezek a beállítások és a teljes kód, a 86. ábrán látható.



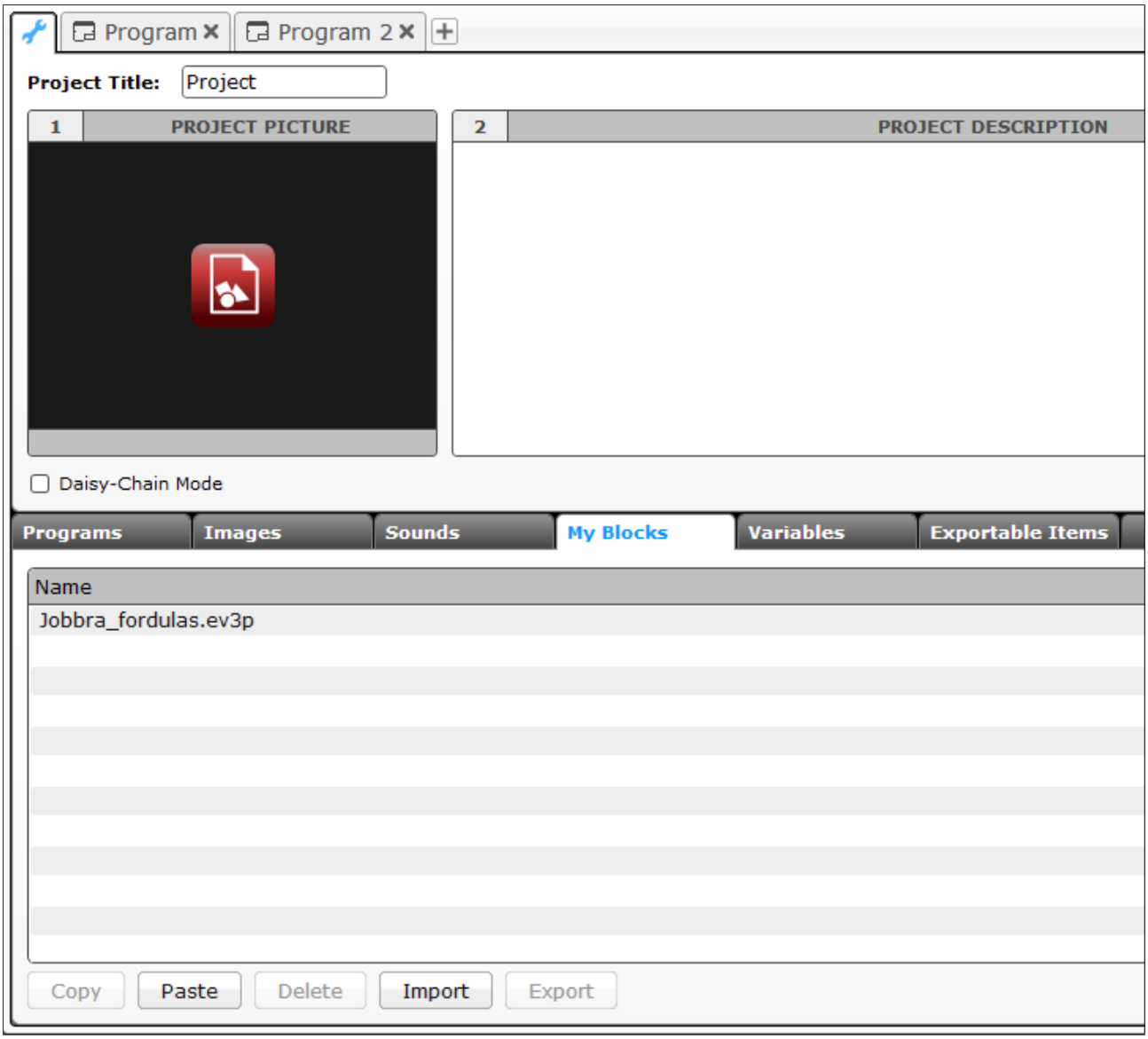
86. ábra. A saját blokkunk kész kódja.

Ha minden módosítás elkészült, akkor mentsük el az aktuális programot, aminek a blokkal azonos a neve és az adott projekten belül, de egy másik programban, már láthatjuk és használhatjuk is az új, saját készítésű blokkunkat. (Látni, már láthatjuk a kód szerkesztése közben is az új blokkot, azonban ott még ne használjuk!) A 87. ábrán látható egy példa a blokk alkalmazására. Az itt szereplő kód bal felső sarkában lévő fogaskerékre kattintva érjük el a blokk szerkesztő felületét, ahol módosíthatjuk a blokk külső megjelenését ill. paramétereinek listáját. Ezen kívül, ha kétszer rákattintunk a blokkra, akkor a szerkesztő felület nélkül megnyithatjuk a blokk mögöt álló kódot.



87. ábra. Példa saját blokk használatára.

A bevezető részben volt szó arról, hogy nemcsak abban a projektben lehet felhasználni a blokkot, ahol elkészítjük, hanem máshol is. Ehhez a a projekt (ahol létrehoztuk az új blokkot) beállításait kell megnyitnunk, amit a Projekten belül, a különböző programjaink bal szélén a fogóra kattintva érünk el. (88. ábra) Ezen a felületen keressük meg a My Blocks menüt! Itt találjuk az összes, ebben a projektben használt saját készítésű blokkunkat. Az exportáláshoz kattintsunk a blokkra, amit máshol is használni szeretnénk, jelen esetben a Jobbra_fordulas.ev3p-t, majd kattintsunk az aktiválódó *Export* gombra. A felugró ablakban válasszuk ki, hogy hova szeretnénk menteni, majd nyomjunk rá a mentésre. Ha egy új projektben szeretnénk felhasználni a blokkot, akkor ugyanezen a felületen tudjuk beimportálni azt. Ebben az esetben az Import gombra kell kattintani, majd a felugró ablakban meg kell keresni a mappát, ahol megtalálható a gépünkön a fájl és a megnyitással be is tudjuk rakni az új projektünkbe. Ha mégsincs szükségünk valamely saját készítésű blokkunkra, akkor a blokk kiválasztása után a Delete gombra kattintva törölhetjük ki.



88. ábra. Saját blokk exportálásának és importálásának felülete.

3.9.2. Feladatok

1. Készíts egy olyan blokkot, amivel jobbra 90°-ot lehet fordulni!
2. Készíts egy olyan blokkot, amivel bármelyik irányba fordulhatunk 90°-ot!
3. Készíts egy olyan blokkot, amivel a gyro szenzort kalibrálod!
4. Készíts egy olyan blokkot, amivel a logikai értékeket átalakítod szöveggé! Oldd meg, hogy hosszú (Igaz/Hamis) és rövid (I/H) formában is felhasználható legyen a szöveg!

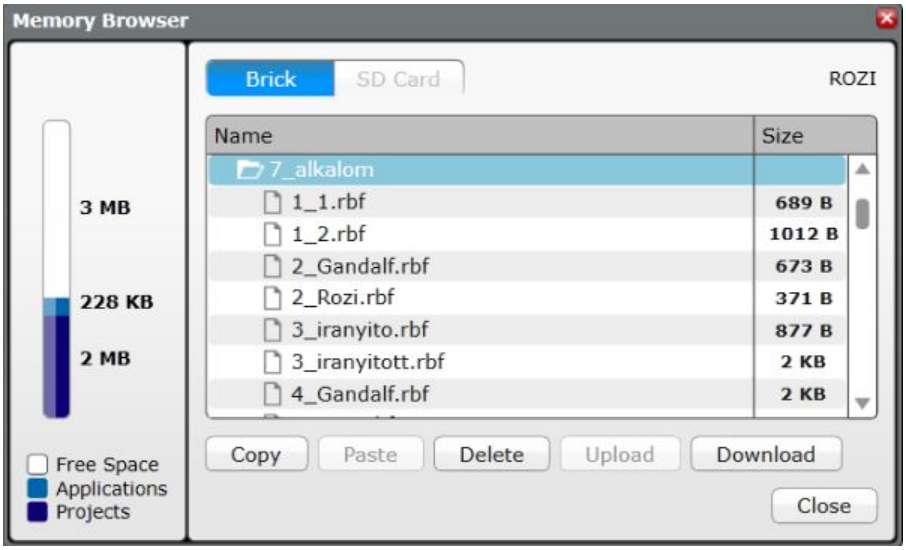
Ezekkel a feladatokkal megnéztük hogyan lehet bemeneti értéket és kimeneti értéket tartalmazó blokkokat készíteni. Készíthetünk a szenzorok értéket lekérő és valamilyen formában megjelenítő blokkot, alakzatok lerajzolását tartalmazó blokkot, de akár hangok variációit lejátszó saját blokkokat is. Mivel, tényleg bármi szerepelhet egy saját blokkban, így a további feladatok kitalálása Rád hárul kedves Kolléga. Milyen általad használt kódot szeretnél leegyszerűsíteni? Mi az a kódrészlet, amit gyakran használsz?

4. Az EV3 belső memóriájának kezelése

EV3 fájlstruktúra

Bár ehhez a részhez érve valószínűleg számos projektet elkészítettél már, mégis fontosnak találtuk, hogy részletesen ismertessük az EV3 fájlstruktúráját. Ez elengedhetetlen ahhoz, hogy diákjainknak precízen taníthassuk a projektek felépítésével kapcsolatos ismereteket. A második tananyag tartalma ezt méginkább indokolttá teszi, hiszen itt foglalkozunk először saját képekkel, hangfájlokkal és az üzenetküldéshez kapcsolódó szöveges dokumentumokkal. Nézzük meg hogy épülnek fel projektjeink, mit is tartalmaznak ezek a fájlok!

A programokat projektekbe vagy applikációkba szervezhetjük. Eddig főleg a projekt opciót használtuk. Az applikáció valójában egy kitüntetett projekt, mely megjelenik a robot "applications" menüpontjában a gyári programok mellett (balról a harmadik fül a robot képernyőjén). Programunkat akkor érdemes projekt helyett applikációba menteni, ha gyakran szeretnénk azt futtatni robotunkon. Ilyen fontos segédprogram lehet például egy gyro szenzor vagy hangérzékelő kalibráló. Utóbbit lefuttathatjuk minden alkalommal mielőtt hangérzékeléssel kapcsolatos programot használnánk, így bemérve az alap zajszintet. Ennek hatására robotunk ezt tekinti az alapzajnak és csak az ennél nagyobb értékeket méri, azokat viszont nagyobb pontossággal. Applikáció létrehozásához először, ahogy eddig is, projektet kell létrehoznunk. Amikor készen vagyunk az összes programmal, a Tools menüből válasszuk a "Download as App" opciót. Ennek hatására programjaink a roboton lévő applikáció menübe kerülnek.



89. ábra. Fájlstruktúra

A szoftverben létrehozott projektek/applikációk egységet alkotnak. Felfoghatjuk őket egyetlen fájl helyett (fájlkezelőben így látjuk) egy egész mappaként. Ezek tartalmazhatnak programokat (.ev3p), hangfájlokat (.rsf), képeket, grafikákat (.rgf), szöveges fájlokat (.rtf). A szoftver memória böngészője mappaként jeleníti meg a projekteket, példaként nézzük meg a 89. ábrát. Itt a projekt neve 7_alkalom. Ezen belül számos programot találunk. Jogosan merül fel benned a kérdés, kedves Olvasó, hogyha ezek programok, akkor miért nem .ev3p kiterjesztésűek. A számítógépen a programoknak valóban ez

a kiterjesztése, de a roboton, az általa futtatható állomány már .rbf kiterjeszt kap. Ettől függetlenül ezek ugyanazok a programok, melyeket a szoftverben létrehoztunk!
A projektek a számítógépen .ev3 fájlként vannak tárolva, mely a korábban említett elemeket tartalmazhatja. Fontos a feltételes mód, mivel nem feltétlenül tartalmazza azokat! Például egy robotunk által generált szöveges fájl csak a robot memóriájába kerül alapértelmezetten, nem lesz benne a projektben, ami számítógépünkön mentésre kerül.

Adattárolás az okostéglán

Minden tégla alapértelmezetten 16 MB memóriával rendelkezik. Azonban itt nem csak az általunk írt programok kapnak helyet, hanem a példaprogramok, beépített grafikák és hangok is. Így valójában csak 5 MB szabad hely áll rendelkezésünkre gyári állapotban. A 90. ábrán látható memory browser segítségével a beépített példaprogramokat törölhetjük, így felszabadíthatunk némi helyet a memóriában.
Nem feledkezhetünk meg azonban az időközönként esedékes frissítésről! Az EV3 szoftverének frissítése mindig visszaállítja a belső memória gyári állapotát. Így elvesznek robotra mentett projektjeink, a példaprogramok pedig, melyektől nagy nehezen megszabadultunk visszakerülnek a téglára.

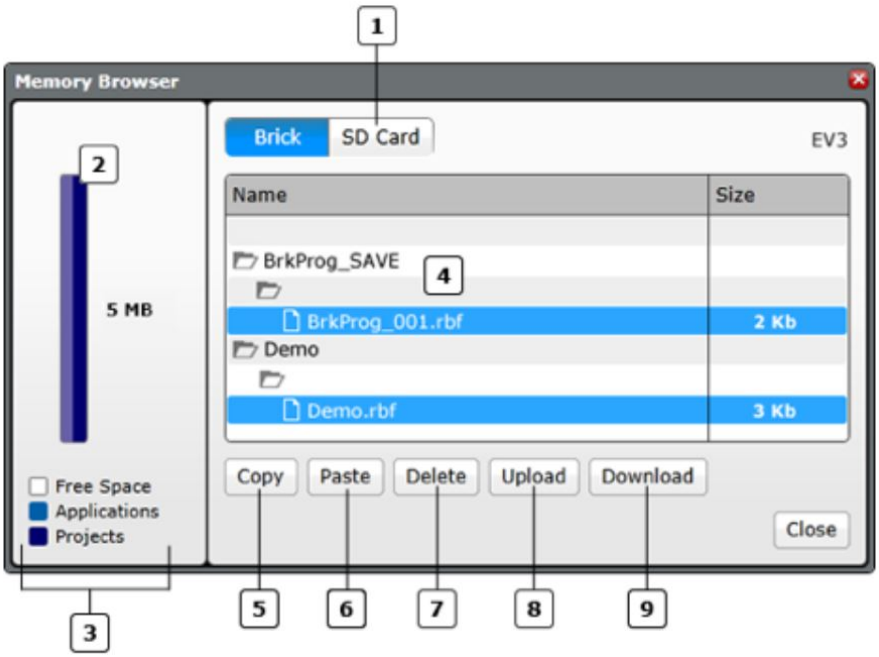
Ha túl kevésnek bizonyul az 5 MB, robotunk memóriáját egy SD-kártyával bővíthetjük. A tégla legfeljebb 32 GB-os kártyát támogat! Ezen ugyanúgy tárolhatjuk projektjeinket, hozzá tartozó fájljainkat. Előnye a sok szabad helyen kívül, hogy ennek tartalma nem veszik el frissítéskor.

Memóriakezelés

A memóriaböngésző átfogó képet ad a robot memóriahasználatáról. A 90. ábrán látható ablakot a Tools menüben, a Memory Browser opcióra kattintva nyithatjuk meg. A bal oldali oszlopban látjuk a robot belső memóriájának állapotát. A fehér rész a fennmaradó szabad helyet mutatja, a világoskék az applikációk által foglalt hely mennyiségét, míg a sötétékék azt, hogy mennyi helyet foglalnak a robot belső memóriájából a projektek. A 89. ábrán jól látszik a három kategóra.

A Memory Browserben a következő műveleteket tudjuk elvégezni:

- másolhatunk projektet vagy egyéb forrásfájlokat a számítógép és a robot között mindkét irányba
- törölhetünk projekteket, programokat vagy fájlokat
- másolhatunk fájlokat projektek között
- mozgathatunk projektet az SD-kártya és belső memória között



90. ábra. Memóriaböngésző felülete

A Memory Browser felületének magyarázata (90. ábra[12]):

1. Eszközválasztó fül: itt válaszhatjuk ki, hogy a robot belső memóriáját, vagy az SD kártyát szeretnénk kezelni
2. Memória grafikon: a tárhely telítettségének grafikus reprezentációja
3. Fájl fajták: Projects és Applications a két helyet foglaló kategória
4. A tárolt fájlok listája
5. Másolás funkció
6. Beillesztés funkció
7. Kijelölt elem törlése
8. Feltöltés: A kijelölt elem feltöltése a számítógépre
9. Letöltés: Ennek a funkciónak használatával a robotra letölthető a számítógépről kiválasztható fájl

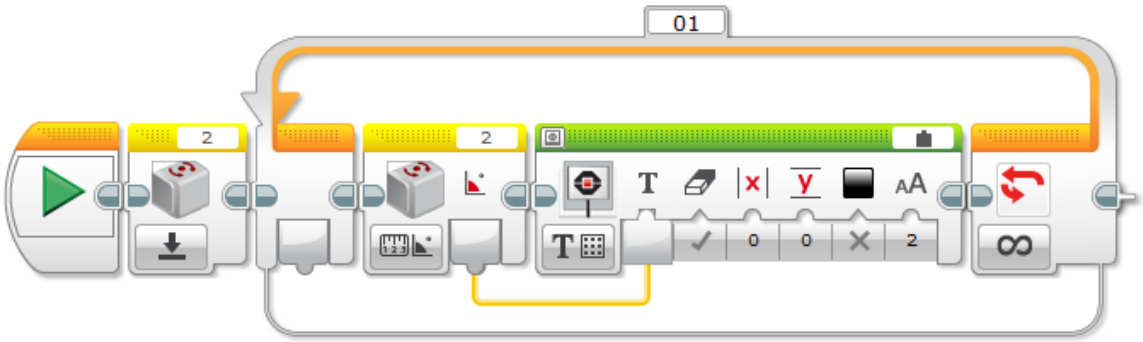
[12] https://ev3-help-online.api.education.lego.com/Retail/en-us/page.html?Path=editor%2FMemoryBrowser.html

5. Feladatok megoldásai

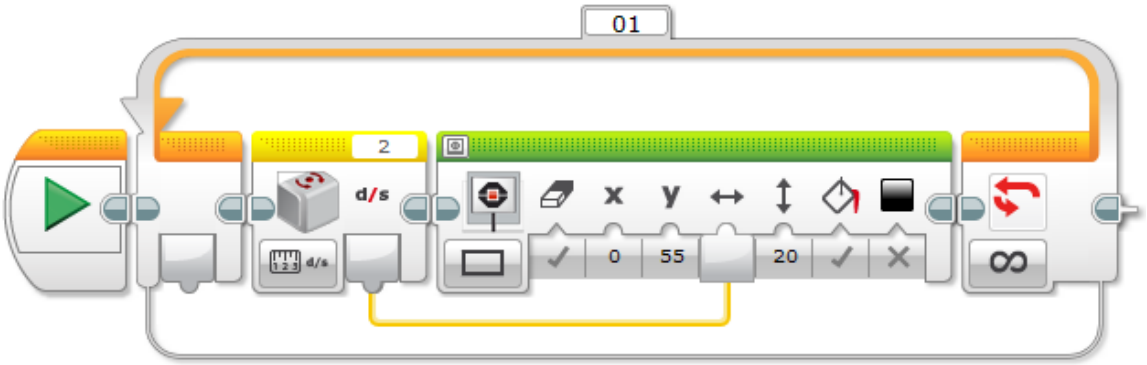
5.1. 1.alkalom

Gyro szenzor használata

1. *Írd ki a kijelzőre az aktuális szögelfordulást! Ügyelj arra, hogy mindig az aktuális értéket jelenítsd meg!*



2. *Jeleníts meg a kijelző közepén a robot sebességével azonos szélességű téglalapot! (A téglalapot színezd ki, hogy jobban látszódjon az eredmény! A teszteléshez kézzel mozgasd a robotot vagy állíts be valamilyen mozgást a képernyőre rajzolás mellé!)*



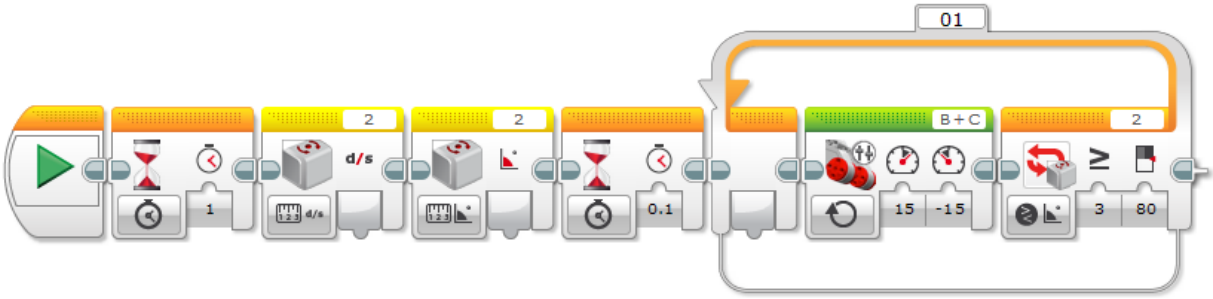
3. *Készíts programot, melyben a robot egy tengelyfordulás után fordul 90°-ot, majd ismét megy előre egy tengelyfordulást! A fordulást a várakozás blokk (gyro szenzor módjának) segítségével hajtsd végre!*

4. *Hogyan lehetne a gyro szenzor más blokkját használva 90°-ot fordulni?*

   Más megoldások is elfogadhatóak, azzal a feltétellel, hogy a gyro szenzor a forgást tényleg 90°-nak érzékeli.

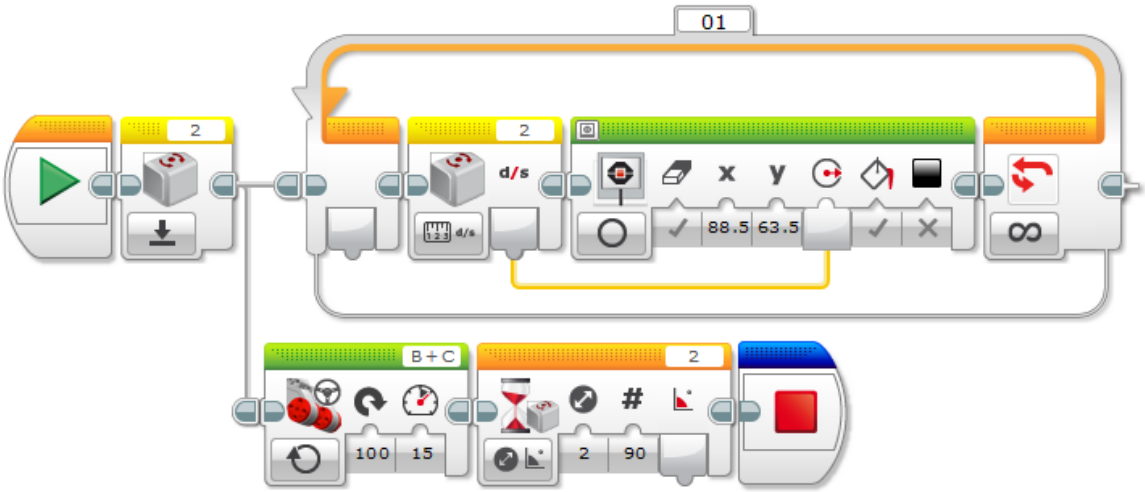
   A megoldásban nem elírás a 80°, ugyanis az általam próbált eszközön ezzel a beállítással érzékel

a szenzor 90°-ot. Ennek az az oka, hogy bizonyos idő eltelik a küszöb átlépése és a motor leállása között, amíg bizony továbbmegy a robot és picit túlfordul. Azért, hogy ezzel a túlfordulással együtt meglegyen a derékszög és ne lépjük túl, hamarabb le kell állítani a forgást. Az, hogy ez hány fokot jelent, eszközönként eltérő lehet, azonban érdemes a sebességet és a fokot is módosítani a próbálkozások során.



Egy másik, ehhez hasonló megoldásként, lehet azt is alkalmazni, hogy kb. 75°-ig 15-ös sebességgel forgunk, majd a maradék forgást (15°) lassabban hajtjuk végre. Ezzel kisebb mértékű hibát érhetünk el, hiszen lassabban forog a motor és emiatt nem fordul annyira túl, mint egy nagyobb sebességnél.

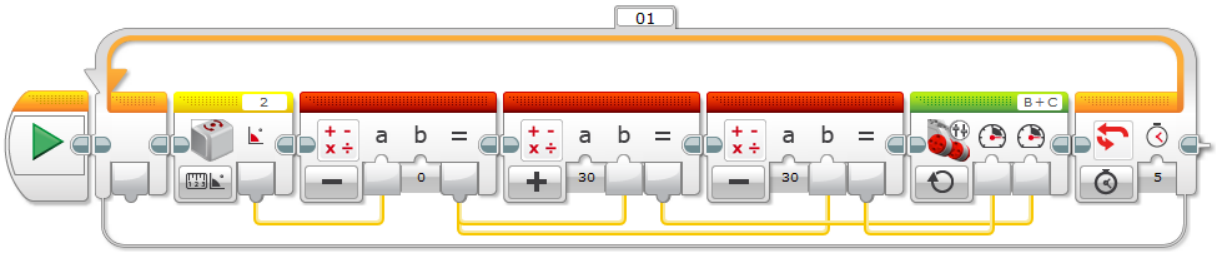
5. *A következő programot futtatva a robot 90°-ot fordul jobbra, miközben a kijelzőre kirajzol a szögsebességének megfelelő sugarú kört. Próbáld ki! Mi lehet az oka annak, hogy nem a korábban leírtakat hajtja végre a robot?*



A program futtatásakor a kijelzőn megjelenik a kör, azonban nem áll meg 90°-nál a robot, hanem tovább forog. Ez amiatt van, hogy az eszköznek párhuzamosan kétféle módban kellene használnia a gyro szenzort, amit próbál teljesíteni, de nem sikerül neki teljes mértékben. (Az, hogy próbálkozik, a szoftver Port View részében látható, ugyanis hol a szögsebességet, hol az elfordulás szögét jeleníti meg.) Tehát vizsgálja a fordulását mértékét, azonban az eredményt nem tudja kivárni, mert a sebességet kell megnéznie, ezért folyamatosan forog.

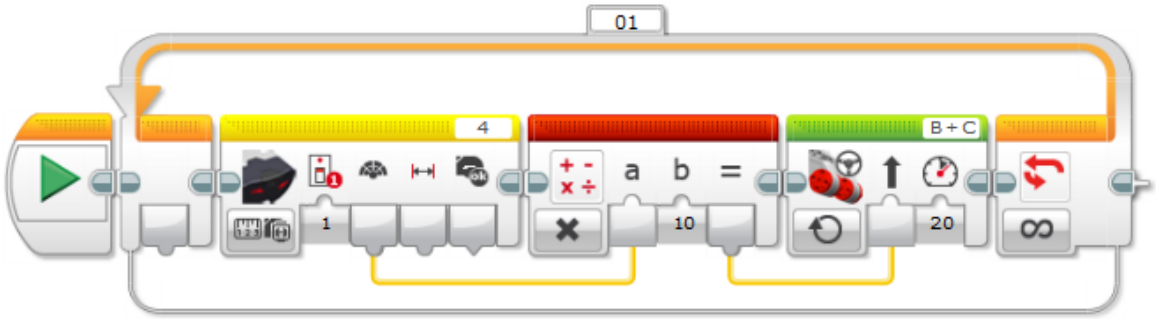
6. *Készíts programot, amelyben a robot pontosabb egyenes vonalú mozgást végez! Ügyelj arra, hogy a program során legfeljebb 1 értékkel térjen el a 0-tól a szögelfordulás mértéke!* [13]

   A feladatnak számos megoldása létezik, én azért a lent láthatót választottam, mert egyrészt, érdekes megoldás, másrészt, nemcsak egyenes irányú haladást tudunk vele elérni, így több esetben is használható. A kódban az első számú matematikai blokkban lévő *b* érték jelöli a kívánt irányú haladásunkat, tehát ha egyenesen szeretnénk menni, akkor a *b* nulla lesz. Azonban, ha a jelenlegi helyzetünkhöz képest 60°-ot kell fordulnunk jobbra és utána kell egyenesen mennünk, akkor a *b*-t 60-ra kell állítani. A következő két matematikai blokkban az *a* értékek a motor kívánt sebességét jelölik. Ezeket azonos értékűekre kell beállítani és majd a szenzor értékétől és a kívánt fordulási szögtől függően fognak változni. A ciklus kilépési feltételét 5 másodpercre állítottam,



Az infravörös szenzorhoz tartozó jeladó használata

1. *Készíts programot, amelyben az eszköz mozgása során, a jeladót követi! (Tipp: A megoldásban használd fel a jeladó irányát. Tipp 2: Használj valamilyen matematikai műveletet az irány felhasználásakor.)*

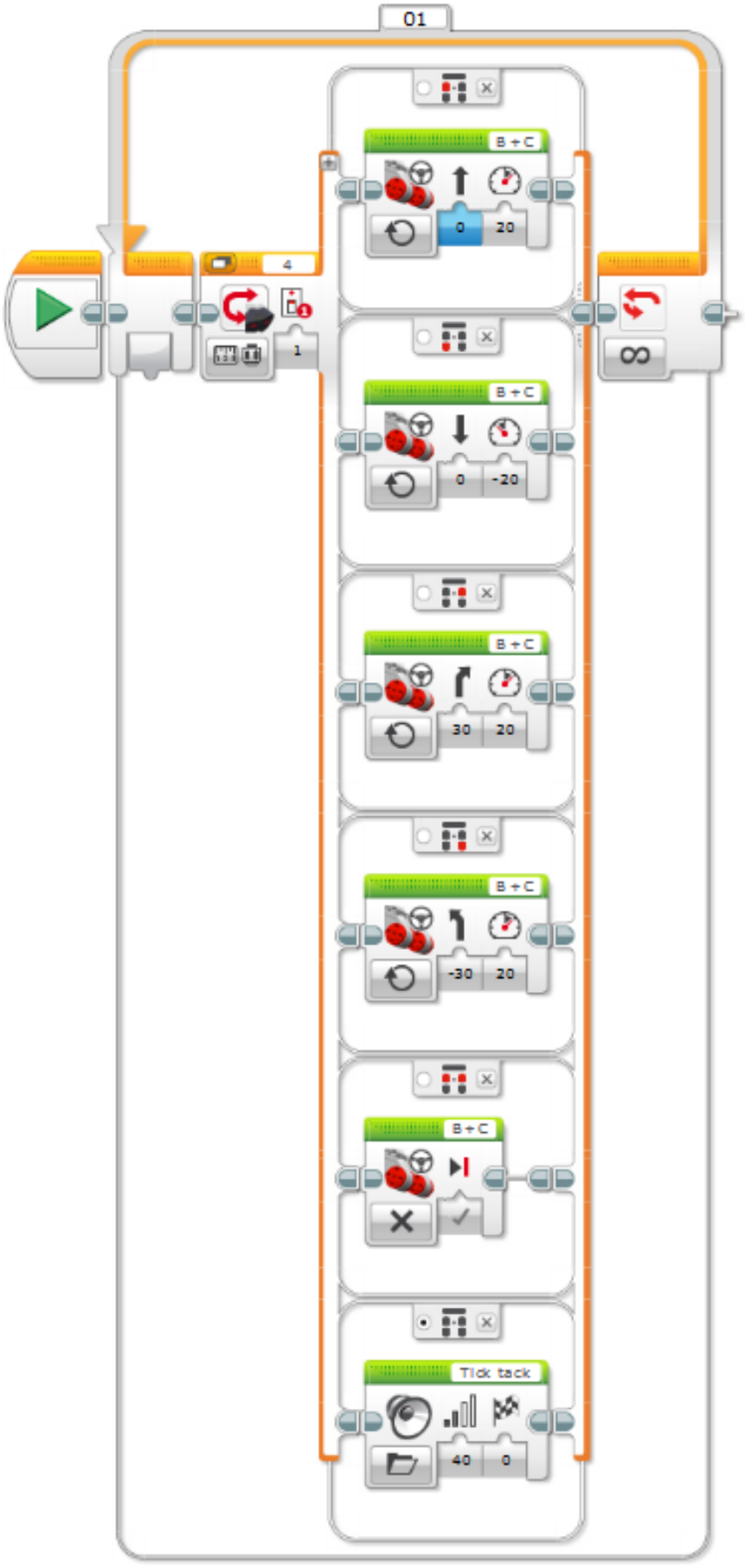


2. *Készíts programot, amelyben a távirányítót használva mozgatod a robotot! Állítsd be, hogy ha nincs lenyomva gomb, akkor a Tick tack hangot adja ki, valamint ha az egyes és hármas gombokat egyszerre lenyomjuk, akkor megálljon a motor. (Pl.: gomb 1 előre, gomb 2 hátra, stb.)*

   Ezzel a feladattal nagyon jól szemléltethető, hogy mennyi idő eltelik az adatok küldése és feldolgozása között. Ugyanis hiába nyomunk meg egy gombot a távirányítón, az nem hajtódik végre azonnal, néhány másodpercet várni kell. Ezen kívül azt is bemutatja a feladat, hogy nem minden esetben érzékeli a távirányító a gombnyomást. Ugyanis, nekem többször is előfordult, hogy megnyomtam valamelyik gombot, azonban nem hajtódott végre az annak megfelelő parancs, csupán a Tik tak hang játszódott le, ami meg azt jelenti, hogy nem érzékel gombnyomást. Tehát min-

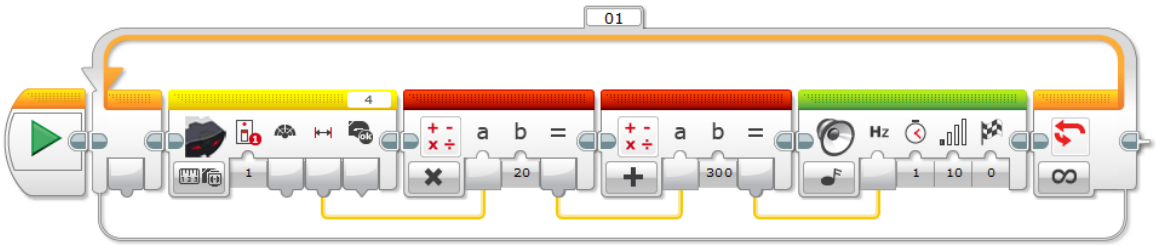
[13] A megoldás Builderdude35 YouTube felhasználó *Gyro Following - More Accurately Control your FLL Robot* c. videója alapján készült.

denképp érdemes kicsit játszani a távirányítású robotunkkal és kipróbálni, hogy melyik esetben működik és melyikben nem a távirányító.



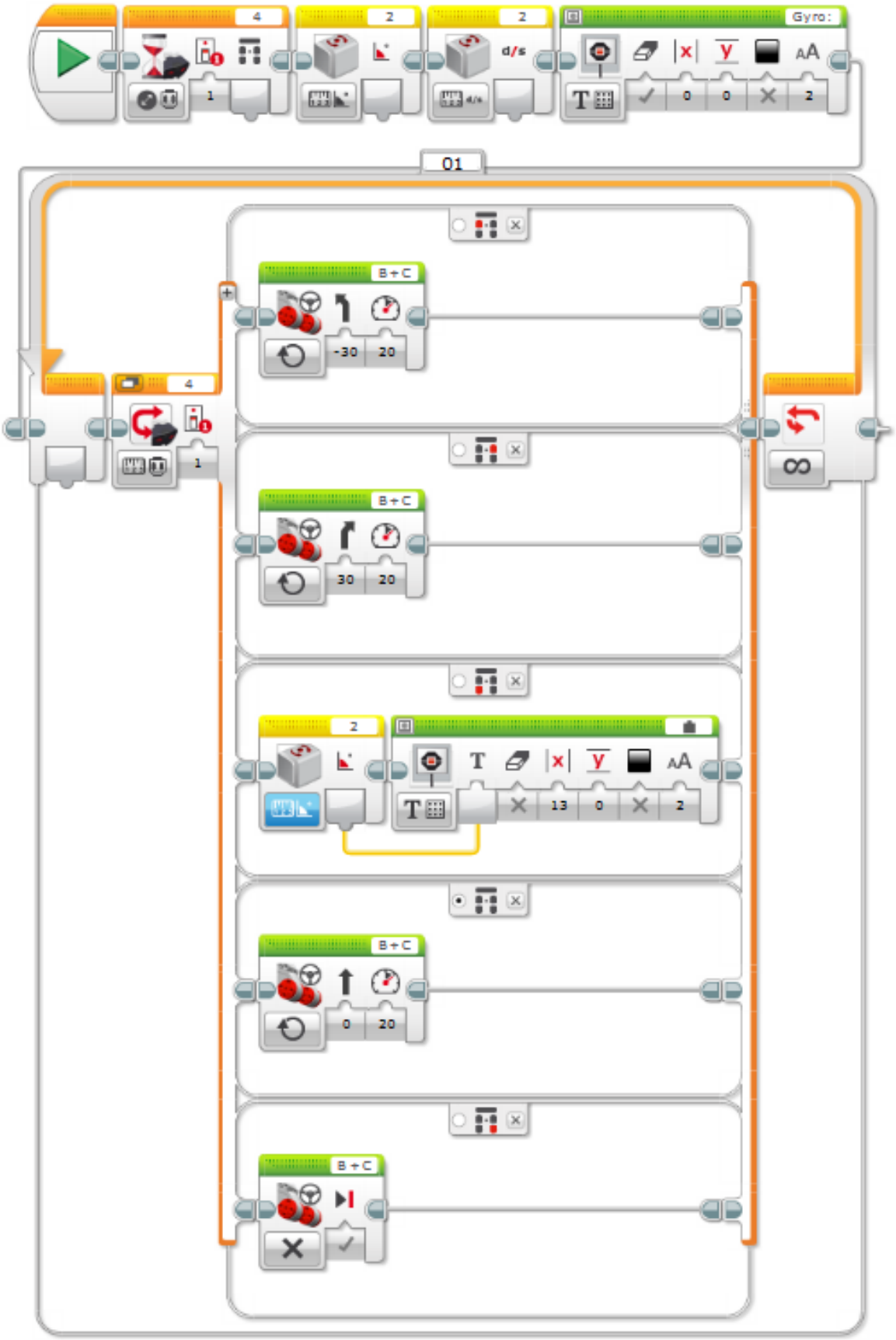
3. *Készíts egy olyan programot, amelyben a jeladó távolságát az érzékelőtől hanggal jelzed! Minél messzebb van a jeladó, annál magasabb legyen a hang.*

   A feladat megoldásában bármilyen matematikai művelettel módosított érték megfelelő, ha a lejátszás során a távolságnak megfelelően változik a hang magassága.



4. *Készíts egy olyan programot, amiben a távirányítóval irányítod a robotot, hogy melyik irányba forduljon el ill. mikor álljon meg! Egy ezektől különböző gomb segítségével írasd ki a kijelzőre a gyro szenzor elfordulási szögét! A program, a távirányító bármely gombjának megnyomásakor induljon el!*

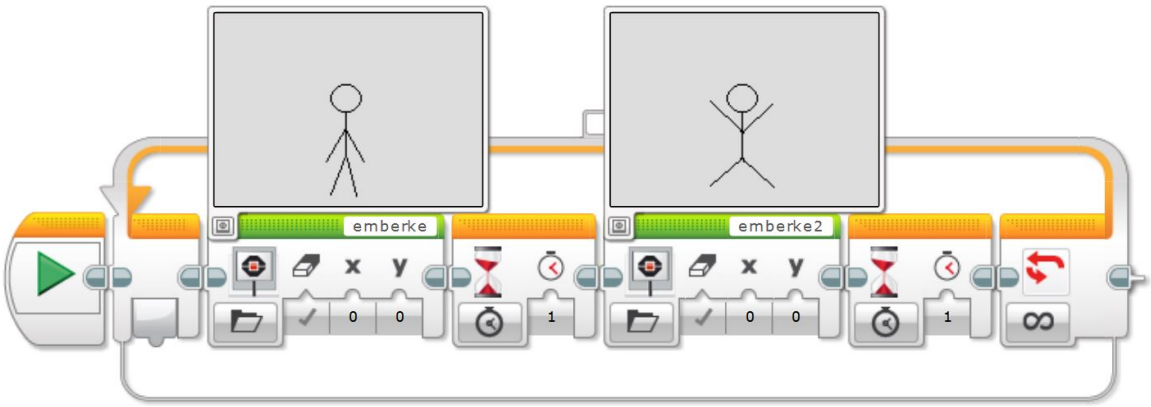
   A megoldás a következő oldalon látható.


Gyro:
01
B+C
-30
20
B+C
30
20
B+C
0
20
B+C

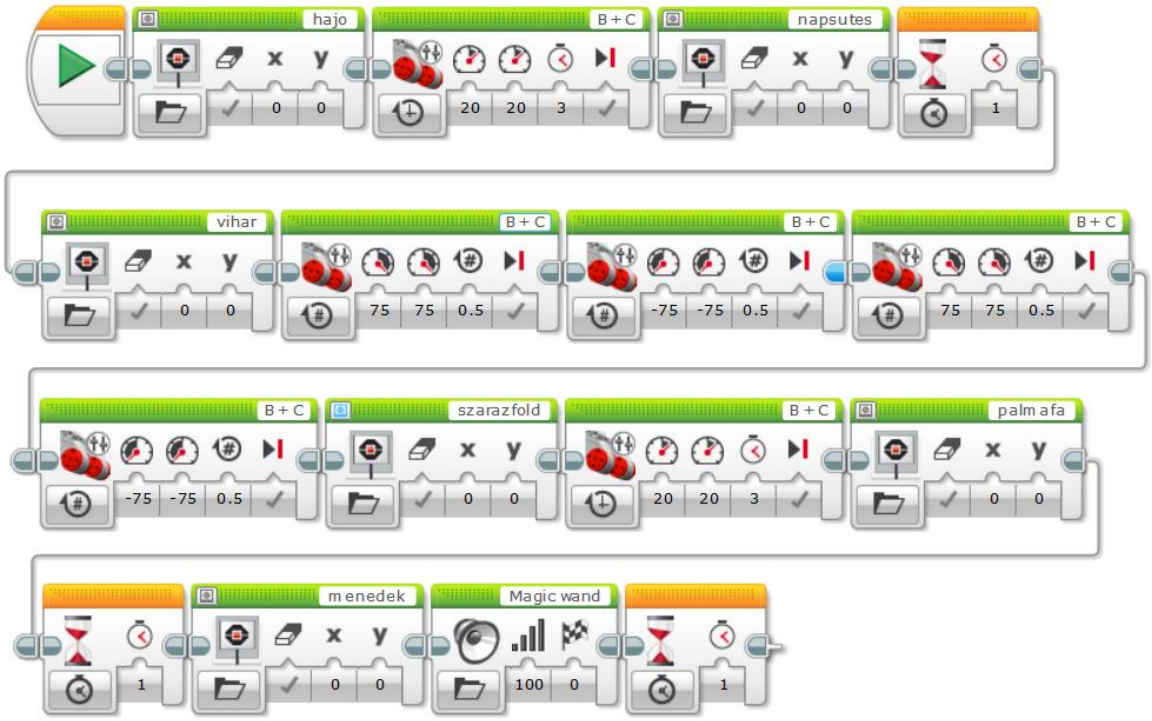
5.2. 2.alkalom

Kép rajzolása és megjelenítése

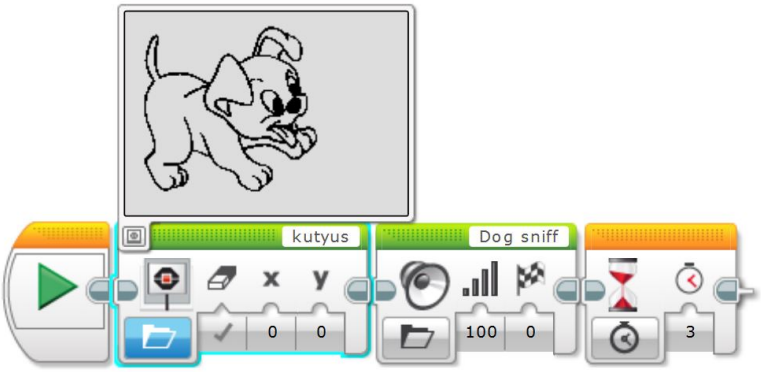
1. *Rajzolj az Image Editor segítségével egy vigyázzállásban álló pálcika emberkét. Munkádat mentsd emberke néven. Rajzolj egy ugyanekkora méretű pálcika emberkét, aki terpeszállásban áll, kezeit felemelve. Ezt a rajzot mentsd emberke2 néven. Alkoss programot, melyben a robot képernyőjén csillagugrást végez pálcika emberkéd. Használd elkészített rajzaid.*



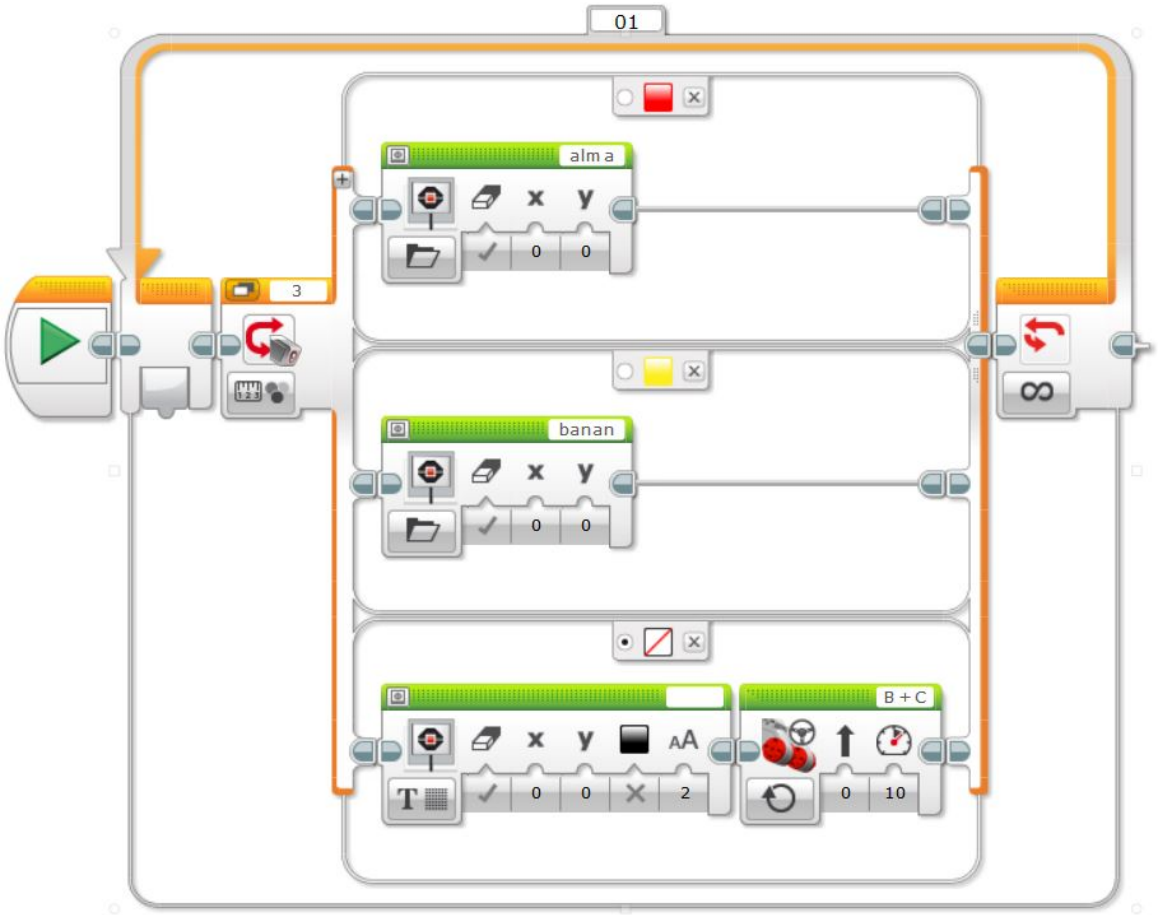
2. *Engedd szabadjára fantáziád! Rajzolj az Image Editorban több képet: háttér, szereplők és a programozás segítségével játssz el velük egy történetet, melyet a robot képernyőjén jeleníts meg! (A megoldás csak mintaként szolgál, példa a rajzok elnevezésére, a számos különböző blokk használatára.)*



3. *Tölts le egy képet az internetről. Szerkeszd az Image Editorban tetszés szerint és jelenítsd meg ezt robotod képernyőjén. Állíts be egy hozzá illő hangot is.*



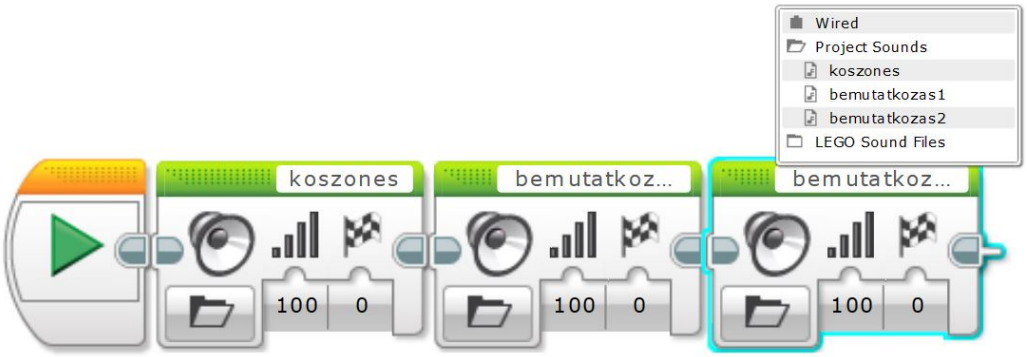
4. *Alkoss egy olyan robotot, ami színekről asszociál valamire (pl.: gyümölcs, állat), ami hasonló színű. Jeleníts meg egy képet arról a dologról, amire "robotod gondol", amikor meglátja az adott színt. Például citromsárga szín láttán banánra, míg piros színt észlelve almára gondol. A képet internetről is letöltheted, de akár saját kezűleg is megrajzolhatod a szoftverben. (Szükséges: pálya színes elemekkel, melyeken a robot végighalad.)*



5.3. 3.alkalom

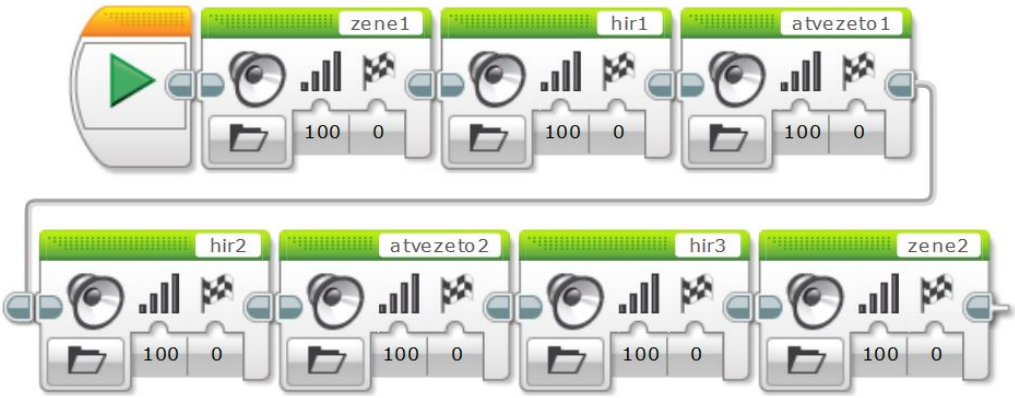
Saját hang készítése és lejátszása

1. *Készíts olyan programot, melyben robotod köszön és röviden bemutatkozik. Elmondja hogy hívják és miket szeret csinálni a szabadidejében. Engedd szabadjára fantáziád, találj ki minél érdekesebb, viccesebb "robot tulajdonságokat". A szöveget több különböző fájlba is mentheted, hiszen egyszerre csak 8 mp rögzítésére van lehetőség.*



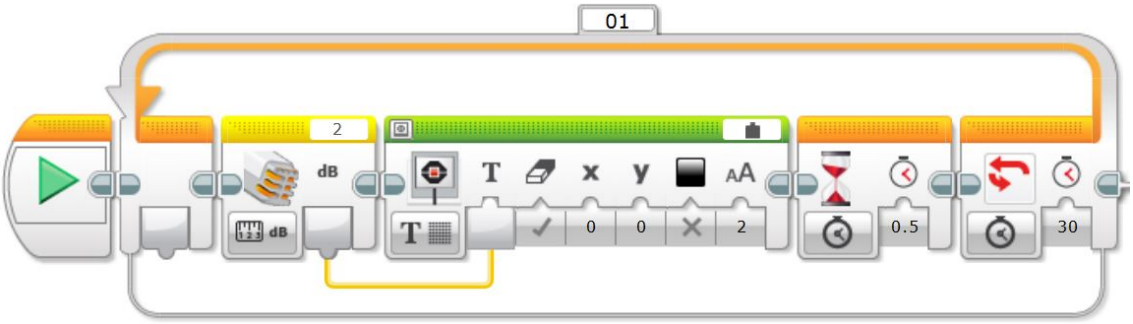
2. *Az előző szakköri alkalmak egyikén saját rajzaidból alkotott történethez készíts mesélő szöveget. Az egyes felvételeket a történet megfelelő pontjaihoz illeszd! A hangfelvételek alatt folyamatosan legyen látható az adott részhez tartozó kép, jelenet is. (A motormozgatást érdemes kivenni a programból, mivel olyan hangos, hogy mellette nem lehet jól érteni a felvett szöveget.)*

3. *Készíts osztályhíradót vagy mókás időjárás jelentést! Használj internetről letöltött hangokat és saját felvett szövegeket egyaránt! A teljes anyag hossza, melyet a robot megszólaltat legalább fél perces legyen! Munkádat akár a robot képernyőjén megjelenített képekkel is színesítheted.*

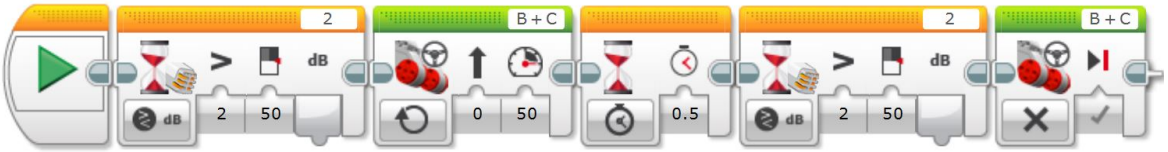


NXT hangérzékelő használata

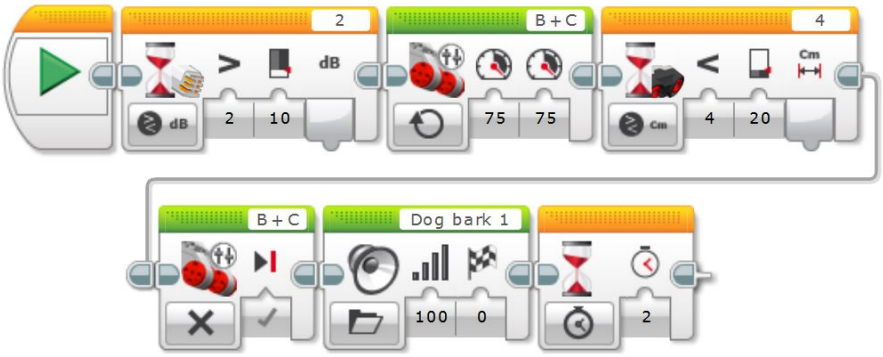
1. *Írasd ki robotod képernyőjére a mért hangerősséget. Az érték kövesse a zajszint változását 30 másodpercen keresztül. Gondoskodj róla, hogy például egy taps esetén az érték leolvasható legyen, ne tűnjön el túl gyorsan. Állapítsd meg mennyi a mért érték suttogás, normál beszéd, illetve taps esetén.*



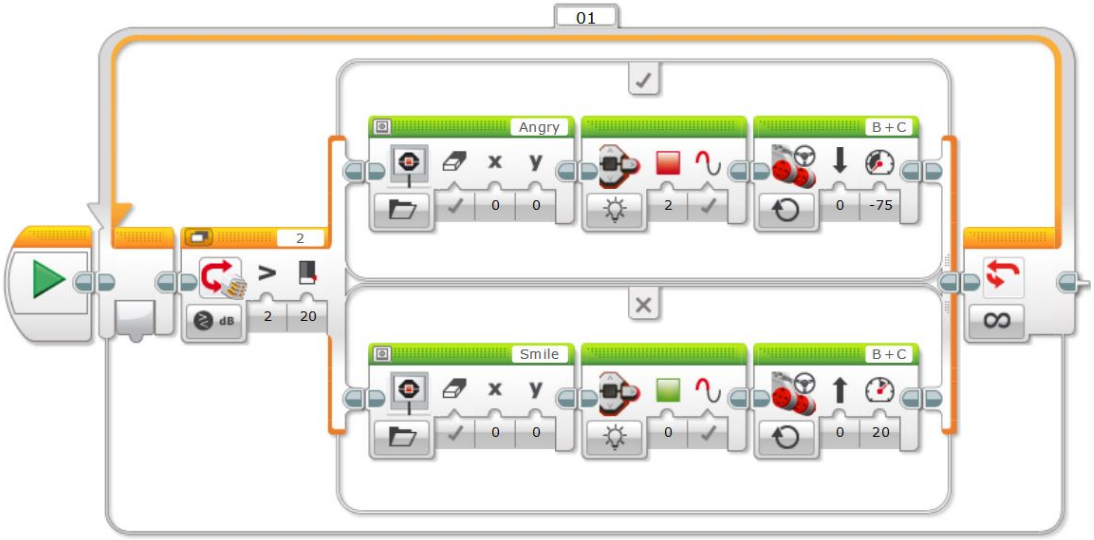
2. *Készíts programot, melynek hatására robotod tapsra elindul. Egyenesen halad, míg újabb tapsot nem hall.*



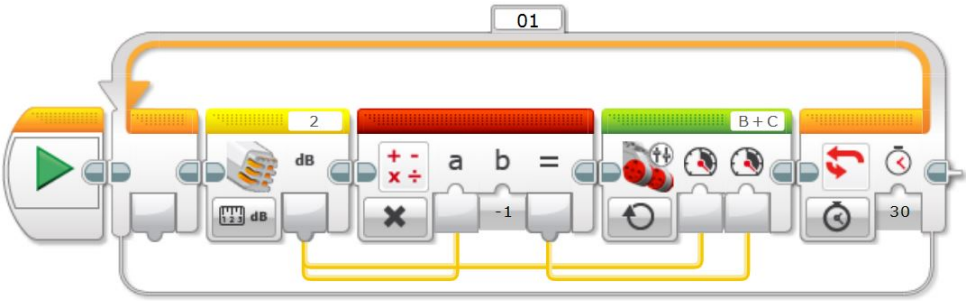
3. *Robotod most egy engedelmes kutyus szerepét öltse magára! Egy helyben vár, amíg nem mondod neki, hogy "Gyere ide!". Ekkor egyenesen elindul feléd, majd körülbelül 20 centiméterrel előtted megáll és ugat. Így örül a gazdájának. :) (Vigyázz! Kutyusod csak akkor engedelmeskedik pontosan parancsodnak, ha csend van körülötted!)*



4. *Alkoss programot, melynek hatására robotod pirosan villogva, dühös arccal gyorsan hátrálni kezd, ha túl nagy a zaj. Ha csendesebb a helyzet akkor zölden villogva, mosolyogva elindul lassan előre.*



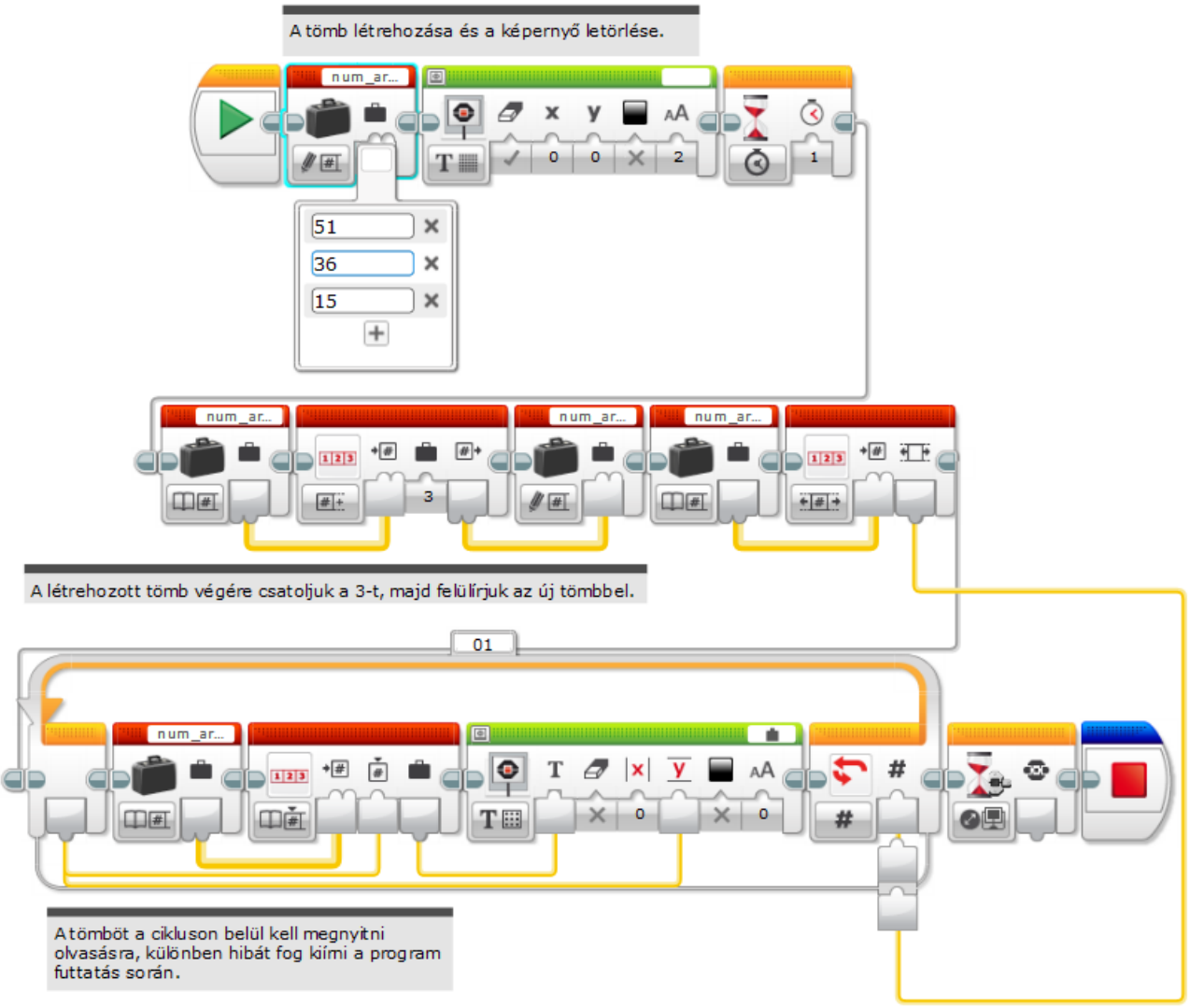
5. *Robotod forogjon körbe saját tengelye körül 30 másodpercig. Sebessége a mért hangerősségtől függjön. Nagy zaj esetén gyorsan, míg egyre halkuló hangok mellett egyre lassabban forogjon.*



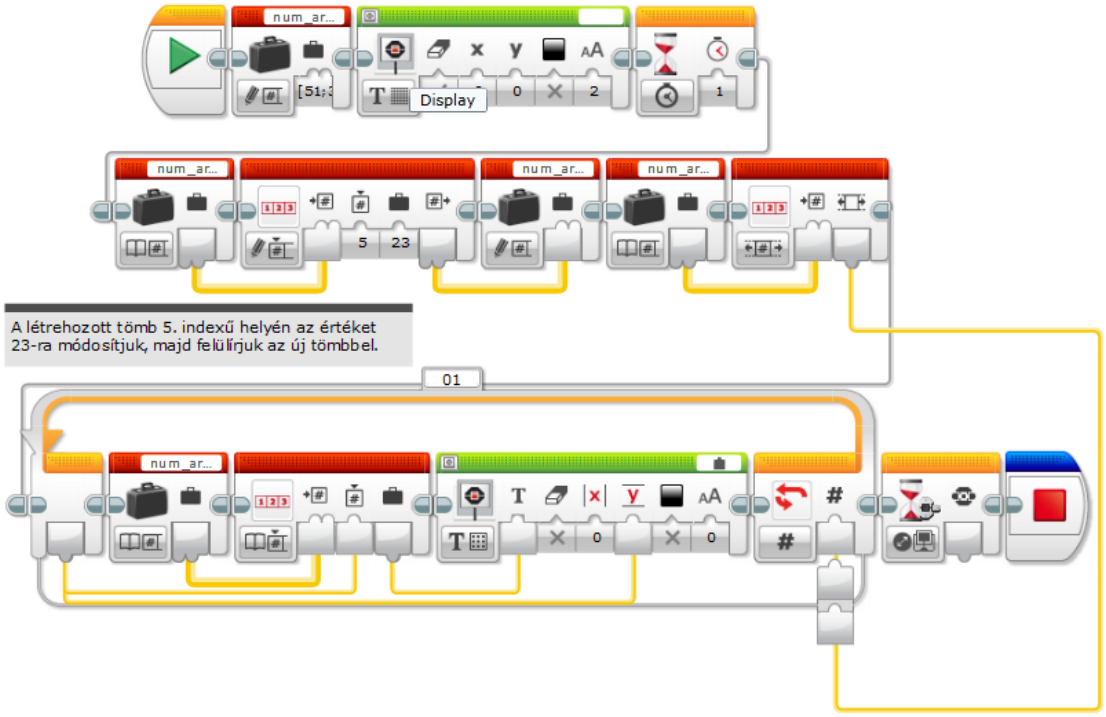
5.4. 4.alkalom

Tömbök és konstansok használata

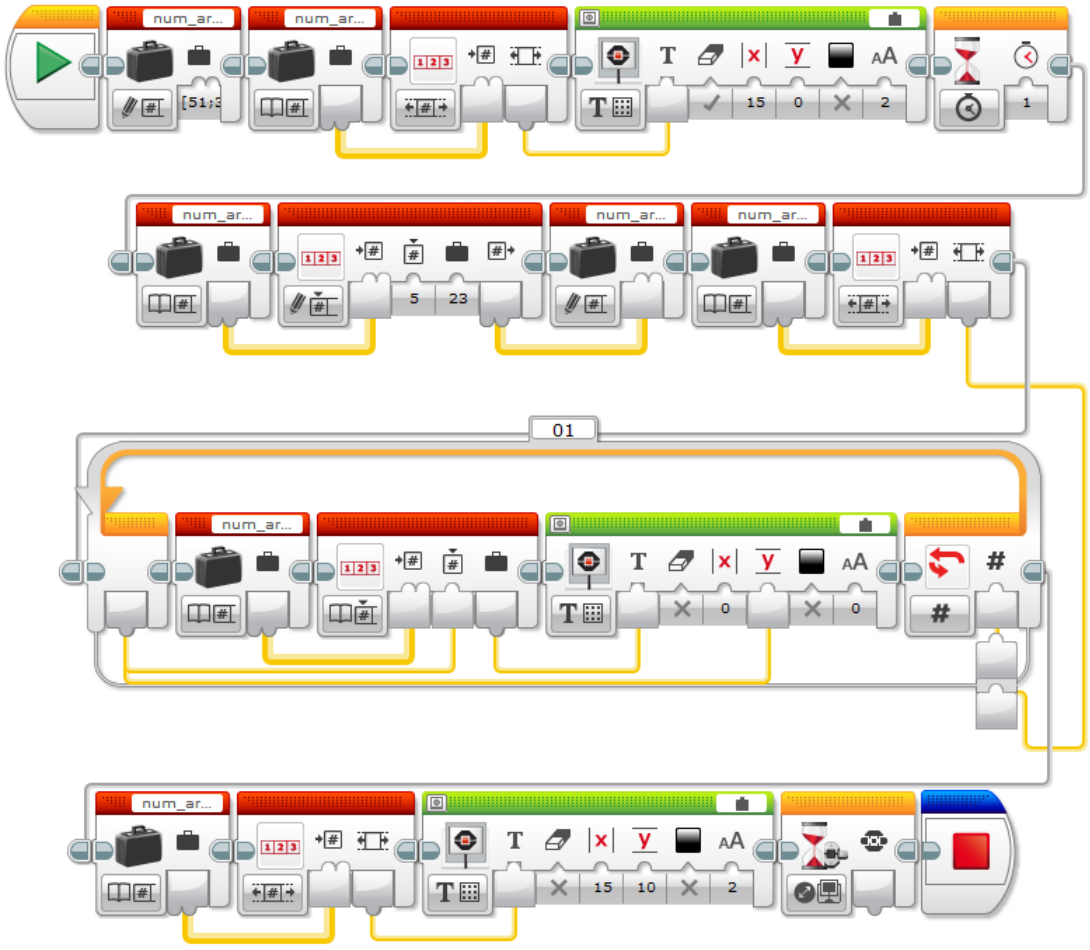
1. *Hozz létre egy tömböt, aminek az elemei a következők: 51, 36, 15! Fűzd a tömb végére a 3-as értéket! Ezt követően jelenítsd meg a kijelzőn az értékeket külön sorokban! Az értékek megjelenítése akkor érjen véget, ha megnyomjuk a tégla valamelyik gombját.*



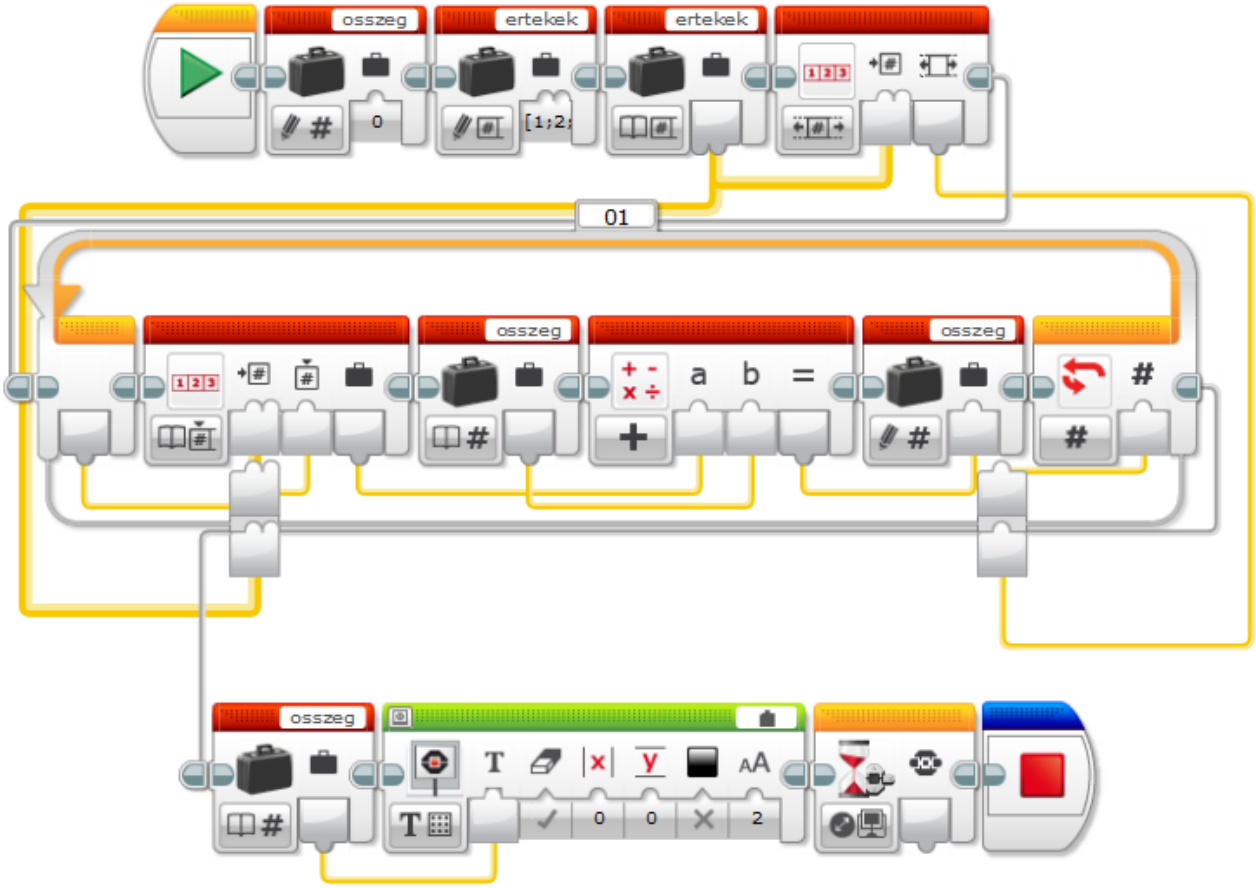
2. *Az előző feladatban létrehozott tömb 5. indexű helyén lévő értéket módosítsd 23-ra! Ezt követően jelenítsd meg az értékeket az előző feladatban leírt módon!*



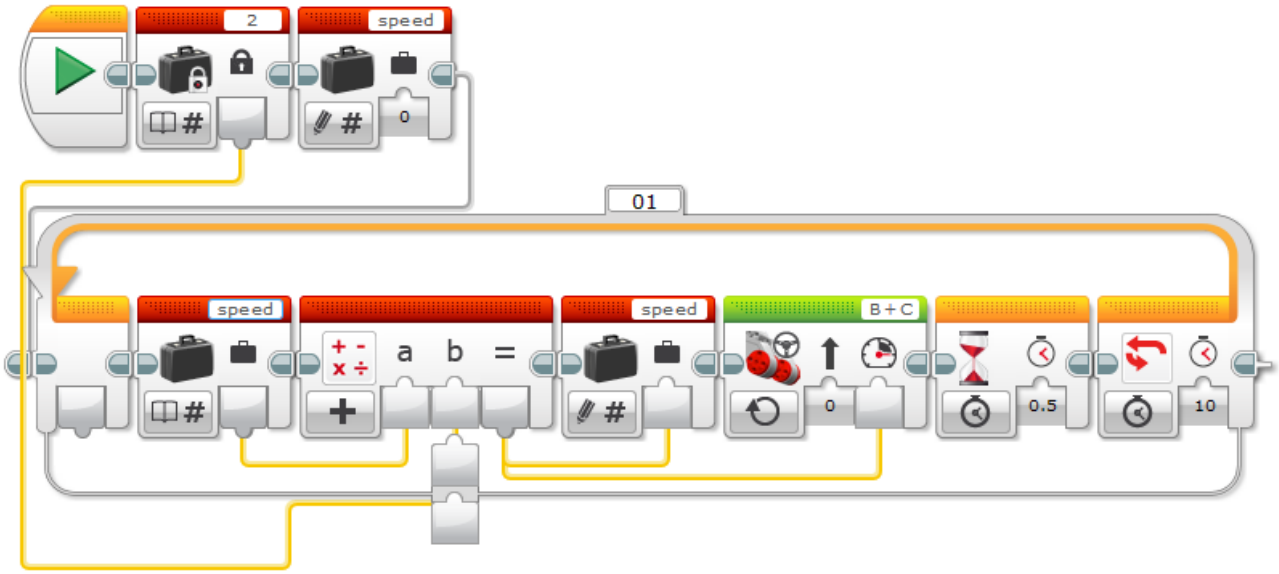
3. *Hány értéke van/volt az előző feladatban szereplő tömbnek a feladat végrehajtása előtt és után? A tömb értékein kívül, jelenítsd meg „mindkét” tömb méretét a kijelző jobb oldalán!*



4. *Készíts egy olyan programot, amiben egy általad megadott értékeket tartalmazó tömbnek, az összes elemét összeadod! A tömbben legalább 5 érték legyen és a végeredményt jelenítsd meg a kijelzőn.*

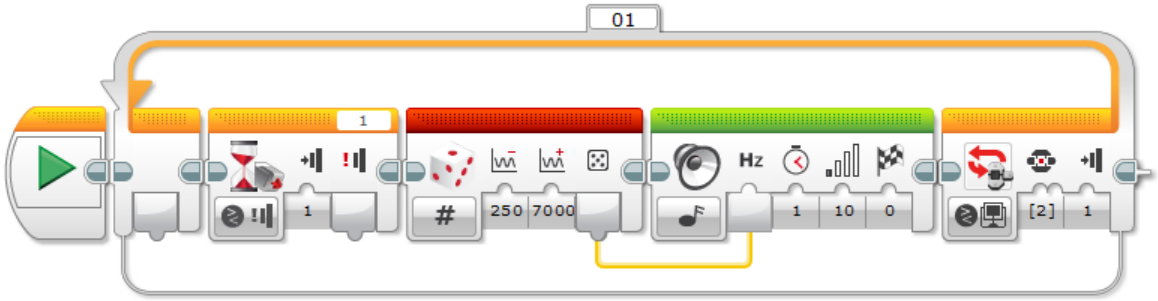


5. *Készíts egy programot, amelyben a robotod sebessége folyamatosan nő! Ehhez használd fel a 2-t, mint konstans értéket!*



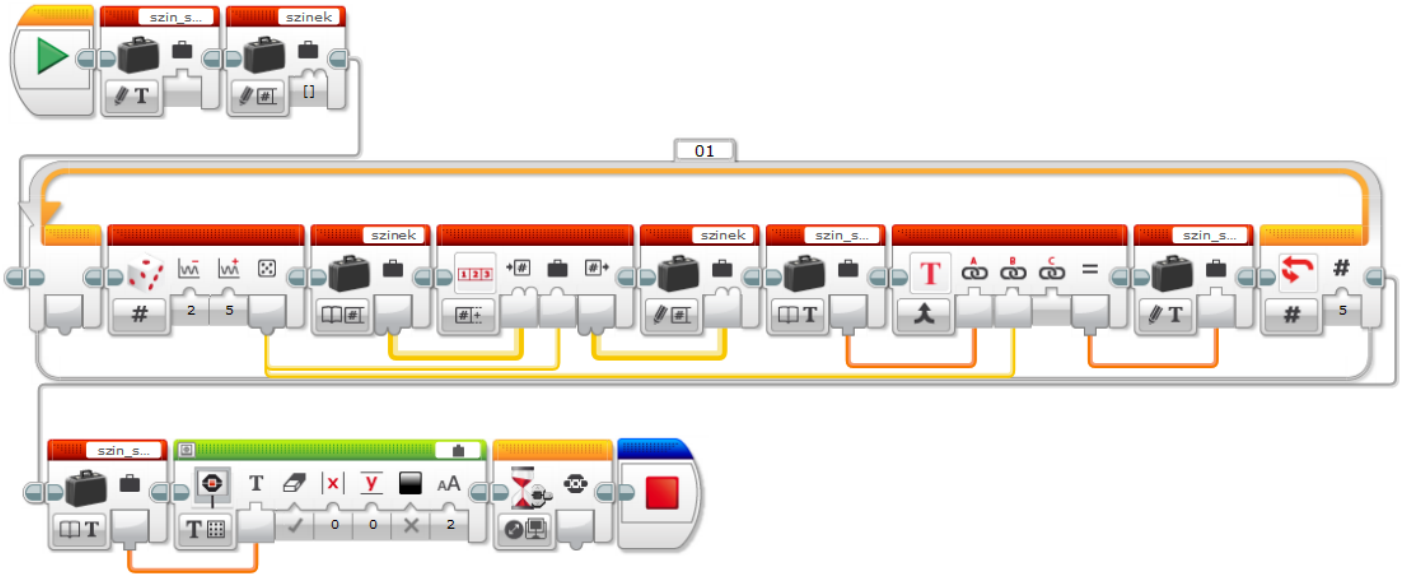
További adatmanipulációs elemek használata

1. *Készíts egy programot, amiben a nyomásérzékelő megnyomására 250 és 7000 Hz közötti hangot játszik le a robot! Ezt addig folytassa, amíg a középső gombot meg nem nyomjuk!*



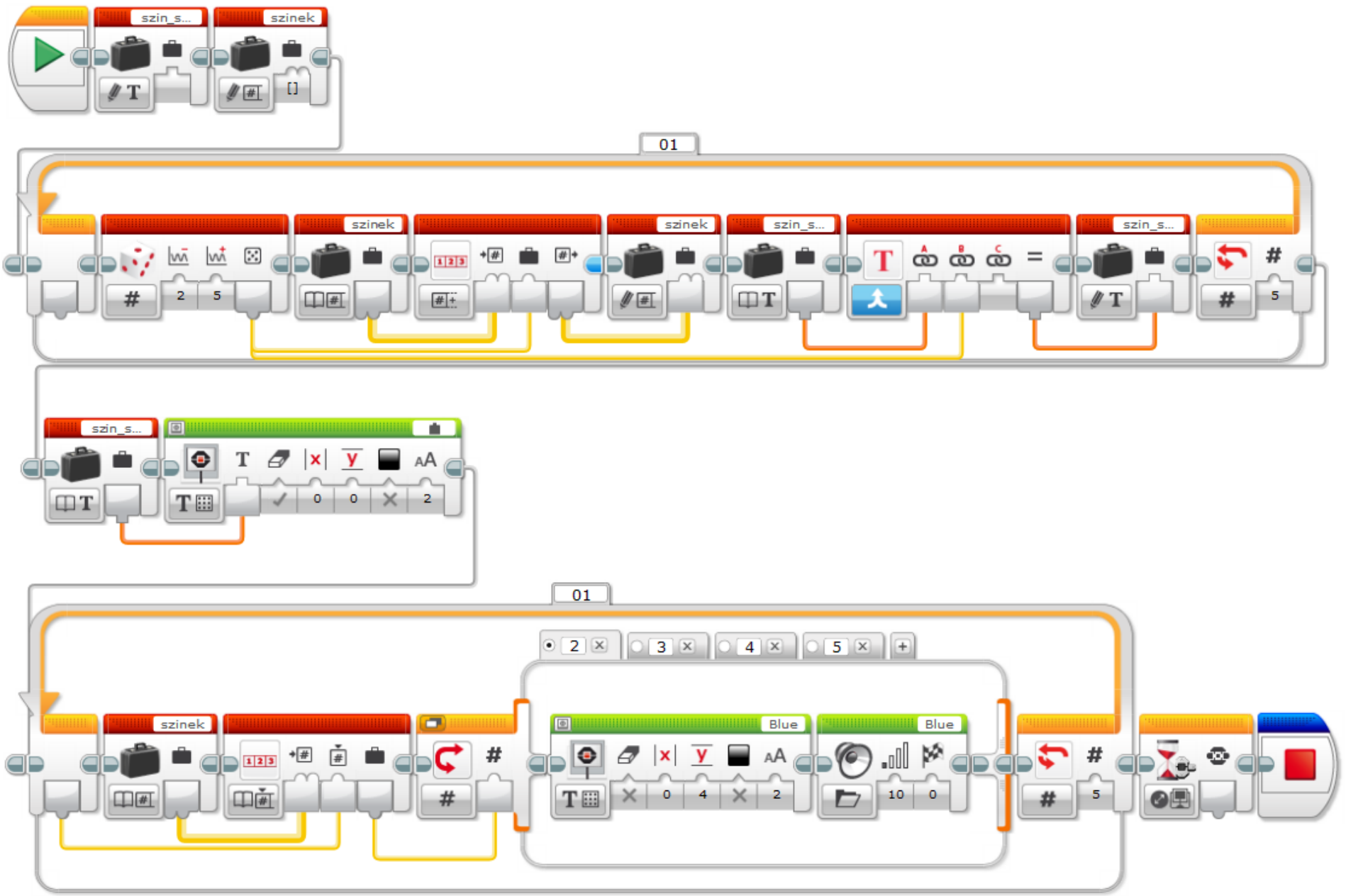
2. *Hozz létre egy ötelemű tömböt, amiben 2 és 5 közötti értékek szerepelnek! Az értékeket közvetlenül egymás után írva jelenítsd meg a kijelzőn!*

   A megoldás olvashatóbb, kiegészített változata a következő oldalon található.



3. *Bővítsük ki az előző feladatot! A tömbben lévő értékeket értelmezzük a színérzékelő által érzékelt színeknek és ez alapján „mondja ki" a robot az adott értékhez tartozó szín nevét angolul. (Ha rendelkezésre áll több idő, akkor javaslom, hogy vegyétek fel magyarul a színek neveit a Sound Editor segítségével és ezt használjátok fel a színek kimondásakor.*

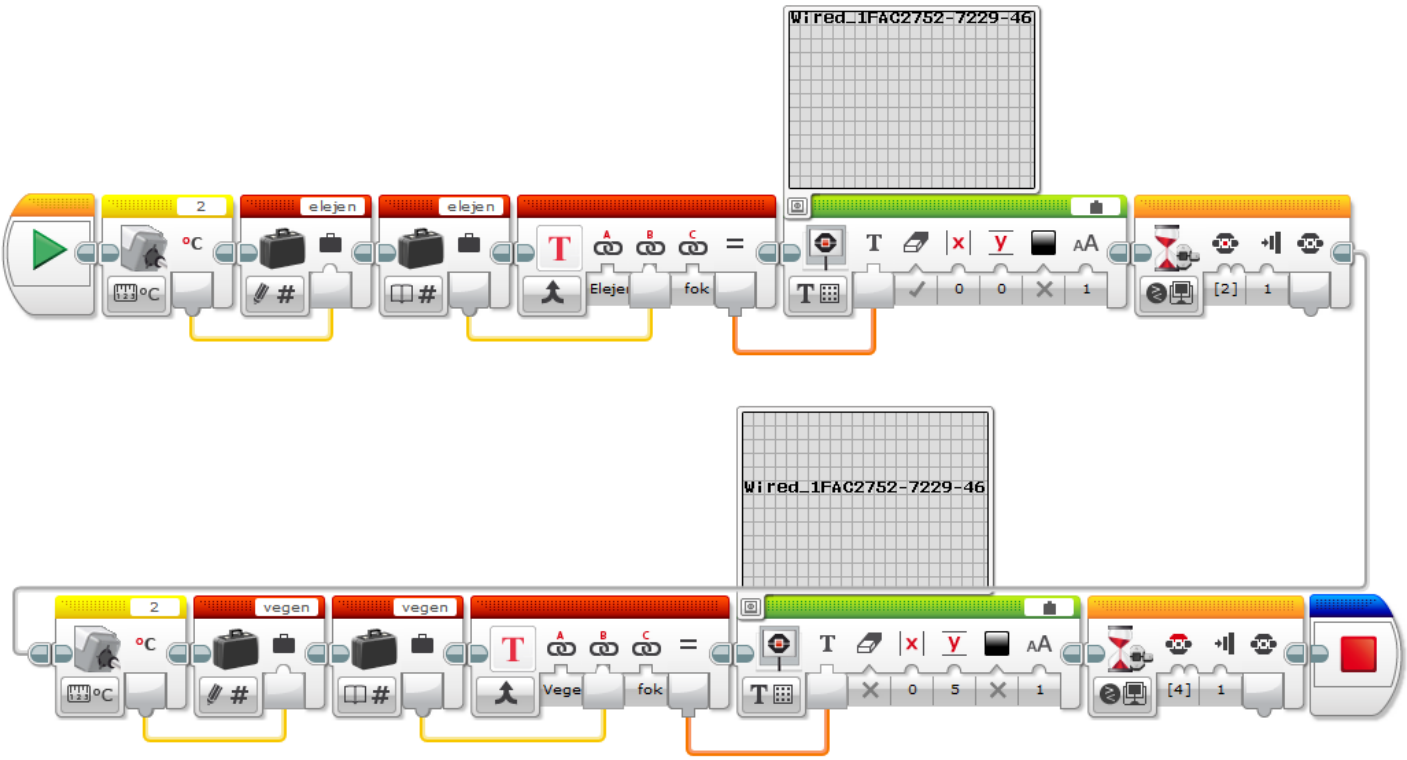
   A megoldás a következő oldalon található.


szin_s...
szinek
01
2
5
szinek
szinek
szin_s...
szin_s...
5
szin_s...
0
0
2
01
2
3
4
5
szinek
Blue
Blue
0
4
2
10
0
5

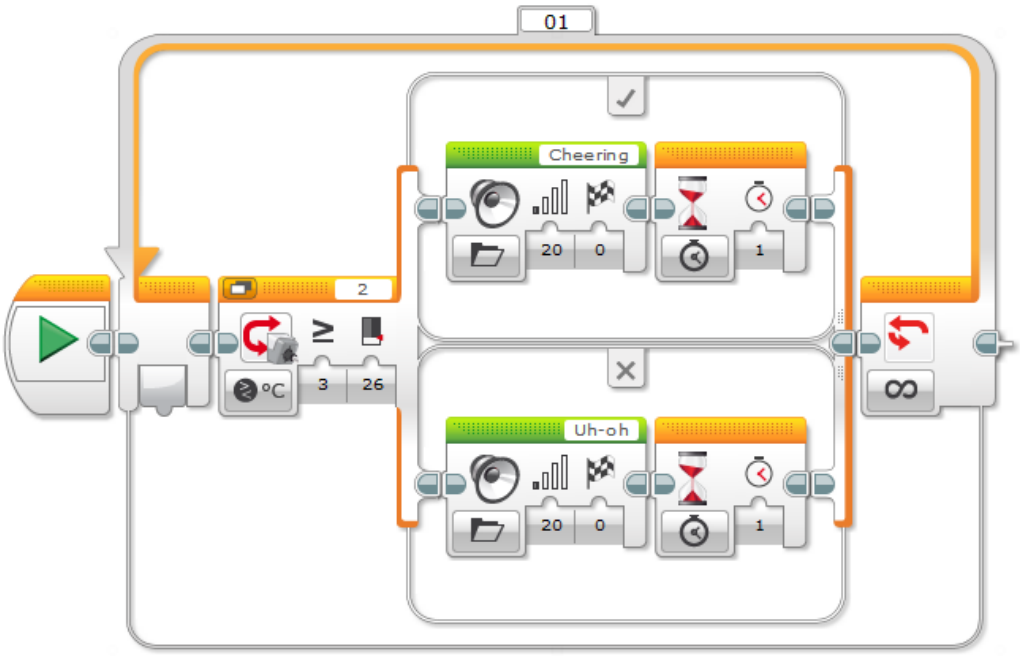
5.5. 5.alkalom

Hőmérsékletmérő használata

1. *Készíts programot, melyben megjeleníted az indításkor érzékelt hőmérsékletet, majd gombnyomás után valamilyen folyadék hőmérsékletét! A program befejezése előtt legyen látható a program elején és végén mért hőmérséklet is!*



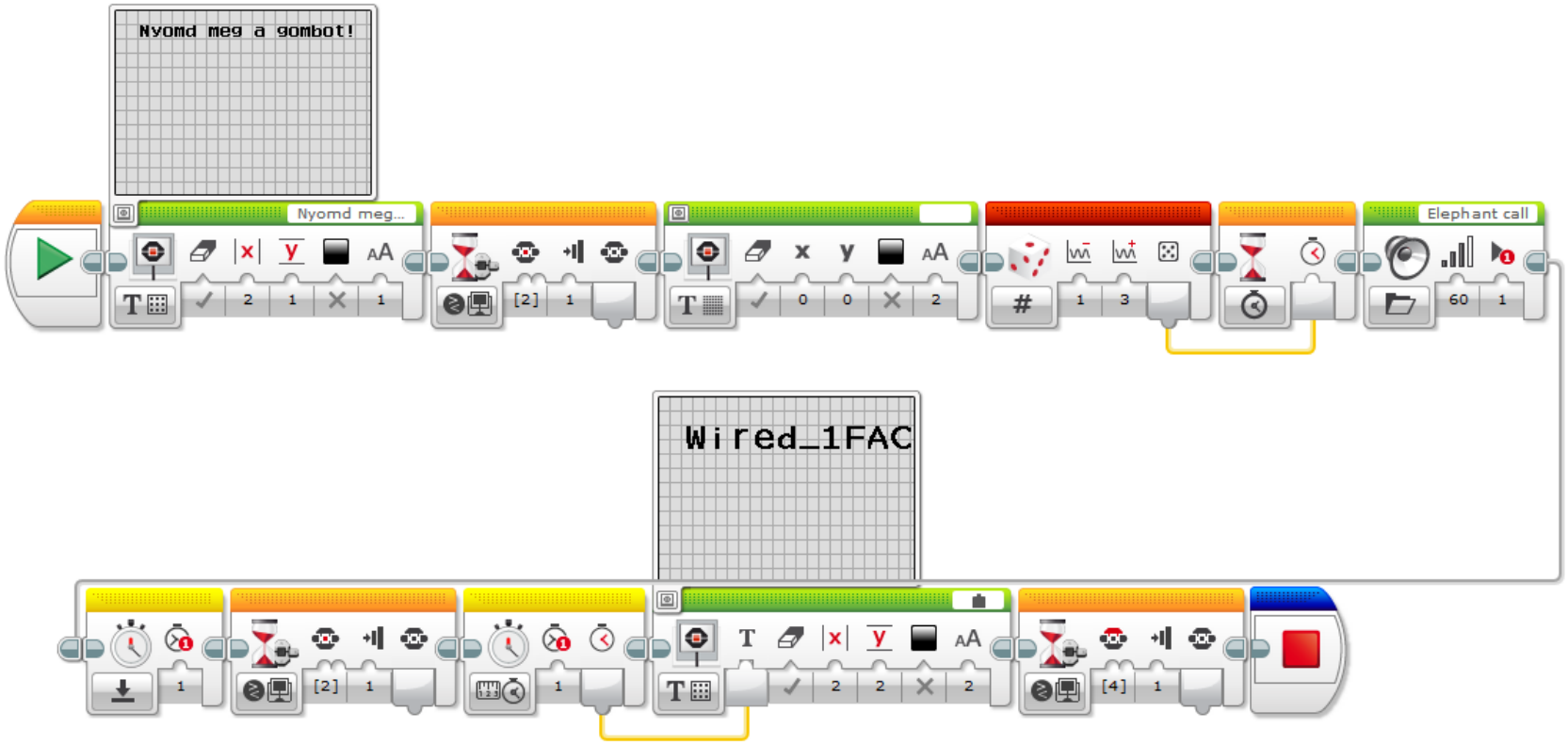
2. *Készíts programot, melyben ha a jelenlegi átlagos hőmérséklettől magasabbat mérsz, akkor pozitív hangjelzést ad a robot, míg ha alacsonyabbat, akkor negatívat hangot játszik le!*
   A hőmérsékletérzékelő szobahőmérsékleten 25-26°C-ot érzékelt, ezért szerepel 26°C a feladatban. A feladat futtattása előtt mindenképp érdemes megnézni, hogy hány fokot érzékel alaphelyzetben a szenzor, és ehhez viszonyítani a feladat megoldásában beállított értékeket.



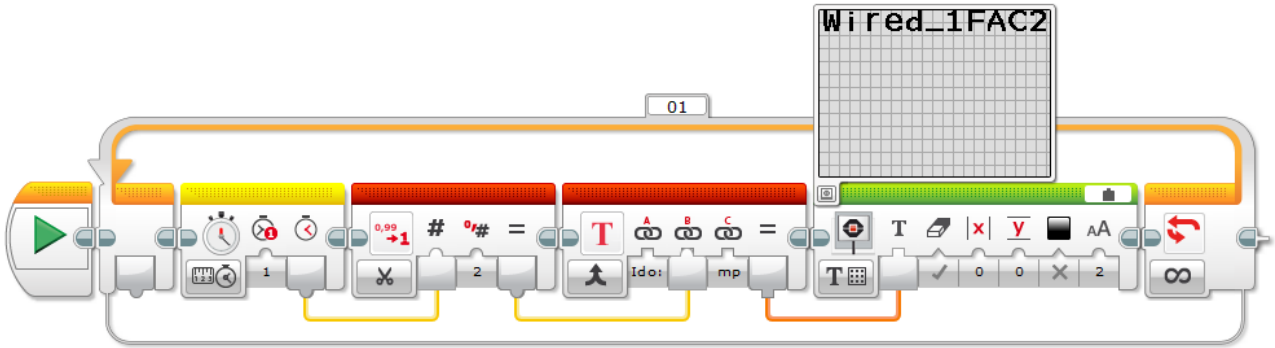
Stopperek használata

1. *Készíts programot, amelyben megméred, hogy egy fordulatot mennyi idő alatt tesz meg a robot! Az időt jelenítsd meg a kijelzőn is.*

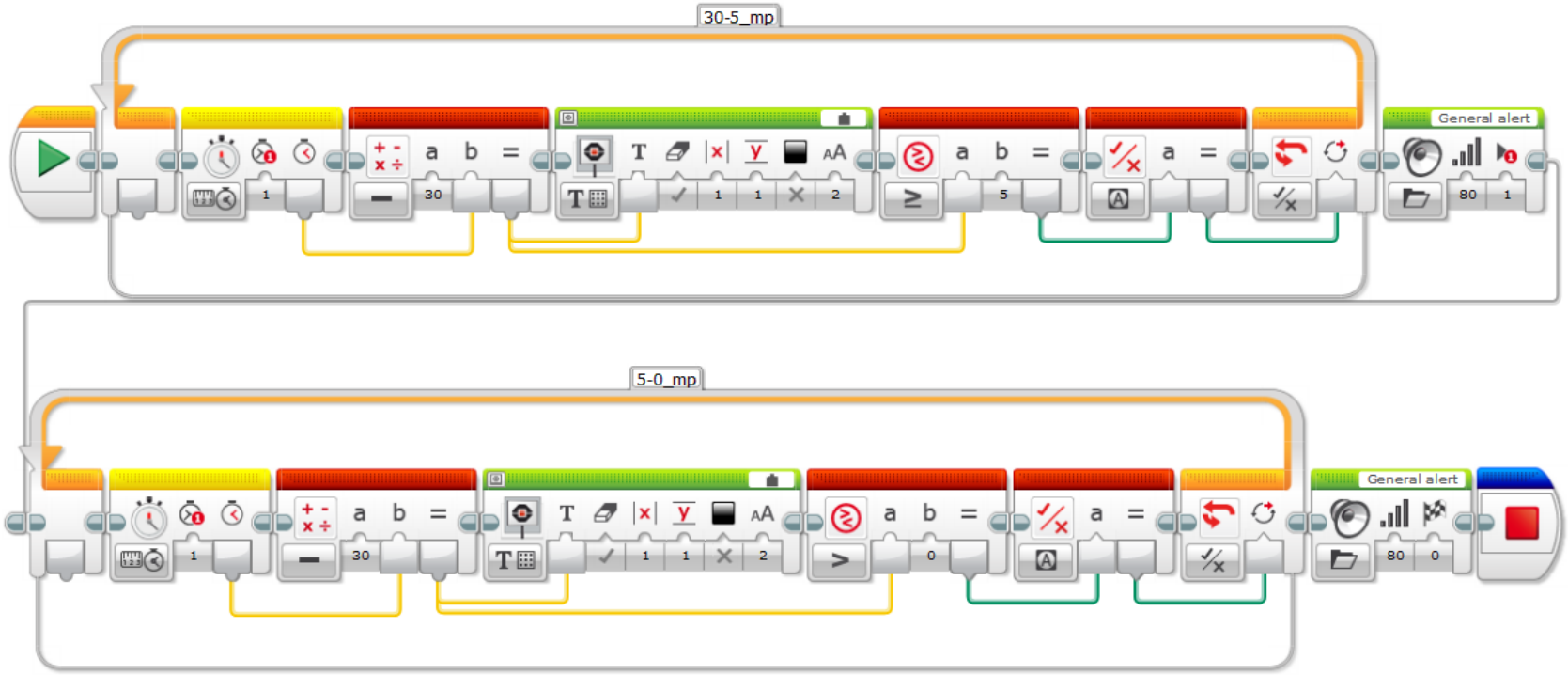
2. *Teszteld a gyorsaságod! Készíts egy reakcióidőt mérő játékot, melyben egy hang lejátszása után, meg kell nyomnod a középső gombját a robotnak. A hang lejátszása és a gomb megnyomása közötti időt mérd meg és jelenítsd meg a kijelzőn! Azért, hogy ne legyen olyan könnyű a játék, oldd meg, hogy véletlen mennyiségű idő teljen el az indítás és a hang lejátszása között! Teszteljétek a játékot! Ki a leggyorsabb?*



3. *Készíts egy programot, amelyben folyamatosan megjeleníted a kijelzőn a futtatás elindítása óta eltelt időt két tizedesjegy pontossággal a következő formátumban: „Ido: mp”! (Az vonal jelöli az eltelt időt.)*



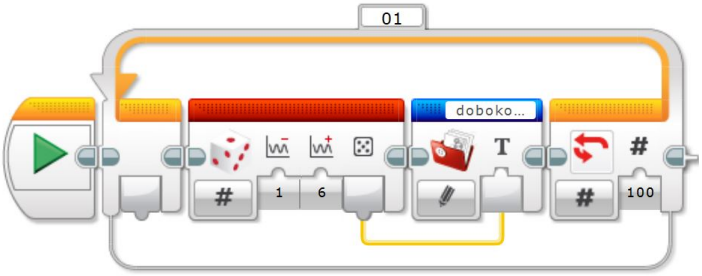
4. *Készíts egy visszaszámlálót, ami 30 másodperctől indul és hangjelzéssel figyelmeztet, ha már csak 5 másodperc van hátra! Ezen kívül akkor is jelezzen, ha lejárt az idő! A rendelkezésünkre álló időt mindneképp jelenítsd meg a kijelzőn! (Tipp: 30-ból vond ki az eltelt időt.)*



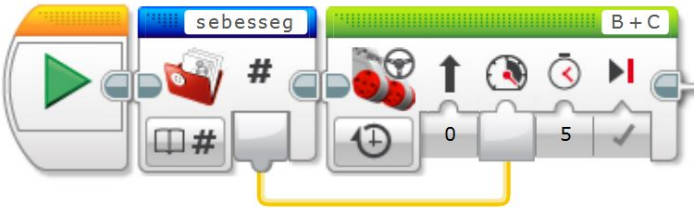
5.6. 6.alkalom

Fájlok elérése és módosítása

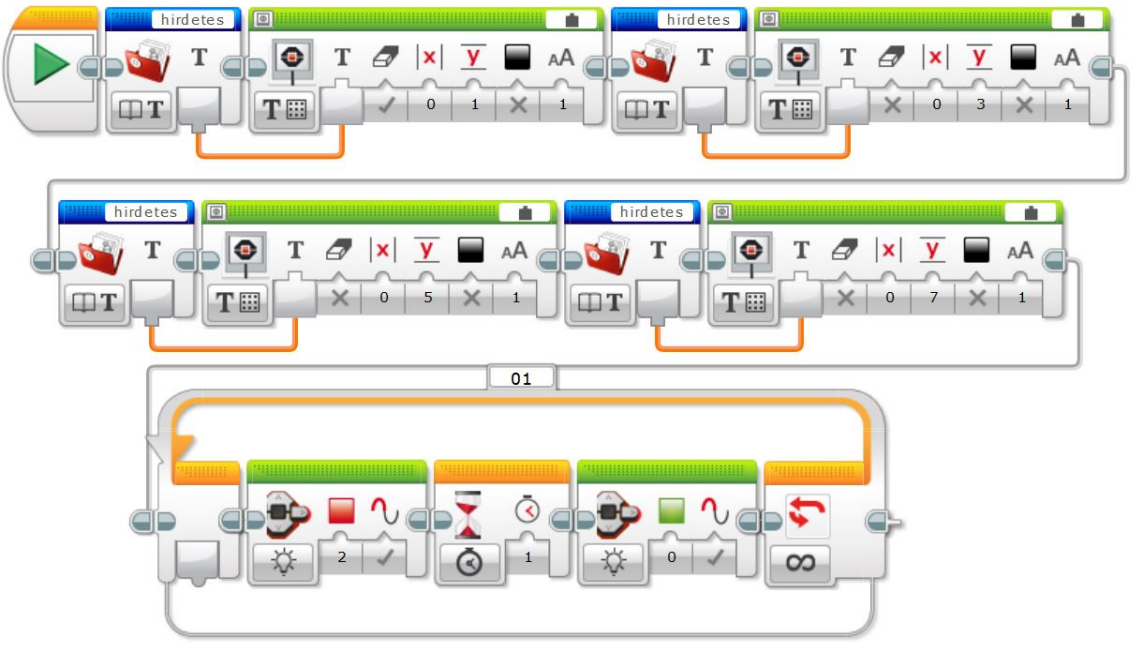
1. ***Dobókocka robot:*** *Robotod hozzon létre egy dobokocka nevű fájlt. Ennek tartalma az általa véletlenszerűen generált 100 db dobókocka dobás eredménye legyen (vagyis 1 és 6 közötti egész számok). Ellenőrizd a fájl tartalmát letöltve azt számítógépedre!*



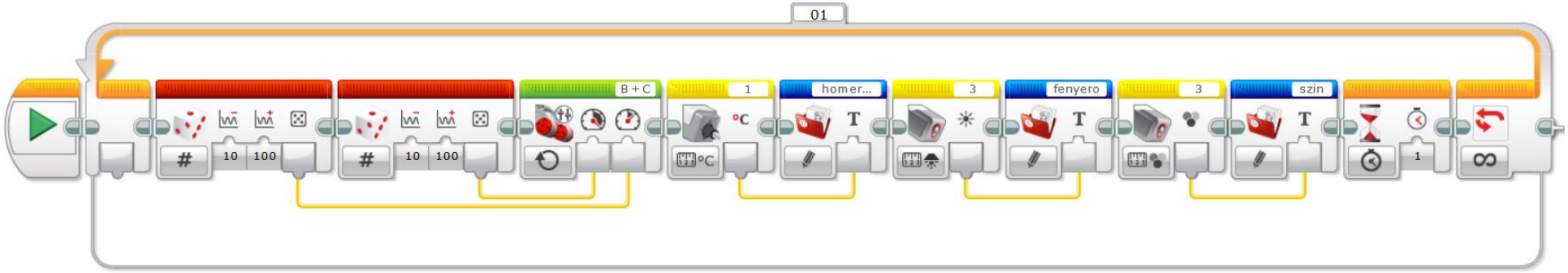
2. *Hozz létre egy sebesseg.rtf nevű fájlt, mely egyetlen szám értéket tartalmaz! Robotod haladjon egyenesen, sebességét a fájlból kiolvasott adat határozza meg!*



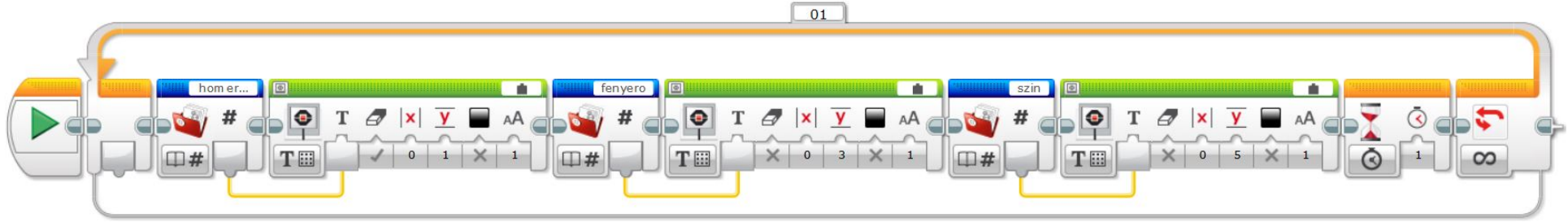
3. ***Hirdetőtábla robot:*** *Találj ki egy rövid hirdetés szöveget és töltsd fel azt robotodra hirdetes.rtf néven! Készíts programot, amely kiolvassa a hirdetes nevű fájl tartalmát és az első négy sort kiírja a robot képernyőjére félkövér betűtípussal! Mivel egy hirdetésnél a lényeg a nagy nézettség, robotod villogtassa ledjét, hogy felhívja magára a figyelmet!*



4. ***Felderítő robot:*** *Robotod nagy kaland előtt áll! Küldetése, hogy felderítést végezzen egy távoli bolygón. Ehhez a következő feladatokat kell teljesítenie: tetszőleges mozgást végezve pásztázzon át egy területet, eközben másodpercenként mérje meg a hőmérsékletet, a környezeti fényerősséget és állapítsa meg milyen színű a talaj! Ezeket az információkat jegyezze fel három különböző fájlba! (A fájloknak beszédes nevet adj, mert még szükség lesz rájuk!)*



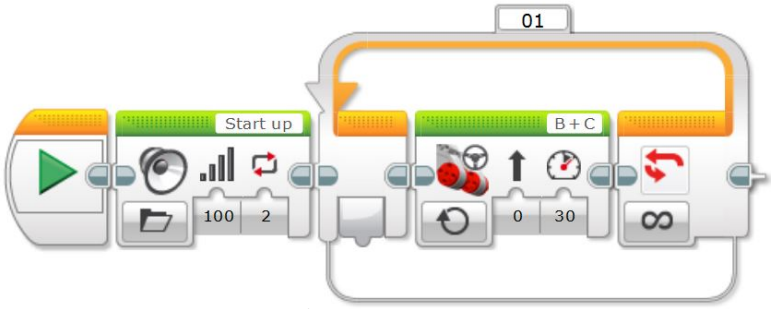
5. ***Felderítő robot 2:*** *Amikor robotod sikeresen végzett a bolygó feltérképezésével, hazatérve beszámol a mérési eredményekről. Másodpercenként kiolvas 1 elmentett hőmérséklet értéket, 1 fényerősséget és egy talajszínt a mentett fájlokból. Ezeket az adatokat egymás alá kiírja képernyőjére. (Továbbfejlesztési lehetőség: a képernyőn az értékeket úgy jeleníti meg, hogy minden sor erején feltünteti a kategória nevét. Pl.: hőmérséklet: mért érték*



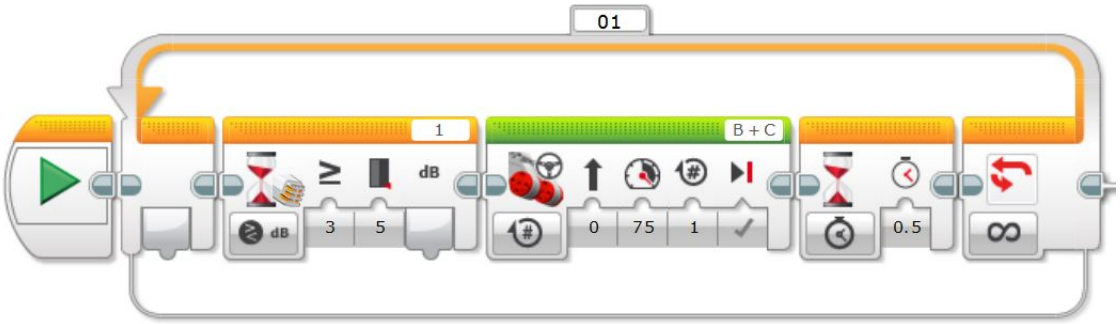
5.7. 7.alkalom

EV3 robotok közötti kommunikáció

1. ***Bemelegítő feladat*** *- Robotok interakciója üzenetküldés nélkül: A következő feladatban két robotra lesz szükséged! Az egyik robot feladata, hogy "énekelve" egyenesen haladjon előre közepes sebességgel. A másik robot nem szereti, ha barátja hangosan énekel a fülébe, így ha közel ér hozzá, a nagy hangerő hatására kicsit távolabb megy. Mindig elmenekül, hogyha hangoskodnak mellette. (A feladat végrehajtásához az éneklő robot a másik robot mögött helyezkedjen el, tőle legalább 1 méteres távolságot hagyva kezdetben!)*

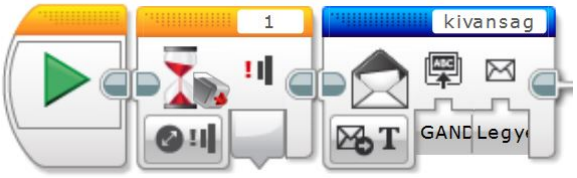


91. ábra. Éneklő robot programja

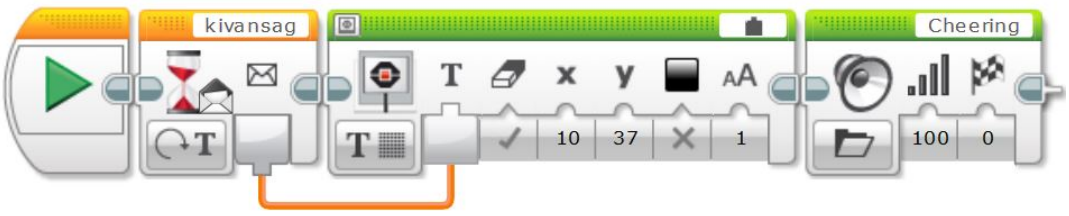


92. ábra. Menekülő robot programja

2. *Készíts programot, melynek hatására Rozi (az egyik robot) a nyomásérzékelő változásának hatására szöveges üzenetet küld barátjának! Gandalf (az üzenetet fogadó robot) az üzenet érkezésekor azonnal kiírja az üzenet szövegét a képernyőjére és hangosan örül barátja jelentkezésének.*

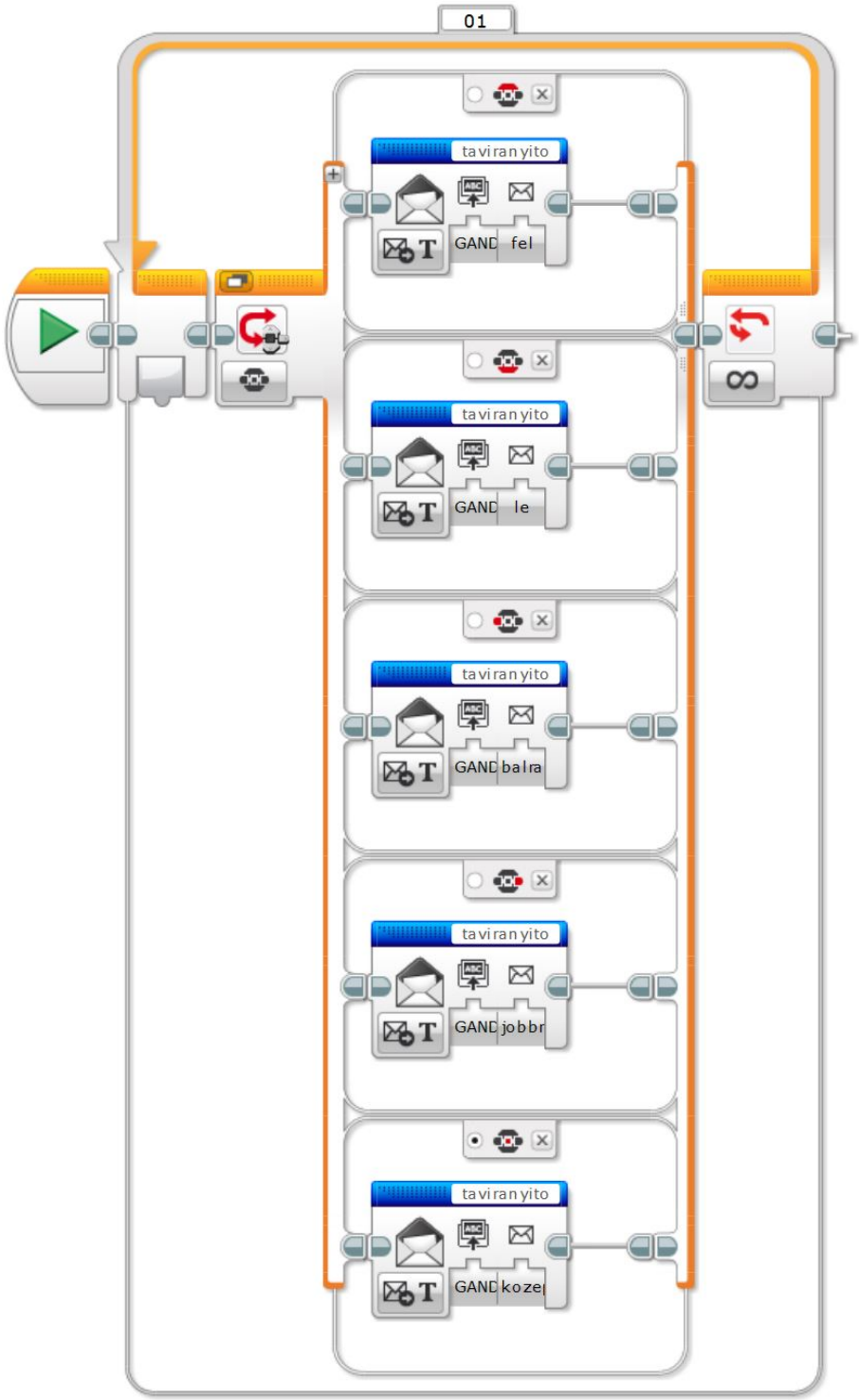


93. ábra. Rozi programja

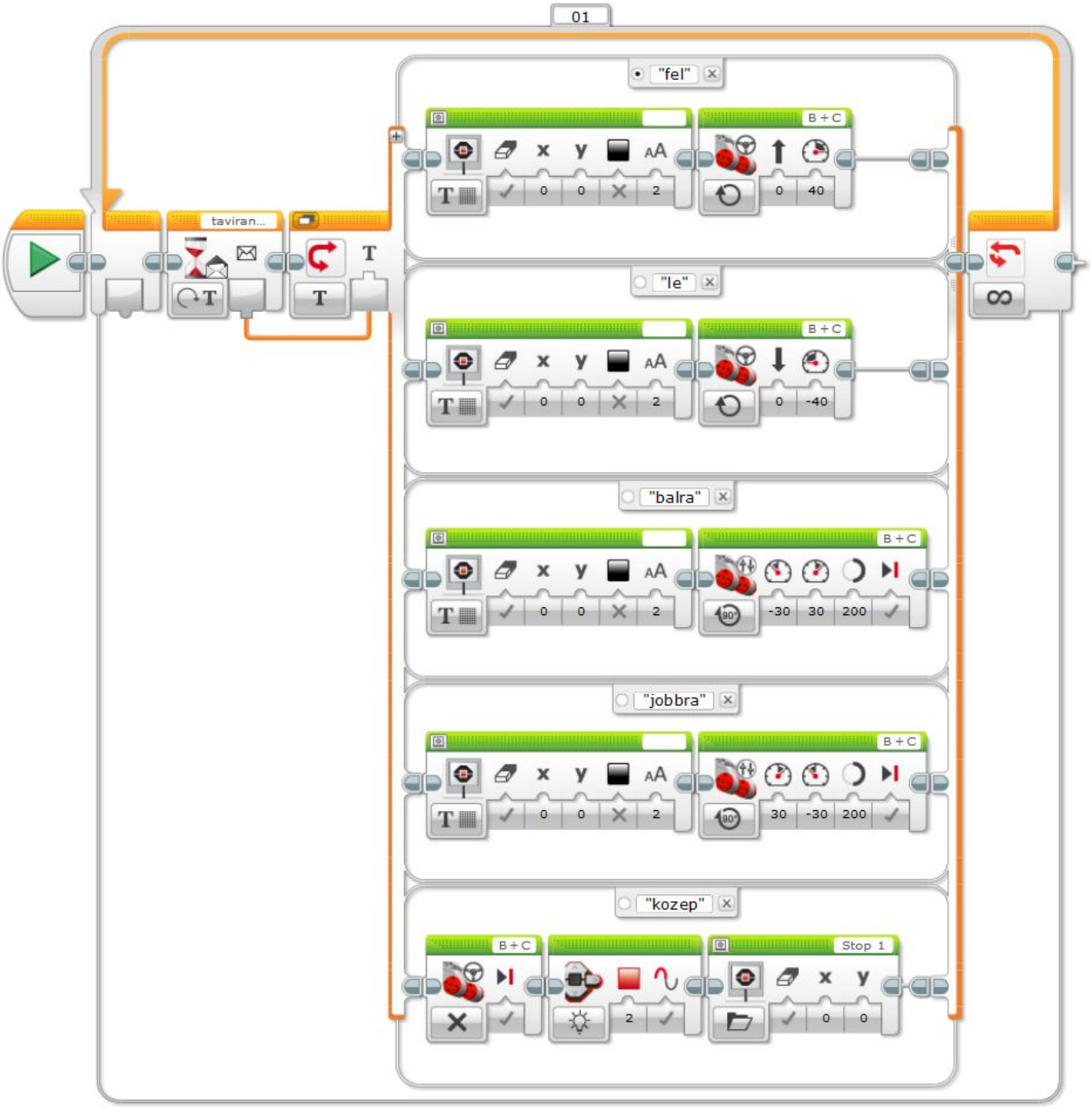


94. ábra. Gandalf programja

3. *Alkoss távirányító robotot! Most az egyik robot távirányítóként fog szolgálni, és ezzel fogjuk irányítani a másik robotot. Használd a tégla programozható gombjait! Amíg nyomva tartjuk az irányító robot felső gombját, az irányított robot menjen folyamatosan előre, az alsó gomb hatására hátra. A bal gombot megnyomva forduljon balra 90°-ot, a jobb gomb hatására jobbra ugyanennyit. Ha a középső gombot nyomjuk meg, villogtassa pirosan ledjét és rajzoljon STOP táblát képernyőjére. Ez a rajz ne legyen a képernyőn, amikor a robot mozgásban van! (Tipp: az elágazás szerkezet Text módjában felhasználhatjuk a kapott szöveges üzenet tartalmát.)*

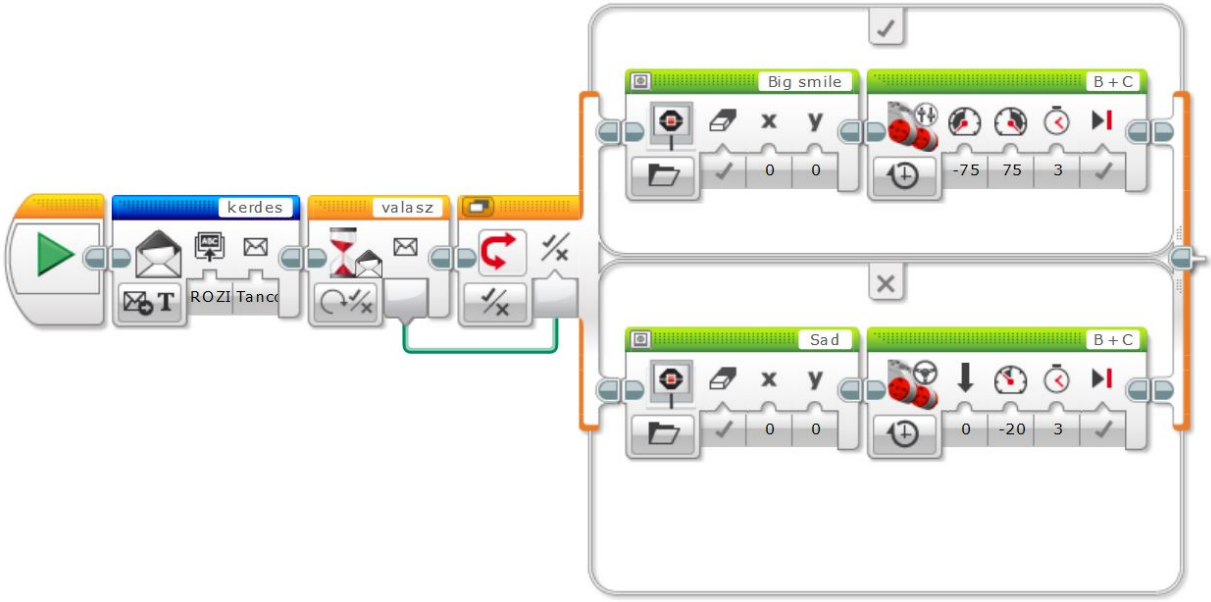


95. ábra. Rozi programja

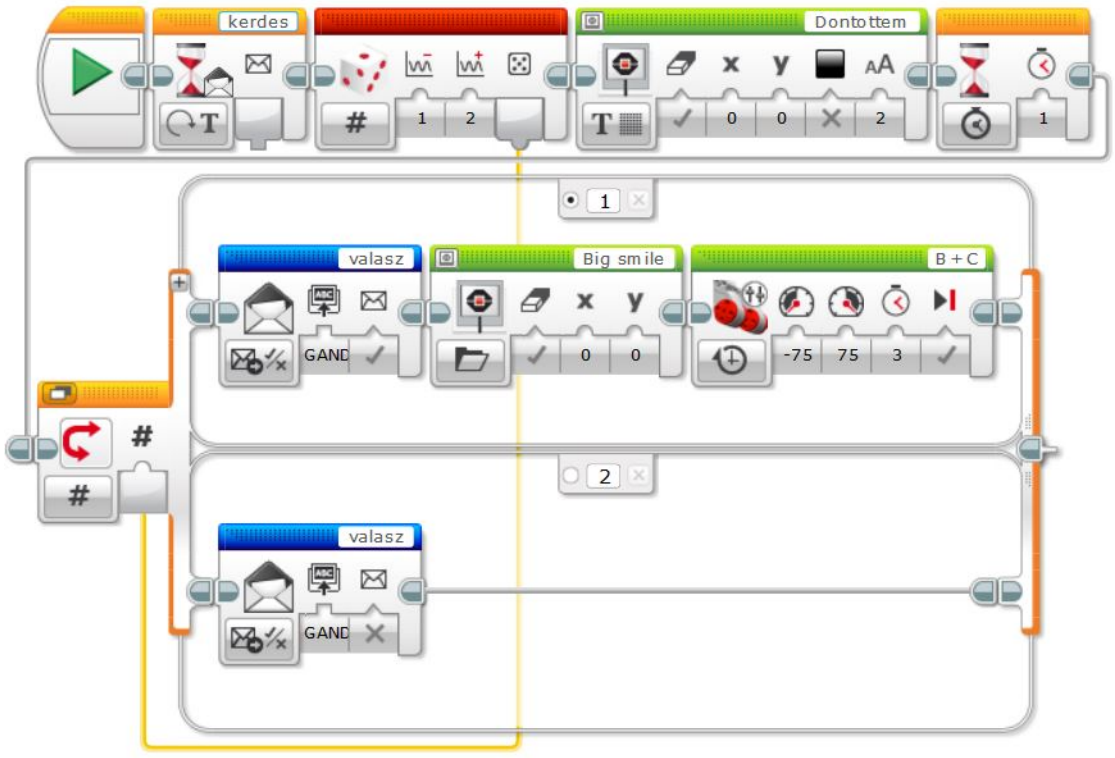


96. ábra. Gandalf programja

4. *A következő feladatban Gandalf üzenetben felkéri Rozit táncolni. Rozi a véletlenre bízza a dolgot, 50% valószínűséggel elfogadja a felkérést. Amikor döntött, kiírja képernyőjére, hogy "dontottem" és válaszol Gandalfnak. Ha Rozi igennel válaszolt, akkor mindketten mosolyognak és körbe forognak 3 másodpercig. Ha Rozi elutasította a felkérést Gandalf szomorú lesz és távolabb megy.*

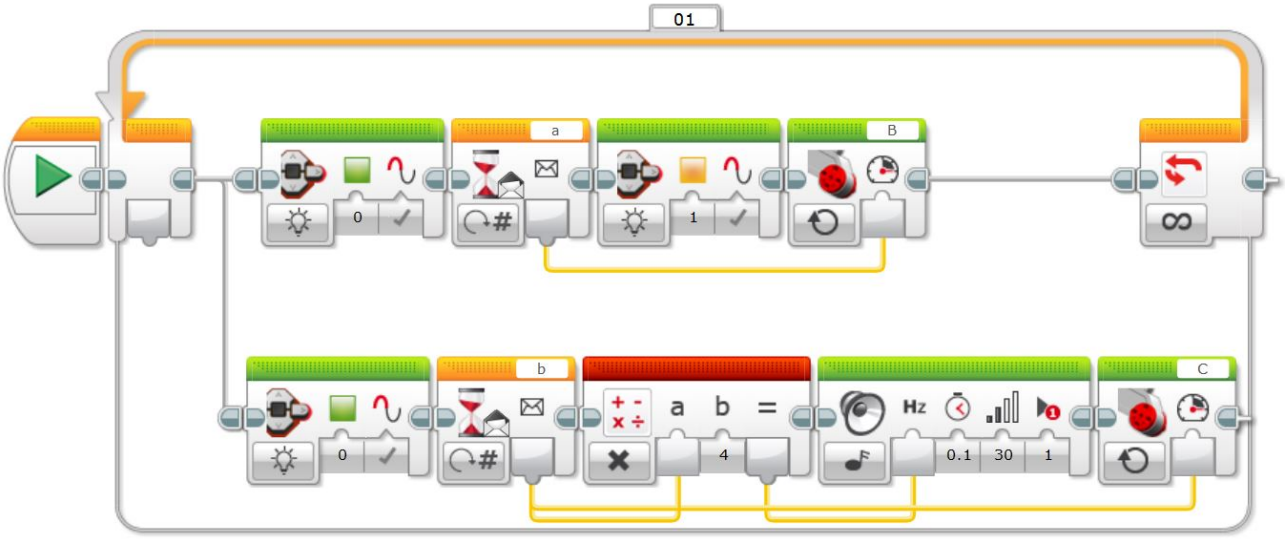


97. ábra. Gandalf programja

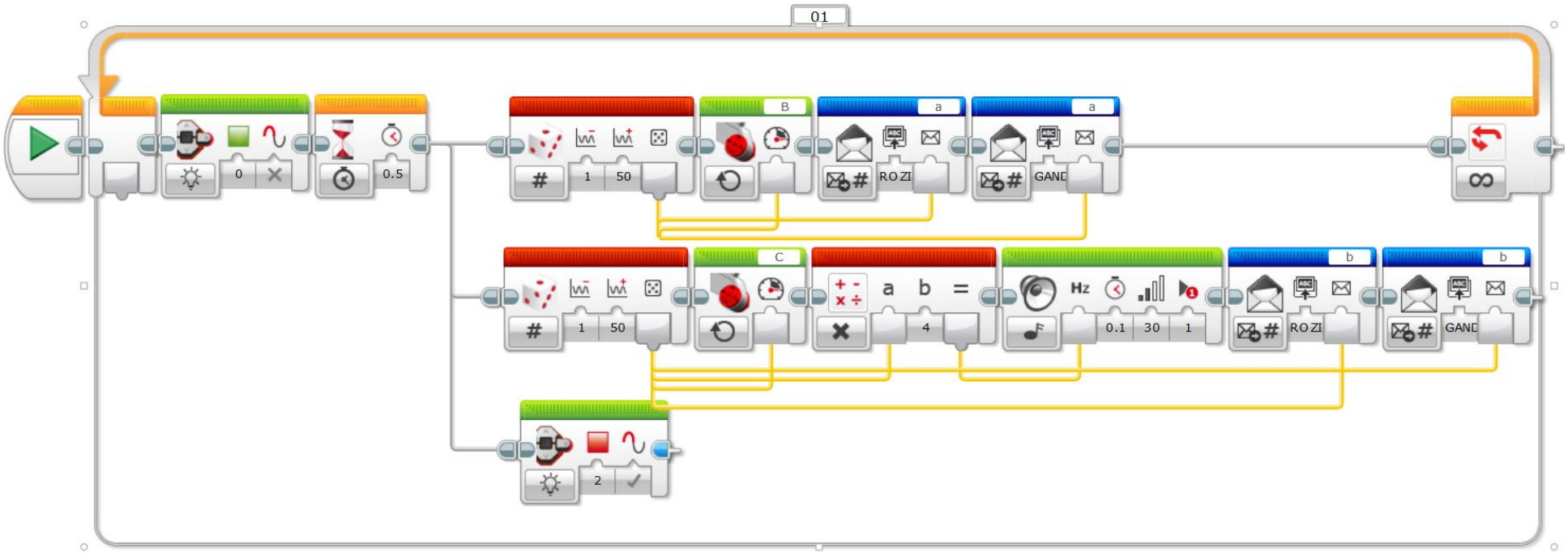


98. ábra. Rozi programja

5. *Táncoló robotok - csak szinkronban! A következő feladathoz akár 3 robotot is használhatsz! Legyen egy kitüntetett robot, aki a legjobb táncos. Ő random generált értékek szerint állítgatja motorjait. Erről információt küld folyamatosan a két "háttértáncosnak" akik próbálják őt ennek segítségével utánozni. Ha jól dolgoztál, a robotok szinkronban táncolnak! :)*



99. ábra. Fogadó, utánzó robotok programja



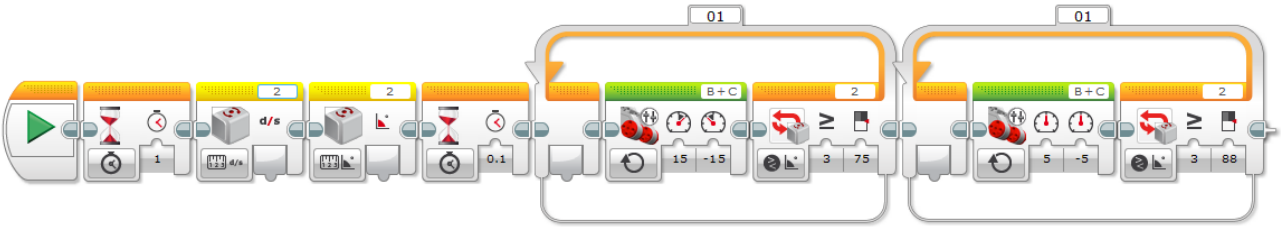
100. ábra. Kitüntetett, koreográfiát küldő robot programja

5.8. 9.alkalom

Saját blokk létrehozása

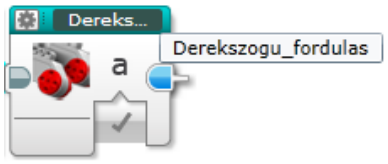
1. *Készíts egy olyan blokkot, amivel jobbra 90°-ot lehet fordulni!*

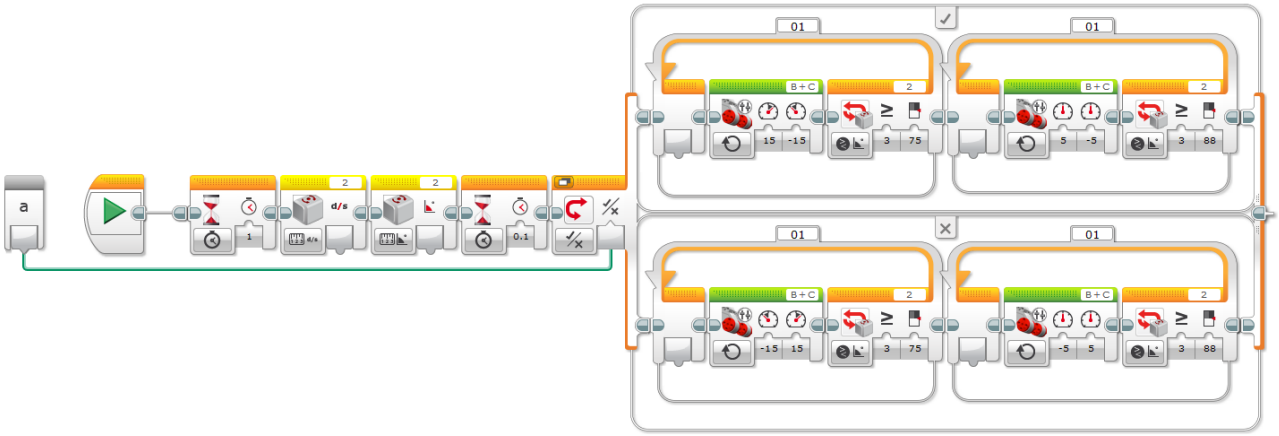
   A blokknak nincs semmilyen paramétere, így nem jelenik meg semmilyen paraméterátadó szürke blokk.



2. *Készíts egy olyan blokkot, amivel bármelyik irányba fordulhatunk 90°-ot!*

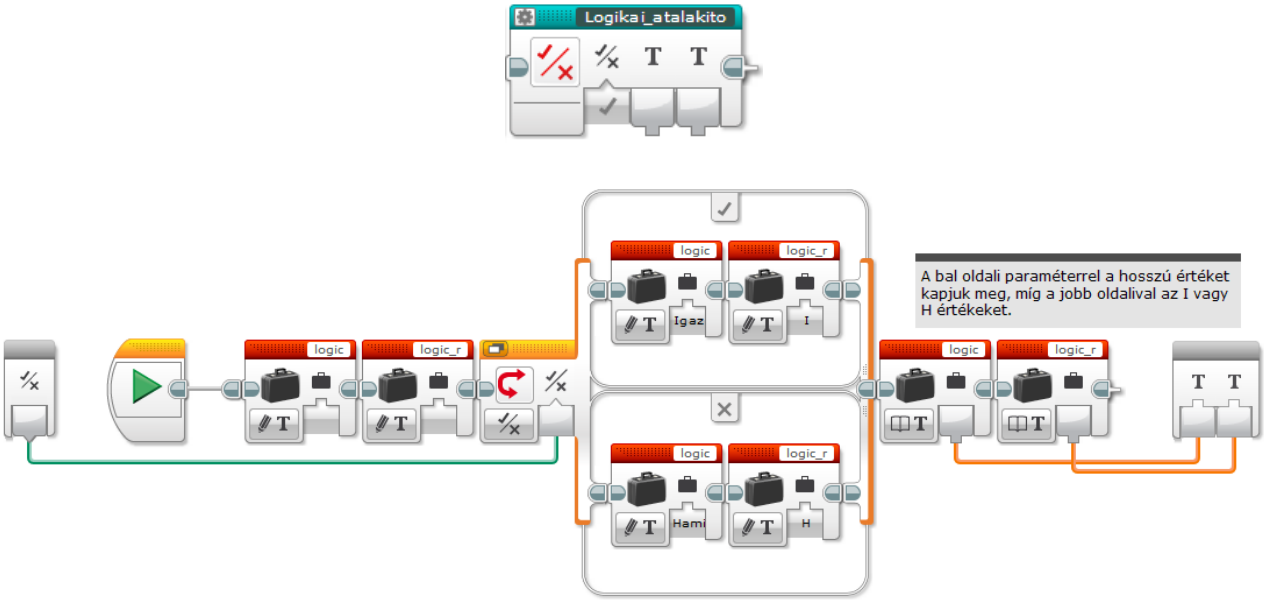
   A blokknak én egy logikai bemenő paramétert adtam (neve: Jobbra?), melyben az alapértelmezett érték az igaz, és a jobbra fordulást jelenti. Ettől eltérő megoldás is elfogadható, ha megfelelően működik.





3. *Készíts egy olyan blokkot, amivel a gyro szenzort kalibrálod!*

4. *Készíts egy olyan blokkot, amivel a logikai értékeket átalakítod szöveggé! Oldd meg, hogy hosszú (Igaz/Hamis) és rövid (I/H) formában is felhasználható legyen a szöveg!*



6. Irodalomjegyzék

[1] Barbalics Dóra Krisztina. Lego mindstorms ev3 robotok programozása ii., szakköri segédanyag tanárok számára. Master's thesis, Eötvös Loránd Tudományegyetem, Informatikai kar, Média- és Oktatásinformatikai Tanszék, 5 2020.

[2] Barbalics Dóra Krisztina; Solymos Dóra. Lego Mindstorms EV3 robotok programozása. ELTE Informatikai Kar, http://tet.inf.elte.hu/tetkucko/wp-content/uploads/2018/12/legomindstorms_szakkorianyag.pdf, 2018. Utoljára megtekintve: 2020.11.02.

[3] A gyro szenzor kalibrálásának módjai. (angol nyelvű). https://ev3lessons.com/en/ProgrammingLessons/advanced/GyroRevisited.pdf. Utoljára megtekintve: 2020.10.29.

[4] Using the infrared sensor beacon mode (angol nyelvű). https://ev3-help-online.api.education.lego.com/Education/en-gb/page.html?Path=editor%2FUsingSensors_Infrared_Beacon.html. Utoljára megtekintve: 2020.11.07.

[5] Giroszkóp (magyar nyelvű). https://hu.wikipedia.org/wiki/Giroszk%C3%B3p. Utoljára megtekintve: 2020.10.03.

[6] Giroszkóp (angol nyelvű). https://en.wikipedia.org/wiki/Gyroscope. Utoljára megtekintve: 2020.10.03.

[7] Az ev3 gyro szenzorának néhány adata. (angol nyelvű). https://education.lego.com/en-us/products/lego-mindstorms-education-ev3-gyro-sensor-/45505. Utoljára megtekintve: 2020.10.28.

[8] A gyro szenzor kalibrálásának rövidebb módjai és azok magyarázata. (angol nyelvű). https://ev3lessons.com/en/ProgrammingLessons/advanced/Gyro.pdf. Utoljára megtekintve: 2020.10.31.